1 Introduzione alla semantica

Qual è il significato di un programma? Quando scriviamo un programma, utilizziamo una sequenza di caratteri da rappresentare

il programma. Ma questa sintassi è proprio il modo in cui rappresentiamo il programma: non è ciò che il programma significa.

Forse potremmo definire il significato di un programma come qualunque cosa accada quando eseguiamo il programma (magari usando un interprete o compilandolo prima). Ma possiamo avere bug negli interpreti e

compilatori! Cioè, un interprete o un compilatore potrebbe non riflettere accuratamente il significato di un programma. Quindi noi

deve cercare altrove una definizione di cosa significhi un programma.

Un posto dove cercare il significato di un programma è nel manuale delle specifiche del linguaggio. Tali manuali

in genere forniscono una descrizione informale dei costrutti linguistici.

Un'altra opzione è quella di dare una definizione formale e matematica della semantica del linguaggio. Una definizione matematica formale può avere i seguenti vantaggi rispetto a una descrizione informale.

• Meno ambiguo. Il comportamento della lingua è più chiaro, utile per chiunque ne abbia bisogno

scrivere programmi nella lingua, implementare un compilatore o interprete per la lingua, aggiungerne uno nuovo

caratteristica alla lingua, ecc.

• Più conciso. Concetti e notazioni matematiche possono descrivere in modo chiaro e conciso un linguaggio, e

restrizioni statali sui programmi legali nella lingua. Ad esempio, la specifica del linguaggio Java (2a

edizione) dedica un capitolo (26 pagine) alla descrizione del concetto di assegnazione definita, la maggior parte del quale lo è

descrivendo, in inglese, un'analisi del flusso di dati che può essere espressa in modo più succinto usando la matematica.

• Argomenti formali. Ancora più importante, una semantica formale ci consente di affermare e dimostrare il programma

proprietà che ci interessano. Ad esempio: possiamo affermare e dimostrare che tutti i programmi in una lingua sono garantiti privi di determinati errori di runtime o privi di determinate violazioni della sicurezza; noi

può dichiarare la specifica di un programma e dimostrare che il programma soddisfa la specifica (cioè, che

il programma è garantito per produrre l'output corretto per tutti i possibili input).

Tuttavia, lo svantaggio della semantica formale è che possono portare a modelli matematici abbastanza complessi,

soprattutto se si tenta di descrivere tutti i dettagli in un linguaggio moderno completo. Poca programmazione reale

le lingue hanno una semantica formale, poiché modellare tutti i dettagli di una lingua del mondo reale è difficile: reale

le lingue sono complesse, con molte caratteristiche. Per descrivere queste caratteristiche, sia la matematica che

la notazione per la modellazione può diventare molto densa, rendendo difficile la comprensione della semantica formale. Infatti,

a volte è necessario sviluppare nuove matematiche e notazioni per modellare accuratamente le caratteristiche del linguaggio.

Quindi, sebbene possano esserci molti vantaggi nell'avere una semantica formale per un linguaggio di programmazione, non lo fanno

tuttavia superano i costi di sviluppo e utilizzo di una semantica formale per i veri linguaggi di programmazione.

Esistono tre approcci principali per specificare formalmente la semantica dei linguaggi di programmazione:

• semantica operativa: descrive come un programma verrebbe eseguito su una macchina astratta;

• semantica denotazionale: modella i programmi come funzioni matematiche;

• Semantica assiomatica: definisce il comportamento del programma in termini di formule logiche soddisfatte

prima e dopo un programma;

Ciascuno di questi approcci presenta vantaggi e svantaggi diversi in termini di modalità matematica

sono sofisticati, quanto sia facile usarli nelle bozze o quanto sia facile implementare un interprete o

compilatore basato su di essi.

2 Un semplice linguaggio di espressioni aritmetiche

Per comprendere alcuni dei concetti chiave della semantica, inizieremo con un linguaggio molto semplice di interi

espressioni aritmetiche, con assegnazione. Un programma in questo linguaggio è un'espressione; esecuzione di un programma

significa valutare l'espressione in un numero intero.

Per descrivere la struttura di questo linguaggio utilizzeremo i seguenti domini:

x, y, z ∈ Var

n, m ∈ Int

e ∈ Esp

Var è l'insieme delle variabili del programma (ad es. foo, bar, baz, i, ecc.). Int è l'insieme di numeri interi costanti (ad esempio, 42,

−40, 7). Exp è il dominio delle espressioni, che specifichiamo usando una grammatica BNF (Backus-Naur Form):

e ::= x | n | e1 + e2 | e1 × e2 | x := e1; e2

Informalmente, l'espressione x := e1; e2 significa che a x viene assegnato il valore di e1 prima di valutare e2. Il

il risultato dell'intera espressione è quello di e2.

Questa grammatica specifica la sintassi per la lingua. Un problema immediato qui è che la grammatica è

ambiguo. Considera l'espressione 1 + 2 × 3. Si possono costruire due alberi sintattici astratti:

+

/ \

1 \*

/ \

2 3

\*

/ \

+ 3

/ \

1 2

Ci sono diversi modi per affrontare questo problema. Uno è riscrivere la grammatica per la stessa lingua

per renderlo inequivocabile. Ma questo rende la grammatica più complessa e più difficile da capire. Un altro

la possibilità è di estendere la sintassi per richiedere parentesi attorno a tutte le espressioni:

x | n | (e1 + e2) | (e1 × e2) | x := e1; e2

Tuttavia, questo porta anche a disordine e complessità inutili.

Invece, separiamo la "sintassi concreta" del linguaggio (che specifica come inequivocabilmente

analizzare una stringa in frasi di programma) dalla "sintassi astratta" del linguaggio (che descrive, possibilmente

ambiguamente, la struttura delle frasi di programma). In questo corso useremo la sintassi astratta e assumeremo

che l'albero della sintassi astratto sia noto. Quando scriviamo espressioni, useremo occasionalmente le parentesi

indicano la struttura dell'albero sintattico astratto, ma le parentesi non fanno parte del linguaggio stesso.

(Per i dettagli sull'analisi, la grammatica e l'eliminazione dell'ambiguità, vedere o seguire il corso per compilatori Computer

Scienza 153.)