

Possibili DOMANDE per l'orale (Modulo Gorrieri)

Capitolo 1 – Macchine astratte, interpreti, compilatori

1. **Che cosa è una macchina astratta?** E in cosa si differenzia da una macchina fisica?
2. **Che cos'è un interprete?** In cosa consiste il ciclo fetch-decode-execute?
3. Cos'è il linguaggio macchina?
4. Possono esistere macchine diverse con lo stesso linguaggio macchina?
5. In quali modi è possibile implementare una macchina astratta? Elencare vantaggi e svantaggi delle varie tecniche.
6. **Che cos'è un compilatore?**
7. Relativamente alla tecnica d'implementazione software, **descrivere la tecnica d'implementazione interpretativa pura e quella compilativa pura.**
8. **Quando un interprete si può dire corretto? Quando un compilatore si può dire corretto?**
9. Confrontare l'implementazione di una macchina astratta su una macchina ospite per mezzo di un interprete o di un compilatore.
10. Come vengono implementate nella realtà le macchine astratte? **Che cos'è la macchina intermedia?**
11. **Quando si dice che una implementazione è di tipo interpretativo e quando di tipo compilativo?** Fare esempi di linguaggi la cui implementazione è di un tipo o dell'altro.
12. L'interprete e il compilatore si possono sempre realizzare?
13. Che cosa è l'implementazione via kernel?
14. Quando si parla di bootstrapping?

Capitolo 2 – Descrivere un linguaggio di programmazione

15. **Quali sono i livelli di descrizione di un linguaggio?**
16. Sintassi: qual è l'aspetto lessicale e quale quello grammaticale di un linguaggio?
17. Cos'è un alfabeto? Cos'è una parola o stringa? **Cos'è A^* ?** Tale insieme è enumerabile?
18. Definizione di potenza di una stringa. Definizione di potenza di un linguaggio. Definizione di chiusura/iterazione (o stella di Kleene) di un linguaggio.
19. **Definizione di grammatica libera da contesto. Come si deriva una stringa? Qual è il linguaggio generato da una grammatica libera?**
20. **Cos'è un albero di derivazione?** Cos'è una derivazione canonica sinistra/destra? Esiste una corrispondenza biunivoca tra alberi di derivazioni e derivazioni canoniche?
21. **Quando una grammatica è ambigua? (fare un esempio) Quando un linguaggio è ambiguo? (fare un esempio)**
22. È possibile rimuovere l'ambiguità dalla grammatica delle espressioni aritmetiche? Come?
23. **Cos'è l'albero di sintassi astratta?** Che differenza c'è tra sintassi concreta e sintassi astratta? Cos'è lo zucchero sintattico?
24. **Fare esempi di vincoli sintattici contestuali.** Possono essere catturati attraverso grammatiche libere?
25. **Cosa s'intende per semantica statica? E per semantica dinamica?**
26. **Elencare le varie fasi in cui si articola un compilatore. Descrivere in dettaglio ogni singola fase.**
27. A chi serve definire la semantica di un linguaggio e perché?
28. Quali tecniche si usano per dare semantica ad un linguaggio di programmazione?

29. **Imparare le regole di semantica operativa SOS per il semplice linguaggio presentato a lezione.** Regole di valutazione interna-sinistra ed esterna-sinistra.
30. Cosa s'intende per pragmatica? Cosa si intende per implementazione di un linguaggio?

Capitolo 3 – Analisi lessicale-Linguaggi regolari

31. Cosa fa l'analizzatore lessicale?

32. Cos'è un token?
33. Cos'è un pattern? Come lo si rappresenta? Cos'è un lessema?

34. Definizione di espressioni regolari (sintassi) e di linguaggio associato (semantica).

35. **Quali sono i linguaggi regolari?** I linguaggi finiti sono tutti regolari? Esistono linguaggi infiniti regolari?
36. Definizione di equivalenza tra espressioni regolari. Elencare alcune leggi di equivalenza.
37. Cos'è una definizione regolare e a cosa serve?
38. **Definizione di NFA** (automa finito nondeterministico). Discutere cosa si intende per nondeterminismo. Mettere in relazione la definizione formale con la rappresentazione grafica come diagramma di transizioni.
39. **Come si definisce il linguaggio accettato da un NFA?** Definizione di equivalenza tra NFA.
40. **Definizione di DFA (automa finito deterministico).** Discutere cosa si intende per determinismo. Mostrare che un DFA è un caso speciale di NFA, ovvero la classe dei DFA è un sottoinsieme della classe degli NFA.
41. Dato un NFA, come si ricava un equivalente DFA? Ovvero **descrivere come è definita la costruzione per sottoinsiemi**. Definizione di epsilon-closure e algoritmo associato. Qual è la complessità della costruzione per sottoinsiemi nel caso pessimo? Ovvero se NFA ha n stati, quanti stati può avere il DFA equivalente?
42. Dato i due punti precedenti, **enunciare il teorema che dice che la classe dei linguaggi riconosciuti da NFA coincide con la classe dei linguaggi riconosciuti da DFA**.
43. **Come si costruisce un NFA a partire da una espressione regolare**, in modo tale che il linguaggio riconosciuto dall'NFA sia lo stesso del linguaggio associato all'espressione regolare?
44. **Definizione di grammatica regolare.**
45. Come si associa ad una grammatica regolare un equivalente NFA?
46. **Dato un DFA, come si costruisce una grammatica regolare equivalente?**
47. **Data una grammatica regolare, come si costruisce un'espressione regolare equivalente?**
48. **Descrivere, con un diagramma riassuntivo, tutte le relazioni fra i formalismi introdotti: NFA, DFA, grammatiche regolari, espressioni regolari.** Questo diagramma dimostra che tutti questi formalismi sono equivalenti e descrivono la classe dei linguaggi regolari.
49. **Definizione di stati equivalenti in un DFA.** Quando due stati di un DFA sono distinguibili?
50. **Come funziona l'algoritmo iterativo con tabella a scala per produrre le classi di equivalenza di stati di un DFA?**
51. Una volta determinate le classi di equivalenza degli stati di un DFA, **come si costruisce l'automa minimo associato?**
52. **Cos'è Lex? Qual è il suo input e il suo output?**
53. Qual è la struttura di un file .l? Come funziona l'analizzatore lessicale prodotto da Lex?

- 54. Come si interfaccia Lex con Yacc?
- 55. **Intestazione e dimostrazione del pumping lemma.**
- 56. **Come si può utilizzare il pumping lemma (a rovescio) per dimostrare che un linguaggio non è regolare?**
- 57. **Quali sono le proprietà di chiusura dei linguaggi regolari?** Quali proprietà si possono decidere (ovvero verificare alitmicamente)?

Capitolo 4: analisi sintattica – linguaggi liberi

- 58. Cosa si intende per analisi sintattica? **Cos'è un parser?**
- 59. **Definizione di automa a pila nondeterministico (PDA).** Definizione di configurazione (o descrizione istantanea), mossa in un passo e mossa in più passi.
- 60. **Definizione di linguaggio accettato da un PDA per stato finale o per pila vuota.** Sono equivalenti queste due modalità di riconoscimento?
- 61. **Mostrare come data una grammatica libera G, sia possibile costruire un PDA P equivalente.** È possibile costruire, dato un PDA P, una grammatica equivalente G? Concludere che la classe dei linguaggi liberi coincide con la classe dei linguaggi riconosciuti da PDA.
- 62. **Quali sono le proprietà di chiusura dei linguaggi liberi? L'intersezione di un linguaggio libero con un regolare è un linguaggio libero?**
- 63. **Intestazione e dimostrazione del "Pumping Theorem".**
- 64. **Come si può utilizzare tale teorema (a rovescio) per dimostrare che un linguaggio non è libero?**
- 65. **Classificazione di Chomsky delle grammatiche e dei linguaggi.** Definizione di grammatica dipendente dal contesto e di grammatica monotona. Quale tipo di automi corrisponde ad ogni classe?
- 66. **Definizione di DPDA (automa a pila deterministico) e di linguaggio libero deterministico.**
- 67. La classe dei linguaggi liberi deterministici è strettamente inclusa in quella dei linguaggi liberi? Contiene strettamente la classe dei linguaggi regolari?
- 68. **Che cosa dice la prefix property e perché è interessante per i DPDA?**
- 69. Usando un endmarker \$, si può riconoscere un linguaggio libero deterministico che non gode della prefix property anche per pila vuota? Come?
- 70. Un linguaggio libero deterministico è ambiguo?
- 71. Proprietà di chiusura dei linguaggi liberi deterministici: chiusi per complementazione, ma non per intersezione né per unione.
- 72. **Da cosa si parte per costruire un analizzatore sintattico (ovvero parser)? Da una espressione regolare? Da una grammatica libera? Da un PDA?**
- 73. **Cosa prende in input e cosa produce in output un parser?**
- 74. Che differenza c'è tra un parser nondeterministico ed uno deterministico?
- 75. Quali sono le due tecniche essenziali per costruire parsers?
- 76. **Le tecniche top-down e bottom-up in che cosa differiscono?**
- 77. Quali tipi di grammatiche non sono adatte al top-down parsing? Quali tipi di produzioni sono poco adatte al bottom-up parsing?
- 78. Cosa sono le produzioni epsilon e cosa sono i simboli non-terminali annullabili?
- 79. **Come si può trasformare una grammatica G che contiene produzioni epsilon in una grammatica G' che non ne contiene, preservando il linguaggio a meno di epsilon?**
- 80. Cosa sono le produzioni unitarie e cosa sono le coppie unitarie?
- 81. **Come si può trasformare una grammatica G che contiene produzioni unitarie in una equivalente G' che non ne contiene?**

82. Data una grammatica G , quali sono i suoi simboli utili? Quali sono i suoi simboli generatori? Quali i suoi simboli raggiungibili?
83. Come si calcolano i generatori? Come si calcolano i raggiungibili?
84. **In che modo si eliminano i simboli inutili di una grammatica? È importante l'ordine delle operazioni da svolgere?**
85. **Quando si dice che una grammatica è ricorsiva sinistra? Come si elimina la ricorsione sinistra immediata? E quella non immediata? Perché serve eliminare la ricorsione sinistra?**
86. **Cosa vuol dire fattorizzare a sinistra una grammatica? Perché serve fattorizzare?**
87. Cos'è un parser a discesa ricorsiva?
88. **Definizione di $\text{First}(\alpha)$ e di $\text{Follow}(A)$.**
89. **Algoritmi per calcolare $\text{First}(\alpha)$ e di $\text{Follow}(A)$.**
90. **Come è fatta e come si riempie una tabella di parsing $\text{LL}(1)$?**
91. **Quando una grammatica G si dice di classe $\text{LL}(1)$? Quali sono le condizioni necessarie e sufficienti per G affinché sia di classe $\text{LL}(1)$?**
92. **Perché un parser di questo tipo è chiamato LL ?**
93. Come funziona il parser $\text{LL}(1)$ con una pila?
94. **È vero che ogni linguaggio regolare è pure $\text{LL}(1)$?**
95. Come sono definiti $\text{First}_k(\alpha)$ e di $\text{Follow}_k(A)$ per $k \geq 2$.
96. Come si definisce una grammatica $\text{LL}(k)$? Ed un linguaggio $\text{LL}(k)$? Come si relazionano tra di loro?
97. Che relazione esiste tra grammatiche $\text{LL}(k)$ e grammatiche ambigue? E con le grammatiche ricorsive sinistre?
98. Esistono linguaggi liberi che non sono $\text{LL}(k)$ per nessun k ? Ed esistono linguaggi liberi deterministici che non sono $\text{LL}(k)$ per nessun k ?
99. **Cos'è un parser bottom-up (o shift-reduce)? Qual è il suo input e il suo output? Perché sono chiamati parser LR?**
100. Che tipo di conflitti si possono presentare in un parser del genere? Quando si presenta un conflitto (shift/reduce o reduce/reduce), quale azione bisogna scegliere?
101. **Cos'è un prefisso viabile? Come lo si definisce in termini di una grammatica?**
102. **Cos'è un item $\text{LR}(0)$? Come si generano tutti gli item di una grammatica (aumentata con un simbolo iniziale nuovo S')?**
103. **Come è fatto il NFA dei prefissi viabili? Come si ricava il DFA dei prefissi viabili, detto anche automa canonico $\text{LR}(0)$?**
104. **Come è fatta una tabella di parsing $\text{LR}(0)$? Come la si riempie a partire dall'automa canonico $\text{LR}(0)$? Quando una grammatica è di classe $\text{LR}(0)$?**
105. **Come è fatto il parser $\text{LR}(0)$ che utilizza la tabella di parsing $\text{LR}(0)$? Quanti stack servono?**
106. Esistono grammatiche non $\text{LR}(0)$? Fare un esempio semplice.
107. **Come è fatta e come si riempie una tabella di parsing $\text{SLR}(1)$? Cosa vuol dire l'acronimo SLR ? Perché si mette 1 come parametro?**
108. Quando una grammatica è di classe $\text{SLR}(1)$? Esistono grammatiche non di classe $\text{SLR}(1)$?
109. **Cos'è un item $\text{LR}(1)$? Come si costruisce il NFA $\text{LR}(1)$? E come l'automa canonico $\text{LR}(1)$?**
110. **Come è fatta e come si riempie una tabella di parsing $\text{LR}(1)$?**
111. **Come è fatta una tabella di parsing $\text{LALR}(1)$? Quando una grammatica è di classe $\text{LALR}(1)$?**
112. Esistono grammatiche $\text{LALR}(1)$ che non sono $\text{SLR}(1)$? Esistono grammatiche $\text{LR}(1)$ che non sono $\text{LALR}(1)$? Mostrare una grammatica che è $\text{LR}(1)$ ma non $\text{LALR}(1)$.

113. Come si può generalizzare l'idea per ogni $k \geq 2$? Ovvero quando una grammatica è di classe LR(k), SLR(k) o LALR(k)?
114. **Come si relazionano le grammatiche della famiglia LR (LR, SLR, LALR) al variare di k? Come si relazionano le grammatiche LR(k) e LL(k) al variare di k? Le grammatiche LR(k) e LL(k) sono sempre non ambigue? Esistono grammatiche ambigue che sono LL(k) o LR(k) per qualche k? Esistono grammatiche che non sono LR(k) per nessun k?**
115. Come si relazionano i linguaggi della famiglia LR(k) rispetto a quelli LL(k)? Esistono linguaggi liberi deterministici che non sono LR(k) per qualche k? **Esistono linguaggi liberi deterministici che non sono LL(k) per qualche k?**
116. **Esiste un linguaggio regolare che non è LR(0)?** Come si relazionano i linguaggi LR(0) rispetto a quelli LL(1)? La prefix property è una condizione necessaria e sufficiente affinché un linguaggio sia di classe LR(0)?
117. **La classe dei linguaggi SLR(1) coincide con la classe dei linguaggi liberi deterministici?**
118. **Cos'è YACC? Qual è il suo input e il suo output?** Come si ottiene un parser eseguibile a partire da un file .y? Come agisce il parser generato da YACC in sintonia con lo scanner generato da Lex?
119. **Come è la struttura di un file .y di YACC?** Cosa sono le regole? Cos'è l'azione semantica?
120. È possibile gestire grammatiche ambigue con YACC, specificando le associatività e le priorità fra gli operatori per risolvere l'ambiguità? Come si comporta YACC in presenza di conflitti?

Capitolo 5 – Fondamenti

121. È possibile costruire un programma Check che, preso in input un qualunque programma P, restituisce 1 se P è corretto e 0 se P è scorretto? Ovvero esiste un qualche compilatore che può scovare tutti i possibili errori di un programma?
122. **Cosa dice il problema della fermata (Halting Problem)?** (L'errore in esame è la possibilità di non terminare il calcolo.) **Come si dimostra che il problema non può essere risolto?**
123. Quando un problema è decidibile?
124. **Quali sono tipici esempi di proprietà indecidibili per i linguaggi di programmazione?**
125. Cos'è una Macchina di Turing (MdT)? Che cosa calcola?
126. **Quando un linguaggio è detto Turing-completo? Cosa afferma la tesi di Church-Turing e perché non si può dimostrare?**
127. I normali linguaggi di programmazione sono Turing-completi? Cosa afferma il teorema di Jacopini-Bohm?
128. Quale relazione esiste tra espressività di un formalismo e decidibilità di proprietà dello stesso? Fare una panoramica prendendo in esame i tre formalismi MdT, PDA, DFA, e le due proprietà $w \in L(M)$ (è w riconosciuta dalla macchina M?) e $L(M_1) = L(M_2)$ (le due macchine sono equivalenti, ovvero riconoscono lo stesso linguaggio?).

Legenda:

- Domande in **rosso**: se non si sa rispondere, non venire assolutamente all'orale.
- Domande in **grassetto nero**: meglio saper rispondere, per venire all'orale.
- Altre domande: possibili in un orale (meglio saperle se si punta ad un bel voto)

Per le domande relative agli altri capitoli del libro, chiedere al prof. Gabbrielli.