Calcolo dei numeri reali esatti in Haskell

Ingegneria dell’Informazione, Informatica e Statistica

Andrea Princic

Sapienza
Università di Roma

Introduzione

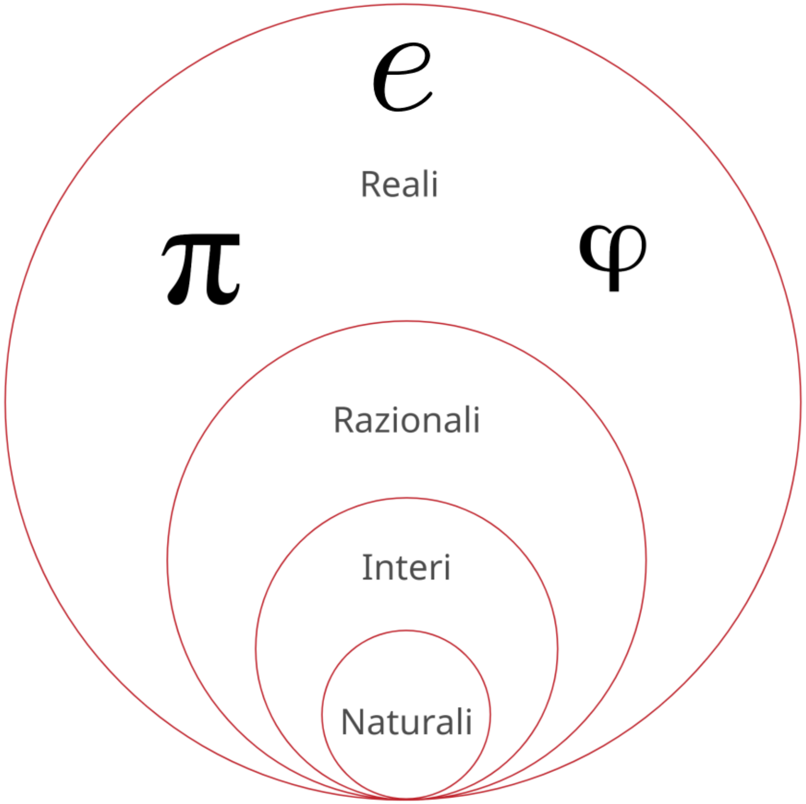
Calcolo dei numeri reali esatti

- Si riferisce alla possibilità di fare operazioni usando numeri in notazione posizionale, con un numero di cifre potenzialmente infinito e non periodico, ad un livello arbitrario di precisione
- Fa uso di rappresentazioni ed algoritmi diversi da quelli classici, che permettono di rappresentare numeri a precisione illimitata usando una quantità finita di memoria per la rappresentazione e per gli algoritmi
- Elimina la perdita di precisione prima, durante e dopo le operazioni, al prezzo di un calo delle prestazioni

I numeri reali nei calcolatori

I numeri reali

- Razionali
  - hanno una parte finale periodica
  - possono essere rappresentati con una quantità finita di memoria
  - teoricamente sarebbe possibile fare calcoli esatti sui numeri razionali
- Irrazionali
  - non sono periodici
  - non possono essere rappresentati con una quantità finita di memoria
  - si possono fare calcoli a precisione arbitraria



Precisione con la rappresentazione standard

- La rappresentazione dei numeri in virgola mobile va bene per fare calcoli che non richiedono livelli di precisione estremi
- Un classico esempio di perdita di precisione durante un calcolo banale:

```
>>> 0.1 + 0.1
0.2
>>> 0.2 + 0.1
0.30000000000000004
```

- Perfino nella rappresentazione di un intero:

```
>>> 2147483647.0
2147483648.0
>>> 2147483647.0 - 64
2147483648.0
>>> 2147483647.0 - 65
2147483520.0
```

Interi a precisione illimitata

- Alcuni linguaggi implementano gli interi a precisione illimitata: un tipo di dato che permette di rappresentare numeri interi di grandezza arbitraria, eliminando il problema dell'overflow
- Questo tipo di interi funziona come un intero normale, però incrementa la sua grandezza in memoria quando raggiunge il limite rappresentabile dalla sua attuale dimensione
- Python e Haskell implementano questo tipo di interi
- Utilizzando questo tipo di interi si potrebbero rappresentare razionali a precisione illimitata

Numeri computabili

Numeri computabili

- Anche potendo calcolare i numeri reali a qualunque livello di precisione, i numeri calcolabili con queste tecniche non sono comunque tutti i numeri reali
- I numeri computabili sono un sottoinsieme dei numeri reali che include tutti i numeri razionali e una quantità numerabile di numeri irrazionali
- In questo insieme si trovano tutti i numeri reali `x` per i quali esiste una macchina di Turing tale che, dato un naturale `n` sul nastro iniziale, termina con l'`n`-esima cifra di `x` sul nastro
- Esiste quindi una quantità non numerabile di numeri irrazionali che non si possono calcolare

Rappresentazioni per i reali

Gli stream

- Per poter rappresentare una quantità arbitraria di informazioni utilizziamo gli stream: una sequenza di elementi di lunghezza illimitata che vengono elaborati soltanto quando necessario
- Grazie a questa caratteristica si possono rappresentare i numeri reali come sequenze infinite di cifre, calcolandone soltanto poche alla volta partendo da sinistra
- Questo fa sì che gli algoritmi sugli stream operino da sinistra a destra, e non da destra a sinistra come si fa di solito
- Anche il riporto, dove necessario, viene portato da sinistra a destra

Gli stream
Sintassi

- Gli stream sono simili a delle liste

$$[x_1,\ x_2,\ x_3,\ \ldots] \equiv x_1 : x_2 : x_3 : \ldots$$

- Concatenazione di cifre in testa ad uno stream

$$x_1 : x_2 : x_3 : x$$

- Stream infinito di una cifra

$$\overrightarrow{x}$$

- Valore numerico di uno stream

$$[\![x]\!]$$

Gli stream
Semantica

- Rappresentando i numeri soltanto con gli stream non c'è modo di distinguere tra parte intera e parte decimale
- Per questo gli stream rappresentano soltanto un intervallo chiuso (dipendente dalla base) di numeri reali
- Il passaggio da numerale a numero viene fatto nel seguente modo

$$[\![x]\!] = \sum_{i=1}^{\infty} d_i \times b^{-i} = \frac{d_1}{b} + \frac{d_2}{b^2} + \frac{d_3}{b^3} + \ldots$$

- In base 2 gli stream di cifre permettono di rappresentare i numeri nell'intervallo [0, 1]

Stream di cifre

- Una semplice rappresentazione dei numeri reali come stream di cifre non può funzionare perché alcune operazioni non sarebbero computabili
- Ad esempio valutando la somma in base 2

$$0 : 1 : 1 : 1 : \ldots \;+\; 0 : 0 : 0 : 0 : \ldots$$

- Servirebbe una quantità potenzialmente infinita di cifre anche solo per stabilire la prima cifra del risultato

Cifre binarie con segno

- Una possibile rappresentazione che rende computabile questo tipo di operazioni è quella delle cifre binarie con segno: alla rappresentazione binaria si aggiunge la cifra -1. Questo amplia l'intervallo di rappresentazione a [-1,1]
- La nuova cifra introduce una ridondanza, ovvero permette di rappresentare uno stesso numero in infiniti modi diversi: in questa rappresentazione gli unici numeri che si possono scrivere in un solo modo sono 1 e -1, e valgono le seguenti identità

$$\begin{aligned} 1 : \overline{1} : x &= 0 : 1 : x \\ \overline{1} : 1 : x &= 0 : \overline{1} : x \end{aligned}$$

Cifre binarie con segno

- In questa rappresentazione la somma

$$0:1:1:1:\ldots \; + \; 0:0:0:0:\ldots$$

- è computabile. Grazie alla cifra -1, infatti, l'algoritmo per la somma può generare la cifra 1 come prima cifra del risultato e in seguito, se il risultato reale si rivelasse più piccolo di quello generato, basterebbe generare una o più cifre -1 per ridurre il valore dell'output
- Per rappresentare numeri sull'intera retta reale si usa una rappresentazione con mantissa (stream) ed esponente

$$[\![x]\!] = [\![(m,\ e)]\!] = [\![m]\!] \times 2^e$$

Basi non naturali

- La base di una rappresentazione non deve per forza essere un numero naturale: qualunque numero computabile va bene, a patto che il numero di cifre usate nella rappresentazione sia strettamente maggiore della base
- Una rappresentazione in base reale è data da un numero naturale `d` e un numero computabile `b` tali che `1 < b < d`. Una sequenza di interi

$$z_0 : z_1 : z_2 : \ldots$$

- in tale base rappresenta il numero

$$x = \sum_{i=0}^{\infty} z_i \times b^{-i}$$

- in cui il primo intero della serie rappresenta la parte intera

Golden notation

Rappresentazione in base ϕ

- Usa come base il numero irrazionale ϕ

$$\phi = \frac{\sqrt{5}+1}{2} \simeq 1.618033988749$$

- In questa base la ridondanza è introdotta dall'uguaglianza

$$\phi^{n+2} = \phi^{n+1} + \phi^{n}$$

$$1.00 = 0.11 = 0.\overline{10}$$

- Il che significa che ogni numero (diverso da 0) ha una rappresentazione non finita con parte periodica non nulla

Golden notation
Notazione semplificata

- Gli stream di cifre in base ϕ permettono di rappresentare i numeri nell'intervallo [0, ϕ]

$$[\![\alpha]\!]_s = \sum_{i=1}^{\infty} \alpha_i \times \phi^{-i}$$

- In questa notazione gli unici numeri che si possono rappresentare in un solo modo sono

$$\phi = \left[\!\left[\overrightarrow{1}\right]\!\right]_s \qquad 0 = \left[\!\left[\overrightarrow{0}\right]\!\right]_s$$

- Gli algoritmi per questa notazione utilizzano alcune cifre di lookahead dalla testa degli stream per generare ricorsivamente cifre in output
- I risultati delle operazioni in questa notazione vengono shiftati di alcune posizioni per evitare overflow

Golden notation
Notazione completa

- Per rappresentare tutti i numeri computabili si usa una rappresentazione simile a quella con mantissa esponente, con una modifica per poter rappresentare i numeri negativi

$$[\![z : \alpha]\!]_f = (-1 + [\![\alpha]\!]_s) \times \phi^{2z} = \left(-1 + \sum_{i=1}^{\infty} \alpha_i \times \phi^{-i}\right) \times \phi^{2z}$$

- In testa allo stream in notazione semplificata si aggiunge un intero che rappresenta la metà dell’esponente
- Al valore dello stream in notazione semplificata si sottrae 1 per rendere possibile la rappresentazione di numeri negativi

Golden notation
Addizione semplificata

$$[\![A(\alpha,\ \beta,\ a,\ b)]\!]_s = \frac{[\![\alpha]\!]_s + [\![\beta]\!]_s + \frac{a}{\phi} + \frac{b}{\phi^2}}{\phi^2}$$

$$\begin{array}{lcl}
A(0:\alpha,\ 0:\beta,\ 0,\ b) & = & 0:A(\alpha,\ \beta,\ b,\ 0) \\
A(0:0:\alpha,\ 0:\beta,\ 1,\ b) & = & 0:A(b:\alpha,\ \beta,\ 1,\ 1) \\
A(0:1:\alpha,\ 0:1:\beta,\ 1,\ 1) & = & 1:0:A(\alpha,\ \beta,\ 0,\ 1) \\
A(0:0:\alpha,\ 1:0:\beta,\ 1,\ 0) & = & 0:1:A(\alpha,\ \beta,\ 1,\ 0) \\
A(0:\alpha,\ 1:\beta,\ 1,\ 1) & = & 1:A(\alpha,\ \beta,\ 0,\ 0) \\
A(1:\alpha,\ 1:\beta,\ 1,\ b) & = & 1:A(\alpha,\ \beta,\ b,\ 1)
\end{array}$$

$$\begin{array}{lcl}
A(1:\alpha,\ 0:\beta,\ a,\ b) & = & A(0:\alpha,\ 1:\beta,\ a,\ b) \\
A(\alpha,\ 1:\beta,\ 0,\ b) & = & A(\alpha,\ 0:\beta,\ 1,\ b) \\
A(a_1:1:\alpha,\ b_1:0:\beta,\ a,\ b) & = & A(a_1:0:\alpha,\ b_1:1:\beta,\ a,\ b) \\
A(\alpha,\ b_1:1:\beta,\ a,\ 0) & = & A(\alpha,\ b_1:0:\beta,\ a,\ 1)
\end{array}$$

Golden notation
Addizione e prodotto completi

- Addizione completa

$$A'(z:\alpha,\ t:\beta) = \begin{cases} (z+1):A(\alpha,\ \beta,\ 1,\ 0) & \text{se } z=t \\ A'((z+1):1:0:\alpha,\ t:\beta) & \text{se } z<t \\ A'(z:\alpha,\ (t+1):1:0:\beta) & \text{se } t<z \end{cases}$$

- Prodotto completo originale

$$P'(z:\alpha,\ t:\beta) = (z+t+2):A(P(\alpha,\ \beta),\ C(A(\alpha,\ \beta,\ 0,\ 0)),\ 1,\ 0)$$

- Prodotto completo corretto

$$P'(z:\alpha,\ t:\beta) = C'((z+t+2):A(C(P(\alpha,\ \beta)),\ A(\alpha,\ \beta,\ 0,\ 0),\ 1,\ 0))$$

Rappresentazione in base ϕ
Numeri di Fibonacci

$$
\begin{aligned}
1 &= 0.11 &= 1 \quad (1)\\
1 &= 1 &= 1 \quad (2)\\
10.01 &= 10.0011 &= 2 \quad (3)\\
100.01 &= 100.01 &= 3 \quad (4)\\
1000.1001 &= 1000.100011 &= 5 \quad (5)\\
10001.0001 &= 10001.0001 &= 8 \quad (6)\\
100010.001001 &= 100010.00100011 &= 13 \quad (7)\\
1000100.010001 &= 1000100.010001 &= 21 \quad (8)\\
10001000.10001001 &= 10001000.1000100011 &= 34 \quad (9)\\
100010001.00010001 &= 100010001.00010001 &= 55 \quad (10)\\
1000100010.0010001001 &= 1000100010.001000100011 &= 89 \quad (11)\\
10001000100.0100010001 &= 10001000100.0100010001 &= 144 \quad (12)
\end{aligned}
$$