ESEMPI DI DOMANDE

Enunciare e discutere la tesi di Church-Turing, chiarendo perché viene formulata come tesi e non come teorema.

Definire le classi: P, NP e indicare le relazioni di contenimento fra le classi

Definire la classe dei problemi NP-completi e indicare le relazioni di contenimento rispetto alle classi P e NP

Indicare un possibile risultato che permetterebbe di stabilire che $P \neq NP$

Indicare un possibile risultato che permetterebbe di stabilire che $P = NP$

Per dimostrare che un problema A è NP-completo cosa è necessario dimostrare?

Definire il problema della soddisfacibilità di una formula logica e cosa è una formula in forma normale congiuntiva (CNF). A quale classe di complessità temporale appartiene il problema della soddisfacibilità di una formula CNF?

Definire il problema della soddisfacibilità di una formula logica e cosa è una formula in forma normale congiuntiva (DNF). A quale classe di complessità temporale appartiene il problema della soddisfacibilità di una formula DNF?

Nel caso di problemi NP-completi non rinunciamo a cercare soluzioni. Illustrare un possibile approccio per il problema della soddisfacibilità di formule logiche.

Nel caso di problemi di ottimizzazione che sono difficili (cioè la versione come problema di riconoscimento è NP-completo) non rinunciamo a cercare soluzioni buone anche se non sempre ottime. Illustrare un possibile approccio per il problema per un problema a vostra scelta.

Illustrare una riduzione – a vostra scelta - che mostra che un problema A è NP-completo.

Per mostrare che un problema B è NP-difficile è sufficiente fornire una riduzione da un problema A a B. Quali proprietà deve verificare A e quali deve verificare la riduzione?

Descrivere le fasi di analisi lessicale e di analisi sintattica di un compilatore; per ciascuna delle due fasi descrivi l'input e l'output e come viene elaborato il programma.

Illustrare la gerarchia di Chomsky delle grammatiche; indicare per ciascuna categoria la proprietà/caratteristica a vostro giudizio maggiormente rilevante.

Descrivi l'algoritmo per eliminare le ricorsioni sinistre di una grammatica di tipo 2.

Descrivere un analizzatore sintattico di tipo top-down

Quali sono le proprietà di una grammatica LL(1) e per quale ragione sono la migliore scelta per descrivere i linguaggi di programmazione.

ESERCIZI

Grammatiche regolari, espressioni regolari e automi a stati finiti

Si considerino le seguenti espressioni regolari sull'alfabeto {a,b,c}:

(1) R1= a (b | aa)* c
(2) R2= abb (a | cc) (ab)*

Per ciascuna costruire un automa a stati finiti deterministico o nondeterministico che riconosce il linguaggio definito dall'espressione regolare.

Data una Grammatica regolare G, è possibile stabilire se G genera il linguaggio vuoto? Come?

Data una Grammatica regolare G, è possibile stabilire se G genera un linguaggio finito o infinito? Come?

(Sugg. Per i due esercizi precedenti utilizzare il fatto che per ogni grammatica regolare G esiste un automa a stati finiti che riconosce tutte e sole le stringhe che appartengono a G)

Alice e Biagio discutono animatamente. Stanno esaminando una grammatica regolare G in cui V_T è l'insieme dei terminali, V_N è l'insieme dei non terminali, S è l'assioma e P l'insieme delle produzioni (di tipo 3). Carla sta per fornire loro un ulteriore insieme di produzioni P', sempre di tipo 3, che tuttavia Alice e Bob non hanno ancora visto. Carla ha avvisato che P è anch'esso basato su V_T e V_N e che chiederà ad Alice e Biagio di esaminare la grammatica regolare G' in cui V_T è l'insieme dei terminali, V_N è l'insieme dei non terminali, S è l'assioma e l'insieme delle produzioni (di tipo 3) è l'unione di P e P'.
Siano L(G) e L(G') i linguaggi generati da G e G'. Alice sostiene che $L(G) \subseteq L(G')$ poiché l'introduzione di nuove produzioni potrebbe consentire di generare nuove stringhe, mentre Biagio pensa che $L(G) \supseteq L(G')$, dato che le nuove produzioni introdurranno restrizioni sulle possibilità di generazione.

Per i seguenti linguaggi definisci un'espressione regolare o una grammatica regolare che li genera:
(1) l'insieme delle stringhe binarie che contengono le stringhe 101 o 010 (o ambedue) come sottostringhe.
(2) l'insieme delle stringhe binarie che contengono le stringhe 00 e 11 come sottostringhe.
(3) l'insieme delle stringhe binarie che contengono la stringa 00 ma non la stringa 11 come sottostringhe.
(4) l'insieme delle stringhe binarie che contengono iniziano con 00 e finiscono con 11.
(5) l'insieme delle stringhe sull'alfabeto {x,y,z} in cui ogni x è immediatamente seguita da una y.
(6) l'insieme delle stringhe sull'alfabeto {x,y,z} che contengono un numero dispari di y.

Sono dati M1 e M2, due automi a stati finiti che riconoscono i linguaggi L1 e L2 rispettivamente;
(1) definire un automa M che riconosce il linguaggio $L1 \cup L2$ (formato dalle stringhe cha appartengono a L1 o a L2 o a entrambi).

Grammatiche libere dal contesto

Considera la seguente grammatica G (assioma S), con simboli terminali {a, b, c}:

S→ aAb | AE
A→ aAbE | ab
B → AB |c
D→ AAcE
E → BA
F→ FA | a

(1) Identifica i simboli non terminali che non sono raggiungibili a partire dall'assioma.
(2) Trasforma la grammatica eliminando le produzioni inutili.

(Risposta (1): D,F - non esiste la possibilità di generare una stringa che contiene D o F a partire dall'assioma S; trasformandolo in una sequenza di terminali;
(2) bisogna eliminare tutte le produzioni che coinvolgono D e F)

Considera la seguente grammatica G (assioma S), con simboli terminali {a, b, c}:
S→ aABb | AC
A→ aAb | ab
B → abB
C → ABC | c

(1) Identifica i simboli non terminali che non sono utilizzati per definire stringhe del linguaggio.
(2) Trasforma la grammatica eliminando le produzioni inutili.

(Risposta (1): B (non esiste la possibilità di eliminare B trasformandolo in una sequenza di terminali; (2) bisogna eliminare tutte le produzioni che coinvolgono B)

Costruire un analizzatore sintattico predittivo a discesa ricorsiva per il linguaggio delle parentesi bilanciate, generato dalla grammatica S (assioma), simboli terminali '(' e ')' e produzioni S→(S) S | ε
E' possibile svolgere l'esercizio anche su una grammatica equivalente a quella data.

Data la grammatica per il linguaggio delle parentesi bilanciate, generato dalla grammatica S (assioma), simboli terminali '(' e ')' e produzioni S→(S) S | ε dare l'albero di derivazione per la stringa (()())()

Si consideri la grammatica S→ aSbS S→ bSaS S→λ
La grammatica è ambigua? Quanti differenti alberi di derivazione esistono per la sequenza abab ? Mostrare le derivazioni sinistre e destre.

Sia data la grammatica - assioma E, simboli nonterminali caratteri maiuscoli, simboli terminali +, -, *, /, (,), id -
E→TQ
Q→+TQ Q→ -TQ Q→ λ
T → FR
R→ *FR R→ /FR R→ λ
F→ (E) F→ id
(a) Fornire l'albero di derivazione per le stringhe
(1) id + ((id* id)- id) * id (2) ((id-id) /id) + (id –(id *id))
(b) Questa grammatica è LL(1) ma la tabella di parsing non può essere calcolata usando solo i simboli FIRST; spiegare perché.

Dimostrare che la seguente grammatica
S→ aB S→ aC S→ B B→ bB B→d C→ B
non è LL(1). Costruire una grammatica LL(1) equivalente alla precedente.
Soluz. La grammatica non è LL(1) a causa delle produzioni con S nella parte sinistra con stesso simbolo FIRST (S→ aB, S→ aC). Infatti non sappiamo scegliere quale delle due produzioni. Osserviamo che si può introdurre un nuovo non terminale T e modificare la grammatica nel seguente modo
S→ aT T→B T→ C S→ B B→ bB B→d C→ B
A questo punto è facile verificare che le produzioni T→C e C→ B equivalgono a T→ B che è già presente; quindi possono essere eliminate. In questo modo otteniamo la grammatica
S→ aT T→B S→ B B→ bB B→d

Verifichiamo che questa grammatica è LL(1); calcoliamo i simboli FIRST
FIRST(S → aT)= {a} FIRST(S→B)= {b,d} FIRST(B→bB)= {b} FIRST(B→d)= {d} FIRST(T→B)= ε
Per produzioni con stessa parte sinistra i simboli FIRST sono disgiunti e quindi la grammatica è LL(1).