Algoritmi e Principi dell'Informatica

Parte I – Modelli e computabilità

Appello dell'8 settembre 2011

Tempo a disposizione: 1h30'

Esercizio 1 (12 punti)

Si consideri il linguaggio L fatto di tutte e sole le stringhe $x \in \{a, b\}^*$ in cui *ogni prefisso y* di *x* è tale che $\#b(y) \geq \#a(y)$, laddove con #a(y) e #b(y) si indica, rispettivamente, il numero di *a* e di *b* nella stringa *y*.

1. Si scriva una grammatica che generi L. La grammatica scritta è a potenza minima tra quelle che generano L? Se no, quale è la classe di grammatiche a potenza minima tra quelle che generano L?
2. Si scriva una rete di Petri che riconosca L. La RdP scritta è a potenza minima tra quelle che riconoscono L?

Esercizio 2 (10 punti)

Si consideri la funzione (parziale) $car_x : N \rightarrow \{s, e, b\}$ che descrive i caratteri di una stringa *x*. Più precisamente, $car_x(i)$ restituisce il carattere in posizione i-esima della stringa *x* (ed è indefinito se *x* non ha carattere in posizione i-esima).
Si consideri la seguente specifica logica di un linguaggio L (le stringhe del linguaggio L, cioè, sono tutte e sole quelle che soddisfano le condizioni scritte sotto):

$$
\begin{aligned}
\exists n \,(\quad & n > 0 \\
& \forall j \,(j \geq n \Rightarrow car_x(j) = \bot) \wedge \\
& \forall j \,(0 \leq j < n \Rightarrow car_x(j) \neq \bot) \wedge \\
& car_x(0) \neq e \wedge \\
& \forall j \,(\quad car_x(j) = e \wedge car_x(j+1) = e \wedge \\
& \qquad car_x(j-1) \neq e \wedge car_x(j+2) \neq e \\
& \qquad \Rightarrow \\
& \qquad car_x(j-1) = s \;))
\end{aligned}
$$

1. Descrivere a parole come è fatto L, e dare almeno 2 esempi di stringhe che appartengono ad L e almeno 2 esempi di stringhe che non appartengono ad L. Gli esempi devono essere tutti significativi, nel senso che devono soddisfare o violare la specifica in modi diversi.
2. Scrivere un automa che riconosce L. L'automa deve essere a potenza minima tra quelli che riconoscono L.

Esercizio 3 (11 punti)

Il `datalog` è un linguaggio di interrogazione per basi di dati. Data una query `datalog` Q, si indica con Q(D) l'insieme di risposte a Q ottenute sul database D.
Si parta dalle seguenti definizioni e fatti noti su `datalog`:

- Si dice che Q1 è *contenuta* in Q2 se $Q1(D) \subseteq Q2(D)$ per ogni database D.
- Due query datalog Q1 e Q2 si dicono invece *equivalenti* se $Q1(D) \subseteq Q2(D)$ e $Q2(D) \subseteq Q1(D)$ per ogni database D.
- Il problema di determinare se due generiche query `datalog` siano equivalenti è indecidibile.

A partire da questi fatti (e solo da questi), è possibile concludere la decidibilità o indecidibilità dei seguenti problemi?

1. Determinare se una generica query `datalog` è contenuta in un'altra generica query `datalog`.
2. Determinare se una generica query `datalog` è equivalente a una query `datalog` prefissata.
3. Determinare se una query `datalog` prefissata è equivalente a un'altra query `datalog` prefissata.

NB: per questo esercizio non è importante conoscere la struttura delle query `datalog`, né il meccanismo di calcolo delle risposte alle query.

Algoritmi e Principi dell'Informatica

Parte II – Complessità, algoritmi, strutture dati

Appello dell'8 settembre 2011

Tempo a disposizione: 1h30

Esercizio 1 (12 punti)

Si consideri la seguente funzione f(x) definita su stringhe $x \in \{a, b\}^*$:

$$f(x) = \text{if } \#a(x) \geq \#b(x) \text{ then a else b}$$

dove #a(x) e #b(x) denotano rispettivamente il numero di *a* ed il numero di *b* nella stringa x. Per esempio, f(aabab)=a, f(abbab)=b, f(baabba)=a, f(ε)=a.

1. Si definisca in dettaglio una macchina di Turing (la più semplice, indipendentemente dalla sua complessità computazionale) che calcola la funzione f.
2. Si delinei una macchina RAM che calcola f (si assuma che la macchina RAM possa gestire caratteri, oltre che numeri).
3. Si valutino le funzioni $T_M(|x|)$, $S_M(|x|)$, $T_{RAM}(|x|)$ e $S_{RAM}(|x|)$, cioè le complessità temporali e spaziali della macchina di Turing e della macchina RAM in funzione della lunghezza della stringa *x* (per la macchina RAM si adotti il criterio di costo uniforme). Si confrontino le due funzioni di complessità e le rispettive classi di complessità $\Theta(T_M)$ e $\Theta(T_{RAM})$: quale è la migliore? E per quel che riguarda $\Theta(S_M)$ e $\Theta(S_{RAM})$?
4. Cosa cambia nella risposta al punto 3 se per la valutazione di $T_{RAM}(|x|)$ e $S_{RAM}(|x|)$ viene usato il criterio di costo logaritmico?

Esercizio 2 (12 punti)

1. Si consideri il seguente algoritmo `p(n)`:

```
p(n)
if n < 2
  return 1
return p(n/2) + p(n/2) + p(n/2)
```

a. Quale è la complessità temporale dell'algoritmo `p` in funzione del parametro *n*?

b. Cambierebbe il risultato prodotto dall'algoritmo se la terza istruzione fosse
`return 3*p(n/2)`
invece di quella scritta sopra?
Cambierebbe la complessità?

2. Si consideri il seguente frammento di pseudocodice:

```
for i := 1 to n
  j := i+1
  while (j / n <= n)
    k := 1
    while (k <= n)
      p(4)
      k := k + 3
    j := j +1
```

laddove `p` è l'algoritmo definito al punto 1.

Dire quale è la complessità temporale del frammento di codice in funzione del parametro *n*.

3. Si consideri il seguente frammento di pseudocodice:

```
i := n
while (i > 0)
  p2(n)
  i := i/2
```

dove `p2(n)` è definito nel seguente modo:

```
p2(n)
if n < 1
  print(n)
else p2(n-1)
```

Dire quale è la complessità temporale del frammento di codice in funzione del parametro *n*.

NB: Tutte le variabili nei frammenti di codice precedenti sono da intendersi di tipo intero, quindi anche le operazioni su di esse sono da intendersi come operazioni sugli interi.

Esercizio 3 (8 punti)

Si consideri il problema di ritornare, dato in ingresso un albero binario *T* contenente numeri interi, la radice del sottoalbero *S* di *T* la cui somma degli elementi è massima (tale cioè per cui non esiste nessun altro sottoalbero di *T* la cui somma dei valori è maggiore di quella degli elementi di *S*).

Scrivere in pesudocodice un algoritmo che risolve il problema desiderato, e darne la complessità.

Soluzioni – Parte I

Esercizio 1

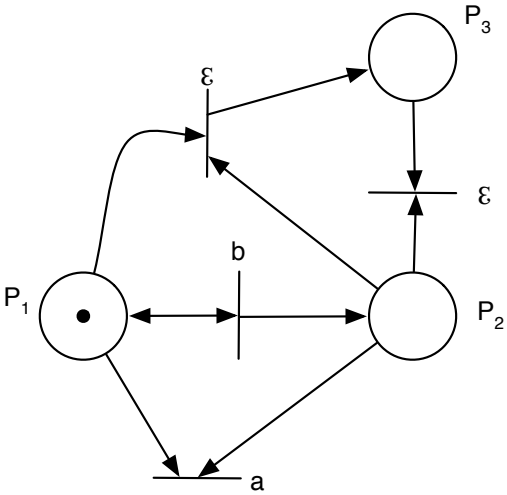
1. Una grammatica a potenza minima che genera il linguaggio desiderato è la seguente:

S → HBS | ε
B → Bb | b
H → bHa | HH | ε

Una grammatica alternativa (sempre a potenza minima) e con meno produzioni, è la seguente:

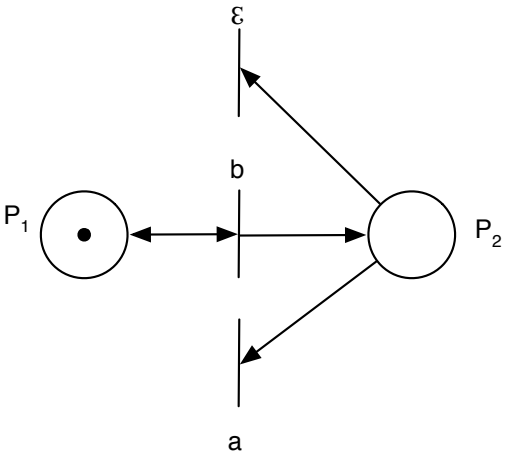
S → bSaS | bB | ε

2. Una rete di Petri che riconosce il linguaggio desiderato è la seguente:



In cui le marcature finali sono 2, M_1, ed M_3, definite nel seguente modo: $M_1(P_1) = 1$, $M_1(P_2) = M_1(P_3) = 0$ e $M_3(P_3) = 1$, $M_3(P_2) = M_3(P_1) = 0$.

In alternativa, un'altra rete di Petri, più compatta, che risolve il problema è la seguente:



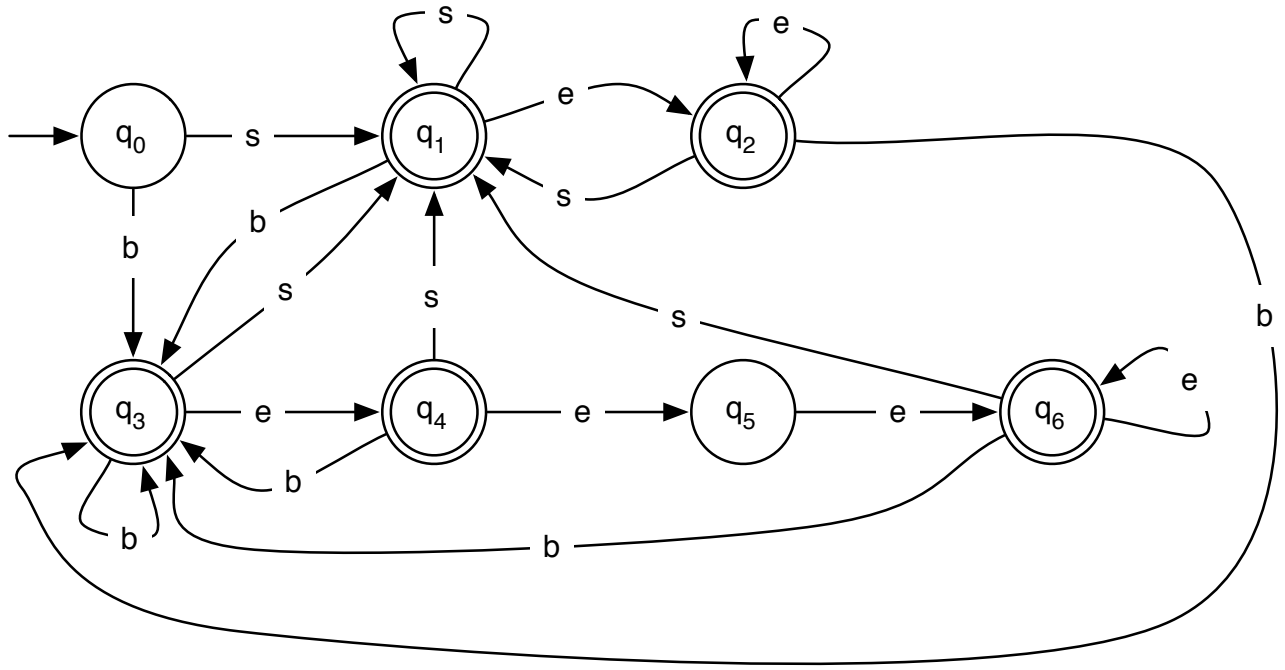
In questo caso la marcatura finale è una sola, e coincide con quella iniziale.

Esercizio 2

Il linguaggio specificato è fatto di tutte e sole le stringhe di almeno 1 carattere che non iniziano per *e*, e tali che ogni volta che si incontrano esattamente 2 *e* consecutive (cioè 2 e precedute e seguite da un simbolo diverso da *e*, eventualmente da ⊥) il simbolo precedente le *e* è una *s*.

2 stringhe che appartengono: bbbbsee, sebbbeeebessseeb
2 stringhe che non appartegono: essss, bsbbeebb

Un automa a stati finiti che riconosce il linguaggio L è il seguente:



Esercizio 3

Il problema è indecidibile. Infatti, se fosse possibile decidere il contenimento tra due generiche query Q1 e Q2 si potrebbe immediatamente decidere anche la loro equivalenza, il che è assurdo. A tal fine si noti che, poiché Q1 è equivalente a Q2 se e solo se Q1 è contenuta in Q2 e Q2 è contenuta in Q1, c'è una riduzione dal problema dell'equivalenza (che è noto essere indecidibile) a quello del contenimento (che è quindi più generale e pertanto indecidibile).

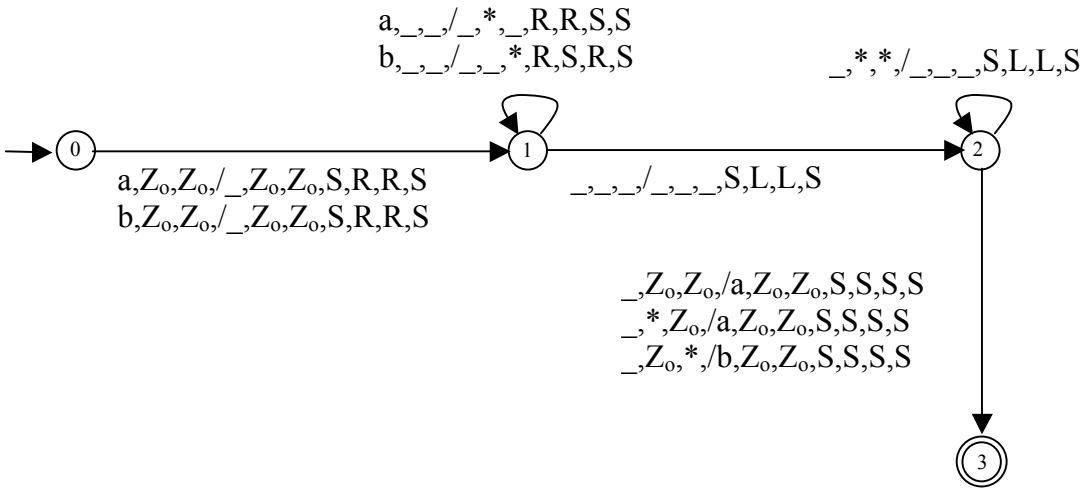
2. In assenza di altre informazioni non è possibile concludere nulla sulla decidibilità di questo problema. Si tratta infatti di un caso specifico di un problema indecidibile (quello dell'equivalenza tra due query generiche).

3. In questo caso il problema è banalmente decidibile in quanto si tratta di una domanda con risposta chiusa sì/no che non dipende da alcun ingresso.

Soluzioni – Parte II

Esercizio 1

1. Una macchina di Turing (a 2 nastri) che calcola la funzione desiderata è la seguente:



2. La macchina RAM legge semplicemente tutti i simboli in input, conta il numero di *a* e *b* in 2 contatori *na* e *nb*, confronta *na* con *nb* ed infine scrive *a* se *na*≥*nb*, *b* altrimenti.
3. La MT percorre l'autoanello sullo stato 1 |x| volte, e quello sullo stato 2 min(#a(x),#b(x)) volte; il caso pessimo è quello in cui #a(x)=#b(x); in questo caso la complessità è $T_M(|x|)=3+(3/2)|x|$; $T_{RAM}(|x|)=k+|x|$ per una (piccola) costante k. T_M è $\Theta(n)$, idem per T_{RAM}. Quindi la complessità T_{RAM} è migliore di T_M, ma la classe Θ è la stessa.
   Per quel che riguarda la complessità **spaziale**, $S_M(|x|)=|x|$, e $S_{RAM}(|x|)=2$,quindi la complessità della macchina RAM è migliore, anche considerando le classi Θ.
4. Poiché il valore dei contatori *na* e *nb* può crescere indefinitamente, i costi delle operazioni di incremento e del confronto finale sono moltiplicati di un fattore log |x|, quindi T_{RAM} è $\Theta(|x| \log |x|)$: la classe di complessità $\Theta(T_{RAM})$ è quindi peggiore di $\Theta(T_M)$. Inoltre, $T_{RAM}(|x|)$ può essere peggiore di $T_M(|x|)$, ma ciò accade per valori abbastanza grandi di |x|.
   Per quel che riguarda la complessità **spaziale**, $S_{RAM}(|x|)=2\cdot\log(|x|)$, quindi la complessità spaziale della macchina RAM è ancora migliore, anche considerando le classi Θ.

Esercizio 2

1a. p è un algoritmo ricorsivo la cui ricorrenza è $T(n) = 3T(n/2)+ \Theta(1)$, la cui soluzione è, applicando il Master theorem, $T(n) = \Theta(n^{\log_2 3})$.

1b. In questo secondo caso la ricorrenza diventa $T(n) = T(n/2)+ \Theta(1)$, in quanto vi è una sola chiamata ricorsiva; la soluzione della ricorrenza è quindi, applicando il Master theorem, $T(n) = \Theta(\log(n))$.

2. Poiché la chiamata all'algoritmo p viene sempre fatta con parametro costante, il suo costo è altrettanto costante. Il costo del frammento di codice dipende quindi solo dai cicli. Quello più interno viene eseguito sempre n/3 volte. Quello intermedio viene eseguito ogni volta $(n-i)^2$ volte, per *i* che va da 1 a n. Il costo totale del frammento è quindi $\sum_{i=1}^{n} i^2 \frac{n}{3}$, che è $\Theta(n^4)$.

3. In questo caso la ricorrenza di p2 è $T(n) = T(n-1) + \Theta(1)$, che non può essere risolta con il Master Theorem. Tuttavia, come visto a lezione, è noto che questa ricorrenza ha soluzione $\Theta(n)$, come si può anche calcolare notando che vengono fatte *n* chiamate ricorsive, ed ognuna ha costo costante. p2 viene invocato *log*(*n*) volte (tante quante sono le iterazioni del ciclo), sempre con parametro *n*, quindi il costo totale del frammento di codice è $\Theta(n \log(n))$.

Esercizio 3
Un algoritmo che risolve il problema desiderato è il seguente:

```
MaxSubtree(T)
[S, sumS, sumT] = MaxSubtreeWalk(T)
return S
```

Dove `MaxSubtreeWalk` è il seguente (dove `-inf` è un valore convenzionale che indica un numero più piccolo di tutti quelli possibili):

```
MaxSubtreeWalk(S)
if S = NIL
  return S,-inf,-inf
else
  [SL, sumSL, sumL] = MaxSubtreeWalk(S.left)
  [SR, sumSR, sumR] = MaxSubtreeWalk(S.right)
  sumS := sumSL+sumSR+S.key
  if sumS >= sumSL and sumS >= sumSR
    return S, sumS, sumS
  else if sumSL >= sumS and sumSL >= sumSR
    return SL, sumSL, sumS
  else return SR, sumSR, sumS
```

L'algoritmo è essenzialmente un post-order tree walk, e la sua complessità è $\Theta(n)$.

Algoritmi e Principi dell'Informatica

Semiunità di Informatica Teorica

Appello dell'8 settembre 2011

Tempo a disposizione: 2h

Esercizio 1 (10 punti)

Si consideri il linguaggio L fatto di tutte e sole le stringhe $x \in \{a, b\}^*$ in cui *ogni prefisso y* di *x* è tale che $\#b(y) \geq \#a(y)$, laddove con #a(y) e #b(y) si indica, rispettivamente, il numero di *a* e di *b* nella stringa *y*.

1. Si scriva un automa che generi L. L'automa scritto è a potenza minima tra quelli che generano L? Se no, quale è la classe di automi a potenza minima tra quelli che generano L?
2. Si scriva una grammatica che generi L. La grammatica scritta è a potenza minima tra quelle che generano L? Se no, quale è la classe di grammatiche a potenza minima tra quelle che generano L?

Esercizio 2 (5 punti)

Si consideri la funzione (parziale) $car_x : N \rightarrow \{s, e, b\}$ che descrive i caratteri di una stringa *x*. Più precisamente, $car_x(i)$ restituisce il carattere in posizione i-esima della stringa *x* (ed è indefinito se *x* non ha carattere in posizione i-esima).
Si consideri la seguente specifica logica di un linguaggio L (le stringhe del linguaggio L, cioè, sono tutte e sole quelle che soddisfano le condizioni scritte sotto):

$$
\begin{aligned}
\exists\, n \,(\quad & n > 0 \\
& \forall j\, (j \geq n \Rightarrow car_x(j) = \bot) \wedge \\
& \forall j\, (0 \leq j < n \Rightarrow car_x(j) \neq \bot) \wedge \\
& car_x(0) \neq e \wedge \\
& \forall j\, (\quad car_x(j) = e \wedge car_x(j+1) = e \wedge \\
& \qquad car_x(j-1) \neq e \wedge car_x(j+2) \neq e \\
& \qquad \Rightarrow \\
& \qquad car_x(j-1) = s \;\;))
\end{aligned}
$$

Descrivere a parole come è fatto L, e dare almeno 2 esempi di stringhe che appartengono ad L e almeno 2 esempi di stringhe che non appartengono ad L. Gli esempi devono essere tutti significativi, nel senso che devono soddisfare o violare la specifica in modi diversi.

Esercizio 3 (9 punti)

Il `datalog` è un linguaggio di interrogazione per basi di dati. Data una query `datalog` Q, si indica con Q(D) l'insieme di risposte a Q ottenute sul database D.
Si parta dalle seguenti definizioni e fatti noti su `datalog`:

- Si dice che Q1 è *contenuta* in Q2 se $Q1(D) \subseteq Q2(D)$ per ogni database D.
- Due query datalog Q1 e Q2 si dicono invece *equivalenti* se $Q1(D) \subseteq Q2(D)$ e $Q2(D) \subseteq Q1(D)$ per ogni database D.
- Il problema di determinare se due generiche query `datalog` siano equivalenti è indecidibile.

A partire da questi fatti (e solo da questi), è possibile concludere la decidibilità o indecidibilità dei seguenti problemi?

1. Determinare se una generica query `datalog` è contenuta in un'altra generica query `datalog`.
2. Determinare se una generica query `datalog` è equivalente a una query `datalog` prefissata.
3. Determinare se una query `datalog` prefissata è equivalente a un'altra query `datalog` prefissata.

NB: per questo esercizio non è importante conoscere la struttura delle query `datalog`, né il meccanismo di calcolo delle risposte alle query.

Esercizio 4 (9 punti)

Si consideri la seguente funzione f(x) definita su stringhe $x \in \{a, b\}^*$:

$$f(x) = \text{if } \#a(x) \geq \#b(x) \text{ then a else b}$$

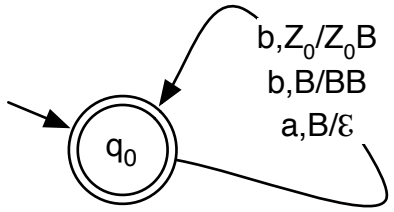
dove #a(x) e #b(x) denotano rispettivamente il numero di *a* ed il numero di *b* nella stringa x. Per esempio, f(aabab)=a, f(abbab)=b, f(baabba)=a, f(ε)=a.

1. Si definisca in dettaglio una macchina di Turing (la più semplice, indipendentemente dalla sua complessità computazionale) che calcola la funzione f.
2. Si delinei una macchina RAM che calcola f (si assuma che la macchina RAM possa gestire caratteri, oltre che numeri).
3. Si valutino le funzioni $T_M(|x|)$, $S_M(|x|)$, $T_{RAM}(|x|)$ e $S_{RAM}(|x|)$, cioè le complessità temporali e spaziali della macchina di Turing e della macchina RAM in funzione della lunghezza della stringa *x* (per la macchina RAM si adotti il criterio di costo uniforme). Si confrontino le due funzioni di complessità e le rispettive classi di complessità $\Theta(T_M)$ e $\Theta(T_{RAM})$: quale è la migliore? E per quel che riguarda $\Theta(S_M)$ e $\Theta(S_{RAM})$?
4. Cosa cambia nella risposta al punto 3 se per la valutazione di $T_{RAM}(|x|)$ e $S_{RAM}(|x|)$ viene usato il criterio di costo logaritmico?

Soluzioni

Esercizio 1

1. Un automa (a potenza minima) che riconosce il linguaggio desiderato è il seguente:



2. Una grammatica a potenza minima che genera il linguaggio desiderato è la seguente:

$S \rightarrow HBS \mid \varepsilon$
$B \rightarrow Bb \mid b$
$H \rightarrow bHa \mid HH \mid \varepsilon$

Una grammatica alternativa (sempre a potenza minima) e con meno produzioni, è la seguente:

$S \rightarrow bSaS \mid bB \mid \varepsilon$

Esercizio 2

Il linguaggio specificato è fatto di tutte e sole le stringhe di almeno 1 carattere che non iniziano per *e*, e tali che ogni volta che si incontrano esattamente 2 *e* consecutive (cioè 2 e precedute e seguite da un simbolo diverso da *e*, eventualmente da $\perp$) il simbolo precedente le *e* è una *s*.

2 stringhe che appartengono: bbbbsee, sebbbeeebessseeb
2 stringhe che non appartegono: essss, bsbbeebb

Esercizio 3

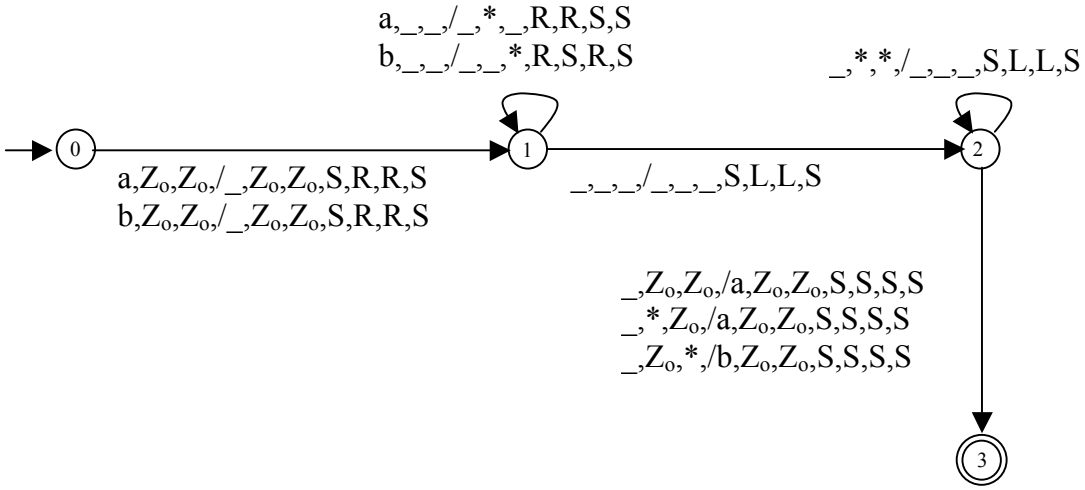
Il problema è indecidibile. Infatti, se fosse possibile decidere il contenimento tra due generiche query Q1 e Q2 si potrebbe immediatamente decidere anche la loro equivalenza, il che è assurdo. A tal fine si noti che, poiché Q1 è equivalente a Q2 se e solo se Q1 è contenuta in Q2 e Q2 è contenuta in Q1, c'è una riduzione dal problema dell'equivalenza (che è noto essere indecidibile) a quello del contenimento (che è quindi più generale e pertanto indecidibile).

2. In assenza di altre informazioni non è possibile concludere nulla sulla decidibilità di questo problema. Si tratta infatti di un caso specifico di un problema indecidibile (quello dell'equivalenza tra due query generiche).

3. In questo caso il problema è banalmente decidibile in quanto si tratta di una domanda con risposta chiusa sì/no che non dipende da alcun ingresso.

Esercizio 4

1. Una macchina di Turing (a 2 nastri) che calcola la funzione desiderata è la seguente:



2. La macchina RAM legge semplicemente tutti i simboli in input, conta il numero di *a* e *b* in 2 contatori *na* e *nb*, confronta *na* con *nb* ed infine scrive *a* se *na*≥*nb*, *b* altrimenti.
3. La MT percorre l'autoanello sullo stato 1 |x| volte, e quello sullo stato 2 min(#a(x),#b(x)) volte; il caso pessimo è quello in cui #a(x)=#b(x); in questo caso la complessità è $T_M(|x|)=3+(3/2)|x|$; $T_{RAM}(|x|)=k+|x|$ per una (piccola) costante k. T_M è $\Theta(n)$, idem per T_{RAM}. Quindi la complessità T_{RAM} è migliore di T_M, ma la classe Θ è la stessa.
   Per quel che riguarda la complessità **spaziale**, $S_M(|x|)=|x|$, e $S_{RAM}(|x|)=2$,quindi la complessità della macchina RAM è migliore, anche considerando le classi Θ.
4. Poiché il valore dei contatori *na* e *nb* può crescere indefinitamente, i costi delle operazioni di incremento e del confronto finale sono moltiplicati di un fattore log |x|, quindi T_{RAM} è $\Theta(|x| \log |x|)$: la classe di complessità $\Theta(T_{RAM})$ è quindi peggiore di $\Theta(T_M)$. Inoltre, $T_{RAM}(|x|)$ può essere peggiore di $T_M(|x|)$, ma ciò accade per valori abbastanza grandi di |x|.
   Per quel che riguarda la complessità **spaziale**, $S_{RAM}(|x|)=2\cdot\log(|x|)$, quindi la complessità spaziale della macchina RAM è ancora migliore, anche considerando le classi Θ.