2. (a) Mostrare che A è Turing-riconoscibile *se e solo se* $A \leq_m A_{TM}$.
(b) Mostrare che A è decidibile *se e solo se* $A \leq_m 0^*1^*$.

2. (a) Dimostriamo separatamente i due versi del se e solo se.

- Supponiamo che A sia Turing-riconoscibile. Allora esiste una Macchina di Turing M che riconosce A. Consideriamo la funzione f tale che $f(w) = \langle M, w \rangle$ per ogni stringa $w \in \Sigma^*$. Questa funzione è calcolabile ed è una funzione di riduzione da A a A_{TM}. Infatti, se $w \in A$ allora anche $\langle M, w \rangle \in A_{TM}$ perché la macchina M accetta le stringhe che appartengono ad A. Viceversa, se $w \notin A$, allora $\langle M, w \rangle \notin A_{TM}$ perché la macchina M rifiuta le stringhe che non appartengono ad A.
- Supponiamo che $A \leq_m A_{TM}$. Sappiamo che A_{TM} è un linguaggio Turing-riconoscibile. Per le proprietà delle riduzioni mediante funzione, possiamo concludere che anche A è Turing-riconoscibile.

(b) Dimostriamo separatamente i due versi del se e solo se.

- Supponiamo che A sia decidibile. Allora esiste una Macchina di Turing M che decide A. Consideriamo la funzione f definita nel modo seguente:

$$f(w) = \begin{cases} 01 & \text{se } M \text{ accetta } w \\ 10 & \text{se } M \text{ rifiuta } w \end{cases}$$

  M è un decisore e la sua computazione termina sempre. Quindi la funzione f può essere calcolata dalla seguente macchina di Turing:

  $F =$ "su input w:
  1. Esegui M su input w.
  2. Se M accetta, restituisci 01, se M rifiuta, restituisci 10."

  f è anche funzione di riduzione da A a 0^*1^*. Infatti, se $w \in A$ allora $f(w) = 01$ che appartiene al linguaggio 0^*1^*. Viceversa, se $w \notin A$, allora $f(w) = 10$ che non appartiene a 0^*1^*.
- Supponiamo che $A \leq_m 0^*1^*$. Sappiamo che 0^*1^* è un linguaggio decidibile. Per le proprietà delle riduzioni mediante funzione, possiamo concludere che anche A è decidibile.

2. (12 punti) Considera il problema di determinare se i linguaggi di due DFA sono l'uno il complemento dell'altro.

(a) Formula questo problema come un linguaggio $COMPLEMENT_{\text{DFA}}$.
(b) Dimostra che $COMPLEMENT_{\text{DFA}}$ è decidibile.

Soluzione.

(a) $COMPLEMENT_{\text{DFA}} = \{\langle A, B \rangle \mid A \text{ e } B \text{ sono DFA e } L(A) = \overline{L(B)}\}$

(b) La seguente macchina N usa la Turing machine M che decide EQ_{DFA} per decidere $COMPLEMENT_{\text{DFA}}$:

$N =$ "su input $\langle A, B \rangle$, dove A e B sono DFA:
1. Costruisci l'automa $\overline{B}$ che riconosce il complementare del linguaggio di B
2. Esegui M su input $\langle A, \overline{B} \rangle$, e ritorna lo stesso risultato di M."

Mostriamo che N è un decisore dimostrando che termina sempre e che ritorna il risultato corretto. Sappiamo che esiste un algoritmo per costruire il complementare di un DFA (basta invertire stati finali e stati non finali nella definizione dell'automa). Di conseguenza, il primo step di N termina sempre. Il secondo step termina sempre perché sappiamo che EQ_{DFA} è un linguaggio decidibile. Quindi N termina sempre la computazione.

Vediamo ora che N dà la risposta corretta:

- Se $\langle A, B \rangle \in COMPLEMENT_{\text{DFA}}$ allora $L(A) = \overline{L(B)}$, e di conseguenza $L(A) = L(\overline{B})$ perché $\overline{B}$ è il complementare di B. Quindi $\langle A, \overline{B} \rangle \in EQ_{\text{DFA}}$, e l'esecuzione di M terminerà con accettazione. N ritorna lo stesso risultato di M, quindi accetta.
- Viceversa, se $\langle A, B \rangle \notin COMPLEMENT_{\text{DFA}}$ allora $L(A) \neq \overline{L(B)}$, e di conseguenza $L(A) \neq L(\overline{B})$ perché $\overline{B}$ è il complementare di B. Quindi $\langle A, \overline{B} \rangle \notin EQ_{\text{DFA}}$, e l'esecuzione di M terminerà con rifiuto. N ritorna lo stesso risultato di M, quindi rifiuta.

2. (12 punti) Considera il seguente problema: dato un DFA D e un'espressione regolare R, il linguaggio riconosciuto da D è uguale al linguaggio generato da R?

(a) Formula questo problema come un linguaggio $EQ_{\text{DFA,REX}}$.

(b) Dimostra che $EQ_{\text{DFA,REX}}$ è decidibile.

Soluzione.

(a) $EQ_{\text{DFA,REX}} = \{\langle D, R\rangle \mid D \text{ è un DFA, } R \text{ è una espressione regolare e } L(D) = L(R)\}$

(b) La seguente macchina N usa la Turing machine M che decide EQ_{DFA} per decidere $EQ_{\text{DFA,REX}}$:

$N =$ "su input $\langle D, R\rangle$, dove D è un DFA e R una espressione regolare:
1. Converti R in un DFA equivalente D_R
2. Esegui M su input $\langle D, D_R\rangle$, e ritorna lo stesso risultato di M."

Mostriamo che N è un decisore dimostrando che termina sempre e che ritorna il risultato corretto. Sappiamo che esiste un algoritmo per convertire ogni espressione regolare in un ε-NFA, ed un algoritmo per convertire ogni ε-NFA in un DFA. Il primo step di N si implementa eseguendo i due algoritmi in sequenza, e termina sempre perché entrambi gli algoritmi di conversione terminano. Il secondo step termina sempre perché sappiamo che EQ_{DFA} è un linguaggio decidibile. Quindi N termina sempre la computazione.

Vediamo ora che N dà la risposta corretta:

- Se $\langle D, R\rangle \in EQ_{\text{DFA,REX}}$ allora $L(D) = L(R)$, e di conseguenza $L(D) = L(D_R)$ perché D_R è un DFA equivalente ad R. Quindi $\langle D, D_R\rangle \in EQ_{\text{DFA}}$, e l'esecuzione di M terminerà con accettazione. N ritorna lo stesso risultato di M, quindi accetta.
- Viceversa, se $\langle D, R\rangle \notin EQ_{\text{DFA,REX}}$ allora $L(D) \neq L(R)$, e di conseguenza $L(D) \neq L(D_R)$ perché D_R è un DFA equivalente ad R. Quindi $\langle D, D_R\rangle \notin EQ_{\text{DFA}}$, e l'esecuzione di M terminerà con rifiuto. N ritorna lo stesso risultato di M, quindi rifiuta.

2. (12 punti) Una variabile A in una grammatica context-free G è *persistente* se compare in ogni derivazione di ogni stringa w in $L(G)$. Data una grammatica context-free G e una variabile A, considera il problema di verificare se A è persistente.

(a) Formula questo problema come un linguaggio $PERSISTENT_{\text{CFG}}$.

(b) Dimostra che $PERSISTENT_{\text{CFG}}$ è decidibile.

Soluzione.

(a) $PERSISTENT_{\text{CFG}} = \{\langle G, A\rangle \mid G \text{ è una CFG, } A \text{ è una variabile persistente}\}$

(b) La seguente macchina N usa la Turing machine M che decide E_{CFG} per decidere $PERSISTENT_{\text{CFG}}$:

$N =$ "su input $\langle G, A\rangle$, dove G è una CFG e A una variabile:
1. Verifica che A appartenga alle variabili di G. In caso negativo, rifiuta.
2. Costruisci una CFG G' eliminando tutte le regole dove compare A dalla grammatica G.
3. Esegui M su input $\langle G'\rangle$, e ritorna lo stesso risultato di M."

Mostriamo che N è un decisore dimostrando che termina sempre e che ritorna il risultato corretto. Verificare che una variabile appartenga alle variabili di G è una operazione che si può implementare scorrendo la codifica di G per controllare se A compare nella codifica. Il secondo passo si può implementare copiando la codifica di G senza riportare le regole dove compare A. Di conseguenza, il primo ed il secondo step terminano sempre. Anche il terzo step termina sempre perché sappiamo che E_{CFG} è un linguaggio decidibile. Quindi N termina sempre la computazione.

Vediamo ora che N dà la risposta corretta:

- Se $\langle G, A\rangle \in PERSISTENT_{\text{CFG}}$ allora A è una variabile persistente, quindi compare in ogni derivazione di ogni stringa $w \in L(G)$. Se la eliminiamo dalla grammatica, eliminando tutte le regole dove compare A, allora otteniamo una grammatica G' dove non esistono derivazioni che permettano di derivare una stringa di soli simboli terminali, e di conseguenza G' ha linguaggio vuoto. Quindi $\langle G'\rangle \in E_{\text{CFG}}$, e l'esecuzione di M terminerà con accettazione. N ritorna lo stesso risultato di M, quindi accetta.
- Viceversa, se $\langle G, A\rangle \in PERSISTENT_{\text{CFG}}$ allora A non è una variabile persistente, quindi esiste almeno una derivazione di una parola $w \in L(G)$ dove A non compare. Se eliminiamo A dalla grammatica, eliminando tutte le regole dove compare, allora otteniamo una grammatica G' che può derivare w, e di conseguenza G' ha linguaggio vuoto. Quindi $\langle G'\rangle \notin E_{\text{CFG}}$, e l'esecuzione di M terminerà con rifiuto. N ritorna lo stesso risultato di M, quindi rifiuta.

2. (12 punti) Dati due DFA, considera il problema di determinare se esiste una stringa accettata da entrambi.

(a) Formula questo problema come un linguaggio $AGREE_{\text{DFA}}$.

(b) Dimostra che $AGREE_{\text{DFA}}$ è decidibile.

Soluzione.

(a) $AGREE_{\text{DFA}} = \{\langle A, B\rangle \mid A, B$ sono DFA, ed esiste una parola w tale che $w \in L(A)$ e $w \in L(B)\}$

(b) La seguente macchina N usa la Turing machine M che decide E_{DFA} per decidere $AGREE_{\text{DFA}}$:

$N =$ "su input $\langle A, B\rangle$, dove A, B sono DFA:

1. Costruisci il DFA C che accetta l'intersezione dei linguaggi di A e B
2. Esegui M su input $\langle C\rangle$. Se M accetta, rifiuta, se M rifiuta, accetta."

Mostriamo che N è un decisore dimostrando che termina sempre e che ritorna il risultato corretto. Sappiamo che esiste un algoritmo per costruire l'intersezione di due DFA. Il primo step di N si implementa eseguendo questo algoritmo, e termina sempre perché la costruzione dell'intersezione termina. Il secondo step termina sempre perché sappiamo che E_{DFA} è un linguaggio decidibile. Quindi N termina sempre la computazione.

Vediamo ora che N dà la risposta corretta:

- Se $\langle A, B\rangle \in AGREE_{\text{DFA}}$ allora esiste una parola che viene accettata sia da A che da B, e quindi il linguaggio $L(A) \cap L(B)$ non può essere vuoto. Quindi $\langle C\rangle \notin E_{\text{DFA}}$, e l'esecuzione di M terminerà con rifiuto. N ritorna l'opposto di M, quindi accetta.
- Viceversa, se $\langle A, B\rangle \notin AGREE_{\text{DFA}}$ allora non esiste una parola che sia accettata sia da A che da B, e quindi il linguaggio $L(A) \cap L(B)$ è vuoto. Quindi $\langle C\rangle \in E_{\text{DFA}}$, e l'esecuzione di M terminerà con accettazione. N ritorna l'opposto di M, quindi rifiuta.

3. (9 punti) Dimostra che un linguaggio è decidibile se e solo se esiste un enumeratore che lo enumera seguendo l'ordinamento standard delle stringhe.

3. Per dimostrare che un linguaggio è decidibile se e solo se esiste un enumeratore che lo enumera seguendo l'ordinamento standard delle stringhe:

(⇒) Se L è decidibile, esiste una TM M che lo decide. Possiamo costruire un enumeratore E che genera tutte le stringhe in ordine lessicografico, le testa con M e produce quelle accettate.

(⇐) Se esiste un enumeratore E che enumera L in ordine standard, possiamo costruire un decisore D per L: Su input w, D simula E. Se E produce w, D accetta. Se E produce una stringa lessicograficamente maggiore di w, D rifiuta. D termina sempre perché E enumera le stringhe in ordine.