Esercizi e Problemi

Sommario: 26 Aprile, 2021

- Trassmissione: Deep e Shallow Binding in Static Scope
- Eccezioni: Controllo di eccezioni in Small21L2.ml
- Tipi: Un sistema di tipi sound per Small21

Esercizio: Trasmissione di funzione

Consideriamo il programma sotto, in linguaggi a blocchi con scope statico ma:

(a) Deep Binding,

(b) Shallow Binding

rispettivamente.

```
{void foo (int f(), int x){
    int fie(){
       return x;
    }
    int z;
    if (x==0) z=f();
    else foo(fie,0);
 }
 int g(){
    return 1;
 }
 foo(g,1);
}
```

Si mostri e si commenti la sequenza di stati attesa dalla computazione di esecutori dei 2 differenti tipi di Linguaggio.

Soluzione in dettaglio – Quadro 1

DEEP BINDING IN STATIC SCOPE

——	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]
g	[Lg,AR0]

Lfoo	foo-Code
Lg	g-Code

MEMORIA

```
{…
 void foo(int f(), int x){
    int fie(){
        return x;
    }
    int z;
    if (x == 0) z = f(x);
    else foo(fie,0);
 }
 int g(){
   return 1;
 }
 foo(g,1);
}
```

SHALLOW BINDING IN STATIC SCOPE

——	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]
g	g

Lfoo	foo-Code
Lg	g-Code

MEMORIA

Sopra i differenti stati durante la trasmissione dell'invocazione `foo(g,1)`.

- Deep Binding: Il valore calcolato per il parametro g è la denotazione di g in AR0.
- Shallow Binding: Il valore calcolato per il parametro g è l'identitificatore g.

Soluzione in dettaglio – Quadro 2

DEEP BINDING IN STATIC SCOPE

––	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]
g	[Lg,AR0]

––	ARfoo
CS	…
CD	…
f	[Lg,AR0]
x	Lx
fie	[Lfie,ARfoo]
z	Lz

Lfoo	foo-Code
Lg	g-Code
Lfie	fie-Code
Lx	1
Lz	

MEMORIA

```
{…
 void foo(int f(), int x){
     int fie(){
           return x;
     }
     int z;
     if (x == 0) z = f(x);
     else foo(fie,0);
 }
 int g(){
   return 1;
 }
 foo(g,1);
}
```

SHALLOW BINDING IN STATIC SCOPE

––	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]
g	g

––	ARfoo
CS	…
CD	…
f	g
x	Lx
fie	[Lfie,ARfoo]
z	Lz

Lfoo	foo-Code
Lg	g-Code
Life	fie-Code
Lx	1
Lz	

Sopra i differenti stati durante l'invocazione di `foo(g,1)`: Prima della valutazione del condizionale.

- Deep Binding: il parametro `f` ha come denotazione la chiusura di g in `AR0`.
- Shallow Binding: Il parametro `f` ha come denotazione l'identitificatore g.

Soluzione in dettaglio – Quadro 3

DEEP BINDING IN STATIC SCOPE

— —	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]
g	[Lg,AR0]

— —	ARfoo
CS	…
CD	…
f	[Lg,AR0]
x	Lx
fie	[Lfie,ARfoo]
z	Lz
x == 0 fie	false [Lfie,ARfoo]

Lfoo	foo-Code
Lg	g-Code
Lfie	fie-Code
Lx	1
Lz	

MEMORIA

```
{…
 void foo(int f(), int x){
    int fie(){
        return x;
    }
    int z;
    if (x == 0) z = f(x);
    else foo(fie,0);
 }
 int g(){
   return 1;
 }
 foo(g,1);
}
```

SHALLOW BINDING IN STATIC SCOPE

— —	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]
g	g

— —	ARfoo
CS	…
CD	…
f	g
x	Lx
fie	[Lfie,ARfoo]
z	Lz
x == 0 fie	false fie

Lfoo	foo-Code
Lg	g-Code
Lx	1
Lz	

Sopra i differenti stati durante l'invocazione di `foo(g,1)`: Valutazione del condizionale e prima dell'invocazione ricorsiva `foo(fie,0)`.

- Deep Binding: Il valore calcolato per il parametro `fie` è la denotazione di `fie` in `ARfoo`.
- Shallow Binding: Il valore calcolato per il parametro `fie` è l'identitificatore `fie`.

Soluzione in dettaglio – Quadro 4

DEEP BINDING IN STATIC SCOPE

— —	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]
g	[Lg,AR0]

— —	ARfoo
CS	…
CD	…
f	[Lg,AR0]
x	Lx
fie	[Lfie,ARfoo]
z	Lz
x == 0 fie	false [Lfie,ARfoo]

— —	ARfoo1
CS	…
CD	…
f	[Lfie,ARfoo]
x	Lx1
fie	[Lfie,ARfoo1
z	Lz1
x == 0 f(x) = x	true 0

Lfoo	foo-Code
Lg	g-Code
Lfie	fie-Code
Lx	1
Lz	
Lx1	0
Lz1	

MEMORIA

```
{…
 void foo(int f(), int x){
     int fie(){
          return x;
     }
     int z;
     if (x == 0) z = f(x);
     else foo(fie,0);
 }
 int g(){
   return 1;
 }
 foo(g,1);
}
```

SHALLOW BINDING IN STATIC SCOPE

— —	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]
g	g

— —	ARfoo
CS	…
CD	…
f	g
x	Lx
fie	[Lfie,ARfoo]
z	Lz
x == 0 fie	false fie

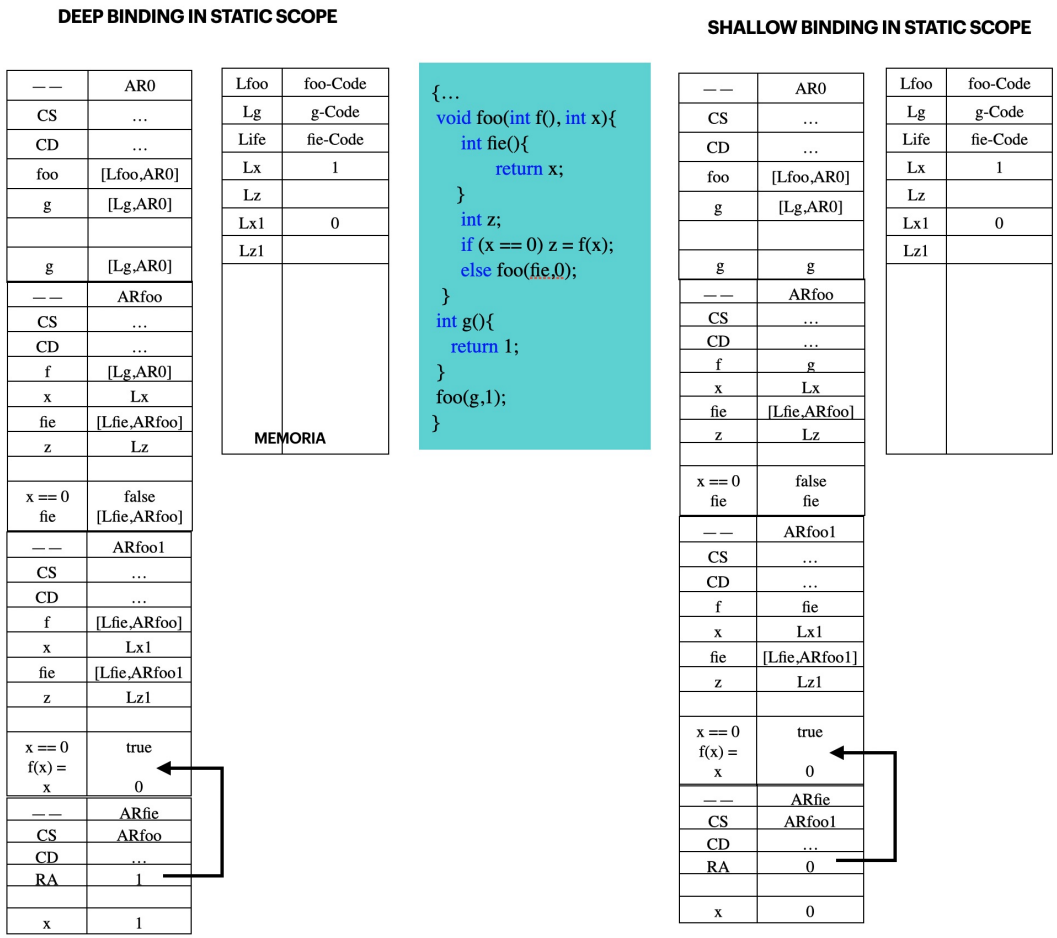
— —	ARfoo1
CS	…
CD	…
f	fie
x	Lx1
fie	[Lfie,ARfoo1]
z	Lz1
x == 0 f(x) = x	true 0

Lfoo	foo-Code
Lg	g-Code
Lfie	fie-Code
Lx	1
Lz	
Lx1	0
Lz1	

Sopra i differenti stati durante l'invocazione di `foo(g,1)`: Invocazione ricorsiva `foo(fie,0)` - Valuatazione del condizionale e trasmissione parametri dell'invocazione `f(x)`.

- Deep Binding: La funzione `f` da invocare è la denotazione di `f`, ovvero la chiusura di `fie` in `ARfoo`.
- Shallow Binding: La funzione `f` da invocare è la funzione legata all'identificatore `fie` in `ARfoo1`, e questa è la denotazione di `fie` in `ARfoo1`, ovvero la chiusura di `fie` in `ARfoo1`.

Soluzione in dettaglio – Quadro 5



Sopra i differenti stati durante l'invocazione di `foo(g,1)`: Invocazione ricorsiva `foo(fie,0)` - invocazione di `f(x)`, dove f è un parametro funzione.

- Deep Binding: Invocazione di `fie` in `ARfoo`.
- Shallow Binding: Invocazione di `fie` in `ARfoo1`.

Soluzione in dettaglio – Quadro 6

DEEP BINDING IN STATIC SCOPE

— —	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]
g	[Lg,AR0]

— —	ARfoo
CS	…
CD	…
f	[Lg,AR0]
x	Lx
fie	[Lfie,ARfoo]
z	Lz
x == 0 fie	false [Lfie,ARfoo]

— —	ARfoo1
CS	…
CD	…
f	[Lfie,ARfoo]
x	Lx1
fie	[Lfie,ARfoo1]
z	Lz1
x == 0 f(x) = x	true 1 0

Lfoo	foo-Code
Lg	g-Code
Life	fie-Code
Lx	1
Lz	
Lx1	0
Lz1	1

MEMORIA

```
{…
 void foo(int f(), int x){
    int fie(){
        return x;
    }
    int z;
    if (x == 0) z = f(x);
    else foo(fie,0);
 }
 int g(){
   return 1;
 }
 foo(g,1);
}
```

SHALLOW BINDING IN STATIC SCOPE

— —	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]
g	g

— —	ARfoo
CS	…
CD	…
f	g
x	Lx
fie	[Lfie,ARfoo]
z	Lz
x == 0 fie	false fie

— —	ARfoo1
CS	…
CD	…
f	fie
x	Lx1
fie	[Lfie,ARfoo1]
z	Lz1
x == 0 f(x) = x	true 0 0

Lfoo	foo-Code
Lg	g-Code
Life	fie-Code
Lx	1
Lz	
Lx1	0
Lz1	0

Sopra i differenti stati durante l'invocazione di `foo(g,1)`: Invocazione ricorsiva `foo(fie,0)` - assegnamento `z = f(x)`.

- Deep Binding: idem.
- Shallow Binding: idem.

Soluzione in dettaglio – Quadro 7

DEEP BINDING IN STATIC SCOPE

— —	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]
g	[Lg,AR0]
— —	ARfoo
CS	…
CD	…
f	[Lg,AR0]
x	Lx
fie	[Lfie,ARfoo]
z	Lz
x == 0 fie	false [Lfie,ARfoo]

Lfoo	foo-Code
Lg	g-Code
Life	fie-Code
Lx	1
Lz	

MEMORIA

```
{…
 void foo(int f(), int x){
     int fie(){
          return x;
     }
     int z;
     if (x == 0) z = f(x);
     else foo(fie,0);
 }
 int g(){
   return 1;
 }
 foo(g,1);
}
```

SHALLOW BINDING IN STATIC SCOPE

— —	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]
g	g
— —	ARfoo
CS	…
CD	…
f	g
x	Lx
fie	[Lfie,ARfoo]
z	Lz
x == 0 fie	false fie

Lfoo	foo-Code
Lg	g-Code
Life	fie-Code
Lx	1
Lz	

Sopra i differenti stati durante l'invocazione di `foo(g,1)`: Invocazione ricorsiva `foo(fie,0)` - ritorno.

- Deep Binding: idem.
- Shallow Binding: idem.

Soluzione in dettaglio – Quadro 8

DEEP BINDING IN STATIC SCOPE

−−	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]

Lfoo	foo-Code
Lg	g-Code

MEMORIA

```
{…
 void foo(int f(), int x){
    int fie(){
          return x;
    }
    int z;
    if (x == 0) z = f(x);
    else foo(fie,0);
 }
 int g(){
   return 1;
 }
 foo(g,1);
}
```

SHALLOW BINDING IN STATIC SCOPE

−−	AR0
CS	…
CD	…
foo	[Lfoo,AR0]
g	[Lg,AR0]

Lfoo	foo-Code
Lg	g-Code

Sopra i differenti stati durante l'invocazione di `foo(g,1)`: Ritorno dall'invocazione e Stato Finale.

- Deep Binding: Computazioni diverse.
- Shallow Binding: Computazioni diverse.
- Coppia (Stato-Iniziale,Stato-Finale) identica: Calcolano la Stessa Funzione Calcolabile?

Esercizio: Controllo di Eccezioni nel codice di Small21L2

Nel materiale distribuito oggi dalla pagina del corso, alla voce Soluzione.ml troviamo il listing con l'ultimo aggiornamento sul Laboratorio. Il listing contiene il codice per risolvere gli esercizi 7 ed 8. L'esercizio 9 mostra alcuni problemi sul dump di ambiente, in particolare il dump conduce al sollevamento di un' eccezione. Si risponda nell'ordine:

(a) Quale eccezione è sollevata e perchè?

(b) Si fornisca una modifica al codice in modo da intercettare tale eccezione e trattarla al fine di ottenere (la) una presentazione (richiesta) per l'ambiente

Esercizio: Un sistema di tipi sound per Small21

Oggi applichiamo a Small21 quanto detto nella Lezione9 sui Tipi e sui sistemi di Tipi per garantire i programmi contro la generazione di stati di "stuck". Nel materiale di oggi, l'allegato SistemaYSmall21 contiene le regole di inferenza di un sistema deduttivo per il controllo di tipi di Small21. Il sistema deve essere completato. Allo scopo:

(a) Considerare e commentare le regole date

(b) Procedere con la definizione delle regole per le seguenti strutture:

- Dichiarazione array monodimensione, modificabile solo nei componenti (oggi);
- Dichiarazione procedura/funzione con al più un parametro per valore o per reference (oggi);
- Completare le regole per le espressioni di Small21
- Completare le regole per i comandi di Small21
- Completare le regole per gli statements di Small21
- Completare le regole per il programma di Small21