+/- Java
Semantica Operazionale

A. Di Marco, P. Inverardi, B. Intrigila, F. Mancinelli, P. Pelliccione

16 marzo 2005

1 La semantica operazionale del +/-Java

A partire da questo paragrafo affronteremo lo studio della semantica operazionale di un frammento di linguaggio di programmazione orientato agli oggetti. Tale linguaggio contiene i principali costrutti del linguaggio Java, ma con delle semplificazioni che ne rendono più semplice la trattazione. Ci riferiremo al linguaggio che tratteremo con il termine +/-Java mantenendo traccia così della sua origine.

2 Struttura dello stato

Precedentemente abbiamo introdotto lo stato per modellare associazioni tra nomi e valori. Nel seguito per poter trattare la programmazione orientata agli oggetti avremo bisogno di una struttura dello stato più sofisticata dove le associazioni nomi valori saranno solo una parte della struttura stato, che chiameremo stato principale. Per poter trattare correttamente caratteristiche più complesse del linguaggio, quale l'annidamento dei blocchi e la chiamata a metodo, è necessario dare allo stato principale una organizzazione più complessa. Intuitivamente invece di trattarlo come una tabella piatta di associazioni tra identificatori e valori, lo tratteremo come una pila ovvero come una sequenza di tabelle, ciascuna delle quali è un insieme di associazioni identificatori valori. Useremo il termine *environment* per far riferimento a una singola tabella. Si consideri la rappresentazione grafica del seguente stato principale ρ (Figura 1) costituito da due environment φ_1 e φ_2.

L'identificatore x ha due valori nello stato principale, uno nell'*environment* φ_1 ed uno nell'environment φ_2. Qual è il valore di x nello stato principale ρ sopra indicato? È il primo valore che troviamo associato a x, ricercando x a partire dall'environment più recente: il più vicino al top della pila.

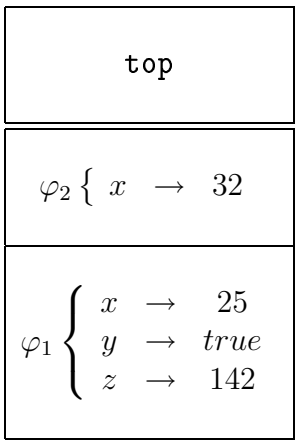

Figura 1: Stato principale ρ

Ricordiamo che gli *environment* vengono aggiunti sempre al top della pila. Nell'esempio precedente φ_2 è stato impilato su φ_1 e il valore di x risulta quindi 32. Da questo momento in poi, chiameremo lo stato principale *stack*. Possiamo formalizzare la nuova struttura dello *stack* nel seguente modo. Un environment φ è una funzione che associa ad un numero finito di identificatori un valore diverso da $\perp$, e $\perp$ a tutti gli altri:

$$\varphi : \texttt{IDE} \rightarrow Val \cup \{\perp\}$$
dove $Val = \mathbb{Z} \cup \mathbb{B}$

Uno *stack* ρ è una sequenza di *environment*. Denotiamo con Ω lo *stack* vuoto. Indichiamo con Φ l'insieme degli environment φ ammissibili. L'insieme $\mathcal{S}$ degli *stack* legali è definito come:

$$\mathcal{S} = \{\Omega\} \cup \{\varphi \cdot \rho \mid \varphi \in \Phi,\ \rho \in \mathcal{S}\}$$

Le operazioni per inserire, eliminare e recuperare l'environment al top dello stack sono definite come:

$$\begin{aligned} push &: \Phi \times \mathcal{S} \rightarrow \mathcal{S} \\ pop &: \mathcal{S} \rightarrow \mathcal{S} \end{aligned}$$

L'accesso ad un identificatore è definito come:

$$\rho(x) = \begin{cases} \perp & \text{se } \rho = \Omega \\ \varphi(x) & \text{se } \rho = \varphi \cdot \rho' \text{ e } \varphi(x) \neq \perp \\ \rho'(x) & \text{se } \rho = \varphi \cdot \rho' \text{ e } \varphi(x) = \perp \end{cases}$$

Definiamo ora la semantica della modifica del valore associato ad un identificatore nello stack ρ.

$$\rho[{}^{v}/_{x}] = \begin{cases} \Omega & \text{se } \rho = \Omega \\ \varphi[{}^{v}/_{x}] \cdot \rho' & \text{se } \rho = \varphi \cdot \rho' \text{ e } \varphi(x) \neq \bot \\ \varphi \cdot \rho'[{}^{v}/_{x}] & \text{se } \rho = \varphi \cdot \rho' \text{ e } \varphi(x) = \bot \end{cases}$$

Osserviamo che le scritture $\rho[{}^{v}/_{x}]$ e $\varphi[{}^{v}/_{x}]$ determinano due comportamenti differenti sullo stack: la seconda modifica sempre lo stack ed esattamente modifica l'environment più recente dello stack, mentre la prima può modificare un environment meno recente o addirittura potrebbe non avere effetto sullo stack se la variabile di cui si vuole modificare il valore associato non è stata mai dichiarata. Infine indicheremo con ω l'environment vuoto che associa $\bot$ a tutti gli identificatori:

$$\omega : \texttt{IDE} \rightarrow \{\bot\} \quad \text{dove } \omega \in \Phi.$$

3 Stato e oggetti

Il modello di stato usato finora consente di trattare variabili con tipo basico: numerico e booleano. In Java ogni tipo complesso, a partire dalle stringhe, è trattato con il paradigma degli oggetti. La natura di questi oggetti è definita delle classi che ne dichiarano sia la struttura sia le operazioni che ne permettono la modifica. Per trattare gli oggetti è necessario arricchire lo stato. Da questo punto in poi lo stato σ verrà visto come una terna così definita:

$$< \rho, \mu, \delta > \text{ dove:}$$

$$\rho : \texttt{IDE} \rightarrow Val^*$$
$$Val^* = Val \cup Loc \cup \{\texttt{null}\} \cup \{\bot\}$$

$$\mu : Loc \rightarrow \Phi \times \Phi$$
$$\delta : \texttt{IDE\_CLASS} \rightarrow \Phi \times \Phi$$

Chiameremo *stack* l'elemento ρ, *heap* l'elemento μ e *library* l'elemento δ.

La *library* viene costruita una volta per tutte utilizzando le dichiarazioni delle varie classi, prima dell'esecuzione del programma vero e proprio. La dichiarazione di una classe ha l'effetto di aggiungere una associazione tra l'identificatore della classe ed una coppia di funzioni $< \varphi_v, \varphi_m >$ contenenti

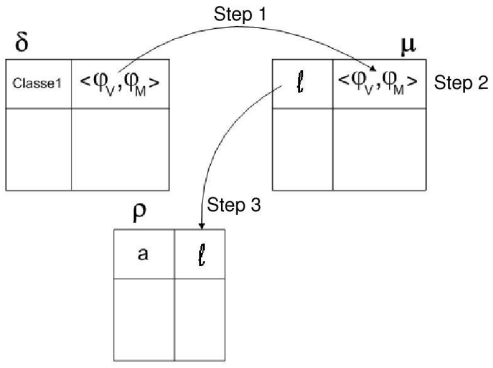

Figura 2: Effetto di `a = new Classe1()`

rispettivamente le variabili ed i metodi definiti all'interno della classe stessa. Il comportamento di ρ è quello definito nella sezione precedente, con un nuovo accorgimento legato alla trattazione degli oggetti. Al momento della dichiarazione di una variabile oggetto, viene aggiunto, come spiegato nella sezione precedente, una coppia nell'environment più recente dell'elemento ρ. La coppia è costituita dall'identificatore della variabile e da un valore riferimento all'oggetto. Se la dichiarazione non prevede l'inizializzazione, il riferimento associato è il riferimento speciale *null*. Tale riferimento rappresenta un puntatore alla struttura effettiva dell'oggetto stesso che viene materialmente memorizzata in μ. A tal proposito bisogna ridefinire opportunamente la funzione φ che, in questo caso, assocerà identificatori a valori o locazioni:

$$\varphi : \texttt{IDE} \rightarrow Val \cup Loc \cup \{\texttt{null}\} \cup \{\perp\}$$

L'istanziazione di un oggetto avviene mediante l'istruzione `a = new Classe1()`. L'esecuzione della `new` ha i seguenti effetti:

Step 1 Creazione della rappresentazione dell'oggetto mediante una coppia di environment $< \varphi_v, \varphi_m >$ contenente una copia delle informazioni reperite da δ ed associate all'identificatore della classe (`Classe1`).

Step 2 Associazione di un riferimento l ($l \in Loc$) nello heap alla rappresentazione dell'oggetto.

Step 3 Inserimento di una nuova coppia nell'environment corrente dello *stack* che associa all'identificatore di variabile `a` il riferimento del passo precedente.

Graficamente quello che avviene è rappresentato in Figura 2.

4 Grammatica

4.1 Semplificazioni del linguaggio

- Le dichiarazioni di classe non hanno modificatori di accesso e sono visibili dalle altre classi dichiarate.
- Non è possibile dichiarare classi annidate.
- Struttura rigida dei programmi: prima le dichiarazioni di classe ed infine una dichiarazione standard della classe `Program` contenente il solo metodo `main` (entry point del programma +/- Java).
- Struttura rigida dei blocchi: prima le dichiarazioni di variabili ed in seguito la sequenza dei comandi.
- Struttura rigida del corpo dei metodi: prima le dichiarazioni di variabili locali, in seguito la sequenza dei comandi ed infine il costrutto `return`.
- Le variabili di istanza sono tutte `private`.
- I metodi definiti dentro una classe sono tutti `public`.
- Restrizione dei tipi basici a `int` e `boolean`.
- Accesso alle variabili di istanza esclusivamente mediante il costrutto `this.IDE`.
- Accesso ai metodi esclusivamente mediante il costrutto `this.METH_IDE`.

4.2 Definizione della grammatica

```
PROGRAM ::= CLASS_DECL_LIST
            public class Program {
              public static void main(String[] IDE) METH_BODY_NO_RETURN
            }

CLASS_DECL_LIST ::= CLASS_DECL CLASS_DECL_LIST |
                    ε

CLASS_DECL ::= class IDE_CLASS { FIELD_DECL_LIST }

FIELD_DECL_LIST ::= FIELD_DECL FIELD_DECL_LIST|
                    ε
```

```
FIELD_DECL ::= FIELD_VAR_DECL |
               METH_DECL |
               CONSTRUCTOR_DECL

FIELD_VAR_DECL ::= private TYPE_NAME IDE;

TYPE_NAME ::= PRIMITIVE_TYPE_NAME |
              IDE_CLASS

PRIMITIVE_TYPE_NAME ::= int |
                        boolean

CONSTRUCTOR_DECL ::= public IDE(PARAM_DECL) METH_BODY_NO_RETURN

METH_DECL ::= public TYPE_NAME METH_IDE(PARAM_DECL) METH_BODY |
              public void METH_IDE(PARAM_DECL) METH_BODY_NO_RETURN

METH_BODY ::= { VAR_DECL_LIST STATEMENT return EXP; }

METH_BODY_NO_RETURN ::= { VAR_DECL_LIST STATEMENT }

PARAM_DECL ::= TYPE_NAME IDE |
               ε

VAR_DECL_LIST ::= VAR_DECL VAR_DECL_LIST |
                  ε

VAR_DECL ::= TYPE_NAME IDE = EXP;

STATEMENT ::= BASIC_STATEMENT STATEMENT |
              ε

BASIC_STATEMENT ::= IDE = EXP; |
                    this.IDE = EXP; |
                    BLOCK |
                    if(EXP) BASIC_STATEMENT else BASIC_STATEMENT |
                    while(EXP) BASIC_STATEMENT |
                    EXP_STATEMENT; |
                    ;

BLOCK ::= { VAR_DECL_LIST STATEMENT }
```

```
EXP ::= CONST_VAL |
        IDE |
        this.IDE |
        EXP OP EXP |
        UNARYOP EXP |
        (EXP) |
        EXP_STATEMENT

EXP_STATEMENT ::= IDE.METH_IDE(PARAM) |
                  this.METH_IDE(PARAM) |
                  new IDE_CLASS(PARAM)

PARAM ::= EXP |
          ε

OP ::= + | - | * | / | == | != | < | <= | > | >= | & | && | ···

UNARYOP ::= !  | -

CONST_VAL ::= NUM | BOOL | null

BOOL ::= true | false
```

Le categorie sintattiche `NUM` e `IDE` rappresentano rispettivamente le stringhe numeriche e quelle alfanumeriche che iniziano con un carattere alfabetico. Le categorie sintattiche `IDE_CLASS` e `METH_IDE` coincidono con la categoria `IDE`.

5 Sistemi di transizione

Def: Un sistema di transizioni `TR` è una tripla $\langle\, \Gamma,\, T,\, \rightarrow \,\rangle$ dove:

- Γ è un insieme i cui elementi sono detti *configurazioni*;
- $T \subseteq \Gamma$ è un insieme di configurazioni dette *configurazioni terminali*
- $\rightarrow$ è un insieme di coppie $\langle\, \gamma,\, \gamma' \,\rangle$ di configurazioni e viene detta *relazione di transizione*. Denoteremo con $\gamma \rightarrow \gamma'$ l'appartenenza della coppia $\langle\, \gamma,\, \gamma' \,\rangle$ alla relazione $\rightarrow$ e chiameremo *transizioni* tali elementi.

Ricordiamo alcune definizioni che verranno utilizzate in seguito.

Φ è l'insieme di tutte le funzioni del tipo:

- φ_v : `IDE` $\rightarrow Val^*$
  dove $Val^* = Val \cup Loc \cup \{\texttt{null}\} \cup \{\bot\}$

- φ_m : IDE $\rightarrow Method$
  dove $Method = \{< I, M > \mid \begin{array}{l} I \in \texttt{IDE} \cup \{\bot\}, \\ M \in \texttt{METH\_BODY} \cup \texttt{METH\_BODY\_NO\_RETURN} \end{array} \}$

- ω : IDE $\rightarrow \{\bot\}$

$\mathcal{S}$, come definito precedentemente, è l'insieme delle funzioni del tipo

- ρ : IDE $\rightarrow Val^*$
  dove $Val^* = Val \cup Loc \cup \{\texttt{null}\} \cup \{\bot\}$

$\mathcal{M}$ è l'insieme delle funzioni del tipo

- $\mu : Loc \rightarrow (\Phi \times \Phi) \cup \{\bot\}$

Δ è l'insieme delle funzioni del tipo

- δ : IDE_CLASS $\rightarrow (\Phi \times \Phi) \cup \{\bot\}$

State è l'insieme degli stati ammissibili $\{\sigma \mid \sigma \in \mathcal{S} \times \mathcal{M} \times \Delta\}$.

Nelle regole seguenti faremo uso delle funzioni *defval* e η.

La funzione *defval* prende un tipo di dato e restituisce il relativo valore di default:

$$\mathit{defval}(\texttt{T}) = \begin{cases} \underline{0} & \text{se} \quad \texttt{T} = \texttt{int} \\ \underline{false} & \text{se} \quad \texttt{T} = \texttt{boolean} \\ \underline{null} & \text{altrimenti} \end{cases}$$

η è la funzione di valutazione semantica che prende una rappresentazione sintattica di un valore costante e restituisce il corrispondente valore semantico:

$$\eta : \texttt{CONST\_VAL} \rightarrow \mathbb{Z} \cup \mathbb{B} \cup \{\underline{null}\}$$

$$\begin{array}{c} \eta(\texttt{0}) = \underline{0} \\ \eta(\texttt{1}) = \underline{1} \\ \eta(\texttt{2}) = \underline{2} \\ \cdots \\ \eta(\texttt{-0}) = \underline{-0} \\ \eta(\texttt{-1}) = \underline{-1} \\ \eta(\texttt{-2}) = \underline{-2} \\ \cdots \\ \eta(\texttt{true}) = \underline{true} \\ \eta(\texttt{false}) = \underline{false} \\ \eta(\texttt{null}) = \underline{null} \end{array}$$

5.1 Il sistema *prog*

$$\begin{array}{rcl} \Gamma_{prog} & = & \{\langle\, \mathtt{P},\, \sigma \,\rangle \mid \mathtt{P} \in \mathtt{PROGRAM},\ \sigma \in \mathtt{State}\} \cup \\ & & \{\sigma \mid \sigma \in \mathtt{State}\} \\ T_{prog} & = & \{\sigma \mid \sigma \in \mathtt{State}\} \end{array}$$

Il sistema *prog* utilizza lo stato $\sigma_0 = \langle\, \rho_0,\, \mu_0,\, \delta_0 \,\rangle$ che rappresenta lo stato iniziale vuoto, così composto:

$\rho_0 = \omega$ dove ω è l'environment vuoto.
$\mu_0 : Loc \to \{\bot\}$ è lo store vuoto che associa $\bot$ a tutte le locazioni della memoria.
$\delta_0 : \mathtt{IDE\_CLASS} \to \{\bot\}$ è la library vuota che associa $\bot$ a tutti gli identificatori di classe.

$$\frac{\begin{array}{c} \sigma_0 = \langle\, \rho_0,\, \mu_0,\, \delta_0 \,\rangle \\ \langle\, \mathtt{C}\,, \sigma_0 \,\rangle \longrightarrow_{decl} \sigma_1 \quad \sigma_1 = \langle\, \rho_0,\, \mu_0,\, \delta_1 \,\rangle \quad \sigma_2 = \langle\, \rho_0[\frac{null}{\mathtt{I}}],\, \mu_0,\, \delta_1 \,\rangle \\ \langle\, \mathtt{D}\,, \sigma_2 \,\rangle \longrightarrow_{local\_decl} \sigma_3 \quad \langle\, \mathtt{S}\,, \sigma_3 \,\rangle \longrightarrow_{com} \sigma' \end{array}}{\begin{array}{l} \mathtt{C\ public\ class\ Program\ \{} \\ \langle\ \ \mathtt{public\ static\ void\ main(String[]\ I)} \\ \ \ \mathtt{\{\ D\ S\}} \\ \mathtt{\}} \end{array}\ , \sigma_0 \,\rangle \longrightarrow_{prog} \sigma'} \quad (prog)$$

5.2 Il sistema *decl*

$$\begin{array}{rcl} \Gamma_{decl} & = & \{\langle\, \mathtt{D},\, \sigma \,\rangle \mid \mathtt{D} \in \mathtt{CLASS\_DECL},\ \sigma \in \mathtt{State}\} \cup \\ & & \{\sigma \mid \sigma \in \mathtt{State}\} \\ T_{decl} & = & \{\sigma \mid \sigma \in \mathtt{State}\} \end{array}$$

$$\frac{\begin{array}{c} \sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \\ \langle\, \mathtt{F}\,, \langle\, \omega,\, \omega \,\rangle \,\rangle \longrightarrow_{cd} \langle\, \varphi_v,\, \varphi_m \,\rangle \\ \delta' = \delta[{}^{\langle\, \varphi_v, \varphi_m \,\rangle}/_{\mathtt{C}}] \quad \sigma' = \langle\, \rho,\, \mu,\, \delta' \,\rangle \end{array}}{\langle\, \mathtt{class\ C\ \{F\}}\,, \sigma \,\rangle \longrightarrow_{decl} \sigma'} \quad (decl_{class})$$

$$\frac{\langle\, \mathtt{D_1}\,, \sigma \,\rangle \longrightarrow_{decl} \sigma_1 \quad \langle\, \mathtt{D_2}\,, \sigma_1 \,\rangle \longrightarrow_{decl} \sigma'}{\langle\, \mathtt{D_1\ D_2}\,, \sigma \,\rangle \longrightarrow_{decl} \sigma'} \quad (decl_{concat})$$

5.3 Il sistema *cd*

$$\begin{aligned}\Gamma_{cd} &= \{\langle \texttt{F}, \langle \varphi_1, \varphi_2 \rangle \rangle \mid \texttt{F} \in \texttt{FIELD\_DECL},\ \langle \varphi_1, \varphi_2 \rangle \in \Phi \times \Phi\} \cup \\ &\quad \{\langle \varphi_1, \varphi_2 \rangle \mid \langle \varphi_1, \varphi_2 \rangle \in \Phi \times \Phi\} \\ T_{cd} &= \{\langle \varphi_1, \varphi_2 \rangle \mid \langle \varphi_1, \varphi_2 \rangle \in \Phi \times \Phi\}\end{aligned}$$

$$\frac{\underline{v} = \mathit{defval}(\texttt{T}) \quad \varphi'_v = \varphi_v[{}^{\underline{v}}/_{\texttt{x}}]}{\langle\, \texttt{private T x;}\,, \langle \varphi_v, \varphi_m \rangle \rangle \longrightarrow_{cd} \langle \varphi'_v, \varphi_m \rangle} \quad (cd_{var})$$

$$\frac{\varphi'_m = \varphi_m[{}^{\langle \bot, \texttt{MB} \rangle}/_{\texttt{m}}]}{\langle\, \texttt{public T}_\texttt{r}\ \texttt{m()\ MB}\,, \langle \varphi_v, \varphi_m \rangle \rangle \longrightarrow_{cd} \langle \varphi_v, \varphi'_m \rangle} \quad (cd_{meth\_nopar})$$

$$\frac{\varphi'_m = \varphi_m[{}^{\langle \texttt{x}, \texttt{MB} \rangle}/_{\texttt{m}}]}{\langle\, \texttt{public T}_\texttt{r}\ \texttt{m(T}_\texttt{p}\ \texttt{x)\ MB}\,, \langle \varphi_v, \varphi_m \rangle \rangle \longrightarrow_{cd} \langle \varphi_v, \varphi'_m \rangle} \quad (cd_{meth})$$

$$\frac{\varphi'_m = \varphi_m[{}^{\langle \bot, \texttt{MB} \rangle}/_{\texttt{C}}]}{\langle\, \texttt{public C()\ MB}\,, \langle \varphi_v, \varphi_m \rangle \rangle \longrightarrow_{cd} \langle \varphi_v, \varphi'_m \rangle} \quad (cd_{constr\_nopar})$$

$$\frac{\varphi'_m = \varphi_m[{}^{\langle \texttt{x}, \texttt{MB} \rangle}/_{\texttt{C}}]}{\langle\, \texttt{public C(T}_\texttt{p}\ \texttt{x)\ MB}\,, \langle \varphi_v, \varphi_m \rangle \rangle \longrightarrow_{cd} \langle \varphi_v, \varphi'_m \rangle} \quad (cd_{constr})$$

$$\frac{\begin{array}{c}\langle\, \texttt{F}_1\,, \langle \varphi_v, \varphi_m \rangle \rangle \longrightarrow_{cd} \langle \varphi_{v_1}, \varphi_{m_1} \rangle \\ \langle\, \texttt{F}_2\,, \langle \varphi_{v_1}, \varphi_{m_1} \rangle \rangle \longrightarrow_{cd} \langle \varphi'_v, \varphi'_m \rangle\end{array}}{\langle\, \texttt{F}_1\ \texttt{F}_2\,, \langle \varphi_v, \varphi_m \rangle \rangle \longrightarrow_{cd} \langle \varphi'_v, \varphi'_m \rangle} \quad (cd_{concat})$$

5.4 Il sistema *local_ decl*

$$\begin{aligned}\Gamma_{local\_decl} &= \{\langle \texttt{D}, \sigma \rangle \mid \texttt{D} \in \texttt{VAR\_DECL},\ \sigma \in \texttt{State}\} \cup \{\sigma \mid \sigma \in \texttt{State}\} \\ T_{local\_decl} &= \{\sigma \mid \sigma \in \texttt{State}\}\end{aligned}$$

$$\frac{\begin{array}{c}\sigma = \langle \rho, \mu, \delta \rangle \quad \langle\, \texttt{e}\,, \sigma \rangle \longrightarrow_{exp} \langle v, \sigma_1 \rangle \\ \sigma_1 = \langle \rho, \mu_1, \delta \rangle \quad \rho = \varphi \cdot \rho_1 \\ \rho' = \varphi[{}^{v}/_{\texttt{x}}] \cdot \rho_1 \quad \sigma' = \langle \rho', \mu_1, \delta \rangle\end{array}}{\langle\, \texttt{T x = e;}\,, \sigma \rangle \longrightarrow_{local\_decl} \sigma'} \quad (local\_decl_{var\_init})$$

$$\frac{\langle\, \texttt{D}_1\,, \sigma \rangle \longrightarrow_{local\_decl} \sigma_1 \quad \langle\, \texttt{D}_2\,, \sigma_1 \rangle \longrightarrow_{local\_decl} \sigma'}{\langle\, \texttt{D}_1\ \texttt{D}_2\,, \sigma \rangle \longrightarrow_{local\_decl} \sigma'} \quad (local\_decl_{concat})$$

5.5 Il sistema *exp*

$$\begin{array}{rcl} \Gamma_{exp} & = & \{\langle\, \mathtt{E},\, \sigma \,\rangle \mid \mathtt{E} \in \mathtt{EXP},\ \sigma \in \mathtt{State}\} \cup \\ & & \{\langle\, e,\, \sigma \,\rangle \mid e \in Val^*\ ,\ \sigma \in \mathtt{State}\} \\ T_{exp} & = & \{\langle\, e,\, \sigma \,\rangle \mid e \in Val^*\ ,\ \sigma \in \mathtt{State}\} \end{array}$$

$$\frac{\eta(\mathtt{c}) = \underline{v}}{\langle\, \mathtt{c}\ , \sigma \,\rangle \longrightarrow_{exp} \langle\, \underline{v},\, \sigma \,\rangle} \quad (exp_{const})$$

$$\frac{\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad \rho(\mathtt{x}) = e}{\langle\, \mathtt{x}\ , \sigma \,\rangle \longrightarrow_{exp} \langle\, e,\, \sigma \,\rangle} \quad (exp_{ide})$$

$$\frac{\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad \rho(\mathtt{this}) = l \quad \mu(l) = \langle\, \varphi_v,\, \varphi_m \,\rangle \quad \varphi_v(\mathtt{x}) = e}{\langle\, \mathtt{this.x}\ , \sigma \,\rangle \longrightarrow_{exp} \langle\, e,\, \sigma \,\rangle} \quad (exp_{ideobj})$$

$$\frac{\begin{array}{c} \langle\, \mathtt{E_1}\ , \sigma \,\rangle \longrightarrow_{exp} \langle\, e_1,\, \sigma_1 \,\rangle \quad \langle\, \mathtt{E_2}\ , \sigma_1 \,\rangle \longrightarrow_{exp} \langle\, e_2,\, \sigma' \,\rangle \\ e = e_1\ op\ e_2 \end{array}}{\langle\, \mathtt{E_1\ OP\ E_2}\ , \sigma \,\rangle \longrightarrow_{exp} \langle\, e,\, \sigma' \,\rangle} \quad (exp_{op}\ (metaregola))$$

$$\frac{\langle\, \mathtt{E_1}\ , \sigma \,\rangle \longrightarrow_{exp} \langle\, e_1,\, \sigma_1 \,\rangle \quad e = unaryop\ e_1}{\langle\, \mathtt{UNARYOP\ E_1}\ , \sigma \,\rangle \longrightarrow_{exp} \langle\, e,\, \sigma' \,\rangle} \quad (exp_{unaryop}\ (metaregola))$$

$$\frac{\langle\, \mathtt{E}\ , \sigma \,\rangle \longrightarrow_{exp} \langle\, e,\, \sigma' \,\rangle}{\langle\, \mathtt{(E)}\ , \sigma \,\rangle \longrightarrow_{exp} \langle\, e,\, \sigma' \,\rangle} \quad (exp_{()})$$

$$\frac{\begin{array}{c} \sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad l = newloc(\mu) \quad \delta(\mathtt{C}) = \langle\, \varphi_v,\, \varphi_m \,\rangle \\ \mu_1 = \mu[{}^{\langle\, \varphi_v, \varphi_m \,\rangle}/_{l}\,] \quad \varphi_m(\mathtt{C}) = \langle\, \bot,\, \mathtt{MB} \,\rangle \quad \mathtt{MB} = \{\mathtt{D\ S}\} \\ \sigma_{local} = \langle\, \omega[{}^{l}/_{\mathtt{this}}\,],\, \mu_1,\, \delta \,\rangle \quad \langle\, \mathtt{D}\ , \sigma_{local} \,\rangle \longrightarrow_{local\_decl} \sigma_2 \\ \sigma_2 = \langle\, \rho_2,\, \mu_2,\, \delta \,\rangle \quad \langle\, \mathtt{S}\ , \sigma_2 \,\rangle \longrightarrow_{com} \sigma_3 \\ \sigma_3 = \langle\, \rho_3,\, \mu_3,\, \delta \,\rangle \quad \sigma' = \langle\, \rho,\, \mu_3,\, \delta \,\rangle \end{array}}{\langle\, \mathtt{new\ C()}\ , \sigma \,\rangle \longrightarrow_{exp} \langle\, l,\, \sigma' \,\rangle} \quad (exp_{new\_nopar})$$

$$\frac{\begin{array}{c} \sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad l = newloc(\mu) \quad \delta(\mathtt{C}) = \langle\, \varphi_v,\, \varphi_m \,\rangle \\ \mu_1 = \mu[{}^{\langle\, \varphi_v, \varphi_m \,\rangle}/_{l}\,] \quad \varphi_m(\mathtt{C}) = \langle\, \mathtt{x},\, \mathtt{MB} \,\rangle \quad \mathtt{MB} = \{\mathtt{D\ S}\} \\ \sigma_1 = \langle\, \rho,\, \mu_1,\, \delta \,\rangle \quad \langle\, \mathtt{p}\ , \sigma_1 \,\rangle \longrightarrow_{exp} \langle\, v_p,\, \sigma_2 \,\rangle \\ \sigma_2 = \langle\, \rho,\, \mu_2,\, \delta \,\rangle \quad \sigma_{local} = \langle\, \omega[{}^{v_p}/_{\mathtt{x}},\, {}^{l}/_{\mathtt{this}}\,],\, \mu_2,\, \delta \,\rangle \\ \langle\, \mathtt{D}\ , \sigma_{local} \,\rangle \longrightarrow_{local\_decl} \sigma_3 \quad \sigma_3 = \langle\, \rho_3,\, \mu_3,\, \delta \,\rangle \\ \langle\, \mathtt{S}\ , \sigma_3 \,\rangle \longrightarrow_{com} \sigma_4 \quad \sigma_4 = \langle\, \rho_4,\, \mu_4,\, \delta \,\rangle \quad \sigma' = \langle\, \rho,\, \mu_4,\, \delta \,\rangle \end{array}}{\langle\, \mathtt{new\ C(p)}\ , \sigma \,\rangle \longrightarrow_{exp} \langle\, l,\, \sigma' \,\rangle} \quad (exp_{new\_par})$$

$$\frac{\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad l = newloc(\mu) \quad \delta(\texttt{C}) = \langle\, \varphi_v,\, \varphi_m \,\rangle \quad \mu' = \mu[{}^{\langle\, \varphi_v, \varphi_m \,\rangle}/_{l}\,] \quad \varphi_m(\texttt{C}) = \bot \quad \sigma' = \langle\, \rho,\, \mu',\, \delta \,\rangle}{\langle\, \texttt{new C()}\,, \sigma \,\rangle \longrightarrow_{exp} \langle\, l,\, \sigma' \,\rangle} \quad (exp_{new\_default})$$

$$\frac{\begin{array}{c}\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad \rho(\texttt{Obj}) = l \quad \mu(l) = \langle\, \varphi_v,\, \varphi_m \,\rangle \\ \varphi_m(\texttt{m}) = \langle\, \bot,\, \texttt{MB} \,\rangle \quad \texttt{MB} = \{\texttt{D S return e;}\} \\ \sigma_{local} = \langle\, \omega[{}^{l}/_{\texttt{this}}\,],\, \mu,\, \delta \,\rangle \quad \langle\, \texttt{D}\,, \sigma_{local} \,\rangle \longrightarrow_{local\_decl} \sigma_1 \\ \sigma_1 = \langle\, \rho_1,\, \mu_1,\, \delta \,\rangle \quad \langle\, \texttt{S}\,, \sigma_1 \,\rangle \longrightarrow_{com} \sigma_2 \\ \sigma_2 = \langle\, \rho_2,\, \mu_2,\, \delta \,\rangle \quad \langle\, \texttt{e}\,, \sigma_2 \,\rangle \longrightarrow_{exp} \langle\, v,\, \sigma_3 \,\rangle \\ \sigma_3 = \langle\, \rho_2,\, \mu_3,\, \delta \,\rangle \quad \sigma' = \langle\, \rho,\, \mu_3,\, \delta \,\rangle\end{array}}{\langle\, \texttt{Obj.m()}\,, \sigma \,\rangle \longrightarrow_{exp} \langle\, v,\, \sigma' \,\rangle} \quad (exp_{mcall})$$

$$\frac{\begin{array}{c}\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad \rho(\texttt{Obj}) = l \quad \mu(l) = \langle\, \varphi_v,\, \varphi_m \,\rangle \\ \varphi_m(\texttt{m}) = \langle\, \texttt{x},\, \texttt{MB} \,\rangle \quad \langle\, \texttt{p}\,, \sigma \,\rangle \longrightarrow_{exp} \langle\, v_p,\, \sigma_1 \,\rangle \\ \sigma_1 = \langle\, \rho,\, \mu_1,\, \delta \,\rangle \quad \texttt{MB} = \{\texttt{D S return e;}\} \\ \sigma_{local} = \langle\, \omega[{}^{v_p}/_{\texttt{x}},\, {}^{l}/_{\texttt{this}}\,],\, \mu_1,\, \delta \,\rangle \\ \langle\, \texttt{D}\,, \sigma_{local} \,\rangle \longrightarrow_{local\_decl} \sigma_2 \\ \sigma_2 = \langle\, \rho_2,\, \mu_2,\, \delta \,\rangle \quad \langle\, \texttt{S}\,, \sigma_2 \,\rangle \longrightarrow_{com} \sigma_3 \\ \sigma_3 = \langle\, \rho_3,\, \mu_3,\, \delta \,\rangle \quad \langle\, \texttt{e}\,, \sigma_3 \,\rangle \longrightarrow_{exp} \langle\, v,\, \sigma_4 \,\rangle \\ \sigma_4 = \langle\, \rho_3,\, \mu_4,\, \delta \,\rangle \quad \sigma' = \langle\, \rho,\, \mu_4,\, \delta \,\rangle\end{array}}{\langle\, \texttt{Obj.m(p)}\,, \sigma \,\rangle \longrightarrow_{exp} \langle\, v,\, \sigma' \,\rangle} \quad (exp_{mcallpar})$$

Un'istanza della metaregola exp_{op}, per esempio nel caso in cui `OP` sia `+`, è la seguente:

$$\frac{\begin{array}{c}\langle\, \texttt{E}_1\,, \sigma \,\rangle \longrightarrow_{exp} \langle\, e_1,\, \sigma_1 \,\rangle \quad \langle\, \texttt{E}_2\,, \sigma_1 \,\rangle \longrightarrow_{exp} \langle\, e_2,\, \sigma' \,\rangle \\ e = e_1 \;+\; e_2\end{array}}{\langle\, \texttt{E}_1 \texttt{ + } \texttt{E}_2\,, \sigma \,\rangle \longrightarrow_{exp} \langle\, e,\, \sigma' \,\rangle} \quad (exp_{+})$$

Nel caso in cui non vi siano dichiarazioni di variabili all'interno del corpo del metodo, le regole exp_{mcall} ed $exp_{mcallpar}$ diventano rispettivamente:

$$\frac{\begin{array}{c}\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad \rho(\texttt{Obj}) = l \quad \mu(l) = \langle\, \varphi_v,\, \varphi_m \,\rangle \\ \varphi_m(\texttt{m}) = \langle\, \bot,\, \texttt{MB} \,\rangle \quad \texttt{MB} = \{\texttt{S return e;}\} \\ \sigma_{local} = \langle\, \omega[{}^{l}/_{\texttt{this}}],\, \mu,\, \delta \,\rangle \quad \langle\, \texttt{S} \,, \sigma_{local} \,\rangle \longrightarrow_{com} \sigma_1 \\ \sigma_1 = \langle\, \rho_1,\, \mu_1,\, \delta \,\rangle \quad \langle\, \texttt{e} \,, \sigma_1 \,\rangle \longrightarrow_{exp} \langle\, v,\, \sigma_2 \,\rangle \\ \sigma_2 = \langle\, \rho_1,\, \mu_2,\, \delta \,\rangle \quad \sigma' = \langle\, \rho,\, \mu_2,\, \delta \,\rangle \end{array}}{\langle\, \texttt{Obj.m()} \,, \sigma \,\rangle \longrightarrow_{exp} \langle\, v,\, \sigma' \,\rangle} \quad (exp_{mcallnodecl})$$

$$\frac{\begin{array}{c}\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad \rho(\texttt{Obj}) = l \quad \mu(l) = \langle\, \varphi_v,\, \varphi_m \,\rangle \\ \varphi_m(\texttt{m}) = \langle\, \texttt{x},\, \texttt{MB} \,\rangle \quad \langle\, \texttt{p} \,, \sigma \,\rangle \longrightarrow_{exp} \langle\, v_p,\, \sigma_1 \,\rangle \\ \sigma_1 = \langle\, \rho,\, \mu_1,\, \delta \,\rangle \quad \texttt{MB} = \{\texttt{S return e;}\} \\ \sigma_{local} = \langle\, \omega[{}^{v_p}/_{\texttt{x}},\, {}^{l}/_{\texttt{this}}],\, \mu_1,\, \delta \,\rangle \\ \langle\, \texttt{S} \,, \sigma_{local} \,\rangle \longrightarrow_{com} \sigma_2 \\ \sigma_2 = \langle\, \rho_2,\, \mu_2,\, \delta \,\rangle \quad \langle\, \texttt{e} \,, \sigma_2 \,\rangle \longrightarrow_{exp} \langle\, v,\, \sigma_3 \,\rangle \\ \sigma_3 = \langle\, \rho_2,\, \mu_3,\, \delta \,\rangle \quad \sigma' = \langle\, \rho,\, \mu_3,\, \delta \,\rangle \end{array}}{\langle\, \texttt{Obj.m(p)} \,, \sigma \,\rangle \longrightarrow_{exp} \langle\, v,\, \sigma' \,\rangle} \quad (exp_{mcallparnodecl})$$

5.6 Il sistema *com*

$$\begin{aligned}\Gamma_{com} &= \{\langle\, \mathtt{S},\, \sigma \,\rangle \mid \mathtt{S} \in \mathtt{STATEMENT},\ \sigma \in \mathtt{State}\} \cup \\ &\quad\ \{\sigma \mid \sigma \in \mathtt{State}\} \\ T_{com} &= \{\sigma \mid \sigma \in \mathtt{State}\}\end{aligned}$$

$$\frac{}{\langle\, ;\, , \sigma \,\rangle \longrightarrow_{com} \sigma} \quad (com_{null})$$

$$\frac{\begin{array}{c}\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \\ \langle\, \mathtt{e}\, , \sigma \,\rangle \longrightarrow_{exp} \langle\, v,\, \sigma_1 \,\rangle \quad \sigma_1 = \langle\, \rho,\, \mu_1,\, \delta \,\rangle \\ \rho' = \rho[^{v}/_{\mathtt{x}}] \quad \sigma' = \langle\, \rho',\, \mu_1,\, \delta \,\rangle\end{array}}{\langle\, \mathtt{x = e;}\, , \sigma \,\rangle \longrightarrow_{com} \sigma'} \quad (com_{assign})$$

$$\frac{\begin{array}{c}\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \\ \langle\, \mathtt{e}\, , \sigma \,\rangle \longrightarrow_{exp} \langle\, v,\, \sigma_1 \,\rangle \quad \sigma_1 = \langle\, \rho,\, \mu_1,\, \delta \,\rangle \\ \rho(\mathtt{this}) = l \quad \mu_1(l) = \langle\, \varphi_v,\, \varphi_m \,\rangle \quad \varphi'_v = \varphi_v[^{v}/_{\mathtt{x}}] \\ \mu_2 = \mu_1[^{\langle\, \varphi'_v, \varphi_m \,\rangle}/_{l}] \quad \sigma' = \langle\, \rho,\, \mu_2,\, \delta \,\rangle\end{array}}{\langle\, \mathtt{this.x = e;}\, , \sigma \,\rangle \longrightarrow_{com} \sigma'} \quad (com_{assign\_this})$$

$$\frac{\langle\, \mathtt{S_1}\, , \sigma \,\rangle \longrightarrow_{com} \sigma_1 \quad \langle\, \mathtt{S_2}\, , \sigma_1 \,\rangle \longrightarrow_{com} \sigma'}{\langle\, \mathtt{S_1\ S_2}\, , \sigma \,\rangle \longrightarrow_{com} \sigma'} \quad (com_{concat})$$

$$\frac{\begin{array}{c}\langle\, \mathtt{E}\, , \sigma \,\rangle \longrightarrow_{exp} \langle\, e,\, \sigma_1 \,\rangle \quad e = \underline{true} \\ \langle\, \mathtt{S_1}\, , \sigma_1 \,\rangle \longrightarrow_{com} \sigma'\end{array}}{\langle\, \mathtt{if(E)\ S_1\ else\ S_2}\, , \sigma \,\rangle \longrightarrow_{com} \sigma'} \quad (com_{if\_true})$$

$$\frac{\begin{array}{c}\langle\, \mathtt{E}\, , \sigma \,\rangle \longrightarrow_{exp} \langle\, e,\, \sigma_1 \,\rangle \quad e = \underline{false} \\ \langle\, \mathtt{S_2}\, , \sigma_1 \,\rangle \longrightarrow_{com} \sigma'\end{array}}{\langle\, \mathtt{if(E)\ S_1\ else\ S_2}\, , \sigma \,\rangle \longrightarrow_{com} \sigma'} \quad (com_{if\_false})$$

$$\frac{\begin{array}{c}\langle\, \mathtt{E}\, , \sigma \,\rangle \longrightarrow_{exp} \langle\, e,\, \sigma_1 \,\rangle \quad e = \underline{true} \\ \langle\, \mathtt{S}\, , \sigma_1 \,\rangle \longrightarrow_{com} \sigma_2 \\ \langle\, \mathtt{while(E)\ S}\, , \sigma_2 \,\rangle \longrightarrow_{com} \sigma'\end{array}}{\langle\, \mathtt{while(E)\ S}\, , \sigma \,\rangle \longrightarrow_{com} \sigma'} \quad (com_{while\_true})$$

$$\frac{\langle\, \mathtt{E}\, , \sigma \,\rangle \longrightarrow_{exp} \langle\, e,\, \sigma' \,\rangle \quad e = \underline{false}}{\langle\, \mathtt{while(E)\ S}\, , \sigma \,\rangle \longrightarrow_{com} \sigma'} \quad (com_{while\_false})$$

$$\frac{\begin{array}{c}\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad \rho(\texttt{Obj}) = l \quad \mu(l) = \langle\, \varphi_v,\, \varphi_m \,\rangle \\ \varphi_m(\texttt{m}) = \langle\, \bot,\, \texttt{MB} \,\rangle \quad \texttt{MB} = \{\texttt{D}\ \texttt{S}\} \\ \sigma_{local} = \langle\, \omega[{}^{l}/_{\texttt{this}}],\, \mu,\, \delta \,\rangle \quad \langle\, \texttt{D}\,, \sigma_{local} \,\rangle \longrightarrow_{local\_decl} \sigma_1 \\ \sigma_1 = \langle\, \rho_1,\, \mu_1,\, \delta \,\rangle \quad \langle\, \texttt{S}\,, \sigma_1 \,\rangle \longrightarrow_{com} \sigma_2 \\ \sigma_2 = \langle\, \rho_2,\, \mu_2,\, \delta \,\rangle \quad \sigma' = \langle\, \rho,\, \mu_2,\, \delta \,\rangle\end{array}}{\langle\, \texttt{Obj.m}()\,, \sigma \,\rangle \longrightarrow_{com} \sigma'} \quad (com_{mcall})$$

$$\frac{\begin{array}{c}\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad \rho(\texttt{Obj}) = l \quad \mu(l) = \langle\, \varphi_v,\, \varphi_m \,\rangle \\ \varphi_m(\texttt{m}) = \langle\, \texttt{x},\, \texttt{MB} \,\rangle \quad \langle\, \texttt{p}\,, \sigma \,\rangle \longrightarrow_{exp} \langle\, v_p,\, \sigma_1 \,\rangle \\ \sigma_1 = \langle\, \rho,\, \mu_1,\, \delta \,\rangle \quad \texttt{MB} = \{\texttt{D}\ \texttt{S}\} \\ \sigma_{local} = \langle\, \omega[{}^{v_p}/_{\texttt{x}},\, {}^{l}/_{\texttt{this}}],\, \mu_1,\, \delta \,\rangle \\ \langle\, \texttt{D}\,, \sigma_{local} \,\rangle \longrightarrow_{local\_decl} \sigma_2 \quad \sigma_2 = \langle\, \rho_2,\, \mu_2,\, \delta \,\rangle \\ \langle\, \texttt{S}\,, \sigma_2 \,\rangle \longrightarrow_{com} \sigma_3 \quad \sigma_3 = \langle\, \rho_3,\, \mu_3,\, \delta \,\rangle \\ \sigma' = \langle\, \rho,\, \mu_3,\, \delta \,\rangle\end{array}}{\langle\, \texttt{Obj.m(p)}\,, \sigma \,\rangle \longrightarrow_{com} \sigma'} \quad (com_{mcallpar})$$

$$\frac{\begin{array}{c}\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad \rho_1 = push(\omega, \rho) \quad \sigma_1 = \langle\, \rho_1,\, \mu,\, \delta \,\rangle \\ \langle\, \texttt{D}\,, \sigma_1 \,\rangle \longrightarrow_{local\_decl} \sigma_2 \quad \langle\, \texttt{S}\,, \sigma_2 \,\rangle \longrightarrow_{com} \sigma_3 \\ \sigma_3 = \langle\, \rho_3,\, \mu_3,\, \delta \,\rangle \quad \rho' = pop(\rho_3) \quad \sigma' = \langle\, \rho',\, \mu_3,\, \delta \,\rangle\end{array}}{\langle\, \{\texttt{D}\ \texttt{S}\}\,, \sigma \,\rangle \longrightarrow_{com} \sigma'} \quad (com_{block})$$

Nel caso in cui nel blocco non vi siano dichiarazioni di variabili, la regola com_{block} diventa:

$$\frac{\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad \langle\, \texttt{S}\,, \sigma \,\rangle \longrightarrow_{com} \sigma' \quad \sigma' = \langle\, \rho',\, \mu',\, \delta \,\rangle}{\langle\, \{\texttt{S}\}\,, \sigma \,\rangle \longrightarrow_{com} \sigma'} \quad (com_{block\_nodecl})$$

6 Esempi

6.1 Dichiarazione di una classe

```
class Rettangolo {
  private int larghezza;
  private int altezza;

  public int area() {
    return this.larghezza * this.altezza;
  }

  public void scala(int fattore) {
    this.larghezza = this.larghezza * fattore;
    this.altezza = this.altezza * fattore;
  }
}
```

La dichiarazione della precedente classe genera le seguenti strutture:

φ_v :

`larghezza`	*defval*(`int`)
`altezza`	*defval*(`int`)

φ_m :

`area`	$\langle\ \perp$, `{return this.larghezza * this.altezza;}` $\rangle$
`scala`	$\langle$ `fattore`, `{ this.larghezza = this.larghezza * fattore; this.altezza = this.altezza * fattore; }` $\rangle$

δ :

`Rettangolo`	$\langle\varphi_v, \varphi_m\rangle$

6.2 Esecuzione di un programma

```
01:class Rettangolo {
02:  private int larghezza;
03:  private int altezza;
04:
05:  public void set_larghezza(int l) {
06:    this.larghezza = l;
07:  }
08:
09:  public void set_altezza(int altezza) {
10:    this.altezza = altezza;
11:  }
12:
13:  public int area() {
14:    return this.larghezza * this.altezza;
15:  }
16:
17:  public void scala(int fattore) {
18:    this.larghezza = this.larghezza * fattore;
19:    this.altezza = this.altezza * fattore;
20:  }
21:}
22:
23:public class Program {
24:  public static void main(String[] args) {
25:    Rettangolo r = new Rettangolo();
26:    int area = 0;
27:
28:    r.set_larghezza(3);
29:    r.set_altezza(4);
30:    area = r.area();
31:  }
32:}
```

Partiamo dallo stato iniziale vuoto $\sigma_0 = \langle\, \rho_0,\, \mu_0,\, \delta_0 \,\rangle$.

1) Applicazione della regola *prog* alle linee `01-32`

1.1) Applicazione della regola $decl_{class}$ alle linee `01-21`

1.1.1) Applicazione della regola cd_{concat} alle linee `02-20`

1.1.1.1) Applicazione della regola cd_{var} alla linea `02`

φ_v :

`larghezza`	0

φ_m : ω

1.1.1.2) Applicazione della regola cd_{var} alla linea `03`

φ_v :

`larghezza`	0
`altezza`	0

φ_m : ω

1.1.1.3) Applicazione della regola cd_{meth} alle linee `05-07`

φ_v :

`larghezza`	0
`altezza`	0

φ_m :

`set_larghezza`	$\langle$ `l, {this.larghezza = l;}` $\rangle$

1.1.1.4) Applicazione della regola cd_{meth} alle linee `09-11`

φ_v :

`larghezza`	0
`altezza`	0

φ_m :

`set_larghezza`	$\langle$ `l, {this.larghezza = l;}` $\rangle$
`set_altezza`	$\langle$ `altezza, {this.altezza = altezza;}` $\rangle$

1.1.1.5) Applicazione della regola cd_{meth_nopar} alle linee `13-15`

φ_v :

`larghezza`	0
`altezza`	0

φ_m :

`set_larghezza`	$\langle$ `l, {this.larghezza = l; }` $\rangle$
`set_altezza`	$\langle$ `altezza, {this.altezza = altezza; }` $\rangle$
`area`	$\langle \perp$, `{return this.larghezza * this.altezza;}` $\rangle$

1.1.1.6) Applicazione della regola cd_{meth} alle linee `17-20`

φ_v :

`larghezza`	0
`altezza`	0

φ_m :

`set_larghezza`	$\langle$ `l, {this.larghezza = l; }` $\rangle$
`set_altezza`	$\langle$ `altezza, {this.altezza = altezza; }` $\rangle$
`area`	$\langle$ $\perp$, `{return this.larghezza * this.altezza;}` $\rangle$
`scala`	$\langle$ `fattore, {this.larghezza = this.larghezza * fattore; this.altezza = this.altezza * fattore; }` $\rangle$

1.1) Al termine dell'applicazione di $decl_{class}$ lo stato è il seguente:

$$\sigma_1 = \langle\, \rho_0,\, \mu_0,\, \delta_1 \,\rangle$$

δ_1 :

`Rettangolo`	$\langle\, \varphi_v,\, \varphi_m \,\rangle$

1.2) Binding del parametro del metodo `main` alla linea 24

$$\sigma_2 = \langle\, \rho_1,\, \mu_0,\, \delta_1 \,\rangle$$

ρ_1 :

`args`	*null*

1.3) Applicazione della regola $local\_decl_{concat}$ alle linee 25-26

1.3.1) Applicazione della regola $local\_decl_{var\_init}$ alla linea 25

1.3.1.1) Applicazione della regola $exp_{new\_default}$ alla linea 25

$$\sigma_2 = \langle\, \rho_2,\, \mu_1,\, \delta_1 \,\rangle$$

ρ_2 :

`args`	*null*
`r`	l_1

μ_1 :

l_1	$\langle$ φ_v : `larghezza` 0, `altezza` 0 , φ_m $\rangle$

1.3.2) Applicazione della regola $local\_decl_{var\_init}$ alla linea 26

1.3.2.1) Applicazione di exp_{const} alla linea 26

1.3.2) Al termine dell'applicazione di $local\_decl_{var\_init}$ lo stato è il seguente:

$$\sigma_3 = \langle\, \rho_3,\, \mu_1,\, \delta_1 \,\rangle$$

ρ_3 :

`args`	*null*
`r`	l_1
`area`	0

μ_1 :

l_1	$\langle$	φ_v :	`larghezza` | 0 `altezza` | 0	, φ_m $\rangle$

1.4) Applicazione della regola com_{concat} alle linee `28-30`

1.4.2) Applicazione della regola $com_{mcallpar}$ alla linea `28`. Lo stato risultante da tale applicazione è il seguente:

$$\sigma_4 = \langle\, \rho_3,\, \mu_2,\, \delta_1 \,\rangle$$

μ_2 :

l_1	$\langle$	φ_v :	`larghezza` | 3 `altezza` | 0	, φ_m $\rangle$

1.4.3) Applicazione della regola $com_{mcallpar}$ alla linea `29`. Lo stato risultante da tale applicazione è il seguente:

$$\sigma_5 = \langle\, \rho_3,\, \mu_3,\, \delta_1 \,\rangle$$

μ_3 :

l_1	$\langle$	φ_v :	`larghezza` | 3 `altezza` | 4	, φ_m $\rangle$

1.4.4) Applicazione della regola com_{assign} alla linea `30`

1.4.4.1) Applicazione della regola exp_{mcall} alla linea `30`

1.4.4) Al termine dell'applicazione di com_{assign} lo stato è il seguente:

$$\sigma_6 = \langle\, \rho_4,\, \mu_3,\, \delta_1 \,\rangle$$

ρ_4 :

`args`	*null*
`r`	l_1
`area`	12

1.4) Al termine dell'applicazione di com_{concat} lo stato è il seguente:

$$\sigma_6 = \langle\, \rho_4,\, \mu_3,\, \delta_1 \,\rangle$$

1) Al termine dell'applicazione di *prog* lo stato è il seguente:

$$\sigma_6 = \langle\, \rho_4,\, \mu_3,\, \delta_1 \,\rangle$$

ρ_4 :

`args`	*null*
`r`	l_1
`area`	12

μ_3 :

l_1	$\langle$	φ_v :	`larghezza` | 3 `altezza` | 4	, $\varphi_m \,\rangle$

δ_1 :

`Rettangolo`	$\langle\, \varphi_v,\, \varphi_m \,\rangle$

6.3 Swap

```
01:class Coord {
02:  private int x;
03:  private int y;
04:
05:  public void set_prima_coordinata(int l) {
06:    this.x = l;
07:  }
08:
09:  public void set_seconda_coordinata(int a) {
10:    this.y = a;
11:  }
12:
13:  public int get_prima_coordinata() {
14:    return this.x;
15:  }
16:
17:  public int get_seconda_coordinata() {
18:    return this.y;
19:  }
20:}
21:
22:class Swap {
23:  public void swap(Coord a, Coord b) {
24:    Coord temp = new Coord();
25:
26:    temp = a;
27:    a = b;
28:    b = temp;
29:  }
30:}
31:
32:public class Program {
33:  public static void main(String[] args) {
34:    Coord obj1 = new Coord();
35:    Coord obj2 = new Coord();
36:    Swap s = new Swap();
37:
38:    obj1.set_prima_coordinata(10);
39:    obj1.set_seconda_coordinata(20);
40:    obj2.set_prima_coordinata(80);
41:    obj2.set_seconda_coordinata(90);
42:
```

```
43:    s.swap(obj1, obj2);
44:  }
45:}
```

Per valutare la semantica di questo programma abbiamo bisogno di una semplice estensione delle regole dei sistemi *cd* e *com*, in particolare delle regole per la dichiarazione di metodi e per la loro invocazione. Questa estensione è necessaria per gestire il metodo `swap` della classe `Swap` che necessita di due parametri:

$$\frac{\varphi'_m = \varphi_m[\,{}^{\langle \mathtt{x_1}, \mathtt{x_2}, \mathtt{MB} \rangle}/_{\mathtt{m}}\,]}{\langle\, \mathtt{public\ T_r\ m(T_{p1}\ x_1,\ T_{p2}\ x_2)\ MB}\,, \langle\, \varphi_v,\, \varphi_m \,\rangle \,\rangle \longrightarrow_{cd} \langle\, \varphi_v,\, \varphi'_m \,\rangle} \quad (cd_{meth2})$$

$$\frac{\begin{array}{c}\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad \rho(\mathtt{Obj}) = l \quad \mu(l) = \langle\, \varphi_v,\, \varphi_m \,\rangle \quad \varphi_m(\mathtt{m}) = \langle\, \mathtt{x_1},\, \mathtt{x_2},\, \mathtt{MB} \,\rangle \\ \langle\, \mathtt{p_1}\,, \sigma \,\rangle \longrightarrow_{exp} \langle\, v_{p1},\, \sigma_1 \,\rangle \quad \sigma_1 = \langle\, \rho,\, \mu_1,\, \delta \,\rangle \\ \langle\, \mathtt{p_2}\,, \sigma_1 \,\rangle \longrightarrow_{exp} \langle\, v_{p2},\, \sigma_2 \,\rangle \quad \sigma_2 = \langle\, \rho,\, \mu_2,\, \delta \,\rangle \\ \mathtt{MB} = \{\mathtt{D\ S}\} \quad \sigma_{local} = \langle\, \omega[\,{}^{v_{p1}}/_{\mathtt{x_1}},\, {}^{v_{p2}}/_{\mathtt{x_2}},\, {}^{l}/_{\mathtt{this}}\,],\, \mu_2,\, \delta \,\rangle \\ \langle\, \mathtt{D}\,, \sigma_{local} \,\rangle \longrightarrow_{local\_decl} \sigma_3 \\ \sigma_3 = \langle\, \rho_3,\, \mu_3,\, \delta \,\rangle \quad \langle\, \mathtt{S}\,, \sigma_3 \,\rangle \longrightarrow_{com} \sigma_4 \\ \sigma_4 = \langle\, \rho_4,\, \mu_4,\, \delta \,\rangle \quad \sigma' = \langle\, \rho,\, \mu_4,\, \delta \,\rangle \end{array}}{\langle\, \mathtt{Obj.m(p_1, p_2)}\,, \sigma \,\rangle \longrightarrow_{com} \sigma'} \quad (com_{mcallpar2})$$

Il sistema *exp* può essere esteso per valutare la chiamata di metodi con due parametri in modo simile:

$$\frac{\begin{array}{c}\sigma = \langle\, \rho,\, \mu,\, \delta \,\rangle \quad \rho(\mathtt{Obj}) = l \quad \mu(l) = \langle\, \varphi_v,\, \varphi_m \,\rangle \quad \varphi_m(\mathtt{m}) = \langle\, \mathtt{x_1},\, \mathtt{x_2},\, \mathtt{MB} \,\rangle \\ \langle\, \mathtt{p_1}\,, \sigma \,\rangle \longrightarrow_{exp} \langle\, v_{p1},\, \sigma_1 \,\rangle \quad \sigma_1 = \langle\, \rho,\, \mu_1,\, \delta \,\rangle \\ \langle\, \mathtt{p_2}\,, \sigma_1 \,\rangle \longrightarrow_{exp} \langle\, v_{p2},\, \sigma_2 \,\rangle \quad \sigma_2 = \langle\, \rho,\, \mu_2,\, \delta \,\rangle \\ \mathtt{MB} = \{\mathtt{D\ S\ return\ e;}\} \quad \sigma_{local} = \langle\, \omega[\,{}^{v_{p1}}/_{\mathtt{x_1}},\, {}^{v_{p2}}/_{\mathtt{x_2}},\, {}^{l}/_{\mathtt{this}}\,],\, \mu_2,\, \delta \,\rangle \\ \langle\, \mathtt{D}\,, \sigma_{local} \,\rangle \longrightarrow_{local\_decl} \sigma_3 \\ \sigma_3 = \langle\, \rho_3,\, \mu_3,\, \delta \,\rangle \quad \langle\, \mathtt{S}\,, \sigma_3 \,\rangle \longrightarrow_{com} \sigma_4 \\ \sigma_4 = \langle\, \rho_4,\, \mu_4,\, \delta \,\rangle \quad \langle\, \mathtt{e}\,, \sigma_4 \,\rangle \longrightarrow_{exp} \langle\, v,\, \sigma_5 \,\rangle \\ \sigma_5 = \langle\, \rho_4,\, \mu_5,\, \delta \,\rangle \quad \sigma' = \langle\, \rho,\, \mu_5,\, \delta \,\rangle \end{array}}{\langle\, \mathtt{Obj.m(p_1, p_2)}\,, \sigma \,\rangle \longrightarrow_{exp} \langle\, v,\, \sigma' \,\rangle} \quad (exp_{mcallpar2})$$

Partiamo dallo stato iniziale vuoto $\sigma_0 = \langle\, \rho_0,\, \mu_0,\, \delta_0 \,\rangle$.

1) Applicazione della regola *prog* alle linee `01-45`

1.1) Applicazione della regola $decl_{concat}$ alle linee `01-30` Al termine dell'applicazione di questa regola lo stato è il seguente:

φ_{v_Coord} :

`x`	0
`y`	0

φ_{m_Coord} :

`set_prima_coordinata`	$\langle$ `l, {this.x = l; }` $\rangle$
`set_seconda_coordinata`	$\langle$ `a, {this.y = a; }` $\rangle$
`get_prima_coordinata`	$\langle$ $\perp$, `{return this.x;}` $\rangle$
`get_seconda_coordinata`	$\langle$ $\perp$, `{return this.y;}` $\rangle$

φ_{v_Swap} : ω

φ_{m_Swap} :

`swap`	$\langle$ `a, b, {Coord temp = new Coord(); temp = a; a = b; b = temp; }` $\rangle$

$$\sigma_1 = \langle\, \rho_0,\, \mu_0,\, \delta_1 \,\rangle$$

δ_1 :

`Coord`	$\langle\, \varphi_{v\_Coord},\, \varphi_{m\_Coord} \,\rangle$
`Swap`	$\langle\, \varphi_{v\_Swap},\, \varphi_{m\_Swap} \,\rangle$

1.2) Binding del parametro del metodo `main` ed applicazione delle regole del sistema *local_decl* alle linee `34-36`

$$\sigma_2 = \langle\, \rho_1,\, \mu_1,\, \delta_1 \,\rangle$$

ρ_1 :

`args`	*null*
`obj1`	l_1
`obj2`	l_2
`s`	l_3

μ_1 :

l_1	$\langle \ \varphi_v\_Coord :$ <table><tr><td>x</td><td>0</td></tr><tr><td>y</td><td>0</td></tr></table> $, \varphi_m\_Coord \rangle$
l_2	$\langle \ \varphi_v\_Coord :$ <table><tr><td>x</td><td>0</td></tr><tr><td>y</td><td>0</td></tr></table> $, \varphi_m\_Coord \rangle$
l_3	$\langle \varphi_v\_Swap : \omega \ , \ \varphi_m\_Swap \rangle$

1.3) Applicazione di com_{concat} alle linee `38-43`

1.3.1) Applicazione di $com_{mcallpar}$ alla linea `38`

1.3.2) Applicazione di $com_{mcallpar}$ alla linea `39`

1.3.3) Applicazione di $com_{mcallpar}$ alla linea `40`

1.3.4) Applicazione di $com_{mcallpar}$ alla linea `41`
Al termine dell'applicazione di questa regola lo stato è il seguente:

$$\sigma_3 = \langle \rho_1, \mu_2, \delta_1 \rangle$$

ρ_1 :

`args`	*null*
`obj1`	l_1
`obj2`	l_2
`s`	l_3

μ_2 :

l_1	$\langle \ \varphi_v\_Coord1 :$ <table><tr><td>x</td><td>10</td></tr><tr><td>y</td><td>20</td></tr></table> $, \varphi_m\_Coord \rangle$
l_2	$\langle \ \varphi_v\_Coord2 :$ <table><tr><td>x</td><td>80</td></tr><tr><td>y</td><td>90</td></tr></table> $, \varphi_m\_Coord \rangle$
l_3	$\langle \varphi_v\_Swap : \omega \ , \ \varphi_m\_Swap \rangle$

1.3.5) Applicazione di $com_{mcallpar2}$ alla linea `43`

$$\rho_1(\mathtt{s}) = l_3$$
$$\mu_2(l_3) = \langle \varphi_v\_Swap, \varphi_m\_Swap \rangle$$
$$\sigma_{local1} = \langle \rho_{local1}, \mu_2, \delta_1 \rangle$$

ρ_{local1} :

`a`	l_1
`b`	l_2
`this`	l_3

μ_2 :

l_1	$\langle \ \varphi_v\_Coord1 :$ `x` 10, `y` 20 $, \varphi_m\_Coord \rangle$
l_2	$\langle \ \varphi_v\_Coord2 :$ `x` 80, `y` 90 $, \varphi_m\_Coord \rangle$
l_3	$\langle \varphi_v\_Swap : \omega \ , \ \varphi_m\_Swap \rangle$

1.3.5.1) Applicazione di $local\_decl_{var\_init}$ alla linea `24`

$$\sigma_{local2} = \langle \rho_{local2}, \mu_3, \delta_1 \rangle$$

ρ_{local2} :

`a`	l_1
`b`	l_2
`this`	l_3
`temp`	l_4

μ_3 :

l_1	$\langle \ \varphi_v\_Coord1 :$ `x` 10, `y` 20 $, \varphi_m\_Coord \rangle$
l_2	$\langle \ \varphi_v\_Coord2 :$ `x` 80, `y` 90 $, \varphi_m\_Coord \rangle$
l_3	$\langle \varphi_v\_Swap : \omega \ , \ \varphi_m\_Swap \rangle$
l_4	$\langle \ \varphi_v\_Coord3 :$ `x` 0, `y` 0 $, \varphi_m\_Coord \rangle$

1.3.5.2) Applicazione di com_{concat} alle linee `26-28`

$$\sigma_{local3} = \langle \rho_{local3}, \mu_3, \delta_1 \rangle$$

ρ_{local3} :

a	l_2
b	l_1
this	l_3
temp	l_1

μ_3 :

l_1	$\langle \ \varphi_v\_Coord1 :$ x 10 / y 20 , $\varphi_m\_Coord \rangle$
l_2	$\langle \ \varphi_v\_Coord2 :$ x 80 / y 90 , $\varphi_m\_Coord \rangle$
l_3	$\langle \varphi_v\_Swap : \omega \ , \ \varphi_m\_Swap \rangle$
l_4	$\langle \ \varphi_v\_Coord3 :$ x 0 / y 0 , $\varphi_m\_Coord \rangle$

1.3.5) Al termine dell'applicazione della regola $com_{mcallpar2}$ lo stato è il seguente:

$$\sigma_4 = \langle \rho_1, \mu_3, \delta_1 \rangle$$

ρ_1 :

args	*null*
obj1	l_1
obj2	l_2
s	l_3

μ_3 :

l_1	$\langle \ \varphi_v\_Coord1 :$ x 10 / y 20 , $\varphi_m\_Coord \rangle$
l_2	$\langle \ \varphi_v\_Coord2 :$ x 80 / y 90 , $\varphi_m\_Coord \rangle$
l_3	$\langle \varphi_v\_Swap : \omega \ , \ \varphi_m\_Swap \rangle$
l_4	$\langle \ \varphi_v\_Coord3 :$ x 0 / y 0 , $\varphi_m\_Coord \rangle$

6.3.1 Swap effettivo

Per realizzare lo swap tra le due istanze dobbiamo modificare il metodo `swap` della classe `Swap` nel seguente modo:

```
...
class Swap {
  public void swap(Coord a, Coord b) {
    Coord temp = new Coord();

    temp.set_prima_coordinata(a.get_prima_coordinata());
    temp.set_seconda_coordinata(a.get_seconda_coordinata());

    a.set_prima_coordinata(b.get_prima_coordinata());
    a.set_seconda_coordinata(b.get_seconda_coordinata());

    b.set_prima_coordinata(temp.get_prima_coordinata());
    b.set_seconda_coordinata(temp.get_seconda_coordinata());
  }
}
...
```