Applicazioni alla traduzione e compilazione

a.a. 2020-2021

Corso di Fondamenti di Informatica - 1 modulo

Corso di Laurea in Informatica

Università di Roma "Tor Vergata"

Prof. Giorgio Gambosi

Fasi del processo di compilazione

1. Analisi lessicale. La sequenza di caratteri viene suddivisa in token, elementi significanti (come simboli terminali) nella definizione della sintassi del linguaggio. Costruzione della tabella dei simboli
2. Analisi sintattica. La sequenza di terminali viene esaminata per derivare la struttura sintattica del programma (albero sintattico, derivazione destra, derivazione sinistra)
3. Generazione del codice intermedio. A partire dalla struttura sintattica del programma e dalla tabella dei simboli, viene generata una prima versione tradotta del programma, in un linguaggio intermedio. Vengono effettuati una serie di controlli su aspetti non context-free del codice, come ad esempio il type checking.
4. Generazione codice finale e ottimizzazione.

Fasi del processo di compilazione

Le prime due fasi sono basate sulla teoria dei linguaggi e degli automi.

- Analisi lessicale. Basata sui linguaggi di tipo 3: la struttura dei vari tipi di token è definita mediante espressioni regolari. Idealmente, l'analisi è effettuata esaminando la sequenza di caratteri per mezzo di tanti ASFD quanti sono i tipi di token.
- Analisi sintattica. Basata sui linguaggi context free: la struttura del linguaggio è definita mediante una grammatica CF. L'analisi è effettuata applicando un opportuno automa a pila per decidere la correttezza del programma (derivabilità nella grammatica del linguaggio) e, in tal caso, il corrispondente albero sintattico (o derivazione destra, o derivazione sinistra).

Analisi lessicale

Basata sulla definizione di un insieme di pattern, mediante espressioni regolari. Ad esempio $[a-z, A-Z][a-z, A-Z, 0-9, .]^*$ può corrispondere alla struttura di un identificatore alfanumerico nel linguaggio, *for* alla parola chiave `for`

Un lessema è sequenza di caratteri che corrisponde a un pattern. Ad esempio 'ao_z' o 'my_counter' per il primo caso, o 'for' per il secondo

L'analizzatore lessicale associa un token (associato al pattern) ad ogni lessema: la sequenza di caratteri viene sostituita da una sequenza di token

Esempio

pattern	token
$[a-z, A-Z][a-z, A-Z, 0-9, \_, .]^*$	`id`
$0 + [1-9][0-9]^*$	`digit`
$==, <, <=, >, >=$	`rel_op`
$=$	`assign`
for	`for`
if	`if`
else	`else`

if a0 > my_counter i=3 else j=0

`if id rel_op id assign digit else id assign digit`

Tabella dei simboli

L'analizzatore lessicale costruisce una tabella dei simboli incontrati, che associa ad ogni token (almeno quelli per i quali è necessario) il corrispondente lessema, insieme eventualmente ad altre informazioni (ad es. il tipo)

L'analisi sintattica fa riferimento ai soli token, mentre la generazione del codice successiva utilizza valori derivati dai lessemi.

Analizzatore lessicale

- Simula un ASF per ogni espressione regolare definita (corrispondente a un pattern e un token)
- Quando un ASF accetta la stringa letta fino ad ora, restituisce il token corrispondente

Dato un insieme di espressioni regolari, è possibile derivare in modo automatico l'insieme di ASF che le accettano, e quindi l'analizzatore lessicale corrispondente: Generatori di analizzatori lessicali

Analisi sintattica

Analizzatore sintattico, o parser:

- è associato ad una grammatica context free non ambigua, che definisce la struttura sintattica del linguaggio utilizzato
- riceve in input una stringa di token (simboli terminali della grammatica)
- verifica se la stringa è derivabile nella grammatica
- in tal caso, restituisce l'unico albero sintattico associato alla stringa

Analisi sintattica

Un parser opera esaminando la stringa da sinistra a destra, in due possibili modi

- in funzione dei primi simboli letti, predice quale è la prossima produzione utilizzata in una derivazione sinistra della stringa, se esiste; l'albero sintattico viene prodotto dall'alto verso il basso (parsing predittivo o top-down)
- in funzione dei primi simboli letti, predice quale produzione è stata utilizzata per produrli, sostituendoli nella stringa con la parte sinistra della produzione: iterando questa operazione, si produce, se esiste, la produzione destra che genera la stringa, al contrario; l'albero sintattico viene prodotto dal basso verso l'alto (bottom-up parsing)

Analisi sintattica

Predizioni non guidate in nessun modo delle produzioni utilizzate ad ogni passo (e quindi delle derivazioni) equivalgono alla simulazione di un NPDA, e comportano tempi di parsing esponenziali nella lunghezza della stringa.

E' necessario prevedere strutture particolari delle grammatiche CF utilizzate, che consentano di utilizzare parser che operano in modo più efficiente, guidati al meglio dai simboli letti