Programma di Paradigmi e possibili domande

Capitolo 1

Definizione di macchina astratta

Una macchina astratta per il linguaggio L detta ML, è un qualsiasi insieme di algoritmi e strutture dati che permettono di eseguire e memorizzare programmi scritti in L.

Capitolo 2

Grammatica libera

Una grammatica libera da contesto è una quadrupla formata da NT (insieme di simboli non terminali) T (insieme di simboli terminali) S (simbolo non terminale iniziale) e R (insieme di produzioni).

Produzioni

Una produzione della forma $V \rightarrow w$ indica una regola che definisce il passaggio da un simbolo non terminale ad una stringa formata da un serie di caratteri terminali (eventualmente anche nessuno) ed una serie di simboli non terminali (eventualmente anche nessuno).

Derivazione

Data una grammatica $G = \{NT, T, S, R\}$ si suppone di avere $v \in NT \cup T$, dunque possiamo dire che da v si deriva w se sostituendo le occorrenze dei simboli terminali presenti nella stringa v si ottiene la stringa w. La stringa w sarà anch'essa composta dai simboli dati da $NT \cup T$ ma potrebbe anche coincidere con una stringa composta dai soli simboli T. La derivazione citata si scrive in $v \rightarrow w$.

Linguaggio Generato

Questo linguaggio L(G), equivale a $\{w \in T^* \mid S \Rightarrow^* w\}$ con w che rappresenta una qualsiasi stringa ottenuta con i simboli T.

Grammatica Ambigua

Una grammatica si definisce ambigua quando ammette per una stringa due o più alberi di derivazione: la compilazione deve essere unica, non devono esserci due modi distinti per compilare.

Stato

In un preciso momento ci si può riferire ad uno stato intendendo una sequenza finita di coppie della forma (X, n) in cui X delimita il nome di una variabile mentre x denota il relativo valore.

Capitolo 3

Ambiente

Per ambiente s'intende l'insieme delle associazioni fra nomi e oggetti denotabili esistenti a run-time o in uno specifico momento, oppure in uno specifico punto (testuale) del programma.

Oggetto denotabile

Un stringa, o meglio, un'insieme di caratteri, può rappresentare un oggetto. In realtà se un nome viene usato per riferirsi ad un dato oggetto, possiamo richiamare tale oggetto facendo sempre riferimento allo stesso nome.

Blocco

Un blocco è una regione testuale del programma delimitata da un segno d'inizio ed uno di fine. Nel suo corpo possono essere contenute dichiarazioni locali. Tali blocchi possono essere associati a procedure, nel qual caso esteso con i parametri formali. Un blocco potrebbe essere anche anonimo, non associato ad una procedura il quale può essere richiamato per esempio con il costrutto GO-TO.

Tipi d'Ambiente

L'ambiente associato ad un blocco è composto da Ambiente Globale, Ambiente Non Locale e Ambiente Locale. L'ambiente globale è costituito dalle associazioni crete all'inizio del programma che risultano visibili a tutti i blocchi presenti. L'ambiente non locale indica tutte le associazioni che sono visibili ad un blocco ma che non sono dichiarate in esso. Infine l'ambiente locale indica l'insieme delle associazioni dichiarate localmente.

Operazioni in entrata/uscita di un blocco

Creazione di associazione tra nome ed oggetto denotato (parametro formale, parametro attuale).
Riferimento ad un oggetto denotato mediante il suo nome.
Disattivazione di un associazione non locale per una nuova associazione locale. L'associazione precedente non viene distrutta ma disattivata.
Riattivazione dell'associazione disattivata al momento dell'uscita dal blocco.
Distruzione delle associazioni create localmente al momento dell'uscita dal blocco.

Scope Statico

Per scope statico s'intende che per cercare il valore di una variabile x non dichiarata localmente, si dovrà accedere al blocco immediatamente esterno al blocco corrente e così via fino ad ottenere il valore corretto di x. Ovviamente è previsto un annidamento dei blocchi semanticamente corretto.

Scopo Dinamico

Al contrario per quanto avviene per lo scope statico il valore di una variabile x non dichiarata localmente è il valore più recente assegnatoli in senso temporale.

Capitolo 5

Gestione Statica

La memoria che viene allocata staticamente è quella allocata dal compilatore prima dell'esecuzione. Gli oggetti per i quali la memoria è allocata staticamente, durante l'esecuzione del programma risiederanno sempre nella stessa locazione di memoria (è il caso delle variabili globali ad esempio). Nel caso in cui il linguaggio non supporti ricorsione la memoria più essere allocata staticamente per tutti gli oggetti denotabili.

Gestione Dinamica mediante PILA

In questo caso in fase di compilazione non viene allocato niente, la memoria è gestita dinamicamente, ossia a run-time. Sostanzialmente nel caso in cui si entri nel blocco A, con un'operazione di PUSH allochiamo spazio in memoria per il blocco A, procedimento inverso avviene all'uscita dal blocco A, ovvero tramite un operazione di POP viene deallocato lo spazio di memoria precedentemente assegnato ad A.

RDA

E' lo spazio di memoria allocato sulla PILA dedicato ad un blocco. Al suo interno troviamo diversi settori per la memorizzazione di RISULTATI INTERMEDI, VARIABILI LOCALI, PUNTATORE CATENA STATICA, PUNTATORE CATENA DINAMICA, INDIRIZZO DI RITORNO, INDIRIZZO DEL RISULTATO e PARAMETRI.

Blocco in-line

Il blocco in-line prevede che l'RDA relativo sia composto da PUNTATORE CATENA DINAMICA, VARIABILI LOCALI e RISULTATI INTERMEDI.

Blocchi Procedura/Funzione

Questi blocchi sono rappresentati mediante RDA costituiti oltre che dai settori menzionati nei blocchi in-line da PUNTATORE CATENA STATICA, INDIRIZZO DI RITORNO, INDIRIZZO RISULTATO, PARAMETRI.

Gestione della PILA

Entrata
- Modifica Program Counter per passare il controllo alla procedura chiamata.
- Allocazione spazio sulla PILA.
- Modifica puntatore al RDA (frame pointer).
- Passaggio dei parametri.
- Salvataggio dei registri (vecchio puntatore RDA, operazioni di controllo).
- Esecuzione codice inizializzazione(solo alcuni linguaggi).

Uscita
- Ripristino valore Program Counter.
- Restituzione valori.
- Ripristino dei registri.
- Esecuzione codice finalizzazione (solo alcuni linguaggi).
- Deallocazione dello spazio dalla PILA.

Gestione dinamica con HEAP

Nel caso in cui il linguaggio di programmazione fornisca dei costrutti per l'allocazione e deallocazione dello spazio di memoria, è necessario gestire la memoria mediante un HEAP.

Heap a blocchi di dimensione fissa

Nonostante questo tipo di gestione dello HEAP sia estremamente semplice, si verificano problematiche inerenti alla frammentazione sia interna che esterna.

Heap a blocchi di dimensione variabile

Con questa tecnica si nota che il fenomeno della frammentazione interna si riduca rispetto ai blocchi di dimensione fissa. Permane il problema della frammentazione esterna.

Heap con unica lista libera

Possiamo considerare lo HEAP come un'unica lista libera che non è inizialmente suddivisa in nessun blocco. Alla richiesta di allocazione di un blocco di dimensione N, si procede con la ricerca con first-fit o best-fit. Con il first-fit viene cercato il primo blocco di dimensione maggiore o uguale a N mentre nel secondo (più costoso) viene cercato tra i blocchi di dimensione maggiore o uguale a N il blocco di dimensioni minore. In alcuni casi (i blocchi devono essere spostabili) si presenta la possibilità di ricompattare la memoria.

Heap con liste libere multiple

Questa tecnica è usata per ridurre il costo di allocazione di un blocco: in tal maniera le liste contengono blocchi di dimensioni diverse. Nel caso in cui i blocchi siano di dimensione variabile ci si riferisce a queste liste come Buddy-List o Fibonacci-List. Sostanzialmente le liste sono suddivise in blocchi di dimensione pari alla potenza di 2 (Buddy-List) e l'allocazione di un blocco di dimensione 24 viene cercato il blocco di dimensione $2^n \geq 24$, ossia 5. Se tale blocco di dimensione 32 è libero questo viene allocato, altrimenti si cerca il blocco di dimensione 2^{n+1} e poi tale blocco viene suddiviso in 2 ed il blocco non utilizzato viene restituito alla lista libera. Successivamente, al momento della deallocazione il blocco si cerca la parte divisa precedentemente (buddy in inglese) e la si ricompatta e la si restituisce alla lista libera. La serie di Fibonacci crescendo più lentamente è un'alternativa per la dimensione dei blocchi in potenze di due che risulta più efficiente per la frammentazione interna.

Regole d'implementazione

Scope Statico

Una volta compilato il programma ci si riconduce ad una struttura che rappresenta l'annidamento dei blocchi: ciò è possibile poiché testualmente si può calcolare il livello di annidamento di ogni blocco. Una volta in possesso di tali informazioni si procede con la realizzazione di un DISPLAY che mostra il modo con il quale una variabile dichiarata non localmente sia visibile ad un blocco di un determinato livello. Per regola, all'entrata di un blocco, l'ambiente non locale viene definito come l'insieme delle associazioni tra nomi e valori ereditate al livello di annidamento superiore. Il costo dell'accesso quindi ad una variabile non locale è proporzionale al numero di blocchi compresi testualmente tra il blocco corrente ed il blocco che contiene la dichiarazione.

Scope Dinamico

Tramite una pila ovviamente è semplice gestire le regole di visibilità. La tecnica per rappresentare questa situazione consiste nella CRT. La CRT è implementabile con una lista che contiene tutte le variabili presenti nel testo del programma, ad ognuna di esse è collegata un'ulteriore lista che rappresenta tutte le associazioni presente nei vari blocchi in senso temporale ovviamente. Il tempo di accesso ad una variabile è costante, poiché ad ogni variabile è associato il relativo valore, ossia l'ultimo assegnamento in senso temporale.

Capitolo 6

Espressione

A basso a livello gestire l'ordine di sequenza delle procedure o funzioni è implementato tramite PC. Nell'ambito della programmazione ad alto livello ovviamente è opportuno gestirle in modo diverso. L'espressione è un' entità sintattica la cui valutazione produce un valore oppure non termina, nel qual caso è detta indefinita.
Può presentarsi in notazione Infissa, Postfissa e Prefissa. La prima, infissa, è necessario introdurre simboli terminali di precedenza affinché non vi sia ambiguità,. Quella prefissa, detta anche polacca, antepone l'operatore agli operandi, non vi sono regole di precedenza e non ammette ambiguità ma purché sia noto il numero di operandi per ogni operatore. La notazione postfissa è analoga a quella precedente ma inversa.

Valutazione Espressione

Un' espressione può essere rappresentata sotto forma di albero:

- ogni nodo non foglia è un operatore.
- ogni sotto-albero di N è un operando per N.
- ogni foglia è un operando.

Le espressioni possono essere valutate da sinistra a destra o viceversa.
Eager e Lazy sono due tecniche di valutazione degli operandi:

- la prima valuta prima tutti gli operandi e poi applica l'operatore ai valori ottenuti.
- la seconda non valuta tutti gli operandi prima del relativo operatore, il quale deciderà quali di questi sono effettivamente necessari (a == b ? b : b/a).

Un altro tipo di valutazione è quella corto circuito, che prevede che ad esempio nel caso di espressioni booleane, si valuti il primo dei due operandi ed in base al suo valore decide se valutare anche il secondo oppure dedurre automaticamente il risultato dell'espressione (a || b, a && b).

Comando

E' un'entità sintattica la cui valutazione non restituisce necessariamente un valore ma può avere un effetto collaterale (cambiamento dello stato).
Le tipologie di comando accettate sono: adibite al controllo di sequenza (goto, return, break), a salti condizionali (if-else, switch-case) e alle iterazioni (for, while, for-each).

Variabile

E' un oggetto denotato da un nome (sequenza di caratteri) il cui contenuto può assumere un valore di svariati tipi appartenenti ad un insieme prefissato.

Assegnamento

E' il comando di base, che permette di modificare il valore della variabile. E' un effetto collaterale dato che modifica lo stato e allo stesso tempo non restituisce nessun valore.

Programmazione strutturata