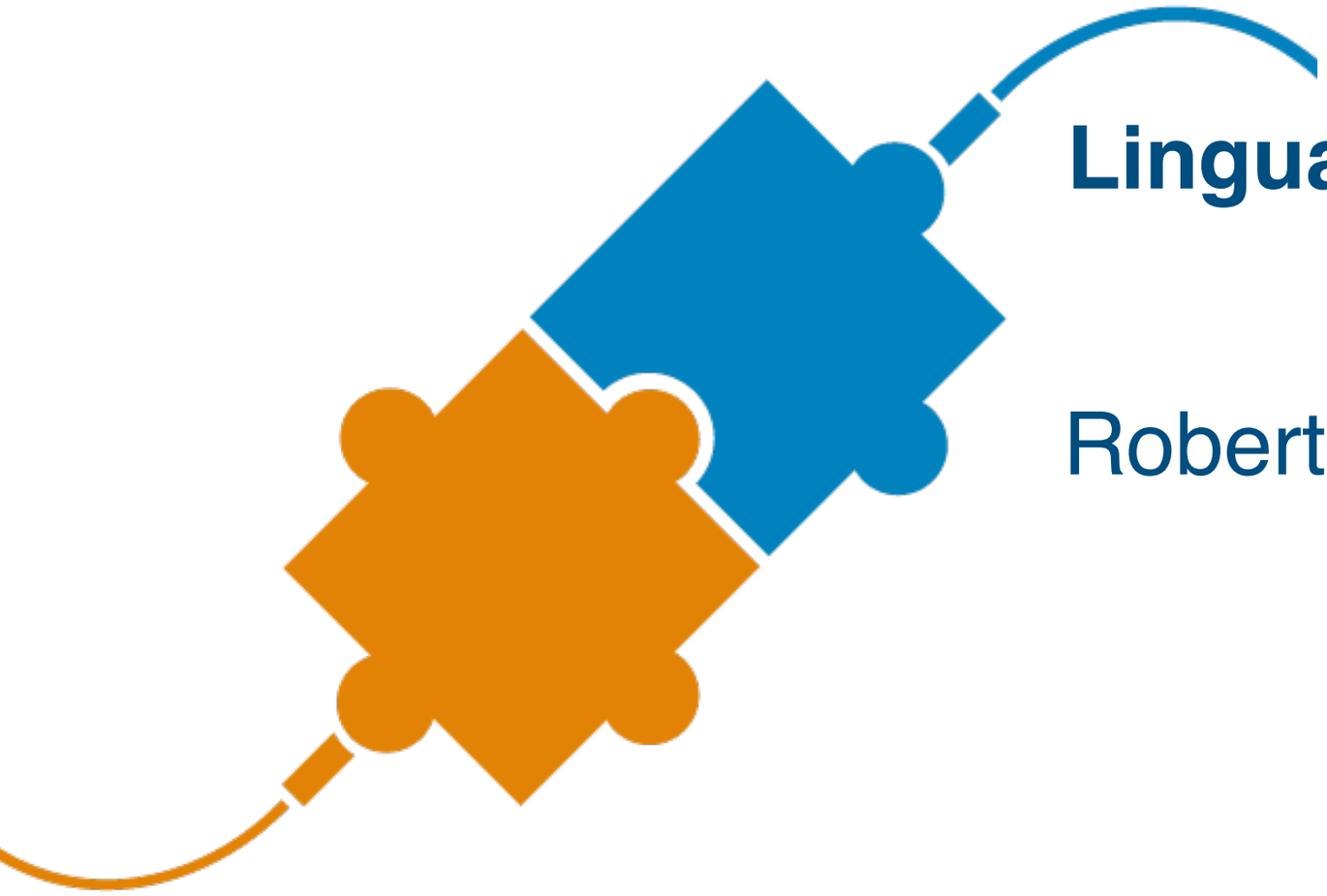


Linguaggi di Programmazione

Roberta Gori

Capitolo 1 (1.1 e 1.2)

Dalla sintassi alla semantica

Linguaggi di programmazione

Quando definiamo un linguaggio di programmazione

ne fissiamo:

1. la sintassi

programmi ben formati,
escludiamo i programmi senza senso

2. i tipi

riduce l'insieme di programmi legali,
aiuta a non commettere errori

3. pragmatica

come si usano i costrutti del linguaggio

4. (semantica)

Il significato dei programmi (ben tipati)

La sintassi

La sintassi di un linguaggio formale

1. L'alfabeto

quali simboli possono essere usati

2. La struttura grammaticale dei programmi

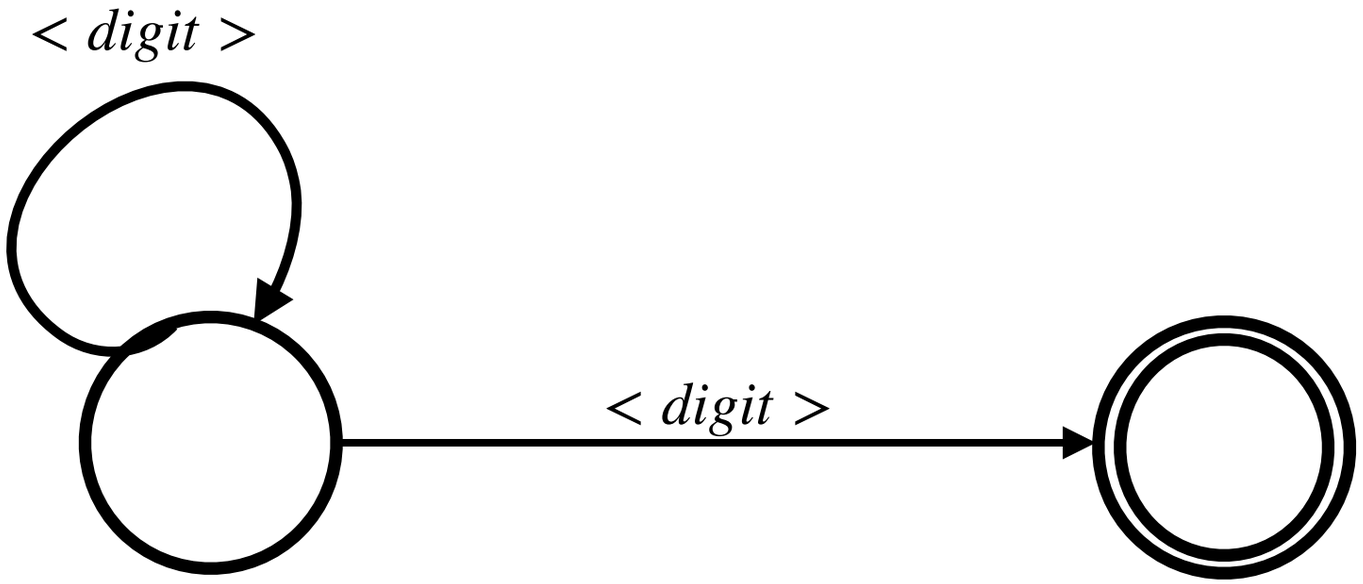
quali sequenze di simboli sono valide, quali devono invece essere segnalate come erronee

Modi per definire la sintassi sono
espressioni regolari, grammatiche libere, la notazione BNF, automi che riconoscono il linguaggio,…

Esempio

Una grammatica in BNF e un automa che riconosce lo stesso linguaggio

```
<numeral> ::= <digit> | <numeral> <digit>
<digit>   ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```



Sistema di tipi

Un sistema di tipi puo' essere usato per

1. ridurre gli errori
2. permettere ottimizzazioni di compilazione
3. scoraggiare le cattive pratiche di programmazione

sistemi di tipi diversi possono essere definiti sullo stesso linguaggio!

I sistemi di tipi sono spesso espressi usando regole logiche

Esempio

```
…
bool b := true;
while (b <= (b && i)) do {
    i := i-1;
}
```

Vantaggi della formalizzazione

Standardizzazione del linguaggio

i programmatori scrivono programmi sintatticamente corretti
gli implementatori scrivono parser corretti

Analisi formale delle proprietà del linguaggio

ambiguità, espressività, comparabilità

Implementazione automatica del front-end del compilatore

yacc, Bison, xtext, …

Esercizio

Consideriamo l'alfabeto $A \triangleq \{ (\ ,\ ) \}$

Definire la gramatica delle stringhe formate da parentesi bilanciate

S ::= ?

Pragmatica

I programmatori dovrebbero capire il codice che scrivono

Ogni manuale di un linguaggio contiene

1. descrizioni in linguaggio naturale dei vari costrutti
2. esempi di frammenti di codice e del loro uso
3. esempi di cattive pratiche

Chiamiamo pragmatiche

1. come sfruttare le varie caratteristiche
2. come dovrebbero essere progettati i compilatori
3. quali strumenti ausiliari sono disponibili

Ma questo e' sufficiente?

Le descrizioni in linguaggio naturale dovrebbero essere

1. più precise possibile
2. comprensibile
3. inequivocabili ma non pedanti

Ma ...

1. è difficile (quasi impossibile) coprire tutti i casi
2. molti punti rimarranno aperti a diverse interpretazioni
3. possono sorgere incongruenze
4. le buone pratiche non eliminano i problemi (li nascondono)

Alcuni problemi

Come dimostrare la conformità a qualche specifica?

Come dimostrare l'assenza di problemi?

Come produrre codice affidabile?

Come provare la conformità ai committenti?

Come provare la correttezza di un'implementazione?

Come definire i risultati corretti dei casi di test?

Come individuare precocemente ambiguità, anomalie, incoerenze?

Come esporre le debolezze?…

Le buone pratiche

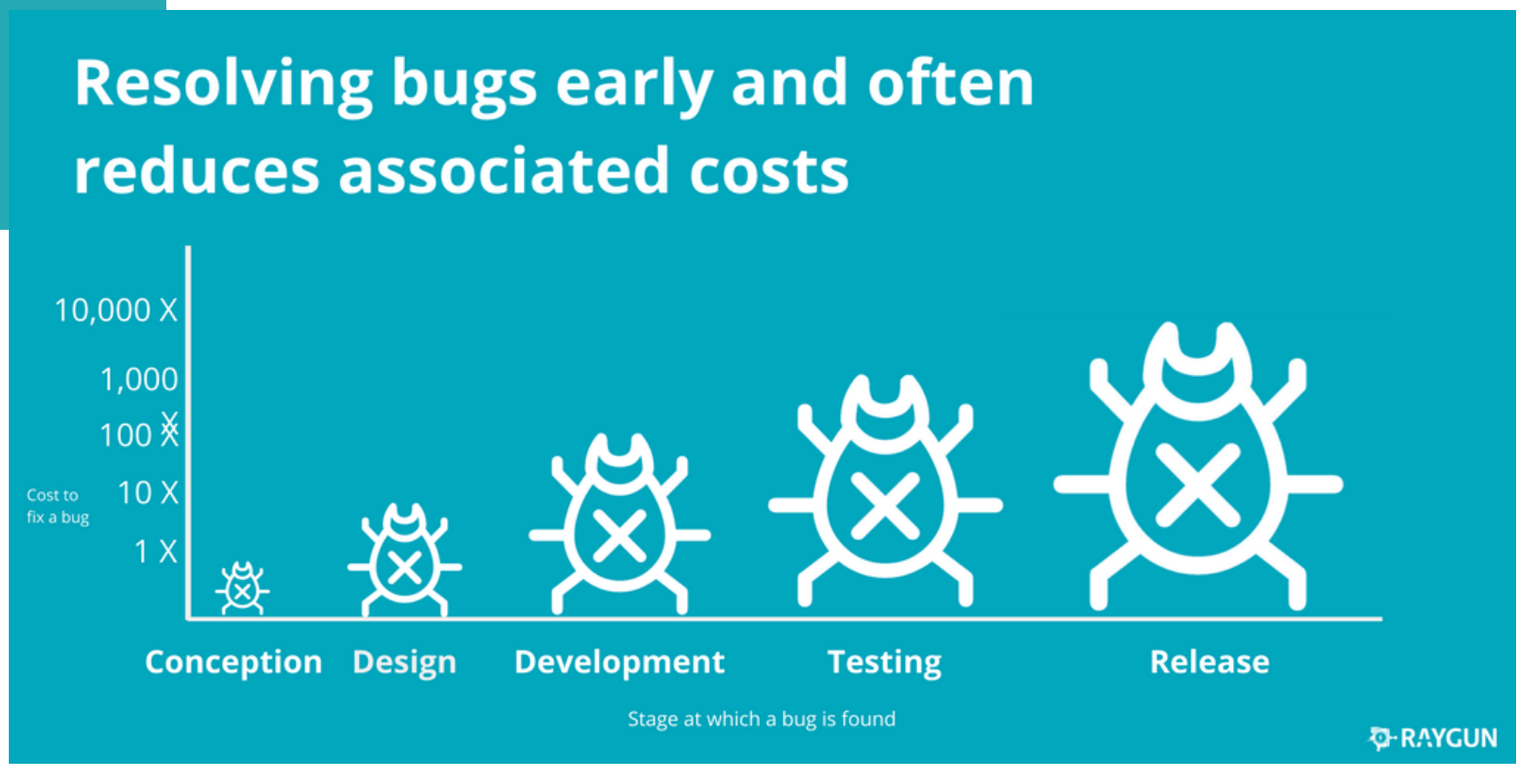
Code reviews

Test-driven development

Ma...

IT WORKS

on my machine



Semantica

La parola semantica è stata introdotta nel 1900:

lo studio di come le parole cambiano il loro significato (M. Bréal)

stranamente anche il suo significato è ora cambiato in

lo studio del legame tra le frasi di una lingua (scritta, parlata o formale) e i loro significati

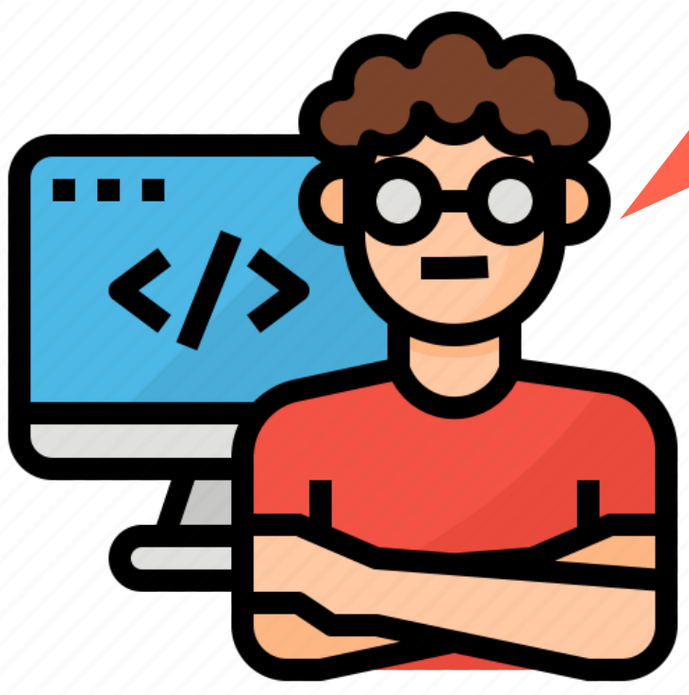
Nell'informatica ha a che fare con

lo studio del significato dei programmi (ben tipati)

La semantica formale assegna un significato rigoroso e non ambiguo ai programmi: dice ai programmatori il significato del codice che scrivono (ad un certo livello di astrazione)

Qualcuno dirà sempre

La (mia) implementazione è la semantica del linguaggio



corretta per definizione

e' indipendente dalla macchina?

e' portabile?

quanto durerà?

astrazione utile per altri programmatori?

come ragionarci sopra?

e quelle dei concorrenti?

…

Vantaggi della formalizzazione

Standardizzazione del modello di riferimento del linguaggio

ufficiale, indipendente dalla macchina

un modello mentale per i programmatori

un punto di riferimento per gli implementatori

Analisi formale delle proprieta' del linguaggio

espressività, correttezza dei tipi,

conformità del programma

Implementazione automatica del back-end del compilatore

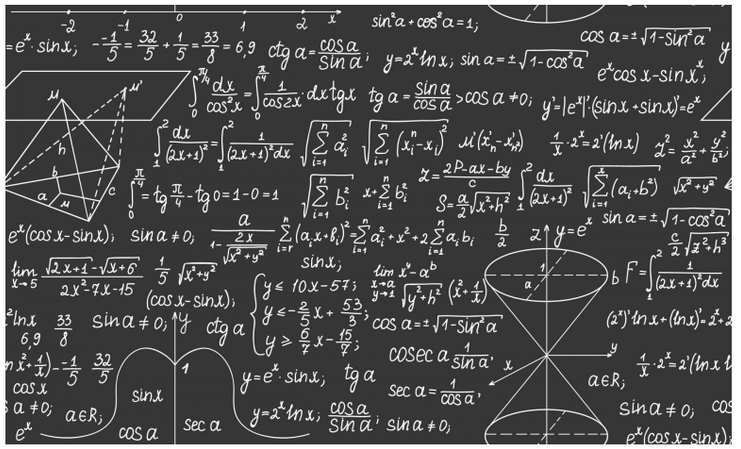
possiamo derivare un interprete per la sperimentazione

Obiezioni...

ma la semantica è più difficile da formalizzare della sintassi!

metodi diversi!

richiede nozioni matematiche e logiche!



$$\frac{\Gamma \vdash A \to B \qquad \dfrac{\Delta \vdash A}{\Delta \vdash !A} \qquad \dfrac{}{B \vdash B}}{\Gamma \vdash !A \multimap B \qquad \Delta, !A \multimap B \vdash B}$$

$$\frac{\Gamma, \Delta \vdash B}{\Gamma \cup \Delta \vdash B}$$

$$\frac{\Gamma \vdash A \to B \qquad \dfrac{\Delta \vdash A}{\Delta \vdash !A} \qquad \dfrac{}{B \vdash B}}{\Gamma \vdash !A \multimap B \qquad \Delta, !A \multimap B \vdash B}$$

$$\frac{\Gamma, \Delta \vdash B}{\Gamma \cup \Delta \vdash B}$$

$$\frac{\Gamma \vdash A \to B \qquad \dfrac{\Delta \vdash A}{\Delta \vdash !A} \qquad \dfrac{}{B \vdash B}}{\Gamma \vdash !A \multimap B \qquad \Delta, !A \multimap B \vdash B}$$

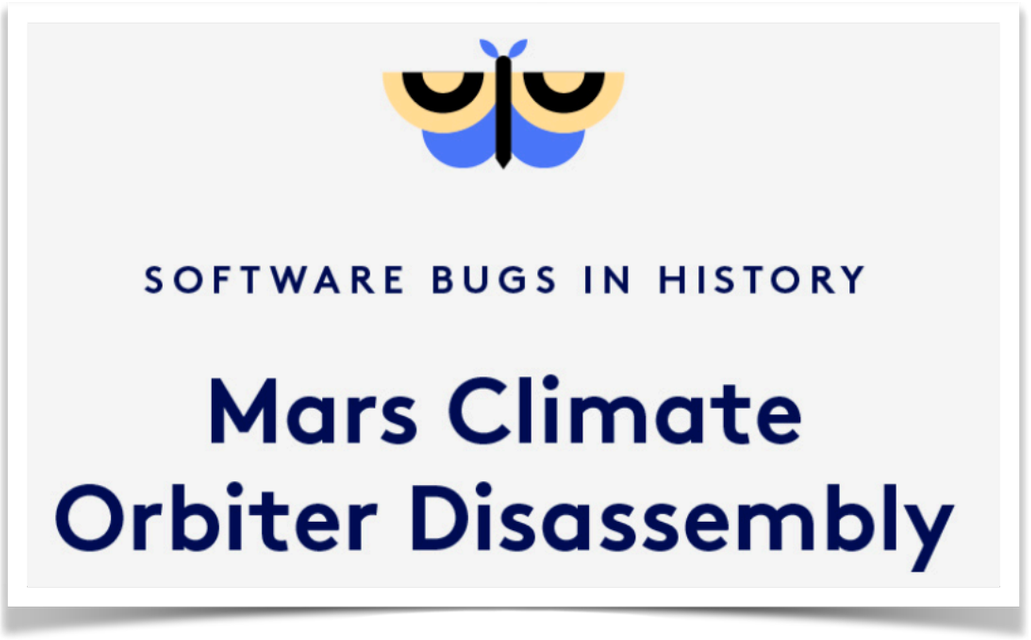
$$\frac{\Gamma, \Delta \vdash B}{\Gamma \cup \Delta \vdash B}$$

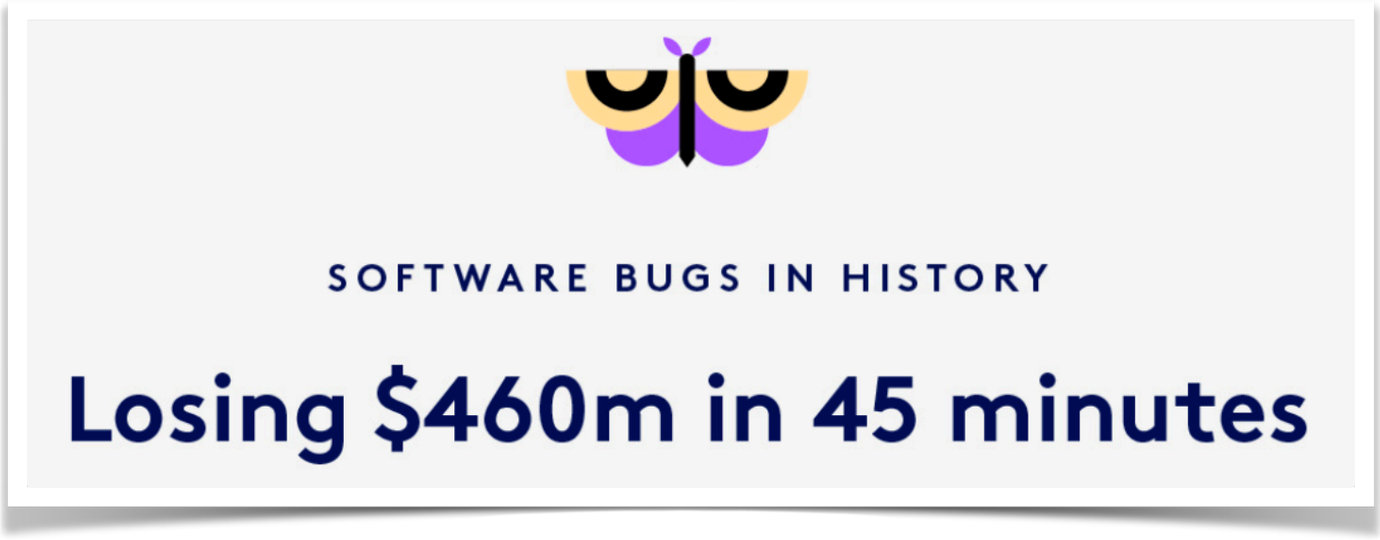
$$\frac{\Gamma \vdash A \to B \qquad \dfrac{\Delta \vdash A}{\Delta \vdash !A} \qquad \dfrac{}{B \vdash B}}{\Gamma \vdash !A \multimap B \qquad \Delta, !A \multimap B \vdash B}$$

$$\frac{\Gamma, \Delta \vdash B}{\Gamma \cup \Delta \vdash B}$$

non farmi perdere tempo, voglio scrivere codice

e di conseguenza ...





e ancora...

Heathrow Airport has apologised for disruption after the west London hub was hit by "technical issues".

One passenger said the situation was "utter chaos" after a problem with the airport's IT system saw staff called in to help passengers get to gates on the second day of the half-term weekend.

Heathrow Airport @HeathrowAirport · 16 feb 2020

Today's technical issue has now been resolved and Heathrow's systems are returning to normal. We apologise for the inconvenience caused. Our teams will continue to monitor our systems and be on hand to provide assistance to passengers as we work to resume our regular operations.

29 36 97

Risposte

[redacted] · 16 feb 2020

In risposta a @HeathrowAirport

Someone turned the system off and back on again?

1

la semantica

Tre differenti approcci

I metodi di definizione della semantica si dividono in tre gruppi principali

1. Operazionale

l' accento e' su come si ottiene il risultato della computazione

2. Denotazionale

solo l'effetto della computazione è interessante, non come si ottiene

3. Assiomatico

l'attenzione si concentra su asserzioni valide nella computazione

Semantica operazionale

opposto ad un dispositivo fisico esistente

Idea:definire un qualche tipo di macchina astratta e descrivere il significato di un programma in termini di passi o istruzioni che questa macchina esegue per eseguire il compito

Motivazione:
spiegare le computazioni

l'enfasi è sugli **stati** e le **transformazioni** tra stati

$$s_0 \to s_1 \to s_2 \to \cdots \to s_n \to r$$

Padri fondatori

['70] small-step: semantica di LISP di John McCarthy (1960) e di Algol 68 (1975)

Recursive Functions of Symbolic Expressions
and Their Computation by Machine, Part I

John McCarthy, Massachusetts Institute of Technology, Cambridge, Mass.

April 1960

['80] approccio SOS: Gordon Plotkin ha introdotto la **semantica operazionale strutturale** (orientata alla sintassi e definita induttivamente) nel 1981 (uno dei rapporti tecnici più citati nell'informatica, pubblicato in una rivista solo più di 20 anni dopo).

$$\frac{\langle e_0, \sigma\rangle \longrightarrow \langle e_0', \sigma\rangle}{\langle e_0 + e_1, \sigma\rangle \longrightarrow \langle e_0' + e_1, \sigma\rangle}$$

['90] big-step:Gilles Kahn ha introdotto la semantica naturale nel 1987, dove il risultato è calcolato in un solo passo

$$s_0 \to r$$

Panoramica

Oggi la relazione di transizione è tipicamente definita induttivamente, da assiomi e regole di inferenza secondo la sintassi del programma (stile SOS).

Vantaggi:
traduzione immediata nelle clausole di Horn della programmazione logica;
prototipo di interprete Prolog (quasi) gratuito;
forti connessioni con la sintassi del linguaggio;
utile per individuare comportamenti sotto specificati;
le regole per i diversi costrutti sono ordinatamente separate;
la matematica coinvolta di solito non è molto complicata;
le descrizioni SOS sono facili da leggere, anche per i non specialisti;

Semantica denotazionale

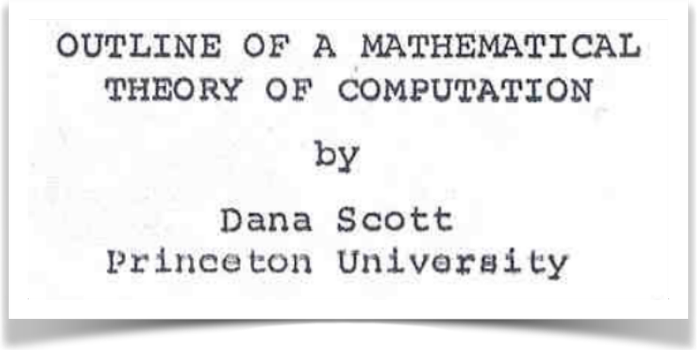
Idea: il significato di un programma è un oggetto matematico (per esempio una funzione da input a output) e i passi fatti per calcolare il risultato non sono importanti

$$[\![\cdot]\!] : \text{Programs} \to \text{Domains}$$

Motivazione: le funzioni sono indipendenti da come vengono calcolate e quindi sono più semplici della sequenza di operazioni passo dopo passo della semantica operazionale

Padri fondatori

[’60/’70]: Christopher Strachey and Dana Scott


OUTLINE OF A MATHEMATICAL
THEORY OF COMPUTATION

by

Dana Scott
Princeton University

Principio di composizione: la semantica e’ una funzione che assegna un elemento di qualche dominio matematico a ogni singolo costrutto in modo tale che

> *il significato di un costrutto composto non dipende dalla forma particolare dei costrutti costituenti, ma solo dai loro significati*

Panoramica

Vantaggi:
matematicamente elegante;
utile per individuare comportamenti sotto specificati;
può essere usato per derivare implementazioni di prototipi;
è servito come ispirazione per molti linguaggi di programmazione;
difficile da applicare a sistemi concorrenti e interattivi

Semantica assiomatica

Idea: descrivere i costrutti di un linguaggio di programmazione fornendo assiomi logici che sono soddisfatti da questi costrutti

Motivazione: dimostrare la correttezza di un programma rispetto ad una data specifica

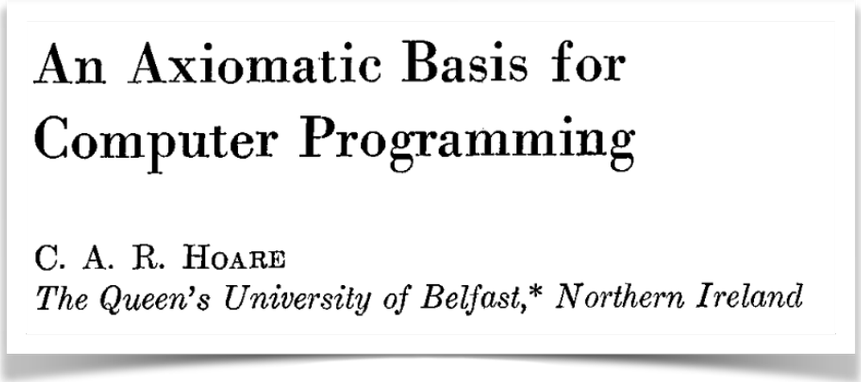
$$\vdash \{P\}\ c\ \{Q\}$$

Padri fondatori

['60]: Robert W. Floyd (1967) and Tony Hoare (1969)


Robert W. Floyd

ASSIGNING MEANINGS TO PROGRAMS[1]


An Axiomatic Basis for Computer Programming

C. A. R. Hoare
The Queen's University of Belfast, Northern Ireland*

Logica di Hoare: una dichiarazione è accompagnata da una precondizione (lo stato prima dell'esecuzione) e una postcondizione (lo stato dopo l'esecuzione)

I significato di un programma è una proposizione logica che afferma alcune proprietà dell'output ogniqualvolta alcune proprietà dell'input sono soddisfatte

Panoramica

Vantaggi:
enfasi sulla correttezza delle prove fin dall'inizio;
sistemi di prove sorprendentemente eleganti;
può essere usato per dimostrare l'assenza di bug;
difficile da applicare a sistemi concorrenti e interattivi

Non fate la guerra

Le diverse semantiche sono spesso viste in opposizione tra loro, ma non dovrebbe essere così!
Possiamo ottenere molto di più dalla loro combinazione!

In questo corso

Ci concentriamo sulla semantica operazionale e denotazionale!

Vedremo le idee e i metodi fondamentali dietro questi approcci e sottolineeremo le loro relazioni, dimostrando alcuni teoremi di corrispondenza rilevanti.

$s_0 \to r$

$[\![\cdot]\!] : \text{Programs} \to \text{Domains}$

Una carrellata dei differenti approcci

Un semplice linguaggio

Sintassi informale delle espressioni numeriche

- ogni numerale N è un espessione;
- se E_1 e E_2 sono espressioni, allora anche $E_1 \oplus E_2$ e' un espressione;
- se E_1 e E_2 sono espressioni, allora anche $E_1 \otimes E_2$ e' un espressione.

N	numerali vs numeri	n
syntassi per scrivere i numeri	5 1 0 1 cinque	$5 \in \mathbb{N}$	oggetti matematici, concetti

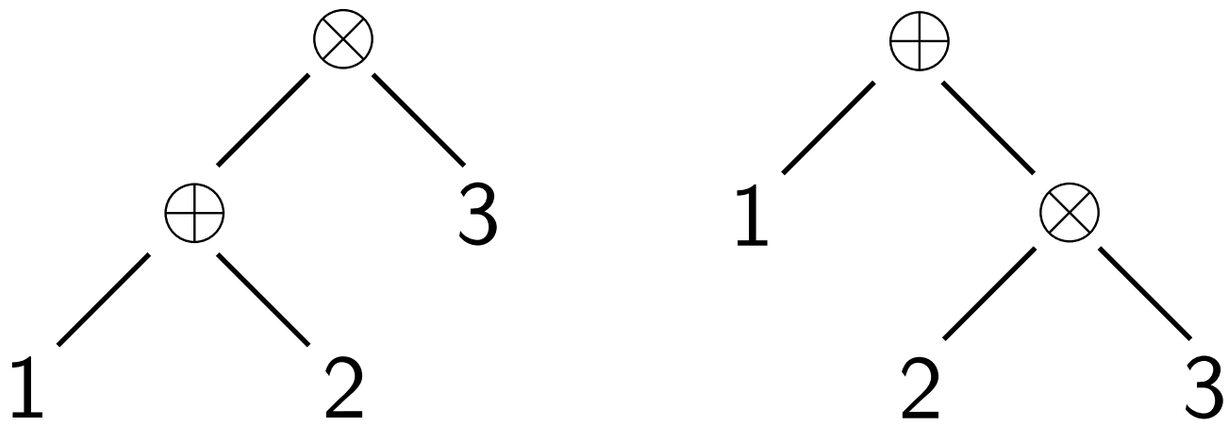
Sintassi formale

$$E ::= N \mid E \oplus E \mid E \otimes E$$

$\oplus 3 \otimes 4$ non e' un espressione numerica ben formata

data la sequenza

$1 \oplus 2 \otimes 3$



ha due diversi alberi di sintassi astratta

usiamo le parentesi

$$1 \oplus (2 \otimes 3)$$

E correggiamo la grammatica per evitare ambiguita'

Dare un significato

$$E ::= N \mid E \oplus E \mid E \otimes E$$

questa e' solo sintassi!

e' N necessariamente un numero?

e' $\oplus$ necessariamente la somma aritmetica?

e' $\otimes$ necessariamente il prodotto aritmetico?

forse stiamo parlando di
matrici con addizione e moltiplicazione

o insiemi con unione e intersezione

o stringhe con concatenazione e minimo prefisso comune

o alberi con due forme di fusione

Semantica informale

Semantica informale delle espressioni numeriche

- un numerale N viene valutato al suo corrispondente valore n;
- per valutare $E_1 \oplus E_2$ valutiamo E_1 e E_2 e poi sommiano i valori;
- per valutare $E_1 \otimes E_2$ valutiamo E_1 e E_2 e poi moltiplichiamo i valori;

Tre regole sono sufficienti per determinare il valore di qualsiasi espressione ben formata, non importa quanto grande

Notate che non stiamo dicendo l'ordine di valutazione degli argomenti: è importante?

Pragmatica

Possiamo illustrare il significato con esempi

- 2 viene valutato a 2;
- $(1 \oplus 2) \otimes 3$ viene valutato a 9;
- $(1 \oplus 2) \otimes (3 \oplus 4)$ viene valutato a 21

Semantica small-steps

Espressioni numeriche runtime

$$\mathsf{E} ::= n \mid \mathsf{N} \mid \mathsf{E} \oplus \mathsf{E} \mid \mathsf{E} \otimes \mathsf{E}$$

Lo stato della macchina astratta può mescolare risultati intermedi con espressioni

un passo $\mathsf{E}_0 \to \mathsf{E}_1$

una valutazione $\mathsf{E}_0 \to \mathsf{E}_1 \to \mathsf{E}_2 \to \cdots \to \mathsf{E}_k \to n$

che puo' essere scritta $\mathsf{E}_0 \to^* n$

ci aspettiamo che $n \not\to$

Come definiamo la relazione di transizione?

regole di inferenza

regole di inferenza

$$(rule\ name)\ \frac{premises}{conclusion}\ side\ condition$$

Se le premesse e la condizione (side condition) sono soddisfatte, allora si può trarre la conclusione

La conclusione è un unica proposizione

Le premesse consistono in uno, nessuno o più proposizioni

La condizione è un predicato logico

Il nome della regola è solo una comoda etichetta

Regole SOS

$$(num)\ \frac{}{\mathsf{N} \to n}$$

$$(sum)\ \frac{}{n_0 \oplus n_1 \to n}\ n = n_0 + n_1$$

$$(sumL)\ \frac{\mathsf{E}_0 \to \mathsf{E}'_0}{\mathsf{E}_0 \oplus \mathsf{E}_1 \to \mathsf{E}'_0 \oplus \mathsf{E}_1}$$

$$(sumR)\ \frac{\mathsf{E}_1 \to \mathsf{E}'_1}{\mathsf{E}_0 \oplus \mathsf{E}_1 \to \mathsf{E}_0 \oplus \mathsf{E}'_1}$$

$$(prod)\ \frac{}{}$$

$$(prodL)\ \frac{}{}$$

$$(prodR)\ \frac{}{}$$

Alcune derivazioni

$$(prodL)\dfrac{(sumL)\dfrac{(num)\dfrac{}{1 \to 1}}{(1 \oplus 2) \to (1 \oplus 2)}}{(1 \oplus 2) \otimes (3 \oplus 4) \to (1 \oplus 2) \otimes (3 \oplus 4)}$$

$$(prodL)\dfrac{(sumR)\dfrac{(num)\dfrac{}{2 \to 2}}{(1 \oplus 2) \to (1 \oplus 2)}}{(1 \oplus 2) \otimes (3 \oplus 4) \to (1 \oplus 2) \otimes (3 \oplus 4)}$$

$$(prodL)\dfrac{(sum)\dfrac{}{(1 \oplus 2) \to 3}\ {\scriptstyle 3 = 1 + 2}}{(1 \oplus 2) \otimes (3 \oplus 4) \to 3 \otimes (3 \oplus 4)}$$

Una computazione

$$
\begin{aligned}
(1 \oplus 2) \otimes (3 \oplus 4) &\rightarrow (\color{blue}{1} \oplus 2) \otimes (3 \oplus 4) \\
&\rightarrow (\color{blue}{1} \oplus \color{blue}{2}) \otimes (3 \oplus 4) \\
&\rightarrow \color{blue}{3} \otimes (3 \oplus 4) \\
&\rightarrow \color{blue}{3} \otimes (\color{blue}{3} \oplus 4) \\
&\rightarrow \color{blue}{3} \otimes (\color{blue}{3} \oplus \color{blue}{4}) \\
&\rightarrow \color{blue}{3} \otimes \color{blue}{7} \\
&\rightarrow \color{blue}{21} \\
&\not\rightarrow
\end{aligned}
$$

$$(1 \oplus 2) \otimes (3 \oplus 4) \rightarrow^* \color{blue}{21}$$

Un altra computazione

$$
\begin{aligned}
(1 \oplus 2) \otimes (3 \oplus 4) &\rightarrow (1 \oplus 2) \otimes (\color{blue}{3} \oplus 4) \\
&\rightarrow (1 \oplus \color{blue}{2}) \otimes (\color{blue}{3} \oplus 4) \\
&\rightarrow (1 \oplus \color{blue}{2}) \otimes (\color{blue}{3} \oplus \color{blue}{4}) \\
&\rightarrow (1 \oplus \color{blue}{2}) \otimes \color{blue}{7} \\
&\rightarrow (\color{blue}{1} \oplus \color{blue}{2}) \otimes \color{blue}{7} \\
&\rightarrow \color{blue}{3} \otimes \color{blue}{7} \\
&\rightarrow \color{blue}{21} \\
&\not\rightarrow
\end{aligned}
$$

$$(1 \oplus 2) \otimes (3 \oplus 4) \rightarrow^* \color{blue}{21}$$

Confluenza?

Abbiamo visto che ci sono molte sequenze di valutazione diverse (non determinismo)

Abbiamo la garanzia che portino tutti allo stesso risultato?

Possiamo cambiare le regole di inferenza per imporre una specifica strategia di valutazione (determinismo)

Per esempio, possiamo imporre una valutazione da sinistra a destra degli argomenti cambiando le regole (sumR) e (prodR)

Strategia di valutazione

$$(num)\ \frac{}{\mathsf{N} \to n}$$

$$(sum)\ \frac{}{n_0 \oplus n_1 \to n}\ n = n_0 + n_1$$

$$(sumL)\ \frac{\mathsf{E}_0 \to \mathsf{E}'_0}{\mathsf{E}_0 \oplus \mathsf{E}_1 \to \mathsf{E}'_0 \oplus \mathsf{E}_1}$$

$$(sumR)\ \frac{\mathsf{E}_1 \to \mathsf{E}'_1}{n_0 \oplus \mathsf{E}_1 \to n_0 \oplus \mathsf{E}'_1}$$

$$(prod)\ \frac{}{}$$

$$(prodL)\ \frac{}{}$$

$$(prodR)\ \frac{}{}$$

Una computazione

$$(1 \oplus 2) \otimes (3 \oplus 4) \rightarrow (1 \oplus 2) \otimes (3 \oplus 4)$$
$$\rightarrow (1 \oplus 2) \otimes (3 \oplus 4)$$
$$\rightarrow 3 \otimes (3 \oplus 4)$$

Ora e' unica!

$$\rightarrow 3 \otimes (3 \oplus 4)$$
$$\rightarrow 3 \otimes (3 \oplus 4)$$
$$\rightarrow 3 \otimes 7$$
$$\rightarrow 21$$
$$\not\rightarrow$$

$$(1 \oplus 2) \otimes (3 \oplus 4) \rightarrow^* 21$$

Semantica big-step

un passo $\mathrm{E}_0 \longrightarrow n$

rappresenta tutta una compitazione!

Come definire la relazione di transizione?

Di solito più semplice delle regole della small-step

Può corrispondere ad un interprete efficiente

SOS rules

$$(num)\ \frac{}{\mathsf{N} \longrightarrow n}$$

$$(sum)\ \frac{\mathsf{E}_0 \longrightarrow n_0 \quad \mathsf{E}_1 \longrightarrow n_1}{\mathsf{E}_0 \oplus \mathsf{E}_1 \longrightarrow n}\ n = n_0 + n_1$$

$$(prod)\ \frac{}{}$$

Una derivazione

$$\dfrac{\dfrac{(num)\dfrac{}{1 \longrightarrow 1} \quad (num)\dfrac{}{2 \longrightarrow 2}}{(sum)\dfrac{}{(1 \oplus 2) \longrightarrow 3}} \quad \dfrac{(num)\dfrac{}{3 \longrightarrow 3} \quad (num)\dfrac{}{4 \longrightarrow 4}}{(sum)\dfrac{}{(3 \oplus 4) \longrightarrow 7}}}{(prod)\dfrac{}{(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21}}$$

non può esprimere calcoli non terminanti

(le derivazioni sono possibili solo per i programmi che terminano)

Semantica denotazionale

un dominio + una funzione di interpretazione

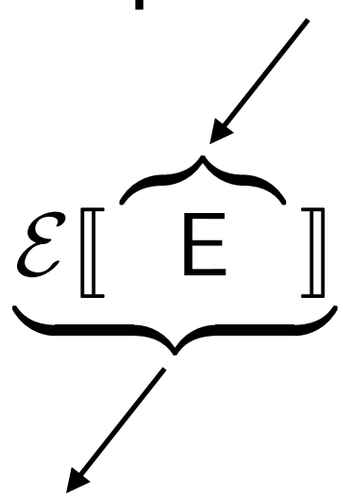
$\mathbb{N}$ $\quad\quad$ $\mathcal{E}[\![\cdot]\!] : Exp \to \mathbb{N}$

la scelta del dominio ha conseguenze immediate: chiunque sa già che le espressioni sono deterministiche e normalizzatili

ogni espressione ha al massimo una risposta

ogni espressione ha una risposta

un termine
(un pezzo di sintassi)



la sua denotazione
(un oggetto semantico)

categorie sintattiche diverse possono richiedere domini diversi!

Induzione strutturale

$$\mathcal{E}[\![\mathsf{N}]\!] = n$$

$$\mathcal{E}[\![\mathsf{E}_0 \oplus \mathsf{E}_1]\!] = \mathcal{E}[\![\mathsf{E}_0]\!] + \mathcal{E}[\![\mathsf{E}_1]\!]$$

$$\mathcal{E}[\![\mathsf{E}_0 \otimes \mathsf{E}_1]\!] = \mathcal{E}[\![\mathsf{E}_0]\!] \cdot \mathcal{E}[\![\mathsf{E}_1]\!]$$

principio di composizionalita'

il significato di un costrutto composto non dipende dalla forma particolare dei costrutti costituenti, ma solo dai loro significati

Una valutazione

$$\begin{aligned}\mathcal{E}[\![(1 \oplus 2) \otimes (3 \oplus 4)]\!] &= \mathcal{E}[\![1 \oplus 2]\!] \cdot \mathcal{E}[\![3 \oplus 4]\!] \\ &= (\mathcal{E}[\![1]\!] + \mathcal{E}[\![2]\!]) \cdot (\mathcal{E}[\![3]\!] + \mathcal{E}[\![4]\!]) \\ &= (1 + 2) \cdot (3 + 4) \\ &= 21\end{aligned}$$

Confronto

$\mathsf{E} \to^* n$ $\quad\quad$ $\mathsf{E} \longrightarrow n$ $\quad\quad$ $\mathcal{E}[\![\mathsf{E}]\!] = n$

normalizzazione / terminazione?

$\forall \mathsf{E}.\ \exists n.\ \mathsf{E} \to^* n$ $\quad\quad$ $\forall \mathsf{E}.\ \exists n.\ \mathsf{E} \longrightarrow n$ $\quad\quad$ $\forall \mathsf{E}.\ \exists n.\ \mathcal{E}[\![\mathsf{E}]\!] = n$

da dimostrare $\quad\quad$ da dimostrare $\quad\quad$ ovvio

Confronto

determininismo?

$$\forall \mathsf{E}, n, m. \left(\begin{array}{l} \mathsf{E} \rightarrow^* n \\ \wedge \\ \mathsf{E} \rightarrow^* m \end{array} \right) \Rightarrow n = m$$

da dimostrare

$$\forall \mathsf{E}, n, m. \left(\begin{array}{l} \mathsf{E} \longrightarrow n \\ \wedge \\ \mathsf{E} \longrightarrow m \end{array} \right) \Rightarrow n = m$$

da dimostrare

$$\forall \mathsf{E}, n, m. \left(\begin{array}{l} \mathcal{E}[\![\mathsf{E}]\!] = n \\ \wedge \\ \mathcal{E}[\![\mathsf{E}]\!] = m \end{array} \right) \Rightarrow n = m$$

ovvio

Confronto

$\mathsf{E} \rightarrow^* n$ $\qquad$ $\mathsf{E} \longrightarrow n$ $\qquad$ $\mathcal{E}[\![\mathsf{E}]\!] = n$

consistenza?

$$\forall \mathsf{E}, n.\ (\mathsf{E} \rightarrow^* n \quad \Leftrightarrow \quad \mathsf{E} \longrightarrow n \quad \Leftrightarrow \quad \mathcal{E}[\![\mathsf{E}]\!] = n)$$

Da dimostrare

Confronto

equivalenze indotte

$$\mathsf{E}_0 \equiv_s \mathsf{E}_1$$
$$\forall n.\ (\mathsf{E}_0 \to^* n \Leftrightarrow \mathsf{E}_1 \to^* n)$$

$$\mathsf{E}_0 \equiv_b \mathsf{E}_1$$
$$\forall n.\ (\mathsf{E}_0 \longrightarrow n \Leftrightarrow \mathsf{E}_1 \longrightarrow n)$$

$$\mathsf{E}_0 \equiv_d \mathsf{E}_1$$
$$\mathcal{E}[\![\mathsf{E}_0]\!] = \mathcal{E}[\![\mathsf{E}_1]\!]$$

coincidono?

Confronto

possiamo provare che sono vere/false:

proprieta’ di specifiche espressioni

$$2 \otimes 6 \equiv_s 3 \otimes 4$$

proprieta’ di espressioni generiche

$$\forall \mathsf{E}, \mathsf{E}_1, \mathsf{E}_2.\ \mathsf{E} \otimes (\mathsf{E}_1 \oplus \mathsf{E}_2) \equiv_d (\mathsf{E} \otimes \mathsf{E}_1) \oplus (\mathsf{E} \otimes \mathsf{E}_2)$$

Esercizio da consegnare

Espressioni con variabili

$$\mathsf{E} ::= x \mid \mathsf{N} \mid \mathsf{E} \oplus \mathsf{E} \mid \mathsf{E} \otimes \mathsf{E}$$

Come valutare espressioni come $(x \oplus 4) \otimes y$?

Abbiamo bisogno della memoria $\mathbb{M} \triangleq \{\sigma \mid \sigma : X \to \mathbb{N}\}$

Gli stati della macchina $\langle \mathsf{E}, \sigma \rangle$

Funzione di interpretazione $\mathcal{E}[\![\cdot]\!] : Exp \to (\mathbb{M} \to \mathbb{N})$

Ridefinite le varie semantiche