Strutturare i Dati

Luca Abeni

March 29, 2017

Strutturazione dei Dati

- Macchina fisica: lavora su sequenze di bit
  - Facile e pratico per un computer...
  - ...Ma non proprio il massimo per un programmatore umano!
- Necessità di associare significati a sequenze di bit
  - Cosa rappresenta la sequenza “`01100001`”?
  - Il numero decimale 87?
  - Il carattere “a”?
  - Altro?
- Necessità di definire operazioni su sequenze di bit
  - Se interpreto “`0010`” e “`0100`” come interi, come ne calcolo la somma?

Tipi di Dato

"*Collezione di valori omogenei ed effettivamente rappresentabili, dotata di un insieme di operazioni che operano su tali valori*"

- Valori omogenei: l'insieme $\{2, 3.1415, true, "AAA"\}$ difficilmente è la base per un tipo...
  - $1, 2, 3, 4, 5, 6, 7, 8, 9$ è già meglio...
- Effettivamente rappresentabili: l'insieme $\mathcal{R}$ dei numeri reali non può costituire un tipo (esempio: π non è rappresentabile su un numero finito di bit)
  - Un insieme di approssimazioni di numeri reali va già meglio...
- Necessità di definire operazioni sui valori

Tipi di Dato

- Una volta definiti vari tipi di dato, come li combino fra loro?
- Sistema di tipi:
  - Insieme di tipi di dato (tipi predefiniti di un linguaggio, ...)
  - Meccanismi che permettono di definire nuovi tipi
  - Meccanismi per il “controllo dei tipi”
    - Regole di equivalenza
    - Regole di compatibilità
    - Meccanismi di inferenza di tipo
  - Regole di controllo dei vincoli sui tipi (controllo statico o dinamico, ...)

Equivalenza fra Tipi

- Tipi equivalenti: diversi fra loro, ma intercambiabili
  - Non distinguibili nel loro uso
  - Posso usare un valore del primo tipo al posto di un valore del secondo e viceversa
- Possono essere visti come due nomi diversi per lo stesso tipo di base...
  - Esempio: “**typedef int** time_t”

Compatibilità fra Tipi

- Il tipo `T` è compatibile col tipo `S` se e solo se un valore di `T` è usabile al posto di un valore di `S`
- Relazione riflessiva, transitiva ma non simmetrica
  - `T1` è compatibile con `T1`
  - Se `T1` è compatibile con `T2` e `T2` è compatibile con `T3`, allora `T1` è compatibile con `T3`
  - Se `T1` è compatibile con `T2` non è detto che `T2` sia compatibile con `T1`
- Equivalenza implica (ovviamente) compatibilità, ma compatibilità non implica equivalenza!

Sistemi Type Safe

- Sistema di tipi type safe (sicuro relativamente ai tipi)
  - "*Nessun programma (sintatticamente corretto) può violare i vincoli sui tipi definiti dal linguaggio*"
  - Nessun programma può, durante la sua esecuzione, generare un errore (non segnalato) derivante dalla violazione di vincoli sui tipi

```
{
    double *p;

    p = malloc(sizeof(int));
    *p = 2.3;
    ...
```

- Il linguaggio C non è type safe!!!

Controlli sui Tipi

- Controlli a tempo di:
  - Compilazione $\rightarrow$ Tipizzazione statica
  - Esecuzione $\rightarrow$ Tipizzazione dinamica
- Differenti trade-off su:
  - Sicurezza (tipizzazione statica: certezza che il programma è sempre corretto)
  - Overhead
  - Tempo di compilazione
  - Flessibilità
- Problemi di decidibilità
  - Controllo statico fallisce anche con programmi che non genereranno mai errore...

Proprietà dei Tipi di Dati

- In realtà, proprietà dei valori che il tipo può assumere (e/o delle operazioni su tali valori)...
- I valori di un tipo di dato possono essere:
  - Denotabili: associabili ad un nome nell'ambiente
  - Esprimibili: risultato di una espressione
  - Memorizzabili: associabili ad una variabile tramite la funzione memoria
- Esempio: le funzioni $f : \texttt{int} \rightarrow \texttt{int}$ sono denotabili? Ed esprimibili? E Memorizzabili?
  - Sempre denotabili
  - Qualche volta esprimibili ("**fn** x => ..." è un'espressione che ritorna una funzione)
  - Qualche volta memorizzabili