

Seminario e Progetto di Esame di Fine Corso 2021

CdL Triennale in Matematica, a.a. 2020/2021

Enrico Calandrini

Dipartimento di Matematica
Università di Pisa

30 Giugno 2021

Scopi del Progetto

- 1 Estendere Small21 con la dichiarazione multipla di variabile, di uno stesso tipo, senza inizializzazione
- 2 Estendere Small21 con la dichiarazione ed invocazione di procedure e funzioni con multipli parametri

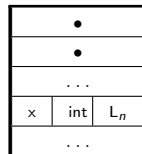
Dichiarazione multipla di variabile

Cosa ci si aspetta dall'aggiunta di questo nuovo costrutto?

Quando si dichiara **una** variabile, viene modificato l'activation record nel modo seguente:

Example

```
int main(){  
...  
Int x;  
...  
}
```



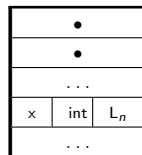
Dichiarazione multipla di variabile

Cosa ci si aspetta dall'aggiunta di questo nuovo costrutto?

Quando si dichiara una variabile, viene modificato l'activation record nel modo seguente:

Example

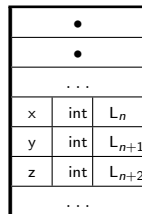
```
int main(){  
...  
Int x;  
...  
}
```



Perciò nella dichiarazione **multipla**, ci si aspetta un comportamento del tipo:

Example

```
int main(){  
...  
Int x, y, z;  
...  
}
```



Dichiarazione multipla di variabile: Osservazioni

- L'aggiunta della dichiarazione multipla di variabile, non aggiunge espressività al linguaggio, ma ha una natura puramente **sintattica**.

Example

```
int main(){  
  ...  
  int x, y, z;  
  ...  
}
```

equivalente

Example

```
int main(){  
  ...  
  int x;  
  int y;  
  int z;  
  ...  
}
```

Dichiarazione multipla di variabile: Sintassi Concreta e Sintassi Astratta

- **Sintassi Concreta:**

$$\text{ideSeq} \rightarrow , \text{ide ideSeq} \quad | \quad \epsilon$$
$$\text{Dcl} \rightarrow \text{Type ide ideseq}$$

- **Sintassi Astratta:**

$$\text{IdeSeq} ::= \text{Ide}[\text{:::}]\text{IdeSeq} \quad | \quad \text{Ide}$$
$$\text{Dcl} ::= [\text{mvar}]\text{Type IdeSeq}$$

Dichiarazione multipla di variabile: Sintassi Concreta e Sintassi Astratta

- Sintassi Concreta:

$$\text{ideSeq} \rightarrow , \text{ide ideSeq} \mid \epsilon$$
$$\text{Dcl} \rightarrow \text{Type ide ideseq}$$

- Sintassi Astratta:

$$\text{IdeSeq} ::= \text{Ide}[\text{:::}]\text{IdeSeq} \mid \text{Ide}$$
$$\text{Dcl} ::= [\text{mvar}]\text{Type IdeSeq}$$

Dichiarazione multipla di variabile: Sistema Y

Per quanto riguarda il **sistema Y**, vengono introdotte le seguenti regole:

$$[Y2'] \frac{\begin{array}{l} t \in \text{Simple} \quad IS = \text{Id}[\text{::}]IR \\ \langle [\text{var}] t \text{ Id } [\text{emptyE}], Y_\rho \rangle \rightarrow_Y \langle [\text{void}], Y_{\rho_1} \rangle \\ \langle [\text{mvar}] t IR, Y_{\rho_1} \rangle \rightarrow_Y \langle [\text{void}], Y_{\rho_2} \rangle \end{array}}{\langle [\text{mvar}] t IS, Y_\rho \rangle \rightarrow_Y \langle [\text{void}], Y_{\rho_2} \rangle}$$

$$[Y2''] \frac{\begin{array}{l} t \in \text{Simple} \quad IS = \text{Id} \\ \langle [\text{var}] t \text{ Id } [\text{emptyE}], Y_\rho \rangle \rightarrow_Y \langle [\text{void}], Y_{\rho_1} \rangle \end{array}}{\langle [\text{mvar}] t IS, Y_\rho \rangle \rightarrow_Y \langle [\text{void}], Y_{\rho_1} \rangle}$$

Dichiarazione multipla di variabile: Sistema Y

Per quanto riguarda il **sistema Y**, vengono introdotte le seguenti regole:

$$\begin{array}{c} t \in \text{Simple} \quad IS = \text{Id}[\:::]IR \\ \langle [\text{var}] t \text{ Id } [\text{emptyE}], Y_\rho \rangle \rightarrow_Y \langle [\text{void}], Y_{\rho_1} \rangle \\ \text{[Y2']} \frac{\langle [\text{mvar}] t IR, Y_{\rho_1} \rangle \rightarrow_Y \langle [\text{void}], Y_{\rho_2} \rangle}{\langle [\text{mvar}] t IS, Y_\rho \rangle \rightarrow_Y \langle [\text{void}], Y_{\rho_2} \rangle} \end{array}$$

Gestione errori di tipo:

$$\text{[E4.1]} \frac{t \notin \text{Simple}}{\langle [\text{mvar}] t IS, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

$$\begin{array}{c} t \in \text{Simple} \quad IS = \text{Id}[\:::]IR \\ \langle [\text{var}] t \text{ Id } [\text{emptyE}], Y_\rho \rangle \rightarrow_Y \langle [\text{void}], Y_{\rho_1} \rangle \\ \text{[E4.3]} \frac{\langle [\text{mvar}] t IR, Y_{\rho_1} \rangle \rightarrow_Y \langle [\text{terr}], Y_{\rho_2} \rangle}{\langle [\text{mvar}] t IS, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle} \end{array}$$

$$\begin{array}{c} t \in \text{Simple} \quad IS = \text{Id} \\ \text{[Y2'']} \frac{\langle [\text{var}] t \text{ Id } [\text{emptyE}], Y_\rho \rangle \rightarrow_Y \langle [\text{void}], Y_{\rho_1} \rangle}{\langle [\text{mvar}] t IS, Y_\rho \rangle \rightarrow_Y \langle [\text{void}], Y_{\rho_1} \rangle} \end{array}$$

$$\text{[E4.2]} \frac{t \in \text{Simple} \quad IS = \text{Id}[\:::]IR \quad \langle [\text{var}] t \text{ Id } [\text{emptyE}], Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_{\rho_1} \rangle}{\langle [\text{mvar}] t IS, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

$$\text{[E4.4]} \frac{t \in \text{Simple} \quad IS = \text{Id} \quad \langle [\text{var}] t \text{ Id } [\text{emptyE}], Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_{\rho_1} \rangle}{\langle [\text{mvar}] t IS, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

Sfruttando le regole per il sistema \mathcal{Y} precedentemente definite, le regole per SEM_{DCL} possono essere scritte semplicemente come:

$$[D2.2] \frac{\begin{array}{l} t \in \text{Simple} \quad IS = \text{Id}[\dots]IR \\ \langle [\text{var}] t \text{ Id } [\text{emptyE}], \sigma \rangle \rightarrow ([\text{void}], \sigma_1) \\ \langle [\text{mvar}] t IR, \sigma_1 \rangle \rightarrow ([\text{void}], \sigma_2) \end{array}}{\langle [\text{mvar}] t IS, \sigma \rangle \rightarrow ([\text{void}], \sigma_2)}$$

$$[D2.3] \frac{\begin{array}{l} t \in \text{Simple} \quad IS = \text{Id} \\ \langle [\text{var}] t \text{ Id } [\text{emptyE}], \sigma \rangle \rightarrow ([\text{void}], \sigma_1) \end{array}}{\langle [\text{mvar}] t IS, \sigma \rangle \rightarrow ([\text{void}], \sigma_1)}$$

Dichiarazione multipla di variabile: modifiche nell'interprete

Di seguito, le principali modifiche all'interprete:

```
59 ideSeq = ide lst
60     and
61
62 dcl =
63     Var of tye * ide * exp
64     | Mvar of tye * ideSeq (*AGGIUNTA*)
65     | Const of tye * ide * exp
66     | Array of tye * ide
67     | SeqD of dcl * dcl
68     | Pcd of tye * ide * seqfpars * blockP (*MODIFICATA_1*)
69     | ED
70     and
```

a sinistra è possibile osservare il nuovo costrutto **ideSeq** e la nuova dichiarazione **mvar**

Dichiarazione multipla di variabile: modifiche nell'interprete

Di seguito, le principali modifiche all'interprete:

```
59 ideSeq = ide l1st
60   and
61
62 dcl =
63   Var of tye * ide * exp
64   | Mvar of tye * ideSeq (*AGGIUNTA*)
65   | Const of tye * ide * exp
66   | Array of tye * ide
67   | SeqD of dcl * dcl
68   | Pcd of tye * ide * seqfpars * blockP (*MODIFICATA_1*)
69   | ED
70   and
```

a sinistra è possibile osservare il nuovo costrutto **ideSeq** e la nuova dichiarazione **mvar**

a destra invece il nuovo caso preso in considerazione nella **Sem_{DCL}**

```
| Mvar(ty,ideSeq) -> (*AGGIUNTA*)
  (match ideSeq with
  | [id1] when (isSimple ty)
    -> (let d1 = Var(ty,id1,EE) in
        dclSem d1 (sk,mu))
  | id1::idR when (isSimple ty)
    -> (let d1 = Var(ty,id1,EE) in
        let (_,sk1,mu1) = dclSem d1 (sk,mu) in
        let d2 = Mvar(ty,idR) in
        dclSem d2 (sk1,mu1))
  | _ -> raise(TypeErrorIS("E4.1: dclSem", ideSeq))
```

Dichiarazione multipla di variabile: modifiche nell'interprete

Di seguito, le principali modifiche all'interprete:

```
59 ideSeq = ide llist
60   and
61
62 dcl =
63   Var of tye * ide * exp
64   | Mvar of tye * ideSeq (*AGGIUNTA*)
65   | Const of tye * ide * exp
66   | Array of tye * ide
67   | SeqD of dcl * dcl
68   | Pcd of tye * ide * seqfpars * blockP (*MODIFICATA_1*)
69   | ED
70   and
```

a sinistra è possibile osservare il nuovo costrutto **ideSeq** e la nuova dichiarazione **mvar**

a destra invece il nuovo caso preso in considerazione nella **Sem_{DCL}**

```
| Mvar(ty,ideSeq) -> (*AGGIUNTA*)
  (match ideSeq with
  | [idi] when (isSimple ty)
    -> (Let d1 = Var(ty,idi,EE) in
        dclSem d1 (sk,mu))
  | idi::idR when (isSimple ty)
    -> (Let d1 = Var(ty,idi,EE) in
        let (_,(sk1,mu1)) = dclSem d1 (sk,mu) in
        let d2 = Mvar(ty,idR) in
        dclSem d2 (sk1,mu1))
  | _ -> raise(TypeErrorIS("E4.1: dclSem", ideSeq))
```

L'unico errore controllato a questo livello è l'E4.1

Dichiarazione multipla di variabile: verifica

Si consideri ora il seguente esempio:

Example


```
Program sum(){  
  int x, y, z;  
  x = 2;  
  y = 4;  
  z = x + y;  
}
```

Dichiarazione multipla di variabile: verifica

Si consideri ora il seguente esempio:

Example

```
Program sum(){  
  int x, y, z;  
  x = 2;  
  y = 4;  
  z = x + y;  
}
```


rappresentato in
Small21 come:

```
let p = Prog ("sum",  
  Block  
    (SeqD  
      (Mvar (Int, ["x";"y";"z"]),  
        ED),  
      SeqC  
        (UnL (Upd (Val "x", N 2)),  
          SeqC  
            (UnL (Upd (Val "y", N 4)),  
              UnL (Upd (Val "z", Plus (Val "x", Val "y"))))))))));
```

Dichiarazione multipla di variabile: verifica

Si consideri ora il seguente esempio:

Example

```
Program sum(){  
  int x, y, z;  
  x = 2;  
  y = 4;  
  z = x + y;  
}
```

⇒
rappresentato in
Small21 come:

```
let p = Prog ("sum",  
  Block  
    (Seq0  
      (Mvar (Int, ["x";"y";"z"]),  
        ED),  
      SeqC  
        (UnL (Upd (Val "x", N 2)),  
          SeqC  
            (UnL (Upd (Val "y", N 4)),  
              UnL (Upd (Val "z", Plus (Val "x", Val "y"))))))))));
```

⇒
lanciando progSem
su tale programma:

```
Stack:  
>{sum,0,[z/(Mint,L2);  
  y/(Mint,L1);  
  x/(Mint,L0)],:cmdNext:[,N]}  
] ]  
Store:  
[L0<-Undef,L1<-Undef,L2<-Undef]
```

```
Stack:  
>{sum,0,[z/(Mint,L2);  
  y/(Mint,L1);  
  x/(Mint,L0)],:cmdNext:[,N]}  
] ]  
Store:  
[L0<-2,L1<-Undef,L2<-Undef]
```



```
Stack:  
>{sum,0,[z/(Mint,L2);  
  y/(Mint,L1);  
  x/(Mint,L0)],:cmdNext:[,N]}  
] ]  
Store:  
[L0<-2,L1<-4,L2<-Undef]
```



```
Stack:  
>{sum,0,[z/(Mint,L2);  
  y/(Mint,L1);  
  x/(Mint,L0)],:cmdNext:[,N]}  
] ]  
Store:  
[L0<-2,L1<-4,L2<-6]
```

```
SUCCESSFUL_TERMINATION  
- : unit/2 = ()
```


Scopi del Progetto

- 1 Estendere Small21 con la dichiarazione multipla di variabile, di uno stesso tipo, senza inizializzazione
- 2 Estendere Small21 con la dichiarazione ed invocazione di procedure e funzioni con multipli parametri

Dichiarazione ed invocazione di procedure e funzioni con multipli parametri: espressività

Si immagini di voler sviluppare il seguente esempio:

Example

```
Program prova(){  
void swap(int a, int b){  
...  
}  
...  
int x, y;  
...  
swap(x,y);  
...  
}
```

Non implementabile al momento

Ad un certo punto
del programma sarà
necessario invertire
due valori!



Bubble sort

Dichiarazione ed invocazione di procedure e funzioni con multipli parametri: espressività

Si immagini di voler sviluppare il seguente esempio:

Example

```
Program prova(){  
void swap(int a, int b){  
...  
}  
...  
int x, y;  
...  
swap(x,y);  
...  
}
```

Non implementabile al momento

Ad un certo punto
del programma sarà
necessario invertire
due valori!



Bubble sort

⚠ al momento attuale non è possibile farlo a meno che non sia già dichiarata una variabile ausiliaria

Dichiarazione ed invocazione di procedure e funzioni con multipli parametri: espressività

Si immagini di voler sviluppare il seguente esempio:

Example

```
Program prova(){  
void swap(int a, int b){  
...  
}  
...  
int x, y;  
...  
swap(x,y);  
...  
}
```

Non implementabile al momento

Ad un certo punto
del programma sarà
necessario invertire
due valori!



Bubble sort

⚠ al momento attuale non è possibile
farlo a meno che non sia già dichiarata
una variabile ausiliaria



↑ ESPRESSIVITÀ

Dichiarazione ed invocazione di procedure e funzioni con multipli parametri: Bubble sort

Sulla base di quanto detto precedentemente, si vorrebbe implementare in Small21 il seguente algoritmo che **ordina un array** sfruttando il meccanismo del Bubble sort:

```
1
2 Prog OrdinaArray()
3 {
4     void swap(ref int a, ref int b)
5     {
6         int tmp = a;
7         a = b;
8         b = tmp;
9     }
10    void bubblesort(int i, int N, bool controllo)
11    {
12        if (i<N)
13            {
14                if (A[i]>A[i+1])
15                    {
16                        swap(A[i],A[i+1]);
17                        controllo = true;
18                    }
19                bubblesort(i+1,N,controllo);
20            }
21        if (controllo = true)
22            bubblesort(0, N-1, false);
23        return;
24    }
25    int A[4];
26    A[0] = 8;
27    A[1] = 5;
28    A[2] = 6;
29    A[3] = 1;
30    bubblesort(0,3,false);
31 }
```

Estendendo Small21 con la dichiarazione di funzione con multipli parametri, ci si aspetta di essere in grado di farlo!

Dichiarazione ed invocazione di procedure e funzioni con multipli parametri: Osservazioni

Quali sono gli aspetti a cui prestare attenzione nell'estensione considerata?

Dichiarazione ed invocazione di procedure e funzioni con multipli parametri: Osservazioni

Quali sono gli aspetti a cui prestare attenzione nell'estensione considerata?

- **dichiarazione di funzione:**
 - 1 Controllo dei tipi (Simple)

Dichiarazione ed invocazione di procedure e funzioni con multipli parametri: Osservazioni

Quali sono gli aspetti a cui prestare attenzione nell'estensione considerata?

- **dichiarazione di funzione:**
 - 1 Controllo dei tipi (Simple)
 - 2 Controllo degli identificatori (Non ripetuti)

Dichiarazione ed invocazione di procedure e funzioni con multipli parametri: Osservazioni

Quali sono gli aspetti a cui prestare attenzione nell'estensione considerata?

- **dichiarazione di funzione:**
 - 1 Controllo dei tipi (Simple)
 - 2 Controllo degli identificatori (Non ripetuti)
 - 3 Chiusura (Lista di tipi)

Dichiarazione ed invocazione di procedure e funzioni con multipli parametri: Osservazioni

Quali sono gli aspetti a cui prestare attenzione nell'estensione considerata?

- **dichiarazione di funzione:**
 - 1 Controllo dei tipi (Simple)
 - 2 Controllo degli identificatori (Non ripetuti)
 - 3 Chiusura (Lista di tipi)
- **invocazione di funzione:**
 - 1 Corrispondenza tra parametri attuali e formali (TrNFun)

Dichiarazione ed invocazione di procedure e funzioni con multipli parametri: Sintassi Concreta

Per quanto riguarda la sintassi Concreta, la forma attuale dei parametri **formali** e **attuali** è già pensata per tale sviluppo:

$$FP \rightarrow Fp \text{ } FPs \quad | \quad \epsilon$$
$$FPs \rightarrow , \text{ } Fp \text{ } FPs \quad | \quad \epsilon$$
$$Fp \rightarrow PPF \text{ } Type \text{ } ide$$
$$PPF \rightarrow \epsilon \quad | \quad ref$$
$$AP \rightarrow Ap \text{ } APs \quad | \quad \epsilon$$
$$APs \rightarrow , \text{ } Ap \text{ } APs \quad | \quad \epsilon$$
$$Ap \rightarrow exp$$

Dichiarazione ed invocazione di procedure e funzioni con multipli parametri: Sintassi Astratta

Sfruttando la sintassi concreta definita sopra, possono essere definiti:

- Un nuovo costruttore [SeqFP]:

$$\text{SFPars} ::= \text{FPars} [\text{SeqFP}] \text{SFPars} \quad | \quad [\text{emptySFP}]$$
$$\text{FPars} ::= [\text{fp}] \text{PPF} \text{Type} \text{Ide}$$
$$\text{PPF} ::= [\text{value}] \quad | \quad [\text{ref}]$$

- Un nuovo costruttore [SeqaP]:

$$\text{SAPars} ::= \text{APars} [\text{SeqAP}] \text{SAPars} \quad | \quad [\text{emptySAP}]$$
$$\text{APars} ::= [\text{ap}] \text{Exp}$$

Dichiarazione ed invocazione di procedure e funzioni con multipli parametri: Sintassi Astratta

Sfruttando la sintassi concreta definita sopra, possono essere definiti:

- Un nuovo costruttore [SeqFP]:

$$\text{SFPars} ::= \text{FPars [SeqFP] SFPars} \quad | \quad [\text{emptySFP}]$$
$$\text{FPars} ::= [\text{fp}] \text{PPF Type Ide}$$
$$\text{PPF} ::= [\text{value}] \quad | \quad [\text{ref}]$$

- Un nuovo costruttore [SeqaP]:

$$\text{SAPars} ::= \text{APars [SeqAP] SAPars} \quad | \quad [\text{emptySAP}]$$
$$\text{APars} ::= [\text{ap}] \text{Exp}$$

Dichiarazione ed invocazione di procedure e funzioni con multipli parametri: Sintassi Astratta

Sfruttando la sintassi concreta definita sopra, possono essere definiti:

- Un nuovo costruttore [SeqFP]:

$$\text{SFPars} ::= \text{FPars} [\text{SeqFP}] \text{SFPars} \quad | \quad [\text{emptySFP}]$$
$$\text{FPars} ::= [\text{fp}] \text{PPF} \text{Type} \text{Ide}$$
$$\text{PPF} ::= [\text{value}] \quad | \quad [\text{ref}]$$

- Un nuovo costruttore [SeqaP]:

$$\text{SAPars} ::= \text{APars} [\text{SeqAP}] \text{SAPars} \quad | \quad [\text{emptySAP}]$$
$$\text{APars} ::= [\text{ap}] \text{Exp}$$

Sono stati eliminati [EFP] ed [EAP]

Dichiarazione di procedure e funzioni con multipli parametri: Sistema Υ

Problemi da controllare:

- 1 Controllo dei tipi
- 2 Controllo degli identificatori
- 3 Chiusura



Soluzione

Modifica della regola [Y5]

$$\begin{array}{l} t \in \text{Simple} \cup \{\text{void}\} \quad F = [\text{fp}] \text{ p } t' \text{ I}' \\ Y_\rho |_0(\text{I}) = \perp \quad t' \in \text{Simple} \\ \text{[Y5]} \frac{\text{I}/[\text{abs}] t[::]t' \otimes Y_\rho = Y_{\rho_1}}{\langle [\text{pcd}] t \text{ I } F \text{ Bs}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_{\rho_1})} \end{array}$$

Come?

Dichiarazione di procedure e funzioni con multipli parametri: Sistema Y

Problemi da controllare:

- 1 Controllo dei tipi
- 2 Controllo degli identificatori
- 3 Chiusura



Soluzione

Modifica della regola [Y5]

$$\begin{array}{l} t \in \text{Simple} \cup \{\text{void}\} \quad F = [\text{fp}] \text{ p } t' \text{ l}' \\ Y_\rho |_0(\text{l}) = \perp \quad t' \in \text{Simple} \\ \text{[Y5]} \frac{[\text{l}/[\text{abs}] t[::]t'] \otimes Y_\rho = Y_{\rho_1}}{\langle [\text{pcd}] t \text{ l } F \text{ Bs}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_{\rho_1})} \end{array}$$

Come?

- Estensione del controllo dei tipi per ogni parametro formale

Dichiarazione di procedure e funzioni con multipli parametri: Sistema Y

Problemi da controllare:

- 1 Controllo dei tipi
- 2 Controllo degli identificatori
- 3 Chiusura



Soluzione

Modifica della regola [Y5]

$$\begin{array}{l} t \in \text{Simple} \cup \{\text{void}\} \quad F = [\text{fp}] \text{ p } t' \text{ I}' \\ Y_\rho |_0(\text{I}) = \perp \quad t' \in \text{Simple} \\ \text{[Y5]} \frac{\text{I}/[\text{abs}] t[::]t' \otimes Y_\rho = Y_{\rho_1}}{\langle [\text{pcd}] t \text{ I } F \text{ Bs}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_{\rho_1})} \end{array}$$

Come?

- Estensione del controllo dei tipi per ogni parametro formale
- Aggiunta del controllo di identificatori non ripetuti

Dichiarazione di procedure e funzioni con multipli parametri: Sistema Υ

Problemi da controllare:

- 1 Controllo dei tipi
- 2 Controllo degli identificatori
- 3 **Chiusura**



Soluzione

Modifica della regola [Y5]

$$\begin{array}{l} t \in \text{Simple} \cup \{\text{[void]}\} \quad F = [\text{fp}] \text{ p } t' \text{ I}' \\ Y_\rho |_0(\text{I}) = \perp \quad t' \in \text{Simple} \\ \text{[Y5]} \frac{\text{I}/[\text{abs } t[::]t'] \otimes Y_\rho = Y_{\rho_1}}{\langle [\text{pcd}] t \text{ I } F \text{ Bs}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_{\rho_1})} \end{array}$$

Come?

- Estensione del controllo dei tipi per ogni parametro formale
- Aggiunta del controllo di identificatori non ripetuti
- **Estensione della catena dei tipi**

Dichiarazione di procedure e funzioni con multipli parametri: Sistema Y

Problemi da controllare:

- 1 **Controllo dei tipi**
- 2 Controllo degli identificatori
- 3 Chiusura



Regola [Y5] modificata:

$$\begin{array}{l}
 t \in \text{Simple} \cup \{\text{void}\} \quad Y_\rho |_0(l) = \perp \\
 SF = F_1[\text{seqFP}] F_2[\text{seqFP}] F_3 \cdots [\text{seqFP}] F_n[\text{seqFP}][\text{emptySFP}] \\
 F_1 = [\text{fp}] p_1 t_1 l_1 \quad t_1 \in \text{Simple} \\
 F_2 = [\text{fp}] p_2 t_2 l_2 \quad t_2 \in \text{Simple} \\
 \quad \quad \quad l_2 \neq l_1 \\
 F_3 = [\text{fp}] p_3 t_3 l_3 \quad t_3 \in \text{Simple} \\
 \quad \quad \quad l_3 \neq l_1 \quad l_3 \neq l_2 \\
 \quad \quad \quad \vdots \\
 F_n = [\text{fp}] p_n t_n l_n \quad t_n \in \text{Simple} \\
 l_n \neq l_1 \quad l_n \neq l_2 \cdots l_n \neq l_{(n-1)} \\
 t_1[::]t_2[::]t_3[::] \cdots [::] t_n = t' \\
 [l/[abs] t[::]t'] \otimes Y_\rho = Y_{\rho_1} \\
 \hline
 \text{[Y5]} \frac{}{\langle [\text{pcd}] t \mid SF \text{ Bs}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_{\rho_1})}
 \end{array}$$

Dichiarazione di procedure e funzioni con multipli parametri: Sistema Y

Problemi da controllare:

- 1 Controllo dei tipi
- 2 Controllo degli identificatori
- 3 Chiusura



Regola [Y5] modificata:

$$t \in \text{Simple} \cup \{\text{void}\} \quad Y_\rho |_0(l) = \perp$$

$$SF = F_1[\text{seqFP}] F_2[\text{seqFP}] F_3 \cdots [\text{seqFP}] F_n[\text{seqFP}][\text{emptySFP}]$$

$$F_1 = [\text{fp}] p_1 t_1 l_1 \quad t_1 \in \text{Simple}$$

$$F_2 = [\text{fp}] p_2 t_2 l_2 \quad t_2 \in \text{Simple}$$

$$l_2 \neq l_1$$

$$F_3 = [\text{fp}] p_3 t_3 l_3 \quad t_3 \in \text{Simple}$$

$$l_3 \neq l_1 \quad l_3 \neq l_2$$

$$\vdots$$

$$F_n = [\text{fp}] p_n t_n l_n \quad t_n \in \text{Simple}$$

$$l_n \neq l_1 \quad l_n \neq l_2 \cdots l_n \neq l_{(n-1)}$$

$$t_1[::]t_2[::]t_3[::] \cdots [::] t_n = t'$$

$$[l/[abs] t[::]t'] \otimes Y_\rho = Y_{\rho_1}$$

$$[Y5] \frac{}{\langle [pcd] t \mid SF \text{ Bs}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_{\rho_1})}$$

Dichiarazione di procedure e funzioni con multipli parametri: Sistema Y

Problemi da controllare:

- 1 Controllo dei tipi
- 2 Controllo degli identificatori
- 3 Chiusura



Regola [Y5] modificata:

$$t \in \text{Simple} \cup \{\text{void}\} \quad Y_\rho |_0(l) = \perp$$

$$SF = F_1[\text{seqFP}] F_2[\text{seqFP}] F_3 \cdots [\text{seqFP}] F_n[\text{seqFP}][\text{emptySFP}]$$

$$F_1 = [\text{fp}] p_1 t_1 l_1 \quad t_1 \in \text{Simple}$$

$$F_2 = [\text{fp}] p_2 t_2 l_2 \quad t_2 \in \text{Simple}$$

$$l_2 \neq l_1$$

$$F_3 = [\text{fp}] p_3 t_3 l_3 \quad t_3 \in \text{Simple}$$

$$l_3 \neq l_1 \quad l_3 \neq l_2$$

$$\vdots$$

$$F_n = [\text{fp}] p_n t_n l_n \quad t_n \in \text{Simple}$$

$$l_n \neq l_1 \quad l_n \neq l_2 \cdots l_n \neq l_{(n-1)}$$

$$t_1 [::] t_2 [::] t_3 [::] \cdots [::] t_n = t'$$

$$[l / [\text{abs}] t [::] t'] \otimes Y_\rho = Y_{\rho_1}$$

$$[Y5] \frac{}{\langle [\text{pcd}] t \mid SF \text{ Bs}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_{\rho_1})}$$

Dichiarazione di procedure e funzioni con multipli parametri: Sistema Υ

Mentre la regola [Y6] viene quasi lasciata inalterata:

$$t \in \text{Simple} \cup \{\{\text{void}\} \quad \text{SF} = [\text{emptySFP}]\}$$
$$[Y6] \frac{Y_\rho |_0(l) = \perp \quad [l/[abs] t] \otimes Y_\rho = Y_{\rho_1}}{\langle [pcd] t \mid \text{SF } Bs, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_{\rho_1})}$$

Dichiarazione di procedure e funzioni con multipli parametri: Sistema Y

Mentre la regola [Y6] viene quasi lasciata inalterata:

$$t \in \text{Simple} \cup \{\text{void}\} \quad \text{SF} = [\text{emptySFP}]$$
$$[\text{Y6}] \frac{Y_\rho |_0(l) = \perp \quad [l / [\text{abs } t]] \otimes Y_\rho = Y_{\rho_1}}{\langle [\text{pcd}] t \mid \text{SF Bs}, Y_\rho \rangle \rightarrow_Y \langle [\text{void}], Y_{\rho_1} \rangle}$$

Gestione errori di tipo

$$[\text{E12}] \frac{t \notin \text{Simple} \cup \{\text{void}\}}{\langle [\text{pcd}] t \mid \text{SF Bs}, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

$$[\text{E14}] \frac{t \in \text{Simple} \cup \{\text{void}\} \quad Y_\rho |_0(l) \neq \perp}{\langle [\text{pcd}] t \mid \text{SF Bs}, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

$$t \in \text{Simple} \cup \{\text{void}\} \quad Y_\rho |_0(l) = \perp$$
$$\text{SF} = F_1[\text{seqFP}] F_2[\text{seqFP}] F_3 \cdots [\text{seqFP}] F_n[\text{seqFP}][\text{emptySFP}]$$
$$F_1 = [\text{fp}] p_1 t_1 l_1 \quad t_1 \in \text{Simple}$$
$$\vdots$$
$$F_k = [\text{fp}] p_k t_k l_k \quad t_k \in \text{Simple}$$
$$l_{k-r} = l_k \quad r \in [1, \dots, (k-1)]$$
$$[\text{E70.1}] \frac{}{\langle [\text{pcd}] t \mid \text{SF Bs}, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

Controllo degli identificatori

Dichiarazione di procedure e funzioni con multipli parametri: Sistema Y

Mentre la regola [Y6] viene quasi lasciata inalterata:

$$[Y6] \frac{t \in \text{Simple} \cup \{\text{void}\} \quad SF = [\text{emptySFP}] \quad Y_\rho |_0(l) = \perp \quad [l / [\text{abs}] t] \otimes Y_\rho = Y_{\rho_1}}{\langle [\text{pcd}] t \mid SF \text{ Bs}, Y_\rho \rangle \rightarrow_Y \langle [\text{void}], Y_{\rho_1} \rangle}$$

Gestione errori di tipo

$$[E12] \frac{t \notin \text{Simple} \cup \{\text{void}\}}{\langle [\text{pcd}] t \mid SF \text{ Bs}, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

$$[E14] \frac{t \in \text{Simple} \cup \{\text{void}\} \quad Y_\rho |_0(l) \neq \perp}{\langle [\text{pcd}] t \mid SF \text{ Bs}, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

$$[E70.2] \frac{\begin{array}{c} t \in \text{Simple} \cup \{\text{void}\} \quad Y_\rho |_0(l) = \perp \\ SF = F_1[\text{seqFP}] F_2[\text{seqFP}] F_3 \cdots [\text{seqFP}] F_n[\text{seqFP}][\text{emptySFP}] \\ F_1 = [\text{fp}] p_1 t_1 l_1 \quad t_1 \\ \vdots \\ F_k = [\text{fp}] p_k t_k l_k \quad t_k \notin \text{Simple} \end{array}}{\langle [\text{pcd}] t \mid SF \text{ Bs}, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

Controllo dei tipi

Dichiarazione di procedure e funzioni con multipli parametri: SEM_{DCL}

⇒ Le nuove regole per la SEM_{DCL} possono essere espresse come:

$$\begin{array}{l}
 t \in \text{Simple} \cup \{\text{void}\} \quad \Delta|_0(l) = \perp \\
 \text{SF} = F_1[\text{seqFP}] F_2[\text{seqFP}] F_3 \cdots [\text{seqFP}] F_n[\text{seqFP}][\text{emptySFP}] \\
 F_1 = [\text{fp}] p_1 t_1 l_1 \quad t_1 \in \text{Simple} \\
 F_2 = [\text{fp}] p_2 t_2 l_2 \quad t_2 \in \text{Simple} \\
 \quad \quad \quad l_2 \neq l_1 \\
 F_3 = [\text{fp}] p_3 t_3 l_3 \quad t_3 \in \text{Simple} \\
 \quad \quad \quad l_3 \neq l_1 \quad l_3 \neq l_2 \\
 \quad \quad \quad \vdots \\
 F_n = [\text{fp}] p_n t_n l_n \quad t_n \in \text{Simple} \\
 l_n \neq l_1 \quad l_n \neq l_2 \cdots l_n \neq l_{(n-1)} \\
 t_1[::]t_2[::]t_3[::] \cdots [::] t_n = t' \\
 \text{tr} = [\text{abs}] t[::]t' \quad \text{Bs} = [\text{BlockS}] d s \\
 \quad \quad \quad \star l, \text{tr}, \text{SF}, d, s, \#\Delta \star = \text{vr} \\
 \text{[D5]} \frac{}{\langle [\text{pcd}] t \mid \text{SF Bs}, (\Delta, \mu) \rangle \rightarrow ([\text{void}], ([l/(\text{tr}, \text{vr})] \otimes \Delta, \mu))} \\
 \text{[D6]} \frac{\begin{array}{l} t \in \text{Simple} \cup \{\text{void}\} \quad \text{SF} = [\text{emptySFP}] \\ \Delta|_0(l) = \perp \quad \text{tr} = [\text{abs}] t \\ [\text{BlockS}] d s \quad \star l, \text{tr}, \text{SF}, d, s, \#\Delta \star = \text{vr} \end{array}}{\langle [\text{pcd}] t \mid \text{SF Bs}, (\Delta, \mu) \rangle \rightarrow ([\text{void}], ([l/(\text{tr}, \text{vr})] \otimes \Delta, \mu))}
 \end{array}$$

Dichiarazione di procedure e funzioni con multipli parametri: modifiche nell'interprete

Di seguito, le principali modifiche all'**interprete** di Small21 in base alle regole definite:

- Introduzione di **seqfpars** e modifica alla dichiarazione **pcd**

```
62 dcl =
63   Var of tye * ide * exp
64   | Mvar of tye * ideSeq (*AGGIUNTA*)
65   | Const of tye * ide * exp
66   | Array of tye * ide
67   | SeqD of dcl * dcl
68   | Pcd of tye * ide * seqfpars * blockP (*MODIFICATA_1*)
69   | ED
70   and
71
72 seqfpars = ESFP (*AGGIUNTA_1*)
73   | SFP of fpars * seqfpars
74   and
75 ..
```

```
| Pcd(ty,ide,sfp,blockP) -> (*MODIFICATA_1*)
  (match (declared sk ide,sfp,blockP) with
  | (_,_,_)
    when not(isSimple ty) && (ty <> Void)
      -> raise(TypeErrorI("E12: dclSen",ide))
  | (true,_,_) -> raise(TypeErrorI("E14: dclSen",ide))
  | (_,SFP(fp,sRfp),BlockP(d,s))
      -> let (_,chaIntye) = chaIntypefpars sfp [] in
          let tr = Abs(ty,chaIntye) in
          let clos = cLosT(ide,tr,sfp,d,s,sizeS sk) in
          let den = DAbs(tr,clos) in
          let skF = binds sk ide den in
          (Void,(skF,mu))
  | (_,ESFP,BlockP(d,s))
      -> (let tr = Abs(ty,[]) in
          let clos = cLosT(ide,ty,ESFP,d,s,sizeS sk) in
          let den = DAbs(tr,clos) in
          let skF = binds sk ide den in
          (Void,(skF,mu))))
```

- Modifica della funzione **dclSem** che fa utilizzo della funzione ausiliaria **chaIntypefpars** ↓

Dichiarazione di procedure e funzioni con multipli parametri: modifiche nell'interprete

Di seguito, le principali modifiche all'**interprete** di Small21 in base alle regole definite:

- Introduzione di **seqfpars** e modifica alla dichiarazione **pcd**

```
62 dcl =
63   Var of tye * ide * exp (*AGGIUNTA*)
64   | Hvar of tye * ideSeq (*AGGIUNTA*)
65   | Const of tye * ide * exp
66   | Array of tye * ide
67   | SeqD of dcl * dcl
68   | Pcd of tye * ide * seqfpars * blockP (*MODIFICATA_1*)
69   | ED
70   and
71
72 seqfpars = ESFP (*AGGIUNTA_1*)
73   | SFP of fpars * seqfpars
74   and
```

```
916 chaintypefpars sfp ides = match sfp with
917   ESFP -> (Void,[])
918   | SFP(fp,sfpr) -> (match fp with
919     | FP(_,t,ide) when (isSimple t) && (not(mem ide ides))
920       -> let ides! = ide::ides in
921         let (_,tr) = chaintypefpars sfpr ides! in
922           (Void,t::tr)
923     | FP(_,t,id) when not(isSimple t)
924       -> raise(TypeErrorI("E70.1: dclSem, type not allowed",id ))
925     | FP(_,t,id) -> raise(TypeErrorI("E70.2: dclSem, duplicated ide",id ))
926   and
```

- Modifica della funzione **dclSem** che fa utilizzo della funzione ausiliaria **chaintypefpars**

Invocazione di procedure e funzioni con multipli parametri: sistema Y

Problemi da controllare:

- 1 Corrispondenza tra parametri attuali e formali



Soluzione

Introduzione della regola [Y35'] e
modifica della regola [Y37]

Invocazione di procedure e funzioni con multipli parametri: sistema Υ

Problemi da controllare:

- 1 Corrispondenza tra parametri attuali e formali



Soluzione

Introduzione della regola [Y35'] e
modifica della regola [Y37]

$$\begin{array}{c} \text{sfps} = \text{fps} [\text{seqFP}] \text{FR} \quad \text{saps} = \text{aps} [\text{seqAP}] \text{AR} \\ \langle \text{fps} \triangleleft \text{aps}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho) \\ \langle \text{FR} \triangleleft_1 \text{AR}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho) \\ \hline [\text{Y35}'] \frac{}{\langle \text{sfps} \triangleleft_1 \text{saps}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho)} \end{array}$$

$$[\text{Y37}] \frac{\text{sfps} = [\text{emptySFP}] \quad \text{saps} = [\text{emptyAFP}]}{\langle \text{sfps} \triangleleft_1 \text{saps}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho)}$$

⚠ Si utilizza anche in questo caso un'analisi induttiva sfruttando le regole [Y35] e [Y36] che controllano a livello base

Invocazione di procedure e funzioni con multipli parametri: sistema \mathcal{Y}

Problemi da controllare:

- 1 Corrispondenza tra parametri attuali e formali



Soluzione

Introduzione della regola [Y35'] e
modifica della regola [Y37]

$$\begin{array}{l} \text{sfps} = \text{fps} \text{ [seqFP]} \quad \text{FR} \quad \text{saps} = \text{aps} \text{ [seqAP]} \quad \text{AR} \\ \langle \text{fps} \triangleleft \text{aps}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho) \\ \langle \text{FR} \triangleleft_1 \text{AR}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho) \\ \text{[Y35']} \frac{}{\langle \text{sfps} \triangleleft_1 \text{saps}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho)} \end{array}$$

$$\text{[Y37]} \frac{\text{sfps} = [\text{emptySFP}] \quad \text{saps} = [\text{emptyAFP}]}{\langle \text{sfps} \triangleleft_1 \text{saps}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho)}$$

⚠ Si utilizza anche in questo caso un'analisi induttiva sfruttando le regole [Y35] e [Y36] che controllano a livello base

Gestione errori di tipo: errori modificati

$$\text{[E62]} \frac{\text{sfps} = [\text{emptySFP}] \quad \text{saps} \neq [\text{emptyAFP}]}{\langle \text{sfps} \triangleleft_1 \text{saps}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$\text{[E63]} \frac{\text{sfps} \neq [\text{emptySFP}] \quad \text{saps} = [\text{emptyAFP}]}{\langle \text{sfps} \triangleleft_1 \text{saps}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

Invocazione di procedure e funzioni con multipli parametri: sistema Υ

Problemi da controllare:

- 1 Corrispondenza tra parametri attuali e formali



$$\begin{array}{l} \text{sfps} = \text{fps} [\text{seqFP}] \text{FR} \quad \text{saps} = \text{aps} [\text{seqAP}] \text{AR} \\ \langle \text{fps} \triangleleft_1 \text{aps}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho) \\ \langle \text{FR} \triangleleft_1 \text{AR}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho) \\ \hline [\text{Y35}'] \frac{}{\langle \text{sfps} \triangleleft_1 \text{saps}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho)} \end{array}$$

Soluzione

Introduzione della regola [Y35'] e
modifica della regola [Y37]

$$[\text{Y37}] \frac{\text{sfps} = [\text{emptySFP}] \quad \text{saps} = [\text{emptyAFP}]}{\langle \text{sfps} \triangleleft_1 \text{saps}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho)}$$

⚠ Si utilizza anche in questo caso un'analisi induttiva sfruttando le regole [Y35] e [Y36] che controllano a livello base

Gestione errori di tipo: nuovi errori

$$[\text{E71}] \frac{\begin{array}{l} \text{sfps} = \text{fps} [\text{seqFP}] \text{FR} \quad \text{saps} = \text{aps} [\text{seqAP}] \text{AR} \\ \langle \text{fps} \triangleleft_1 \text{aps}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho) \end{array}}{\langle \text{sfps} \triangleleft_1 \text{saps}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

$$[\text{E72}] \frac{\begin{array}{l} \text{sfps} = \text{fps} [\text{seqFP}] \text{FR} \quad \text{saps} = \text{aps} [\text{seqAP}] \text{AR} \\ \langle \text{fps} \triangleleft_1 \text{aps}, Y_\rho \rangle \rightarrow_Y ([\text{void}], Y_\rho) \\ \langle \text{FR} \triangleleft_1 \text{AR}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho) \end{array}}{\langle \text{sfps} \triangleleft_1 \text{saps}, Y_\rho \rangle \rightarrow_Y ([\text{terr}], Y_\rho)}$$

Invocazione di procedure e funzioni con multipli parametri: TrNFun

Con lo stesso procedimento utilizzato nel sistema Y, si introduce la nuova regola TrNFun per **il controllo della corrispondenza** tra parametri attuali e formali:

$$\begin{array}{l} \text{sfps} = \text{fps} [\text{seqFP}] \text{FR} \quad \text{saps} = \text{aps} [\text{seqAP}] \text{AR} \\ \langle \text{fps} \triangleleft_1 \text{aps}, (\Delta, \Delta_c, \mu) \rangle \rightarrow_{\text{TR1}} (\Delta_{c_1}, \mu_1) \\ \text{[S16]} \frac{\langle \text{FR} \triangleleft_1 \text{AR}, (\Delta, \Delta_{c_1}, \mu_1) \rangle \rightarrow_{\text{TRN}} (\sigma_r, \text{epi}_r)}{\langle \text{sfps} \triangleleft_1 \text{saps}, (\Delta, \Delta_c, \mu) \rangle \rightarrow_{\text{TRN}} (\sigma_r, \text{epi}_r)} \\ \\ \text{sfps} = [\text{emptySFP}] \quad \text{saps} = [\text{emptyAFP}] \\ \text{[S17.1]} \frac{\sigma_r = (\Delta_c, \mu) \quad \text{epi}_r = []}{\langle \text{sfps} \triangleleft_1 \text{saps}, (\Delta, \Delta_c, \mu) \rangle \rightarrow_{\text{TRN}} (\sigma_r, \text{epi}_r)} \end{array}$$

Osservazione

Quando invocata a seguito di una Call o di un Apply, la TrNFun si occupa anche di inserire i **legami** nel nuovo activation record generato per la procedura o funzione invocata

Invocazione di procedure e funzioni con multipli parametri: Sistema Y per Call ed Apply

Per quanto riguarda il sistema Y per **Call** ed **Apply**, grazie all'introduzione della TrNFun, è possibile limitare il controllo di queste regole alla sola verifica dell'identificatore già definito:

$$[Y25] \frac{Y_\rho |_0(l) = [\text{abs}] [\text{void}] [::] t}{\langle [\text{call}] \mid \text{sapars}, Y_\rho \rangle \rightarrow_Y \langle [\text{void}], Y_\rho \rangle}$$

$$[Y26] \frac{Y_\rho |_0(l) = [\text{abs}] t [::] t' \quad t \in \text{Simple}}{\langle [\text{apply}] \mid \text{sapars}, Y_\rho \rangle \rightarrow_Y \langle [\text{void}], Y_\rho \rangle}$$

Gestione errori di tipo:

$$[E45] \frac{Y_\rho |_0(l) \neq [\text{abs}] [\text{void}] [::] t}{\langle [\text{call}] \mid \text{sapars}, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

$$[E46] \frac{Y_\rho |_0(l) = \perp}{\langle [\text{call}] \mid \text{sapars}, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

$$[E32] \frac{Y_\rho |_0(l) \neq [\text{abs}] t [::] t'}{\langle [\text{apply}] \mid \text{sapars}, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

$$[E33] \frac{Y_\rho |_0(l) = \perp}{\langle [\text{apply}] \mid \text{sapars}, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

$$[E34.1] \frac{Y_\rho |_0(l) = [\text{abs}] t [::] t' \quad t \notin \text{Simple}}{\langle [\text{apply}] \mid \text{sapars}, Y_\rho \rangle \rightarrow_Y \langle [\text{terr}], Y_\rho \rangle}$$

- Regola di inferenza finale per lo statement **Call**:

$$\begin{array}{l}
 \sigma = (\Delta, \mu) \quad \Delta(l = (t_r, v_r)) \\
 v_r = \star l, t, \text{sfps}, \text{dcl}, \text{sts}, k \star \\
 t = [\text{abs}] t' [::] t'' \quad t' = [\text{void}] \\
 \text{ar} = \{l, \# \Delta - k + 1, [], [], []\} \quad \Delta_c = \text{ar} + \Delta \\
 \langle \text{saps} \triangleleft_1 \text{faps}, (\Delta, \Delta_c, \mu) \rangle \rightarrow_{\text{TRN}} (\sigma_r, \text{epi}_r) \\
 \langle \text{dcl}, \sigma_r \rangle \rightarrow ([\text{void}], (\text{ar}_2 + \Delta_2, \mu_2)) \\
 \text{ar}_2 = \{l, \text{lc}, \text{fr}, \text{cnt}, v\} \\
 \{l, \text{lc}, \text{fr}, \text{sts}, v\} = \text{ar}_3 \\
 \langle \text{ar}_3 + \Delta_2, \mu_2 \rangle \rightarrow_{R^*} (\text{ar}_4 + \Delta_4, \mu_4) \\
 \text{[S14]} \frac{}{\langle [\text{call}] l \text{ saps}, \sigma \rangle \rightarrow ([\text{void}], (\Delta_4, \mu_4))}
 \end{array}$$

- Regola di inferenza finale per l'espressione **Apply**:

$$\begin{array}{l}
 \sigma = (\Delta, \mu) \quad \Delta(l = (t_r, c_r)) \\
 c_r = *l, t, sfps, dcl, sts, k* \\
 t = [abs] t'[::] t'' \quad t' \in \text{Simple} \\
 ar = \{l, \# \Delta - k + 1, [], [], []\} \quad \Delta_c = ar + \Delta \\
 \langle saps \triangleleft_1 faps, (\Delta, \Delta_c, \mu) \rangle \rightarrow_{\text{TRN}} (\sigma_r, epi_r) \\
 \langle dcl, \sigma_r \rangle \rightarrow ([void], (ar_2 + \Delta_2, \mu_2)) \\
 ar_2 = \{l, lc, fr, cnt, v\} \\
 \{l, lc, fr, sts, v\} = ar_3 \\
 (ar_3 + \Delta_2, \mu_2) \rightarrow_{R*} (ar_4 + \Delta_4, \mu_4) \\
 ar_4 = \{l, l, f, c, v_r\} \\
 \text{[S15]} \frac{}{\langle [apply] l saps, \sigma \rangle \rightarrow (t', v_r, (\Delta_4, \mu_4))}
 \end{array}$$

Invocazione di procedure e funzioni con multipli parametri: modifiche nell'interprete

- Principali modifiche apportate all'interprete di Small21 in accordo alle regole definite (per semplicità sono mostrate solamente quelle per Apply, in quanto quelle per Call sono analoghe):

- Introduzione di `seqapars` e modifica all'espressione `apply`

```
87 exp =
88     Val of ide
89     | Arrow1 of ide * exp
90     | N of num
91     | Plus of exp * exp
92     | Minus of exp * exp
93     | LT of exp * exp
94     | GT of exp * exp
95     | Eq of exp * exp
96     | B of bool
97     | Or of exp * exp
98     | Apply of ide * seqapars (*MODIFICATA_1*)
99     | EE
100 and
101
102 seqapars = (*AGGIUNTA_1*)
103     ESAP
104     | SAP of apars * seqapars
105 and
---
```

Invocazione di procedure e funzioni con multipli parametri: modifiche nell'interprete

- Principali modifiche apportate all'interprete di Small21 in accordo alle regole definite (per semplicità sono mostrate solamente quelle per Apply, in quanto quelle per Call sono analoghe):

```
| Apply(ide,saps)
-> match gets sk ide with
  (* include check for E33 *)
  | DAbs(Abs(tr,aa),ClosT(_,_ ,sfps,dcl,sts,k)) (* no overloading *)
    when (isSimple tr)
      ->(let ar = mkAR5 (Name ide) ((sizeS sk)-k+1)
          (emptyEnv()) [] None in
          let skc = push sk ar in
          let (sgR,epiR) = trNFun sfps saps sk skc mu in
            (* include check for E34 *)
            let (_,(ar2sk2,mu2)) = dclSem dcl sgR in
            let ar3sk2 = resetC ar2sk2 [UnL sts] in
            let (_,(sk4,mu4)) = nextCmd(ar3sk2,mu2) in
            let sgF = (pop sk4,mu4) in
            (tr,getR sk4,sgF))
  | DAbs(Abs(tr,aa),ClosT(_,_ ,sfps,dcl,sts,k))
      -> raise(TypeErrorE("E34.1: expSem",Apply(ide,saps)))
  | _ -> raise(TypeErrorE("E32: expSem",Apply(ide,saps)))
```

- Modifica della funzione **expSem** che fa utilizzo della funzione ausiliaria **TrNFun** ↓

Invocazione di procedure e funzioni con multipli parametri: modifiche nell'interprete

- Principali modifiche apportate all'interprete di Small21 in accordo alle regole definite (per semplicità sono mostrate solamente quelle per Apply, in quanto quelle per Call sono analoghe):

- Introduzione della nuova funzione **TrNFun**:

```
1246 trNFun sfps saps sk skc mu =
1247     match (sfps,saps) wltH
1248     | (SFP(fp,sfpsR),SAP(ap,sapsR))
1249       -> let (sk1,mu1) = tr1Fun fp ap sk skc mu ln
1250           trNFun sfpsR sapsR sk sk1 mu1
1251     | (ESFP,ESAP)
1252       -> let sgr = (skc,mu) and epr = [] ln
1253           (sgr,epr)
1254     | (ESFP,_)
1255       -> raise(TypeError("E62: Trasmisssion: Too Many Parans"))
1256     | (_,ESAP)
1257       -> raise(TypeError("E63: Trasmisssion: Too Few Parans"))
1258     and
```

Dichiarazione ed Invocazione di procedure e funzioni con multipli parametri: verifica

Tornando sull'esempio del Bubble sort descritto precedentemente:

```
1
2 Prog OrdinaArray()
3 {
4     void swap(ref int a, ref int b)
5     {
6         int tmp = a;
7         a = b;
8         b = tmp;
9     }
10    void bubblesort(int i, int N, bool controllo)
11    {
12        if (i < N)
13        {
14            if (A[i] > A[i+1])
15            {
16                swap(A[i], A[i+1]);
17                controllo = true;
18            }
19            bubblesort(i+1, N, controllo);
20        }
21        if (controllo == true)
22            bubblesort(0, N-1, false);
23        return;
24    }
25    int A[4];
26    A[0] = 8;
27    A[1] = 5;
28    A[2] = 6;
29    A[3] = 1;
30    bubblesort(0, 3, false);
31 }
```

Dichiarazione ed Invocazione di procedure e funzioni con multipli parametri: verifica

È possibile ora esprimere tale programma in Small21 come:

```
let p = Prog ("OrdinaArray",
  Block
    (SeqD
      (Pcd (Void, "swap", SFP (FP (Ref, Int, "a"), SFP (FP (Ref, Int, "b"), ESFP)),
        BlockP
          (Var (Int,"tmp",Val "a"),
            SeqS (Upd (Val "a", Val "b"), Upd (Val "b", Val "tmp")))),
        SeqD
          (Pcd (Void, "bubblesort", SFP (FP (Value, Int, "l"), SFP (FP (Value, Int, "N"), SFP (FP (Value, Bool, "controllo"), ESFP))),
            BlockP
              (ED,
                SeqS
                  (IFT (LT (Val "l", Val "N"),
                    SeqS
                      (IFT (GT (Arrow1 ("Ak", Val "l"), Arrow1 ("Ak", Plus (Val "l", N 1))),
                        SeqS (Call ("swap", SAP (AP (Arrow1 ("Ak", Val "l")), SAP (AP (Arrow1 ("Ak", Plus (Val "l", N 1))), ESAP))),
                          Upd (Val "controllo", B True))),
                      SeqS
                        (Call ("bubblesort", SAP (AP (Plus (Val "l", N 1)), SAP (AP (Val "N"), SAP (AP (Val "controllo"), ESAP))),
                          Return EE))),
                        IFT (Eq (Val "controllo", B True),
                          Call ("bubblesort", SAP (AP (N 0), SAP (AP (Minus (Val "N", N 1)), SAP (AP (B False), ESAP))))))))),
                SeqD
                  (Array (Arr (Int, 4), "Ak"), ED))),
            SeqC
              (UnL (Upd ((Arrow1 ("Ak", N 0), N 0))),
                SeqC
                  (UnL (Upd ((Arrow1 ("Ak", N 1), N 1))),
                    SeqC
                      (UnL (Upd ((Arrow1 ("Ak", N 2), N 2))),
                        SeqC
                          (UnL (Upd ((Arrow1 ("Ak", N 3), N 1))),
                            UnL (Call ("bubblesort", SAP (AP (N 0), SAP (AP (N 3), SAP (AP (B False), ESAP))))))))))))))));
```

Osservazioni

Si è fatto più volte utilizzo in tale esempio della dichiarazione di funzioni con multipli parametri (sia per valore che per reference) e dell'invocazione di tali funzioni.

Dichiarazione ed Invocazione di procedure e funzioni con multipli parametri: verifica

Lanciando **progSem** su tale programma costruito, si ottiene il seguente risultato:

- Prima inizializzazione dello stack e dello Store inizialmente vuoto

```
# progSem p;;  
  
Stack:  
>{OrdinaArray,0,[Ak/(:Mint[4],L0);  
  bubblesort/([void::int,int,bool],$bubblesort,[void::int,int,bool],:fpar::,cmd:,1$);  
  swap/([void::int,int],$swap,[void::int,int],:fpar::,cmd:,1$)],:cmdNext:[N]}  
]  
Store:  
[L0<-Undef,L1<-Undef,L2<-Undef,L3<-Undef]
```

Dichiarazione ed Invocazione di procedure e funzioni con multipli parametri: verifica

Lanciando **progSem** su tale programma costruito, si ottiene il seguente risultato:

```
Stack:
>{OrdlnaArray,0,[Ak/(:Mint[4],L0);
bubblesort/([void::int,int,bool],$bubblesort,[void::int,int,bool],:fpar:::cmd:,1$);
swap/([void::int,int],$swap,[void::int,int],:fpar:::cmd:,1$)],:cmdNext:[,N]}
]
Store:
[L0<-8,L1<-5,L2<-6,L3<-1]

Stack:
>{swap,2,[tmp/(Mint,L7);
b/(Mint,L1);
a/(Mint,L0)],:cmdNext:[,N]}
{bubblesort,1,[controllo/(Mbool,L6);
N/(Mint,L5);
i/(Mint,L4)],:cmdNext:[,N]}
{OrdlnaArray,0,[Ak/(:Mint[4],L0);
bubblesort/([void::int,int,bool],$bubblesort,[void::int,int,bool],:fpar:::cmd:,1$);
swap/([void::int,int],$swap,[void::int,int],:fpar:::cmd:,1$)],:cmdNext:[,N]}
]
Store:
[L0<-5,L1<-5,L2<-6,L3<-1,L4<-0,L5<-3,L6<-false,L7<-8]

Stack:
>{swap,2,[tmp/(Mint,L7);
b/(Mint,L1);
a/(Mint,L0)],:cmdNext:[,N]}
{bubblesort,1,[controllo/(Mbool,L6);
N/(Mint,L5);
i/(Mint,L4)],:cmdNext:[,N]}
{OrdlnaArray,0,[Ak/(:Mint[4],L0);
bubblesort/([void::int,int,bool],$bubblesort,[void::int,int,bool],:fpar:::cmd:,1$);
swap/([void::int,int],$swap,[void::int,int],:fpar:::cmd:,1$)],:cmdNext:[,N]}
]
Store:
[L0<-5,L1<-8,L2<-6,L3<-1,L4<-0,L5<-3,L6<-false,L7<-8]
```

- Prima invocazione di Bubble sort e primo swap

Dichiarazione ed Invocazione di procedure e funzioni con multipli parametri: verifica

Lanciando **progSem** su tale programma costruito, si ottiene il seguente risultato:

- Stack e store finali: **le prime 4 locazioni di memoria rappresentanti l'array sono effettivamente ordinate!**

```
Stack:
~(OrdinaArray_0,[Ak/(:Mint[4],L0);
  bubblesort/([void::int,int,bool],$bubblesort,[void::int,int,bool],:fpar::,cmd:,1$);
  swap/([void::int,int],$swap,[void::int,int],:fpar::,cmd:,1$),:cmdNext:,[N])
]
Store:
[L0<-1,L1<-5,L2<-6,L3<-8,L4<-0,L5<-3,L6<-true,L7<-8,L8<-1,L9<-3,L10<-true,L11<-8,L12<-2,L13<-3,L14<-true,L15<-8,L16<-3,L17<-3,L18<-true,L19<-0,
L20<-2,L21<-false,L22<-1,L23<-2,L24<-true,L25<-6,L26<-2,L27<-2,L28<-true,L29<-0,L30<-1,L31<-true,L32<-5,L33<-1,L34<-1,L35<-true,L36<-0,L37<-0,L
38<-false]

SUCCESSFUL_TERMINATION
- : unit/2 = ()
# []
```

Grazie per l'attenzione!