Ambiente e Memoria nella AM di LP

Sommario: 11 marzo, 2015

- Naming: Uso degli identificatori nei LP
- Blocchi: in-line e Procedura
- Ambiente Locale, NonLocale, e Globale
- Scope: Statico e Dinamico
- Memoria: Statica, Stack RDA, Dinamica
- Struttura del Record Di Attivazione, RDA
- Struttura dell'Heap nella gestione Dinamica

Identficatori e Naming

- **Identificatori**
  - Nel Lambda-Calcolo sono usati per nominare i parametri delle funzioni (gli astratti)
  - Nei Linguaggi di Programmazione sono usati per nominare i **valori denotabili**

- **Valori Denotabili** sono Strutture che possono essere introdotte nel programma e riferite attraverso nomi.
  - V.D. hanno struttura e proprietà anche molto diverse tra loro, a seconda del linguaggio
  - V.D. permessi formano una prima caratteristica del linguaggio e dei programmi che possiamo scrivere con esso:
    - valori costanti;
    - valori modificabili;
    - funzioni, procedure e loro parametri
    - tipi;
    - ...

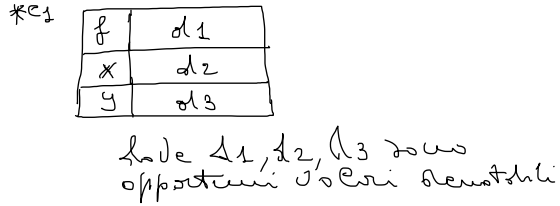
Ambienti

Definition (Ambiente)

è l'insieme delle associazioni (identificatore,valore-denotabile) che esistono a run-time, in uno specifico punto del programma e in uno specifico momento dell'esecuzione

- **Dichiarazione** L'introduzione di queste associazioni avviene attraverso costrutti di dichiarazione

```
int f(){
    return 10;
}
int main(void){
    int *x, *y; (* c1)
    x=(int *)malloc(sizeof(int));
    *x=5;
    y=x;
    *y=f();
    printf("*x=%d\n",*x);
    return 0;
}
```



- Indichiamo: (a) dichiarazioni, (b) le associazioni, (c) l'ambiente in 3 punti a scelta, nel programma C sopra.

Aliasing nell'ambiente

Definition (Aliasing)

tra identificatori che condividono uno stesso valore denotabile.

- Da evitare quando il valore denotato è un *valore modificabile*

```
int f(){
    return 10;
}
int main(void){
    int *x, *y;
    x=(int *)malloc(sizeof(int));
    *x=5;
    y=x;
    *y=f();
    printf("*x=%d\n",*x);
    return 0;
}
```

sorgente di ALIASING

- Indichiamo: (a) la presenza di aliasing, (b) aliasing su valore modificabile

Blocchi

Definition (blocco)

è una regione testuale di programma identificata da un delimitatore di inizio e uno di fine che può contenere dichiarazioni che chiameremo dichiarazioni locali del blocco

- **Blocco in-line** racchiude in modo anonimo sezioni di testo.
  - Sono attraversati come un comando (composto).
  - Possono essere composti tra loro, in sequenza o per annidamento.
- **Blocco Procedura** sono dichiarazioni di procedura/funzione.
  - Hanno un nome e una lista di parametri associato al blocco che forma il corpo della procedura/funzione.
  - Sono attraversati mediante il meccanismo di invocazione.

Blocchi e Ambienti

Definition (tipi di ambiente)

Ad ogni blocco è associato un ambiente formato da 3 parti:

- **Locale** delle dichiarazioni locali
- **Non Locale** degli identificatori visibili ma dichiarati in altri blocchi
- **Globale** degli identificatori visibili in tutti i blocchi del programma

- **Ambiente Locale** dipende unicamente dal blocco
- **Ambiente Globale** è unico per il programma, coincide con il locale di un blocco (quello del blocco più esterno del)programma.
- **Ambiente Non Locale** Il vero problema.

Ambiente Non Locale

- **Ambiente Non Locale** dipende da:
  - tipo di blocco (in-line, procedura)
  - annidamento di blocchi
  - scope statico o dinamico utilizzato dal linguaggio

```
A:{int a =1;

   B:{int b = 2;
      int c = 2;

      C:{int c =3;
         int d;
         d = a+b+c;
         write(d)
      }

      D:{int e;
         e = a+b+c;
         write(e)
      }
   }
}
```

- Solo blocchi in-line. Dipende dall'annidamento: Sono non locali di un blocco, tutti gli identificatori di blocchi che lo racchiudono e non ridefiniti in blocchi intermedi (incluso il blocco stesso)
- (a) chi sono `a`, `b`, `c` nel blocco `D`? (b) utilizzando tabelle per gli ambienti, mostrare gli ambienti locali

Regole di Scope

- **Ambiente Non Locale** Quando abbiamo anche blocchi procedura, dobbiamo considerare lo scope dell'identificatore
- **Scope di un identificatore** è l'insieme delle regioni di un programma, in cui tale identificatore (e quindi, il suo legame al valore denotato) è visibile
- I linguaggi usano uno tra due meccanismi di scope:
  - **Scope Statico** l'identificatore è visibile in ogni blocco in-line annidato e ogni blocco procedura dichiarata nel blocco, in cui non sia ridefinito.
  - **Scope Dinamico** l'identificatore è visibile in ogni blocco in-line annidato e ogni blocco procedura di invocazioni nel blocco, in cui non sia ridefinito.

```
{int x = 0;
 void fie(int n){
    x = n+1;
 }
 fie(3);
 write(x);
    {int x = 0;
     fie(3);
     write(x);
    }
 write(x);
}
```

- Assunto che il linguaggio usi scope statico: Cosa stampa il programma?
- Assunto che il linguaggio usi scope dinamico: Cosa stampa il programma?

Scope: Confronto

- **Scope Statico**
  - Usato dalla quasi totalità dei L.P.
  - Il significato degli identificatori non locali di procedure e funzioni non cambia quando invocate in punti diversi
  - Più complesso da gestire, MA
  - Esistono tecniche particolarmente efficienti che lo rendono il migliore (accesso in tempo costante ai non locali) se utilizzato da un compilatore
- **Scope Dinamico**
  - Introdotto dal linguaggio Lisp e utilizzato in dialetti di Lisp
  - Il significato degli identificatori non locali di procedure e funzioni dipende da dove sono invocate.
  - Più semplice da gestire da interpreti del linguaggio, MA
  - accesso alle non locali non è in tempo costante (e dipende dal livello di annidamento dei blocchi nel programma)

Memoria

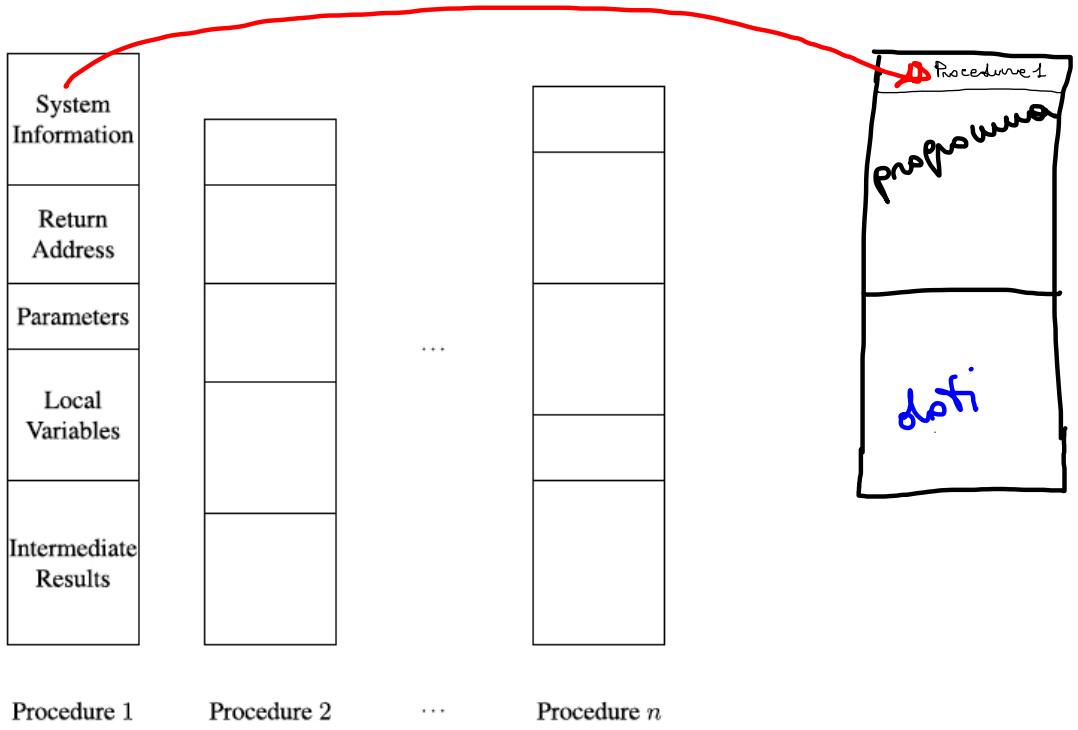
- **Memoria**
  - Deve contenere
    - (a) il programma (Codice Oggetto o Abstract Syntax Tree)
    - (b) tutte le informazioni per gestire l'esecuzione (stato del programma, Activation Record)
    - (c) i dati su cui opera il programma
  - Tre principali tipi di memoria:
    - Memoria Statica
    - Memoria Dinamica a Stack
    - Memoria Dinamica a Heap
  - Le Macchine Astratte dei Linguaggi di oggi, richiedono tutti e tre questi tipi
  - che possono altrimenti, essere emulati su Macchine Astratte con memoria del solo tipo Statica.

Memoria Statica

- **Memoria Statica.**
  - L'allocazione delle strutture è fatta prima dell'esecuzione del programma.
  - L'allocazione delle memoria non può essere modificata dell'esecuzione del programma.
- Per quale dei contenuti (a), (b), (c) (vedi slide precedente) è adatta?
  - (a). Adatta a memorizzare il programma (se il linguaggio non permette riflessione sui programmi)
  - Ma per (b) e (c) in generale, la risposta è non adatta
- Quali caratteristiche di un LP per poter utilizzare Memoria statica anche in (b) e in (c)?
  - (b) nessun meccanismo di ricorsione
  - (c) nessun meccanismo di allocazione dinamica di valori (malloc, liste,..)

Macchina Astratta con sola Memoria Statica

- **Organizzazione della memoria quando**
  - (b) nessun meccanismo di ricorsione
  - (c) nessun meccanismo di allocazione dinamica di valori



- La figura mostra solo (b) e per i soli blocchi procedura, completarla per:
- (b) per blocchi in-line
- (a) per mem. programma; (c) per mem. dati