UNIVERSITÀ DI PISA

Linguaggi di Programmazione con Laboratorio

Seminario di fine corso:

Tipi e valori Record in Small21

Francesco Baldino

Università di Pisa, Dipartimento di Matematica

29 Ottobre 2021

Introduzione

Definizione del progetto

- Estensione a valori Record, ovvero valori strutturati costanti.
- Non esprimibili o calcolabili, ma utilizzabili per esprimere o accedere alle singole componenti

Introduzione

Definizione del progetto

- Estensione a valori Record, ovvero valori strutturati costanti.
- Non esprimibili o calcolabili, ma utilizzabili per esprimere o accedere alle singole componenti

Definizione in sintassi concreta

$$\texttt{struct \{ } t_1 \texttt{ : } sel_1 \texttt{ ... } t_n \texttt{ : } sel_n \texttt{ \} ide;}$$

dove (t_i : sel_i) è la componente i-esima di tipo t_i e di selettore di accesso sel_i.

Vincoli

Nello sviluppo del progetto sono stati assunti i seguenti vincoli:

Vincoli

Nello sviluppo del progetto sono stati assunti i seguenti vincoli:

- I valori Record sono statici, non esprimibili e non assegnabili

Vincoli

Nello sviluppo del progetto sono stati assunti i seguenti vincoli:

- I valori Record sono statici, non esprimibili e non assegnabili
  - Non sono valori calcolabili

Vincoli

Nello sviluppo del progetto sono stati assunti i seguenti vincoli:

- I valori Record sono statici, non esprimibili e non assegnabili
  - Non sono valori calcolabili
- I tipi delle componenti possono essere solo Simple

Vincoli

Nello sviluppo del progetto sono stati assunti i seguenti vincoli:

- I valori Record sono statici, non esprimibili e non assegnabili
  - Non sono valori calcolabili
- I tipi delle componenti possono essere solo Simple
- I valori Record non possono essere passati come parametro a funzione

Confronto con altri valori strutturati: gli Array

- Entrambi hanno componenti allocate in location successive della memoria

Confronto con altri valori strutturati: gli Array

- Entrambi hanno componenti allocate in location successive della memoria
- Entrambi hanno accesso in tempo costante

Confronto con altri valori strutturati: gli Array

- Entrambi hanno componenti allocate in location successive della memoria
- Entrambi hanno accesso in tempo costante
- Gli Array hanno componenti iterabili, i Record no

Confronto con altri valori strutturati: gli Array

- Entrambi hanno componenti allocate in location successive della memoria
- Entrambi hanno accesso in tempo costante
- Gli Array hanno componenti iterabili, i Record no
- I Record possono avere componenti di tipi differenti, gli Array no

Espressività nell'estensione attuale

Questa estensione porta a un linguaggio sicuramente più comodo (ex: dichiarazione multipla di variablili) ma non veramente più espressivo

- La dichiarazione

  ```
  struct { int:x, int:y, int:z } pos;
  ```

  è più concisa di

  ```
  int posX;
  int posY;
  int posZ;
  ```

Espressività nell'estensione attuale

Questa estensione porta a un linguaggio sicuramente più comodo (ex: dichiarazione multipla di variablili) ma non veramente più espressivo

- La dichiarazione

  ```
  struct { int:x, int:y, int:z } pos;
  ```

  è più concisa di

  ```
  int posX;
  int posY;
  int posZ;
  ```

- L'impossibilità di esprimere un valore Record rende di fatto le due dichiarazioni precedenti equivalenti

Espressività nell'estensione attuale

Questa estensione porta a un linguaggio sicuramente più comodo (ex: dichiarazione multipla di variablili) ma non veramente più espressivo

- La dichiarazione

  ```
  struct { int:x, int:y, int:z } pos;
  ```

  è più concisa di

  ```
  int posX;
  int posY;
  int posZ;
  ```

- L'impossibilità di esprimere un valore Record rende di fatto le due dichiarazioni precedenti equivalenti (ancora di più se in Small21 non esteso ammettiamo il carattere "." come ammissibile negli identificatori)

Espressività in ulteriori estensioni

Estensione possibile: aggiungere i Record ai valori trasmissibili come parametro a funzione

- Visto il vincolo di al più un parametro per funzione, l'espressività aumenterebbe molto

Espressività in ulteriori estensioni

Estensione possibile: aggiungere i Record ai valori trasmissibili come parametro a funzione

- Visto il vincolo di al più un parametro per funzione, l'espressività aumenterebbe molto
- Visto il vincolo sulla calcolabilità dei Record, sarebbe equivalente all'estensione a trasmissione di parametri multipli

Espressività in ulteriori estensioni

Estensione possibile: aggiungere i Record ai valori calcolabili

- Grande aumento di espressività: equivalente al poter ritornare più valori (atomici)

Espressività in ulteriori estensioni

Estensione possibile: aggiungere i Record ai valori calcolabili

- Grande aumento di espressività: equivalente al poter ritornare più valori (atomici)
- In realtà potendo trasmettere valori per reference, è più debole dell'estensione a trasmissione di parametri multipli

Espressività in ulteriori estensioni

Estensione possibile: aggiungere i Record ai valori calcolabili

- Grande aumento di espressività: equivalente al poter ritornare più valori (atomici)
- In realtà potendo trasmettere valori per reference, è più debole dell'estensione a trasmissione di parametri multipli

- Unendo queste due estensioni proposte, indipendentemente da altre possibili sulla trasmissione di valori, si otterrebbe un linguaggio più espressivo e comodo

Espressività in ulteriori estensioni

Estensione possibile: rimozione del vincolo Simple ai tipi dei componenti dei Record

Espressività in ulteriori estensioni

Estensione possibile: rimozione del vincolo Simple ai tipi dei componenti dei Record

- Record con componenti Array

Espressività in ulteriori estensioni

Estensione possibile: rimozione del vincolo Simple ai tipi dei componenti dei Record

- Record con componenti Array
- Nested Struct

Espressività in ulteriori estensioni

Estensione possibile: rimozione del vincolo Simple ai tipi dei componenti dei Record

- Record con componenti Array
- Nested Struct
- Espressività comunque subordinata ad altre estensioni

Modifiche alla Sintassi Concreta

La sintassi concreta del linguaggio esteso riporta le seguenti aggiunte:

Modifiche alla Sintassi Concreta

La sintassi concreta del linguaggio esteso riporta le seguenti aggiunte:

- `Type` $\rightarrow$ `...` | `struct { Rc Rcs }` | ...

Modifiche alla Sintassi Concreta

La sintassi concreta del linguaggio esteso riporta le seguenti aggiunte:

- `Type` $\rightarrow$ `...` | `struct { Rc Rcs }` | ...
- `Rcs` $\rightarrow$ `Rc Rcs` | ϵ
- `Rc` $\rightarrow$ `Simple : ide`

Modifiche alla Sintassi Concreta

La sintassi concreta del linguaggio esteso riporta le seguenti aggiunte:

- `Type` $\rightarrow$ `...` | `struct { Rc Rcs }` | ...
- `Rcs` $\rightarrow$ `Rc Rcs` | ϵ
- `Rc` $\rightarrow$ `Simple : ide`

- `DExp` $\rightarrow$ `...` | `ide.ide`

Modifiche alla Sintassi Concreta

La sintassi concreta del linguaggio esteso riporta le seguenti aggiunte:

- `Type` $\rightarrow$ `...` | `struct { Rc Rcs }` | ...
- `Rcs` $\rightarrow$ `Rc Rcs` | ϵ
- `Rc` $\rightarrow$ `Simple : ide`

- `DExp` $\rightarrow$ `...` | `ide.ide`

  (già esiste `Exp` $\rightarrow$ `...` | `DExp` | `...`)

Modifiche alla Sintassi Astratta

```
Type ::= ...  | [rcrd] RecCompSeq | ...
RecCompSeq ::= RecComp [::]  RecCompSeq | RecComp
RecComp ::= Type Ide
```

```
43
44  type tye =
        ...
51      | Rcrd of (tye * ide) list
        ...
55      and
56
```

Modifiche alla Sintassi Astratta

```
Type ::= ...  | [rcrd] RecCompSeq | ...
RecCompSeq ::= RecComp [::]  RecCompSeq | RecComp
RecComp ::= Type Ide
```

```
Dcl ::= ...  | [record] Type Ide | ...
```

```
43
44  type tye =
        ...
51      | Rcrd of (tye * ide) list
        ...
55      and
56
60
61  dcl =
        ...
65      | Record of tye * ide
        ...
69      and
70
```

Modifiche alla Sintassi Astratta

```
Type ::= ...  | [rcrd] RecCompSeq | ...
RecCompSeq ::= RecComp [::]  RecCompSeq | RecComp
RecComp ::= Type Ide
```

```
Dcl ::= ...  | [record] Type Ide | ...
```

```
DExp ::= ...  | Ide [sel] Ide
```
(già esiste `Exp ::= ...  | DExp | ...`)

```
43
44  type tye =
        ...
51      | Rcrd of (tye * ide) list
        ...
55      and
56
60
61  dcl =
        ...
65      | Record of tye * ide
        ...
69      and
70
83
84  exp =
86      ...
87      | Sel of ide * ide
97      ...
98      and
99
```

Modifiche al Sistema di tipi Y

L'estensione aggiunge le seguenti regole con relative gestione di errori:

Modifiche al Sistema di tipi Y

L'estensione aggiunge le seguenti regole con relative gestione di errori:

$$\frac{\begin{array}{c}\texttt{t} = \texttt{[rcrd]}\ ((\texttt{t}_{\mathbf{0}}, \texttt{i}_{\mathbf{0}}), \ldots, (\texttt{t}_n, \texttt{i}_n)) \\ \texttt{t}_j \in \texttt{Simple}\ \forall j \\ \texttt{i}_h \neq \texttt{i}_j\ \forall h \neq j \\ \texttt{Y}_\rho\,|_{\mathbf{0}}(\texttt{I}) = \bot\end{array}}{\texttt{[Ye1]}\quad \texttt{<[record] t I, Y}_\rho\texttt{>} \rightarrow_Y (\texttt{[void], [I/[rcrd] (([mut] t}_{\mathbf{0}}\texttt{, i}_{\mathbf{0}}\texttt{), ..., ([mut] t}_n\texttt{, i}_n\texttt{))])} \otimes \texttt{Y}_\rho)}$$

Modifiche al Sistema di tipi Y

L'estensione aggiunge le seguenti regole con relative gestione di errori:

$$\frac{\begin{array}{c}\texttt{t} = \texttt{[rcrd]}\ ((\texttt{t}_{\mathbf{0}}, \texttt{i}_{\mathbf{0}}), \ldots, (\texttt{t}_n, \texttt{i}_n)) \\ \texttt{t}_j \in \texttt{Simple}\ \forall j \\ \texttt{i}_h \neq \texttt{i}_j\ \forall h \neq j \\ \texttt{Y}_\rho\,|_{\mathbf{0}}(\texttt{I}) = \bot\end{array}}{\texttt{<[record] t I, Y}_\rho\texttt{>} \rightarrow_Y \texttt{([void], [I/[rcrd] (([mut] t}_{\mathbf{0}}\texttt{, i}_{\mathbf{0}}\texttt{), ..., ([mut] t}_n\texttt{, i}_n\texttt{))]) } \otimes \texttt{ Y}_\rho} \quad \text{[Ye1]}$$

$$\frac{\begin{array}{c}\texttt{t} = \texttt{[rcrd]}\ ((\texttt{t}_{\mathbf{0}}, \texttt{i}_{\mathbf{0}}), \ldots, (\texttt{t}_n, \texttt{i}_n)) \\ \exists j\ \texttt{t}_j \notin \texttt{Simple}\end{array}}{\texttt{<[record] t I, Y}_\rho\texttt{>} \rightarrow_Y \texttt{([terr], Y}_\rho\texttt{)}} \quad \text{[Ee1]}$$

Modifiche al Sistema di tipi Y

L'estensione aggiunge le seguenti regole con relative gestione di errori:

$$\frac{\begin{array}{c}\texttt{t} = \texttt{[rcrd]}\ ((\texttt{t}_{\mathbf{0}}, \texttt{i}_{\mathbf{0}}), \ldots, (\texttt{t}_n, \texttt{i}_n)) \\ \texttt{t}_j \in \texttt{Simple}\ \forall j \\ \texttt{i}_h \neq \texttt{i}_j\ \forall h \neq j \\ \texttt{Y}_\rho\,|_{\mathbf{0}}(\texttt{I}) = \bot\end{array}}{\texttt{<[record] t I, Y}_\rho\texttt{>} \rightarrow_Y \texttt{([void], [I/[rcrd] (([mut] t}_{\mathbf{0}}\texttt{, i}_{\mathbf{0}}\texttt{), ..., ([mut] t}_n\texttt{, i}_n\texttt{))]) } \otimes \texttt{ Y}_\rho\texttt{)}}\ [\text{Ye1}]$$

$$\frac{\begin{array}{c}\texttt{t} = \texttt{[rcrd]}\ ((\texttt{t}_{\mathbf{0}}, \texttt{i}_{\mathbf{0}}), \ldots, (\texttt{t}_n, \texttt{i}_n)) \\ \exists j\ \texttt{t}_j \notin \texttt{Simple}\end{array}}{\texttt{<[record] t I, Y}_\rho\texttt{>} \rightarrow_Y \texttt{([terr], Y}_\rho\texttt{)}}\ [\text{Ee1}]$$

$$\frac{\begin{array}{c}\texttt{t} = \texttt{[rcrd]}\ ((\texttt{t}_{\mathbf{0}}, \texttt{i}_{\mathbf{0}}), \ldots, (\texttt{t}_n, \texttt{i}_n)) \\ \texttt{t}_j \in \texttt{Simple}\ \forall j \\ \exists h \neq j\ \texttt{i}_h = \texttt{i}_j\end{array}}{\texttt{<[record] t I, Y}_\rho\texttt{>} \rightarrow_Y \texttt{([terr], Y}_\rho\texttt{)}}\ [\text{Ee2}]$$

Modifiche al Sistema di tipi Y

L'estensione aggiunge le seguenti regole con relative gestione di errori:

$$\frac{\texttt{t} = \texttt{[rcrd]}\ ((\texttt{t}_\mathbf{0}, \texttt{i}_\mathbf{0}), \ldots, (\texttt{t}_n, \texttt{i}_n)) \quad \texttt{t}_j \in \texttt{Simple}\ \forall j \quad \texttt{i}_h \neq \texttt{i}_j\ \forall h \neq j \quad \texttt{Y}_\rho\,|_\mathbf{0}(\texttt{I}) = \bot}{\texttt{<[record] t I, Y}_\rho\texttt{>} \rightarrow_Y \texttt{([void], [I/[rcrd] (([mut] t}_\mathbf{0}\texttt{, i}_\mathbf{0}\texttt{), ..., ([mut] t}_n\texttt{, i}_n\texttt{))]) } \otimes \texttt{ Y}_\rho\texttt{)}}\ [\texttt{Ye1}]$$

$$\frac{\texttt{t} = \texttt{[rcrd]}\ ((\texttt{t}_\mathbf{0}, \texttt{i}_\mathbf{0}), \ldots, (\texttt{t}_n, \texttt{i}_n)) \quad \exists j\ \texttt{t}_j \notin \texttt{Simple}}{\texttt{<[record] t I, Y}_\rho\texttt{>} \rightarrow_Y \texttt{([terr], Y}_\rho\texttt{)}}\ [\texttt{Ee1}]$$

$$\frac{\texttt{t} = \texttt{[rcrd]}\ ((\texttt{t}_\mathbf{0}, \texttt{i}_\mathbf{0}), \ldots, (\texttt{t}_n, \texttt{i}_n)) \quad \texttt{t}_j \in \texttt{Simple}\ \forall j \quad \exists h \neq j\ \texttt{i}_h = \texttt{i}_j}{\texttt{<[record] t I, Y}_\rho\texttt{>} \rightarrow_Y \texttt{([terr], Y}_\rho\texttt{)}}\ [\texttt{Ee2}]$$

$$\frac{\texttt{Y}_\rho\,|_\mathbf{0}(\texttt{I}) \neq \bot}{\texttt{<[record] t I, Y}_\rho\texttt{>} \rightarrow_Y \texttt{([terr], Y}_\rho\texttt{)}}\ [\texttt{Ee3}]$$

Modifiche al Sistema di tipi Y

$$\text{[Ye2]}\quad \frac{\begin{array}{c}\texttt{Y}_\rho\texttt{(I)} = \texttt{[rcrd] ((t}_\mathbf{0}\texttt{, i}_\mathbf{0}\texttt{), ..., (t}_n\texttt{, i}_n\texttt{))} \\ \texttt{J = i}_j \\ \texttt{t}_j = \texttt{[mut] t}\end{array}}{\texttt{<I [Sel] J, Y}_\rho\texttt{>} \rightarrow_Y \texttt{(t, Y}_\rho\texttt{)}}$$

Modifiche al Sistema di tipi Y

$$\frac{\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[rcrd]}\ ((\mathtt{t_0}, \mathtt{i_0}), \ldots, (\mathtt{t}_n, \mathtt{i}_n)) \quad \mathtt{J} = \mathtt{i}_j \quad \mathtt{t}_j = \mathtt{[mut]}\ \mathtt{t}}{\mathtt{<I\ [Sel]\ J,\ Y}_\rho\mathtt{>} \rightarrow_Y \mathtt{(t,\ Y}_\rho\mathtt{)}} \text{ [Ye2]}$$

$$\frac{\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[rcrd]}\ ((\mathtt{t_0}, \mathtt{i_0}), \ldots, (\mathtt{t}_n, \mathtt{i}_n)) \quad \nexists j\ \mathtt{J} = \mathtt{i}_j}{\mathtt{<I\ [Sel]\ J,\ Y}_\rho\mathtt{>} \rightarrow_Y \mathtt{([terr],\ Y}_\rho\mathtt{)}} \text{ [Ee4]}$$

Modifiche al Sistema di tipi Y

$$\text{[Ye2]}\quad \frac{\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[rcrd]}\ ((\mathtt{t_0}, \mathtt{i_0}), \ldots, (\mathtt{t}_n, \mathtt{i}_n)) \quad \mathtt{J} = \mathtt{i}_j \quad \mathtt{t}_j = \mathtt{[mut]}\ \mathtt{t}}{\mathtt{<I\ [Sel]\ J,\ Y}_\rho\mathtt{>} \rightarrow_\mathcal{Y} (\mathtt{t},\ \mathtt{Y}_\rho)}$$

$$\text{[Ee4]}\quad \frac{\mathtt{Y}_\rho(\mathtt{I}) = \mathtt{[rcrd]}\ ((\mathtt{t_0}, \mathtt{i_0}), \ldots, (\mathtt{t}_n, \mathtt{i}_n)) \quad \nexists j\ \mathtt{J} = \mathtt{i}_j}{\mathtt{<I\ [Sel]\ J,\ Y}_\rho\mathtt{>} \rightarrow_\mathcal{Y} (\mathtt{[terr]},\ \mathtt{Y}_\rho)}$$

$$\text{[Ee5]}\quad \frac{\mathtt{Y}_\rho(\mathtt{I}) \neq \mathtt{[rcrd]}\ ((\mathtt{t_0}, \mathtt{i_0}), \ldots, (\mathtt{t}_n, \mathtt{i}_n))}{\mathtt{<I\ [Sel]\ J,\ Y}_\rho\mathtt{>} \rightarrow_\mathcal{Y} (\mathtt{[terr]},\ \mathtt{Y}_\rho)}$$

Modifiche a SEM_{DCL}

$$\frac{\begin{array}{c}\texttt{t} = \texttt{[rcrd]}\ ((\texttt{t}_\mathbf{0}, \texttt{i}_\mathbf{0}), \ldots, (\texttt{t}_n, \texttt{i}_n)) \\ \texttt{t}_j \in \texttt{Simple}\ \forall j \\ \texttt{i}_h \neq \texttt{i}_j\ \forall h \neq j \\ \Delta|_\mathbf{0}(\texttt{I}) = \perp \\ \triangleright(\mu, n+1) = (\texttt{loc}_s, \mu_a) \\ \Delta_I = \texttt{[I/[DRec]}\ \texttt{([rcrd]}\ \texttt{(([mut]}\ \texttt{t}_\mathbf{0}, \texttt{i}_\mathbf{0}), \ldots, \texttt{([mut]}\ \texttt{t}_n, \texttt{i}_n)), \texttt{loc}_s)\texttt{])} \otimes \Delta\end{array}}{\texttt{<[record]}\ \texttt{t}\ \texttt{I}, (\Delta, \mu)\texttt{>} \rightarrow_{DCL} \texttt{([void]},(\Delta_I, \mu_a))}\ \texttt{[De]}$$

Modifiche a SEM_{DCL}

```
841
842 let rec dclSem dcl (sk,(Store(d,g)as mu)) =
843     match dcl with
        ...
896     | Record(ty,ide) ->
897         (match ty with
898             | Rcrd(cps)
899                 when (for_all isSimple (getCmpTys cps)) &&
900                     not(checkDup (getCmpIds cps)) &&
901                     not(declared sk ide)
902                 -> (let num = length cps in
903                     let (loca,muF) = allocate mu num in
904                     let den = DRec(Rcrd(map (fun (ty, id) -> (Mut ty, id)) cps), loca) in
905                     let skF = bindS sk ide den in
906                     (Void,(skF,muF)))
907             | Rcrd(cps)
908                     when not(for_all isSimple (getCmpTys cps))
909                     -> raise(TypeErrorI("Ee1: dclSem",ide))
910             | Rcrd(cps)
911                     when (checkDup (getCmpIds cps))
912                     -> raise(TypeErrorI("Ee2: dclSem",ide))
913             | Rcrd(_)
914                     when declared sk ide
915                     -> raise(TypeErrorI("Ee3: dclSem",ide))
        ...
942     and
943
```

Modifiche a SEM_{DEXP}

$$\frac{\Delta(\mathtt{I}) = (\mathtt{t},\ \mathtt{loc}_s) \quad \mathtt{t} = \mathtt{[rcrd]}\ ((\mathtt{[mut]}\ \mathtt{t_0},\ \mathtt{i_0}),\ \ldots,\ (\mathtt{[mut]}\ \mathtt{t}_n,\ \mathtt{i}_n)) \quad \mathtt{J} = \mathtt{i}_j \quad \mathtt{loc}_s = \mathtt{Loc\ s} \quad \mathtt{loc}_j = \mathtt{Loc(s+j)}}{\mathtt{<I\ [Sel]\ J,}\ (\Delta, \mu)\mathtt{>} \rightarrow_{DEXP} \lfloor \mathtt{[mut]}\ \mathtt{t}_j,\ \mathtt{loc}_j,\ (\Delta, \mu) \rfloor} \quad \text{[Xe1]}$$

```
1104
1105 dexpSem dexp (sk,(Store(d,g)as mu)) =
1106     match dexp with
         ...
1134     | Sel(idR,idC) ->
1135         (match getS sk idR with
1136             | DRec(Rcrd(cps), locS)
1137               when (mem idC (getCmpIds cps))
1138               -> (let Loc s = locS in
1139               let ind = (getCmpIndex idC (getCmpIds cps)) in
1140               let locr = Loc(s + ind) in
1141               let tr = getNthCmp ind (getCmpTys  cps) in
1142               (tr, locr, (sk, mu)))
1143             | DRec(Rcrd(cps), locS)
1144               when not(mem idC (getCmpIds cps))
1145               -> raise(TypeErrorE("Ee4.1: expSem: - ", dexp))
1146             | _
1147               -> raise(TypeErrorE("Ee5.1: expSem: - ", dexp)))
         ...
1151     and
1152
```

Modifiche a SEM_{EXP}

$$\text{[Xe2]}\quad \frac{\texttt{<I [Sel] J, } (\Delta, \mu)\texttt{>} \rightarrow_{DEXP} \lfloor \texttt{[mut] } t_r\texttt{, } loc_r\texttt{, } (\Delta_r, \mu_r)\rfloor \qquad v_r = \mu_r(loc_r)}{\texttt{<I [Sel] J, } (\Delta, \mu)\texttt{>} \rightarrow_{EXP} \lfloor t_r\texttt{, } v_r\texttt{, } (\Delta_r, \mu_r)\rfloor}$$

```
948
949  expSem exp (sk,(Store(d,g)as mu)) =
950      match exp with
             ...
1067         | Sel(idR,idC) ->
1068             (match getS sk idR with
1069                 | DRec(Rcrd(cps), locS)
1070                   when (mem idC (getCmpIds cps))
1071                   -> (let Loc s = locS in
1072                   let ind = (getCmpIndex idC (getCmpIds cps)) in
1073                   let loc1 = Loc(s + ind) in
1074                   let vr = mTOe(getStore mu loc1) in
1075                   (match (getNthCmp ind (getCmpTys cps)) with
1076                           | Mut tr -> (tr, vr, (sk, mu))
1077                           | _ -> raise(SystemError("Ill-Defined Record?: dclSem"))))
1078                 | DRec(Rcrd(cps), locS)
1079                   when not(mem idC (getCmpIds cps))
1080                   -> raise(TypeErrorE("Ee4: expSem: - ", exp))
1081                 | _
1082                   -> raise(TypeErrorE("Ee5: expSem: - ", exp)))
1083      and
1084
```

Esempio di programma ottenibile

```
let prog =
    Prog(
        "progExample",
        Block(
            SeqD(
                Pcd(
                    Void, "changeVal", FP (Ref, Int, "arg"),
                    BlockP(ED, Upd(Val "arg", N 5))),
            SeqD(
                Pcd(
                    Void, "dontChangeVal", FP (Value, Int, "arg"),
                    BlockP(ED, Upd(Val "arg", N 6))),
                Record (Rcrd([(Int, "a"); (Int, "b"); (Bool, "c")]), "example")
            )),

            SeqC(
                UnL (Upd (Sel ("example", "a"), N 3)),
            SeqC(
                UnL (Upd (Sel ("example", "b"), Sel ("example", "a"))),
            SeqC(
                UnL (Upd (Sel ("example", "c"), B True)),
            SeqC(
                UnL (Call ("changeVal", AP (Sel ("example", "a")))),
                UnL (Call ("dontChangeVal", AP (Sel ("example", "b"))))
            ))))
        )
    );;
```

```
(*
    Program progExample
    {
        void changeVal(ref int arg)
        {
            arg = 5;
        }
        void dontChangeVal(value int arg)
        {
            arg = 6;
        }
        struct { int:a, int:b, bool:c } example;
        example.a = 3;
        example.b = example.a;
        example.c = true;
        changeVal(example.a);
        dontChangeVal(example.b);
    }
*)
```

```
Stack:
>{progExample,0,[example/(struct { Mint:a, Mint:b, Mbool:c },L0);
 dontChangeVal/([void::int],$dontChangeVal,[void::int],:fpar:,:cmd:,1$);
 changeVal/([void::int],$changeVal,[void::int],:fpar:,:cmd:,1$)],:cmdNext:,[N]}
]

Store:
[L0<-5,L1<-3,L2<-true,L3<-6]
```

Gestione degli errori

```
(*
    Program progExample
    {
        struct { int[10]:a } example;
    }
*)

let prog =
    Prog(
        "progExample",
        Block(
            Record (
                Rcrd([(Arr(Int, 10), "a")]),
                "example"),
            UnL(ES)
        )
    );;

(*
    Exception: TypeErrorI ("Ee1: dclSem", "example").
*)
```

Gestione degli errori

```
(*
    Program progExample
    {
        struct { int[10]:a } example;
    }
*)

let prog =
    Prog(
        "progExample",
        Block(
            Record (
                Rcrd([(Arr(Int, 10), "a")]),
                "example"),
            UnL(ES)
        )
    );;

(*
    Exception: TypeErrorI ("Ee1: dclSem", "example").
*)
```

```
(*
    Program progExample
    {
        struct { int:a, bool:a } example;
    }
*)
let prog =
    Prog(
        "progExample",
        Block(
            Record (
                Rcrd([(Int, "a"); (Bool, "a")]),
                "example"),
            UnL(ES)
        )
    );;

(*
    Exception: TypeErrorI ("Ee2: dclSem", "example").
*)
```

Gestione degli errori

```
(*
    Program progExample
    {
        struct { int[10]:a } example;
    }
*)

let prog =
    Prog(
        "progExample",
        Block(
            Record (
                Rcrd([(Arr(Int, 10), "a")]),
                "example"),
            UnL(ES)
        )
    );;

(*
    Exception: TypeErrorI ("Ee1: dclSem", "example").
*)
```

```
(*
    Program progExample
    {
        struct { int:a, bool:a } example;
    }
*)
let prog =
    Prog(
        "progExample",
        Block(
            Record (
                Rcrd([(Int, "a"); (Bool, "a")]),
                "example"),
            UnL(ES)
        )
    );;

(*
    Exception: TypeErrorI ("Ee2: dclSem", "example").
*)
```

```
(*
    Program progExample
    {
        struct { int:a } example;
        struct { bool:a } example;
    }
*)
let prog =
    Prog(
        "progExample",
        Block(
            SeqD(
                Record (Rcrd([(Int, "a")]), "example"),
                Record (Rcrd([(Bool, "b")]), "example")
            ),
            UnL(ES)
        )
    );;

(*
    Exception: TypeErrorI ("Ee3: dclSem", "example").
*)
```

Gestione degli errori

```
(*
    Program progExample
    {
        struct { int[10]:a } example;
    }
*)

let prog =
    Prog(
        "progExample",
        Block(
            Record (
                Rcrd([(Arr(Int, 10), "a")]),
                "example"),
            UnL(ES)
        )
    );;

(*
    Exception: TypeErrorI ("Ee1: dclSem", "example").
*)
```

```
(*
    Program progExample
    {
        struct { int:a, bool:a } example;
    }
*)
let prog =
    Prog(
        "progExample",
        Block(
            Record (
                Rcrd([(Int, "a"); (Bool, "a")]),
                "example"),
            UnL(ES)
        )
    );;

(*
    Exception: TypeErrorI ("Ee2: dclSem", "example").
*)
```

```
(*
    Program progExample
    {
        struct { int:a } example;
        struct { bool:a } example;
    }
*)
let prog =
    Prog(
        "progExample",
        Block(
            SeqD(
                Record (Rcrd([(Int, "a")]), "example"),
                Record (Rcrd([(Bool, "b")]), "example")
            ),
            UnL(ES)
        )
    );;

(*
    Exception: TypeErrorI ("Ee3: dclSem", "example").
*)
```

```
(*
    Program progExample
    {
        struct { int:a } example;
        int x = example.b;
    }
*)
let prog =
    Prog(
        "progExample",
        Block(
            SeqD(
                Record (Rcrd([(Int, "a")]), "example"),
                Var(Int, "x", Sel("example", "b"))
            ),
            UnL(ES)
        )
    );;

(*
    Exception: TypeErrorE ("Ee4: expSem: - ", Sel ("example", "b")).
*)
```

Grazie per l'attenzione!