Exercise 3 (8 pts)

We define a *monotonic grammar* as an unrestricted grammar that satisfies the following constraints for every rule of the form $\alpha \rightarrow \beta$:

- $\alpha \in V_N^+$ where V_N is the set of nonterminal symbols
- $\beta \in (V_N \cup V_T)^+$ where V_T is the set of terminal symbols
- $|\alpha| \leq |\beta|$

Notice that the notions of derivation and language generated by the grammar are defined as usual.

Answer the following questions as precisely and clearly as possible, providing suitable justifications.

1. Is the language generated by a monotonic grammar decidable? Is it semidecidable?
2. What changes concerning the decidability and semidecidability of the generated language if the above constraints are removed?

Exercise 4 (8 pts)

The following code fragment, written in a high-level programming language pseudo code, reads two positive integer values into variables *n* and *k*, and a sequence of *n* integer values that are stored in the array *a*. The program outputs 1 if there is a pair of values stored in array *a* such that their sum is equal to *k*, otherwise it outpus 0.

```
read(n);
read(k);
for i from 1 to n do read(a[i]);  endfor;
for i from 1 to n do
        for j from 1 to n do
                if a[i]+a[j] = k then
                        write(1); halt;
                endif;
        endfor;
endfor;
write(0);
halt;
```

Assume that the above fragment is translated into RAM machine code and executed.

1. Evaluate the time and space complexity class of its execution, as a function of the input size, i.e., of the number of integer values that are read from the input. Use the constant cost criterion.
2. What changes in the above complexity evaluations if the logarithmic cost criterion is adopted? Do the complexity figures change significantly?

1

a ε, abba, ababbaba, abababbababa

b $L(G) = \{(ab)^n (ba)^n \mid n \geq 0\}$

c $src(\varepsilon) = \varepsilon$ $src(abba) = abab$

$src(ababbaba) = abababab$

$src(ababbababbab) = abababababab$

d $L = \{(ab)^{2n} \mid n \geq 0\}$

e $S \to aPb \mid \varepsilon$ $P \to bSa$

f NON VALE LA CHIUSURA. SI CONSIDERI AD ESEMPIO LO STESSO L(G) DEFINITO IN (b). ESSO RICHIEDE UNA GRAMMATICA NON CONTESTUALE

2 L_4

$(a(x) \vee b(x)) \wedge (last(x) \to (L(x) \vee d(x))) \wedge ((a(x) \vee b(x)) \to$

$(C(x+1) \vee d(x+1))) \wedge ((\neg last(x) \wedge (C(x) \vee S(x))) \to (a(x+1) \vee b(x+1)))$

$X_2 = ab \in L(\varphi_1)$ E, PIÙ FORMALMENTE, $ab \not\models \varphi_3$ POICHÈ LA CONDIZIONE $(last(x) \to (L(x) \vee d(x)))$ NON È RISPETTATA

L_2

DEFINIAMO IL PREDICATO $P(x)$ COME "SIRINGA CON NUM. PARI DI CALZETTE"

$$((a(0) \vee b(0)) \wedge \neg P(0)) \wedge (last(x) \rightarrow ((C_b(x) \vee C(x)) \wedge P(x))) \wedge$$

$$\wedge ((P(x) \wedge \neg last(x)) \rightarrow ((a(x+1) \vee b(x+1)) \wedge \neg P(x))) \wedge$$

$$\wedge ((\neg P(x)) \rightarrow ((b(x+1) \vee C(x+1))))$$

$aab \notin L(\varphi_2)$ E, PIÙ FORMALMENTE, $aab \not\models \varphi_2$ POICHÈ LA CONDIZIONE

$(last(x) \rightarrow ((C_b(x) \vee C(x)) \wedge P(x)))$ NON È RISPETTATA

$abb \notin L(\varphi_2)$ E, PIÙ FORMALMENTE, $abb \not\models \varphi_2$ POICHÈ LA CONDIZIONE

$(last(x) \rightarrow \ldots \wedge P(x))$ NON È RISPETTATA

3

a Il linguaggio generato risulta **decidibile**. Infatti, usando una Macchina di Turing non deterministica, legge x e verifica la sua appartenenza al linguaggio nel seguente modo:

Per ogni stringa in $(V_N \cup V_T)^+$ vengono eseguite tutte le possibili derivazioni delle stringhe già generate.

Se una stringa in $(V_N \cup V_T)^+$ è già generata, la generazione termina con rigetto.

Se trovo un carattere terminale, la generazione termina ed x è accettata se e solo se è uguale alla stringa generata.

Essendo decidibile, è anche **semidecidibile**

b Si avrebbe una semplice grammatica non ristretta. Per definizione non decidibile e semidecidibile

4

a $T(n) = \tilde{\Theta}(n^2)$ (DAL DOPPIO FOR)

$S(n) = \tilde{\Theta}(n)$ (DAL PRIMO FOR)

b L'INFLUENZA PRINCIPALE È DATA DAGLI INCREMENTI DEI CONTATORI.

$$T(n) = n \cdot \sum_{i=1}^{n} \ell(i) \in \tilde{\Theta}(n \cdot n \cdot \log n) = \tilde{\Theta}(n^2 \log n)$$

$$S(n) = 3 \cdot \ell(n) + 1 + n \in \tilde{\Theta}(n)$$

Theoretical Computer Science Exam, February 13, 2019

The exam consists of **4 exercises**. Available time: 1 hr 40 min.; you may write your answers in Italian, if you wish.

All exercises must be addressed for the written exam to be considered sufficient.

F.NAME …….. L.NAME …….. MATRICOLA ……

I have passed the midterm exam and I intend to skip the first part (Ex.1-2) (select one): **YES NO**

Exercise 1 (8 pts)

1. Design a least general grammar, and a least powerful automaton, that respectively generate and accept the following language

$$L_1 = \{\, a^m\, b^k\, c^n\, d \;\mid\; m > k + n \,\}$$

2. Design a least general grammar, and a least powerful automaton, that respectively generate and accept the following language

$$L_2 = \{\, a^m\, b^k\, c^n \;\mid\; m > 0 \;\wedge\; n > 0 \;\wedge\; m + n \le k \le 2m + 2n \,\}$$

Exercise 2 (8 pts)

Specify, by means of suitable pre- and post-conditions, written in the usual first-order logic for program specification, a subprogram ***searcheck*(*v*, *n*, *e*, *c*)** where: ***v*** and ***n*** are input parameters representing, respectively, an array of integers and the number of its elements; ***e*** is an integer input parameter, and ***c*** is an output integer parameter**,** whose value is determined according to the following clauses (assume that indices in the array start from 0).

- ***c*** = -1 iff ***e*** is not present in ***v*** ;
- ***c*** = 0 iff ***e*** is present in ***v*** with exactly one occurrence;
- if there are at least 2 distinct occurrences of ***e*** in ***v***, then
  - ***c*** = 1 if for every two distinct occurrences of ***e*** inside ***v*** all the elements of ***v*** included between them are > ***e*** (example: ***e*** = 4 and ***v***=[1, 2, 3, 4, 7, 8, 9, 5, 4, 2, 1]);
  - ***c*** = 2 if for every two distinct occurrences of ***e*** inside ***v*** all the elements of ***v*** included between them are < ***e*** (example: ***e*** = 4 and ***v***=[1, 2, 3, 4, 1, 2, -1, -2, 4, 2, 1]);
  - ***c*** = 3 if for every two distinct occurrences of ***e*** inside ***v*** some (i.e., at least one) element of ***v*** included between them is < ***e*** and some (i.e., at least one) element of ***v*** included between them is > ***e*** (example: ***e*** = 4 and ***v***=[1, 2, 3, 4, 1, 6, -1, -2, 4, 2, 1]).

Say if the above specification is *complete*, i.e., if the value of the output parameter is defined for all possible values of the input parameters.

Exercise 3 (8 pts)

Consider the following set S_1

$$S_1 = \{\, y \mid \forall n \text{ the Turing machine } M_y \text{ does not accept any string of length } n \,\}$$

State, providing suitable clear and precise justifications, whether set S_1 is decidable or semidecidable.

Consider next the following set S_2

$$S_2 = \{\, y \mid \forall n \text{ the Turing machine } M_y \text{ accepts some string of length } n \text{ in a number of steps} \le n \,\}$$

State, providing suitable clear and precise justifications, whether at least one of the two sets, S_2 and $\neg S_2$, is semidecidable (remember that $\neg S_2$ denotes the complement of set S_2).

Exercise 4 (8 pts)

Design a Turing machine that accepts ***in real time*** the language

$$\{\, x\#y\# \mid x, y \in \{a, b, c, d\}^+ \text{ and } x \text{ is an anagram of } y \,\}$$

(Let us remind that a string x is an anagram of a string y if the two strings contain the same letters, each letter having in each string the same number of occurrences, e.g., as in *badacdbbd* and *bdcadbadb*).

Sketch a RAM machine that accepts the same language, and compare the time and space complexity of the two machines, according to both the constant and the logarithmic cost criterion.

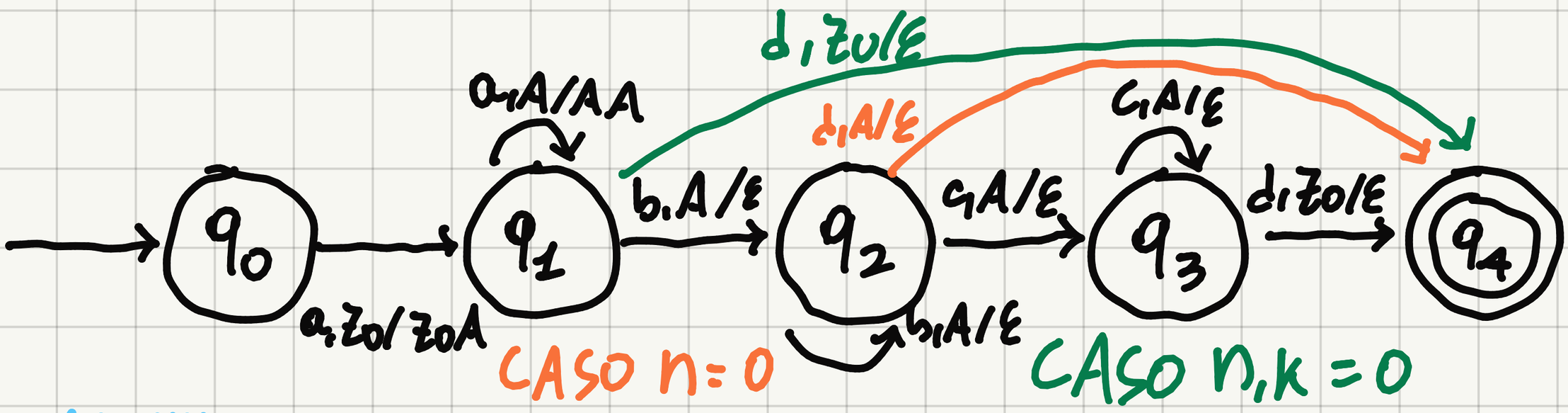
1

a GRAMMATICA:

$S \to Ad$ $\quad A \to aAc \mid aB$ MI ASSICURO CHE $m > n$

$B \to aBb \mid aB \mid \varepsilon$ MI ASSICURO CHE $m > n+k$

AUTOMA: LE CONDIZIONI SUGLI ESPONENTI RICHIEDONO L'USO DI UN AUTOMA A PILA



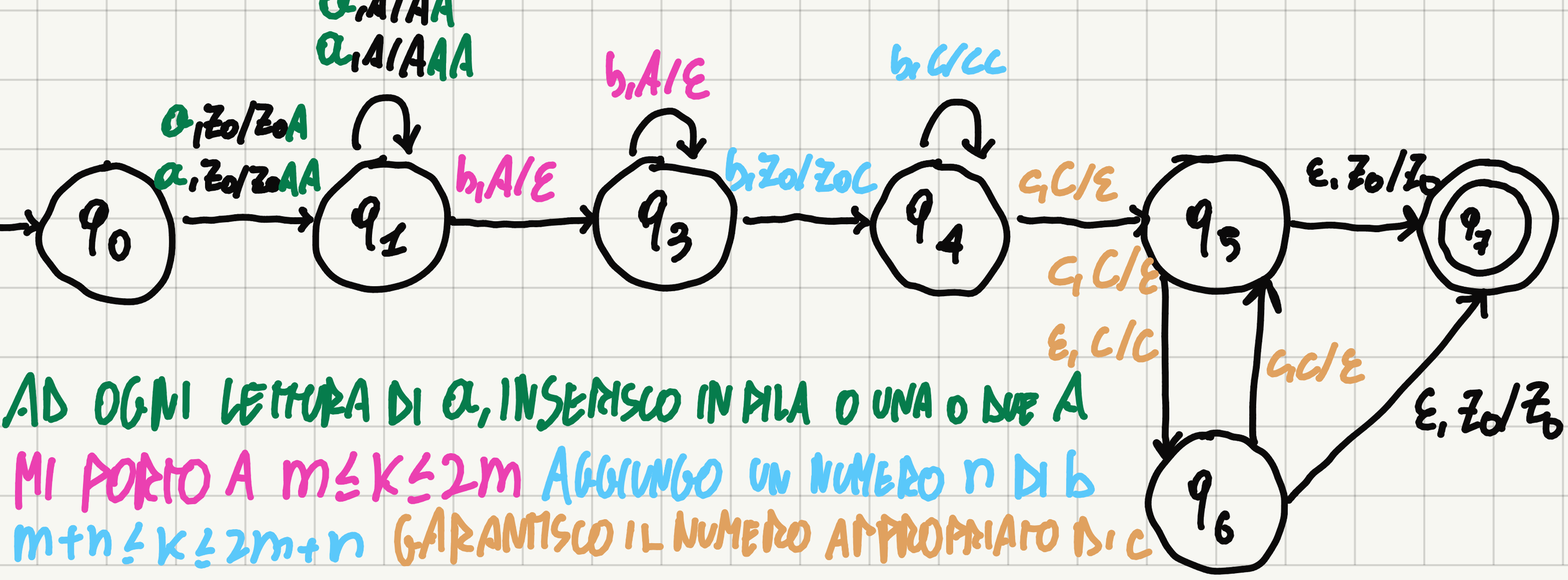
b GRAMMATICA:

$S \to AB$

$A \to aAb \mid aAbb \mid ab \mid abb$

$B \to bBc \mid bbBc \mid bc \mid bbc$

AUTOMA: LE CONDIZIONI SUGLI ESPONENTI RICHIEDONO L'USO DI UN AUTOMA A PILA. INOLTRE, LA CONDIZIONE SUK RICHIEDE UN AUTOMA NON DETERMINISTICO



AD OGNI LETTURA DI a, INSERISCO IN PILA O UNA O DUE A

MI PORTO A $m \leq k \leq 2m$ AGGIUNGO UN NUMERO n DI b

$m+n \leq k \leq 2m+n$ GARANTISCO IL NUMERO APPROPRIATO DI c

2 INPUT:

- v ARRAY DI DIMENSIONE n
- $n>0$
- e INTERO

OUTPUT:

- c INTERO

PRECONDIZIONI] $n>0$

POSTCONDIZIONI] $((c=-1)\leftrightarrow \neg\exists i(0\leq i<n \wedge v[i]=e)) \wedge$

$\wedge ((c=0)\leftrightarrow \exists i(0\leq i<n \wedge v[i]=e \wedge \forall k(0\leq k<n \rightarrow$

$\rightarrow (v[k]=e \leftrightarrow k=i))) \wedge$

$\wedge ((\exists i \exists j(0\leq i<j<n \wedge v[i]=e \wedge v[j]=e) \rightarrow$

$(\forall i \forall j(0\leq i<j<n \wedge v[i]=e \wedge v[j]=e \rightarrow \forall k(i<k<j \rightarrow v[k]>e)] \rightarrow c=1) \wedge$

$(\forall i \forall j(0\leq i<j<n \wedge v[i]=e \wedge v[j]=e \rightarrow \forall k(i<k<j \rightarrow v[k]<e)] \rightarrow c=2) \wedge$

$(\forall i \forall j$

$(0\leq i<j<n \wedge v[i]=e \wedge v[j]=e \rightarrow$

$((\exists k(i<k<j \wedge v[k]\leq e)) \wedge (\exists k(i<k<j \wedge v[k]>e))) \rightarrow c=3)$

LE DEFINIZIONI SONO INCOMPLETE, POICHÈ MANCA IL CASO DI "DUE E CONSECUTIVE", O ANCHE CASI CON ZERO O PIÙ E

3 S_1

NON DECIDIBILE. INFATTI, LE FUNZIONI CONSIDERATE SONO RELATIVE A M.T. CHE NON ACCETTANO STRINGHE DI LUNGHEZZA n. QUESTO NON È NE L'INSIEME VUOTO E NE L'INSIEME DI TUTTE LE FUNZIONI COMPUTABILI. PERTANTO S_1 È NON DECIDIBILE PER RICE THEOREM

SEMIDECIDIBILE? STUDIAMO IL COMPLEMENTARE $\neg S_1 = \{y \mid \exists n$ PER CUI M_y ACCETTA STRINGHE DI LUNGHEZZA $n\}$. QUESTO È UN INSIEME SEMIDECIDIBILE: SIMULANDO L'ESECUZIONE DI UNA M.T., PRESA LA STRINGA SI VERIFICA SE VIENE ACCETTATA → S_1 NON SEMIDECIDIBILE

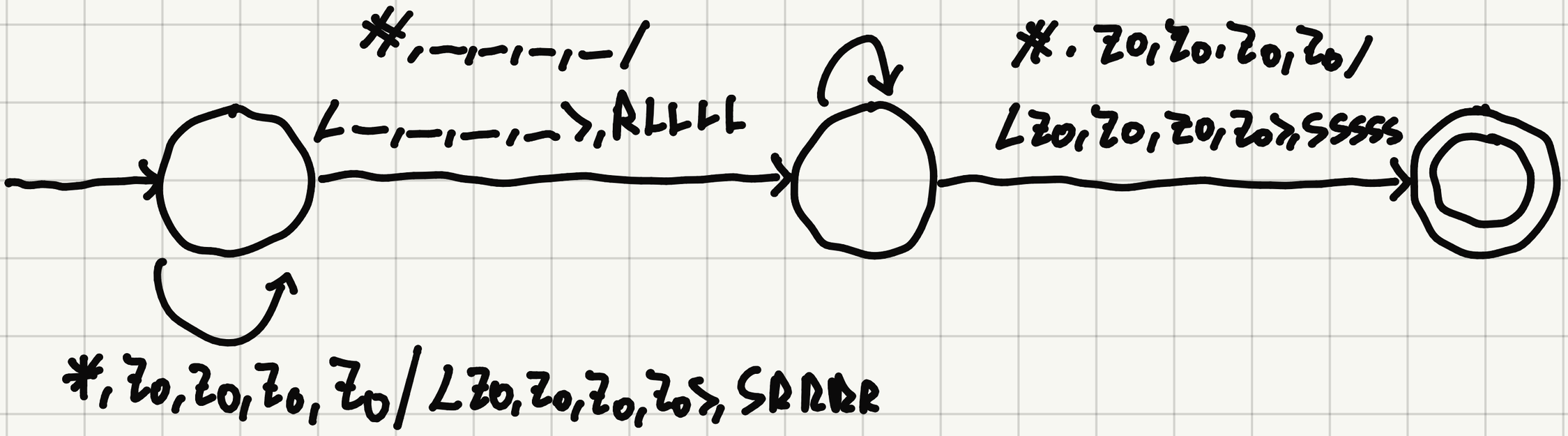
S_2

S_2 È NON DECIDIBILE PER RICE THEOREM, PER CUI SOLO UNO TRA S_2 E $\neg S_2$ È SEMIDECIDIBILE

$\neg S_2 = \{y \mid \forall n$ LA MACCHINA DI TURING NON ACCETTA STRINGHE DI LUNGHEZZA n IN $\leq n$ STEP$\}$ È SEMIDECIDIBILE (CONSIDERAZIONI ANALOGHE A $\neg S_1$), PER CUI S_2 È NON SEMIDECIDIBILE

4 OCCORRE UNA MACCHINA DI TURING A 4 NASTRI

a, *, -, -, - / L, -, -, -, -> , RLSSS
b, -, *, -, - / L, -, -, -, -> , RSLSS
c, -, -, *, - / L, -, -, -, -> , RSSLS
d, -, -, -, * / L, -, -, -, -> , RSSSL



a, -, -, -, - / L A, -, -, -> , RRSSS
b, -, -, -, - / L, -, B, -, -> , RSRSS
c, -, -, -, - / L, -, -, C, -> , RSSRS
d, -, -, -, - / L, -, -, -, D> , RSSSR

UNA MACCHINA RAM LEGGE LA STRINGA IN INPUT E USA 4 CONTATORI PER MEMORIZZARE IL NUMERO DI OCCORRENZE DEI 4 POSSIBILI CARATTERI IN X È LETTA //, VERIFICA SE GLI STESSI NUMERI FORMANO IN Y. DETTA n LA LUNGHEZZA DELLA STRINGA:

COSTO UNIFORME:

- $T(n) = \Theta(n)$
- $S(n) = \Theta(1)$

COSTO LOGARITMICO:

- $T(n) = \Theta(n \log n)$
- $S(n) = \Theta(\log n)$

Theoretical Computer Science Exam, June 26, 2019

The exam consists of **4 exercises**. Available time: 1 hr 40 min.; you may write your answers in Italian, if you wish.

All exercises must be addressed for the written exam to be considered sufficient.

F.NAME …….. L.NAME …….. MATRICOLA ……

I have passed the midterm exam and I intend to skip the first part (Ex.1-2) (select one): **YES NO**

Exercise 1 (8 pts)

1. Design a least general grammar that generates the language $L = L_1^*$, where $L_1 = \{ xx^R \mid x \in \{a, b\}^* \}$
2. Write a derivation for the string $aabbaa \in L$ according to the above defined grammar.
3. Design a least powerful automaton that accepts language L defined above.
4. Write a computation of the above automaton that accepts the string $aabb \in L$

Exercise 2 (8 pts)

1. Write a Monadic First Order (or Monadic Second Order if necessary) logic formula that characterizes the language of nonempty strings over the alphabet $I = \{a, b\}$, such that the total number of *a*'s included in the string is even.
2. Write a Monadic First Order (or Monadic Second Order if necessary) logic formula that characterizes the language of nonempty strings over the alphabet $I = \{a, b\}$, such that *a*'s occur only at positions with an even index value (remember that in the word structure the index of position starts with the value 0).
3. Write a Monadic First Order (or Monadic Second Order if necessary) logic formula that characterizes the language of nonempty strings over the alphabet $I = \{a, b\}$, such that it cannot happen that the *j*-th occurrence of an *a* symbol, with *j* odd (that is, the first, third, fifth, … occurrence) occurs at a position whose index is even.

Exercise 3 (8 pts)

Consider the following language *L* over the binary alphabet $\Sigma = \{ 0, 1 \}$:

$L = \{ x \mid x \in \Sigma^*$ and x is the binary encoding of a positive integer that is divisible by three $\}$

Answer the following questions, providing suitable clear and precise justifications.

1. Is the language *L* decidable? Is it semidecidable?
2. Is it possible to build a Turing Machine that enumerates *L*?
3. Given a generic Turing machine *m*, is it possible to decide whether *m* accepts *L*?
4. Given a generic Turing machine *m*, is it possible to decide whether *m* enumerates *L*?
5. Consider the following set S

   $S = \{ x \mid \forall n \, (f_x(n) \neq \perp \rightarrow f_x(n) = 3 \cdot n \,) \}$

   Describe in words the set *S* and argue whether *S* is decidable or semidecidable.

Exercise 4 (8 pts)

Design, or at least sketch in a clear, sufficiently detailed way, a Turing machine that recognizes strings of the following kind:

$$a^n b a^{n-1} b \ldots a^2 b a b b \quad \text{with } n > 1$$

an example of an acceptable input string is *aaaabaaabaababb.*

The time complexity of the Turing machine should be as reduced as possible.

The pseudocode below encodes an algorithm (to be executed by the RAM) to recognize the strings described above.

```
-- read first sequence of type a^j b
m = 0;
do
    read(c);    if ( c == 'a' ) then m = m+1; endif;
while ( c == 'a' );
if ( c == 'b' and m > 0 ) then mPrev = m; else write('no'); halt; endif;
do
    -- read next sequence of type a^j b
    m = 0;
    do
            read(c);   if ( c == 'a' ) then m = m + 1;   endif;
    while ( c == 'a' );
    if  ( c == 'b' and  m == mPrev - 1 )  then mPrev = m;   else write('no');  halt;  endif;
while ( m > 0 );
write('yes'); halt;
```

Evaluate in a precise way the time complexity class of the above program in the case of an accepted string, with both the uniform and the logarithmic cost criteria.

Compare the complexity of the Turing and RAM machines when the logarithmic time criterion is adopted.

1

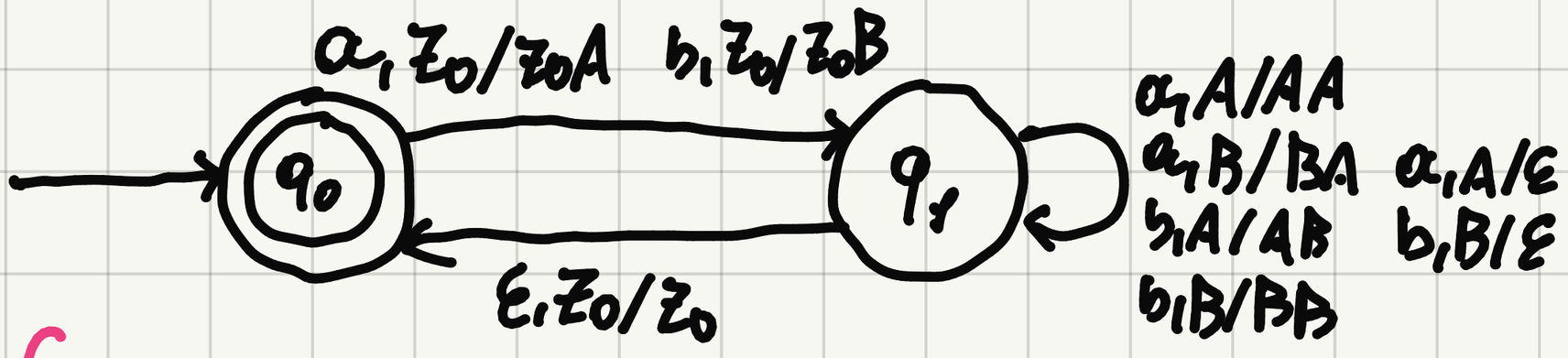
a $S \to SS_1 \mid \varepsilon \qquad S_1 \to aS_1a \mid bS_1b \mid \varepsilon$

b $S \to aSa \to aaSaa \to aabSbaa$

$\to aabbSbbaa \to aabbbaa$

c PER AVERE MEMORIA DI X, E POI INVERTIRLO, OCCORRE UN AUTOMA A PILA DETERMINISTICO

INOLTRE, NON CONOSCENDO IL VALORE DI PARTENZA, OCCORRE UN NON .



SI NOTI CHE $L^* \neq L$

d $aabb$. POSSIAMO IMMAGINARLO COME $aa \cdot bb$

$\langle q_0, aabb, Z_0 \rangle \vdash \langle q_1, abb, Z_0A \rangle \vdash \langle q_1, bb, Z_0 \rangle$

$\vdash \langle q_0, bb, Z_0 \rangle \vdash \langle q_1, b, Z_0B \rangle \vdash \langle q_1, \varepsilon, Z_0 \rangle \vdash \langle q_0, \varepsilon, Z_0 \rangle$

2

a DEFINIAMO $N(x)$ IL PREDICATO "NUMERO PARI DI a"

$(first(x) \leftrightarrow a(x) \vee b(x)) \wedge$ STRINGHE NON VUOTE

$(first(x) \to N(x)) \wedge \forall x((\neg last(x) \wedge N(x) \wedge a(x) \to \neg N(x+1)) \wedge$

$(\neg last(x) \wedge \neg N(x) \wedge a(x) \to N(x+1))) \wedge (last(x) \leftrightarrow N(x))$

b DEFINIAMO $P(x)$ IL PREDICATO "POSIZIONE DI INDICE PARI"

$$(first(x) \rightarrow P(x)) \wedge (a(x) \rightarrow P(x))$$

! NON SE E SOLO SE IN QUANTO ANCHE LE b POSSONO COMPARIRE IN POSIZIONI PARI

$$\wedge (\forall x ((P(x) \wedge \neg last(x)) \rightarrow \neg P(x+1)) \wedge$$

$$\wedge ((\neg P(x) \wedge \neg last(x)) \rightarrow P(x+1)) \wedge$$

$$\wedge (last(x) \leftrightarrow P(x)))$$

c RIUSIAMO I PREDICATI $D(x)$ E $P(x)$ DEFINITI IN PRECEDENZA

$$(first(x) \rightarrow \neg D(x) \wedge P(x)) \wedge$$

$$\forall x((P(x) \wedge \neg last(x)) \rightarrow \neg P(x+1) \wedge$$

$$((\neg P(x) \wedge \neg last(x)) \rightarrow P(x+1) \wedge$$

$$((\neg D(x) \wedge a(x) \wedge \neg last(x)) \rightarrow D(x+1)) \wedge$$

$$((D(x) \wedge a(x) \wedge \neg last(x)) \rightarrow \neg D(x+1) \wedge$$

$$\neg \exists z ((a(z) \wedge D(z) \wedge P(z)))))$$

3

α QUELLO DATO È L'INSIEME DI TUTTE LE FUNZIONI COMPUTABILI, IN QUANTO È SEMPRE DEFINIBILE UNA FUNZIONE CHE, DATO UN NUMERO BINARIO IN INPUT, CON UNA SEQUENZA FINITA DI OPERAZIONI RIESCE A STABILIRE SE IL RISPETTIVO DECIMALE È DIVISIBILE PER 3. QUINDI, IL LINGUAGGIO È DECIDIBILE. E QUINDI SEMIDECIDIBILE. PER RICE THEOREM

b SÌ, PER LA SECONDA PARTE DELLA TESI DI CHURCH (ULTERIORI DETTAGLI A PAG. 306 DEL LIBRO)

c DOVENDO CONSIDERARE INFINITI CASI POSSIBILI, IL PROBLEMA CONSIDERATO RISULTA IRRISOLVIBILE

d NO, PORCHÈ QUESTO È COMUNQUE UN MODO PER VALUTARE SE M ACCETTA L

e S È L'INSIEME DI FUNZIONI CHE, DATO N IN INPUT, NE RESTITUISCE IL MULTIPLO DI 3. LA FUNZIONE È CHIARAMENTE NON DECIDIBILE PER RICE THEOREM, NON ESSENDO NÈ INSIEME VUOTO E NÈ L'INSIEME DI TUTTE LE FUNZIONI COMPUTABILI.

È SEMIDECIDIBILE. ANALIZZIAMO IL COMPLEMENTARE $\neg S = \{x \mid \exists n (\varphi_x(n) = \perp \rightarrow \varphi_x(n) \neq 3n)\}$. CON LA SIMULAZIONE DOVETAILING, È IMMEDIATO CONCLUDERE CHE ESISTONO FUNZIONI

PER CUI, LA FUNZIONE $\neg S$ È SEMIDECIDIBILE, E QUINDI S NON LO È

4

È SUFFICIENTE UNA MACCHINA DI TURING A SINGOLO NASTRO. NOTO IL VALORE DI n, VENGONO LETTE n CL ED EFFETTUATI n MOVIMENTI VERSO DESTRA DAL NASTRO CHE CONTA IL NUMERO DI CL. ALLA LETTURA DI b, SI SCRIVE DI UNA POSIZIONE A SINISTRA IN MODO DA RISTAMPARE n-1 CL ETC. SIA x LA LUNGHEZZA DELLA STRINGA

$T(x) = \Theta(x)$, CON $x = \Theta(n^2)$

LA COMPLESSITÀ TEMPORALE DELLA MACCHINA RAM DIPENDE DAL NUMERO DI LETTURE EFFETTUATE, PER CUI $T(x) = \Theta(x)$ ADOTTANDO IL CRITERIO DI COSTO UNIFORME.

CON IL CRITERIO LOGARITMICO, IL FATTORE DOMINANTE LO DÀ L'ISTRUZIONE "m=m+1". SIA p LA POSIZIONE IN MEMORIA IN CUI È SALVATA m.

IL NUMERO DI OPERAZIONI DIPENDE DA:

LOAD p	$\ell(p)+\ell(M(p))$	} ESEGUITO n VOLTE
ADD = 1	$\ell(M[0])+\ell(1)$	
STORE p	$\ell(M[0])+\ell(p)$	

DEFINITO i COME IL VALORE DI p PRIMA DELL'INCREMENTO.

IL COSTO COMPLESSIVO È DATO DA

$$\ell(P) + \ell(i) + \ell(i+1) + \ell(1) + \ell(i) + \ell(P) =$$

$$= 2\ell(P) + 2\ell(i) + \ell(i+1) \simeq 2\ell(P) + 3\ell(P+1)$$

COMPLESSIVAMENTE,

$$T(x) = \sum_{k=1}^{n} \cdot \sum_{i=1}^{k} (2\ell(P) + 3\ell(i)) =$$

$$= \sum_{k=1}^{n} (2k\,\ell(P) + \sum_{i=1}^{k} 3\ell(i)) =$$

$$= 2\ell(P) \cdot \frac{n(n+1)}{2} + 3\sum_{k=1}^{n} (k\log(k) - k)$$

$$= \Theta(k\log(k) - k)$$

$$= \Theta(x \log \sqrt{x})$$

QUINDI, CON COSTO LOGARITMICO LA COMPLESSITÀ TEMPORALE AUMENTA

Theoretical Computer Science Exam, July 16, 2019

The exam consists of **4 exercises**. Available time: 1 hr 40 min.; you may write your answers in Italian, if you wish.

All exercises must be addressed for the written exam to be considered sufficient.

F.NAME …….. L.NAME …….. MATRICOLA ……

I have passed the midterm exam and I intend to skip the first part (Ex.1-2) (select one): **YES NO**

Exercise 1 (8 pts)

1. Design a least general grammar that generates the language $L = \{ x \in \{a, b\}^* \mid \#_a(x) \bmod 3 = 2 \}$, where $\#_a(x)$ denotes the number of symbols *a* occurring in the string *x*, and *mod* is the usual operator that returns the remainder of integer division (e.g., 7 mod 3 = 1, 14 mod 3 = 2, 6 mod 3 = 0).
2. Write a derivation for the string $abba \in L$ according to the above defined grammar.
3. Design a least powerful automaton that accepts language L defined above.
4. Write a computation of the above automaton that accepts the string $abba \in L$
5. What is the least general type of grammar that generates the language $L_1 = L^*$, where L is the language considered in the above points, and the least powerful type of automaton accepting the same language L_1?
6. Consider language $L_2 = \{ x \in \{a, b\}^* \mid \#_a(x) = 2 \cdot \#_b(x) \}$. Design a least powerful automaton that accepts language L_2, and write the computations of the above automaton that accept the strings $baa, aba, aab \in L$

Exercise 2 (8 pts)

Specify, by means of suitable pre- and post-conditions, written in the usual first-order logic for program specification, a subprogram ***somAll* (*v, n, p, s, a*)** where:

- ***v*** and ***n*** are input parameters representing, respectively, an array of integers and the number (greater than 2) of its elements; it is assumed that the integers stored in ***v*** are not all equal; for instance, [1, 2] and [2, 2, 2] are both invalid values for ***v***
- ***p*** is an integer input parameter, such that $p>1$, assumed to be prime
- ***s*** and ***a*** are two output Boolean parameters**,** whose value, after the execution of the *somAll* subprogram, is determined by the following clauses.
  - ***s*** is true if and only if the input prime number ***p*** is an integer divisor of *some* element of ***v***
  - ***a*** is true if and only if the input prime number ***p*** is an integer divisor of *all* the elements of ***v***

Exercise 3 (8 pts)

Consider any family F of sets such that F has the following properties

a. F is not empty
b. F is *closed under complementation* (for every set $S \in F$ also $\neg S \in F$, $\neg S$ being the complement of set S)
c. $\emptyset \notin F$ (all sets in F are non-empty)

For each of the following statements,

1. All sets of the family F are decidable
2. All sets of the family F are semidecidable and not decidable
3. No set of the family F is semidecidable
4. Some set of the family is semidecidable and not decidable

say if it is:

- necessarily true (it holds for every family F with the above features), or
- necessarily false (it does not hold for any family F with the above features), or
- possibly true/false (it holds for some family but not for some other).

Suitably motivate your answer.

Do the answers you provided above change if, in addition, the elements of the family F are sets of indices of computable functions? Suitably motivate your answer.

Exercise 4 (8 pts)

Design a Turing machine that accepts as input a string of type a^n, with $n > 1$, and writes on the output tape the string:

$$c^n b c^{n-1} b^2 c^{n-2} \ldots b^{n-2} c^2 b^{n-1} c\, b^n$$

for example if the input is *aaa* then the output is *cccbccbbcbbb.*

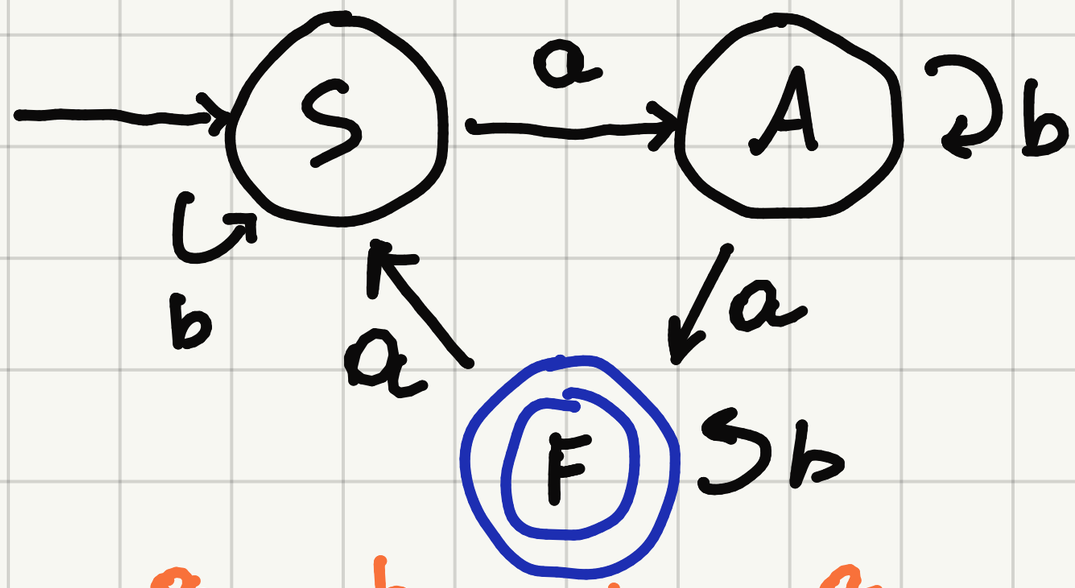
Identify the time complezity function of the designed TM and determine its time complexity class. Solutions where the time complexity of the Turing machine is small are preferred.

Briefly elaborate on the time complexity of a RAM machine that computes (in a reasonably efficient way) the same function and compare it with that of the TM, especially in the case when the logarithmic cost is considered for the RAM.

1

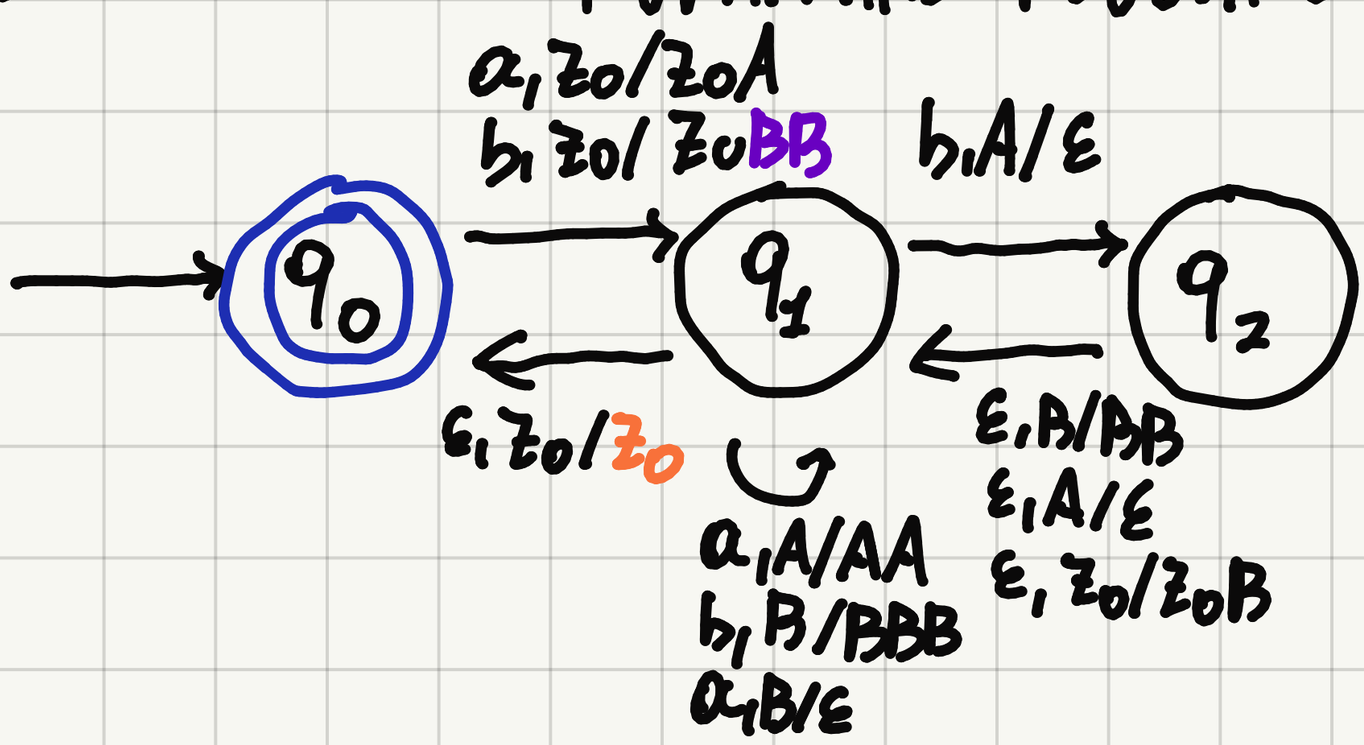
a $S \to bS \mid aA \qquad A \to bA \mid aF \qquad F \to \varepsilon \mid bF \mid aS$

b $S \to aA \to abA \to abbA \to abbaF \to abba$

c 

d $S \xrightarrow{a} A \xrightarrow{b} A \xrightarrow{b} A \xrightarrow{a} F$

e POICHÈ È STATO SUFFICIENTE UN AUTOMA A STATI FINITI, E QUINDI UNA GRAMMATICA REGOLARE, L È REGOLARE → L* È REGOLARE ED È GENERATO DA UNA GRAMMATICA REGOLARE

f 

! SI NOTI CHE È POSSIBILE INSERIRE PIÙ DI UNA B

ALL'INIZIO, $\#a(x) = \#b(x) = 0$

LASCIAMO INVARIATO PER PERMETTERE EVENTUALI ALTRI INPUT

$\langle q_0, bbaa, Z_0 \rangle \vdash \langle q_1, aa, Z_0BB \rangle \vdash \langle q_1, a, Z_0B \rangle \vdash \langle q_1, \varepsilon, Z_0 \rangle \vdash \langle q_0, \varepsilon, Z_0 \rangle$

$\langle q_0, abba, Z_0 \rangle \vdash \langle q_1, bba, Z_0A \rangle \vdash \langle q_2, ba, Z_0 \rangle \vdash \langle q_1, a, Z_0B \rangle \vdash \langle q_1, \varepsilon, Z_0 \rangle \vdash \langle q_0, \varepsilon, Z_0 \rangle$

$\langle q_0, aaab, Z_0 \rangle \vdash \langle q_1, aab, Z_0A \rangle \vdash \langle q_1, ab, Z_0AA \rangle \vdash \langle q_2, b, Z_0A \rangle \vdash \langle q_1, \varepsilon, Z_0 \rangle \vdash \langle q_0, \varepsilon, Z_0 \rangle$

2 INPUT:

- ARRAY v DI DIMENSIONE $n > 2$
- $p > 1$ NUMERO PRIMO

OUTPUT:

- s, a BOOLEANI

PRECONDIZIONI] $(n > 2) \wedge \exists i j (0 \leq i < n \wedge 0 \leq j < n \wedge i \neq j \wedge$

$\wedge v[i] \neq v[j]) \wedge \neg \exists k h (1 < k < p \wedge 1 < h < p \wedge p = kh)$

POSTCONDIZIONI] $(s = T \leftrightarrow [\exists i (0 \leq i < n \rightarrow \exists k (k \neq 1 \wedge k \neq v[i] \wedge v[i] = k \cdot p)$

$\wedge (a = T \leftrightarrow (\forall i (0 \leq i < n \rightarrow \exists k (k \neq 1 \wedge k \neq v[i] \wedge v[i] = kp)$

3 a USANDO IL RICE THEOREM:

- $F = \emptyset$ NON SI VERIFICA
- F PUÒ ESSERE L'INSIEME DI TUTTE LE FUNZIONI COMPUTABILI.

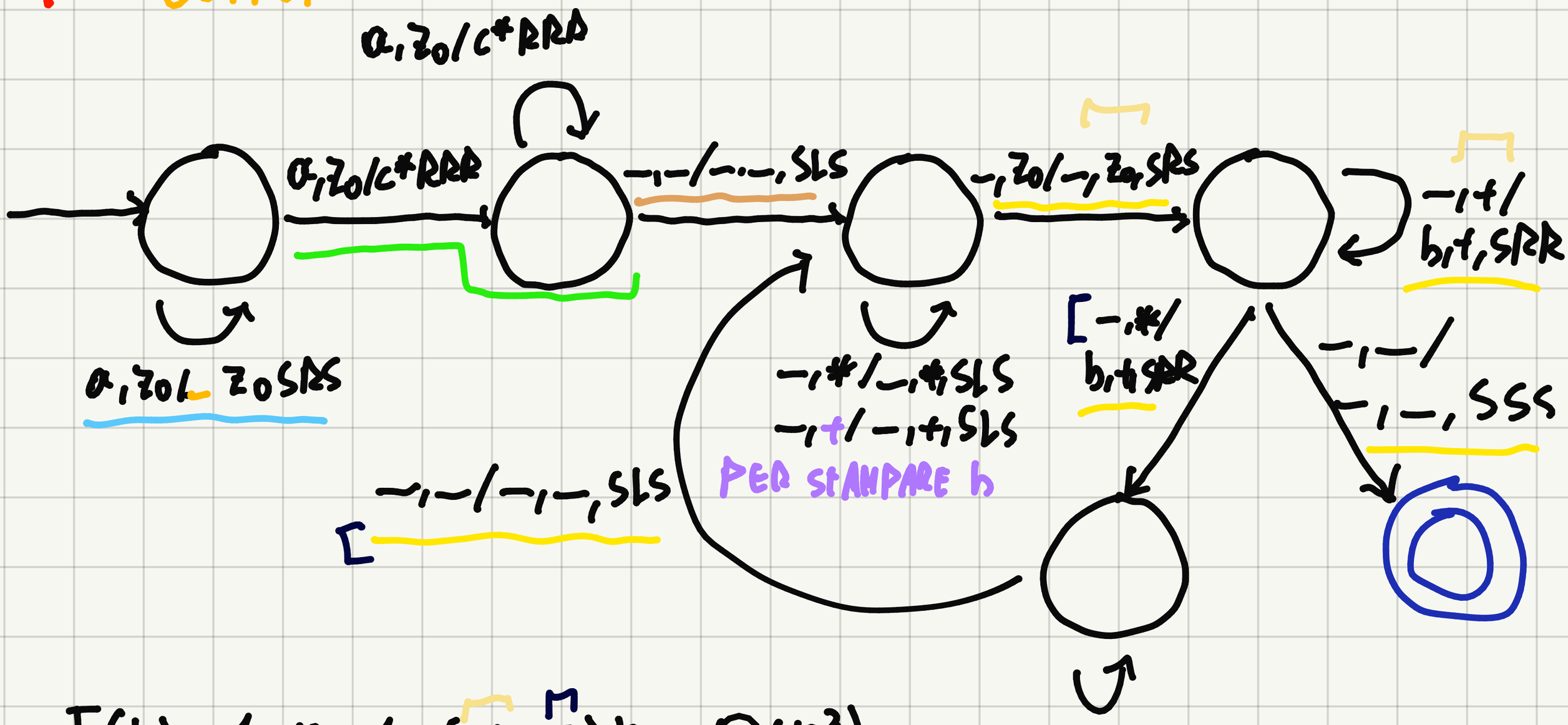
  → T/F

b SE SUPPONIAMO LA NON DECIDIBILITÀ, SOLO UNO TRA S E $\neg S$
ENTRAMBI → F
È SEMIDECIDIBILE MA, PER PROPRIETÀ DI F, LO DOVREBBERO ESSERE
NESSUNO È SEMI

c PER LE STESSE CONSIDERAZIONI IN b. T/F (CASO ESTREMO IN CUI

d PER LE STESSE CONSIDERAZIONI IN b. T/F

4 OUTPUT



$$T(n) = \underline{1} + \underline{n} + \underline{1} + \overbrace{(\underline{2n+2}}^{m})n = \Theta(n^2)$$

UNA MACCHINA RAM UTILIZZA UN CONTATORE PER "MISURARE" LA LUNGHEZZA DELLA STRINGA IN INPUT ED UNO PER L'OUTPUT

COSTO UNIFORME:

$T(n) = \Theta(n^2)$

COSTO LOGARITMICO:

$T(n) = \Theta(n^2 \log n)$

Theoretical Computer Science Exam, January 24, 2020

The exam consists of **4 exercises**. Available time: 1 hr 40 min.; you may write your answers in Italian, if you wish.

All exercises must be addressed for the written exam to be considered sufficient.

F.NAME …….. L.NAME …….. MATRICOLA ……

I have passed the midterm exam and I intend to skip the first part (Ex.1-2) (select one): **YES NO**

Exercise 1 (8 pts)

Consider a given alphabet Σ. Given any two strings $x, y \in \Sigma^*$, let us say that y is a substring of x iff y can be obtained from x by removing any of its characters (also all or none of them). So, for instance, the substrings of $x = abaa$ are exactly the following: ε, a, b, ab, ba, aa, aba, aaa, baa, $abaa$.

Given any language L on alphabet Σ, that is, $L \subseteq \Sigma^*$, let us define ***SUBSEQ(L)*** as the language that includes exactly the substrings of all the strings of L :

$$SUBSEQ(L) = \{ y \in \Sigma^* \mid y \text{ is a substring of } x, \text{ for some } x \in L \}$$

Now let us consider a unary alphabet Σ, that is, $|\Sigma| = 1$. Prove, or disprove, the following statement:

For ***every*** language L on a unary alphabet, $SUBSEQ(L)$ is a regular language.

Please be at the same time clear and concise in your answer. (for additional space, use the last page)

Exercise 2 (8 pts)

Write a Monadic First Order (or Monadic Second Order only if necessary) logic formula characterizing the following languages

1) $L_1 = \{ x \in \{a, b, c\}^* \mid x$ contains (at least) one pair of consecutive a's $\}$ For instance, $cbcaab \in L_1$, $aba \notin L_1$, $cbaabcaab \in L_1$, $cbaaaab \in L_1$.
2) $L_2 = \{ x \in \{a, b, c\}^* \mid x$ contains no more than one pair of consecutive a's $\}$ For instance, $cbcaab \in L_2$, $aba \in L_2$, $cbaabcaab \notin L_2$, $cbaaaab \notin L_2$.
3) $L_3 = \{ x \in \{a, b, c\}^* \mid x$ contains an odd number of pairs of consecutive a's $\}$ For instance, $aabca \in L_3$, $aba \notin L_3$, $baaabc \notin L_3$, $cbaaaab \in L_3$. For this language, in addition, show the value of the predicates used in the formula at each position of the following words: $aabca$, $baaabc$, and $cbaaaab$.

Exercise 3 (8 pts)

Prove that the family of semidecidable sets is closed under intersection, i.e., for every set S_1 and S_2, both semidecidable, their intersection $S_{12} = S_1 \cap S_2$ is also semidecidable. To this end, answer the following questions.

1. Outline a procedure $semDec_{S12}$ that "semidecides" S_{12}, i.e., such that, for $x \in S_{12}$ $semDec_{S12}(x)$ terminates and returns true, while for $x \notin S_{12}$ $semDec_{S12}(x)$ does not terminate; you can assume the existence of similar procedures $semDec_{S1}$ and $semDec_{S2}$ that semidecide S_1 and S_2.
2. Outline a procedure that enumerates S_{12}.
3. Under the assumption that $S_{12} \neq \emptyset$, write a definition of the generating function g_{S12} of S_{12} (hint: exploit the fact that the generating functions g_{S1} of S_1 and g_{S2} of S_2 exist; make sure that g_{S12} is total computable and its image is S_{12}).

Exercise 4 (8 pts)

1. Describe a 1-tape Turing machine that takes as input a string of type $a^n b^m$, with $n = 3^k$, $k \geq m$, $m > 0$, and outputs a^h, with $h = \frac{3^k}{3^m} = 3^{k-m}$. For instance, when the input is *aaab* the output is *a*; when the input is $a^{27}b^2$ the outpt is a^3. For the sake of simplicity, ignore the case in which the input is not a string of the specified type. Analyze the time and space complexity of this Turing machine, still ignoring the case in which the input is not a string of the specified type.
2. Describe a single tape Turing machine that accepts the language $\{ a^n \mid n = 3^k, k \geq 0\}$. Analyze its time and space complexity.

1 L'ENUNCIATO È VERO. CONSIDERIAMO, AD ESEMPIO, $\Sigma = \{a\}$ E IL LINGUAGGIO L COSTRUITO SU Σ. L PUÒ ESSERE O FINITO O INFINITO

FINITO

LA LUNGHEZZA MASSIMA DI UNA STRINGA È DEFINITA $\max_{x \in L} |x| = n$ PER DEGLI $n \in \mathbb{N}$ E $a^n \in L$. LE SOTTOSTRINGHE DI a^n SONO DEFINITE NELL'INSIEME $S = \{a^k \mid 0 \leq k \leq n\} = SUBSEQ(L)$ ED, ESSENDO FINITO, $SUBSEQ(L)$ È NECESSARIAMENTE REGOLARE

INFINITO

$\forall n \in \mathbb{N}\ \exists n' > n \mid a^n \notin L$. OSSIA, PER OGNI n ESISTE UN NUMERO n' MAGGIORE TALE CHE $\{a^k \mid 0 \leq k \leq n'\}$ È UN SOTTOINSIEME DI $SUBSEQ(L)$, PER CUI $SUBSEQ(L) = \{a^k \mid k \in \mathbb{N}\}$, CHE ACCADE NON È CHE Σ^* STESSO, PER CUI $SUBSEQ(L)$ È REGOLARE.

2 a $\exists x (\neg last(x) \wedge a(x) \wedge a(x+1))$

b $\exists x (\neg last(x) \wedge a(x) \wedge a(x+1) \wedge \forall y (\neg last(y) \wedge a(y) \wedge a(y+1)) \leftrightarrow y = x)$

c DEFINIAMO IL PREDICATO $D(x)$ "NUMERO DISPARI DI COPPIE DI a"

$\neg D(0) \wedge \exists x ((\neg last(x) \wedge a(x) \wedge a(x+1) \wedge \neg D(x)) \rightarrow D(x+1)) \wedge$
$\leftrightarrow D(x))$
$\wedge ((\neg last(x) \wedge a(x) \wedge a(x+1) \wedge D(x) \rightarrow \neg D(x+1)) \wedge (last(x)$

3 a SIANO semDecs1 E semDecs2 CHE VERIFICANO SE UN INPUT $x \in S_1$ E S_2 (RISPETTIVAMENTE). semDecs1 RESTITUIRÀ 1 QUANDO SIA semDecs1 CHE semDecs2 RESTITUISCONO 1

b LA VALUTAZIONE DI semDecs12 PUÒ PASSARE APPLICANDO LA SIMULAZIONE DOVETAILING

c $g_{S_{12}}(x) = \begin{cases} g_{S_1}(x) & g_{S_1}(x) = g_{S_2}(y) \text{ O } g_{S_2}(y) \\ K & \text{ALTRIMENTI} \end{cases}$

4 a È SUFFICIENTE UNA MACCHINA DI TURING A SINGOLO NASTRO CHE LEGGE LA STRINGA IN INPUT E, PER OGN' $n \in \mathbb{N}$, IL NASTRO SI SPOSTA DI UNA POSIZIONE A DESTRA PER GARANTIRE CHE $m \leq K$ (E QUINDI $m \leq \log n$) E STAMPARE a^h. NEL CASO PESSIMO, $m = K$ (E $h = 0$). $T(n) = \Theta(n \cdot m) = \Theta(n \log n)$ $S(n) = \Theta(h)$

b ESEGUITE K SCANSIONI DELL'INPUT E, PER OGNI a, CI SI ASPETTA DI "CONTARNE ALTRE DUE". $T(n) = \Theta(n \log n)$ E $S(n) = \Theta(n)$

Theoretical Computer Science Exam, February 18, 2020

The exam consists of **4 exercises**. Available time: 1 hr 50 min.; you may write your answers in Italian, if you wish.

All exercises must be addressed for the written exam to be considered sufficient.

F.NAME …….. L.NAME …….. MATRICOLA ……

I have passed the midterm exam and I intend to skip the first part (Ex.1-2) (select one): **YES NO**

Exercise 1 (8 pts)

1) Let L be any regular language on alphabet A, and let a be any symbol of A. Then consider the language L' that includes exactly (i.e., all and only) the strings of L that include at least one occurrence of symbol a. Is language L' regular? Suitably motivate your answer.
2) Consider now the context-free language $L_1 = \{ (a \mid b)^n c^n \mid n > 0 \}$.
   a) Design a PDA that accepts L_1.
   b) Consider next the language L_1' that includes exactly (i.e., all and only) the strings of L_1 that include at least one occurrence of symbol a. Is language L_1' regular? Suitably motivate your answer and design the minimum power automaton that accepts L_1'.
   c) (Optional) Generalize your previous answer by considering any language $L_i = L_c \cap L_r$ equal to the intersection between a context-free language L_c and a regular language L_r : what is the smallest language family to which L_i belongs? Suitably motivate your answer.

Exercise 2 (8 pts)

Consider a subprogram having four parameters: an array of integers *in*, its size *nIn*, an array of integers *out*, its size *nOut*. The input parameters are: *in*, *nIn*, and *nOut*, whereas the array *out* is an output parameter. The size (i.e., number of components) of the two arrays can be assumed to have the same value, greater than 2, and all the elements of array *in* are assumed to be strictly positive.

After execution of the subprogram all even-valued elements of *in* must be stored at some odd-index position (index 1, 3, etc.) of array out, and all odd-valued elements of *in* must be stored at some even-index position (index 0, 2, etc.) of array out. Notice that indices of array positions are assumed to start from 0. For instance, the specification is satisfied by the following values: *in*=[2, 5, 3, 4], *out*=[3, 4, 5, 2]; it is *not* satisfied by the following ones: *in*=[2, 5, 3, 4], *out*=[3, 5, 4, 2]. Answer the following questions:

1. write pre- and post-conditions specifying the requirements of the above described procedure;
2. is the specification always satisfiable? If not, say in which cases it is not satisfiable;
3. modify the precondition in such a way that the postcondition can always be made true by a suitable correct implementation.

Exercise 3 (8 pts)

Is it decidable whether a generic sentence of the Monadic First Order logic on strings is satisfiable or not? Explain clearly your answer.

Exercise 4 (8 pts)

Consider the following language

$$L = \{\, a^{h_1} b^{k_1} a^{h_2} b^{k_2} \ldots a^{h_n} b^{k_n} \mid n > 0 \ \wedge \forall i\, (1 \le i \le n \rightarrow h_i > k_i) \wedge \forall i\, (1 \le i < n \rightarrow k_i > h_{i+1}) \,\}$$

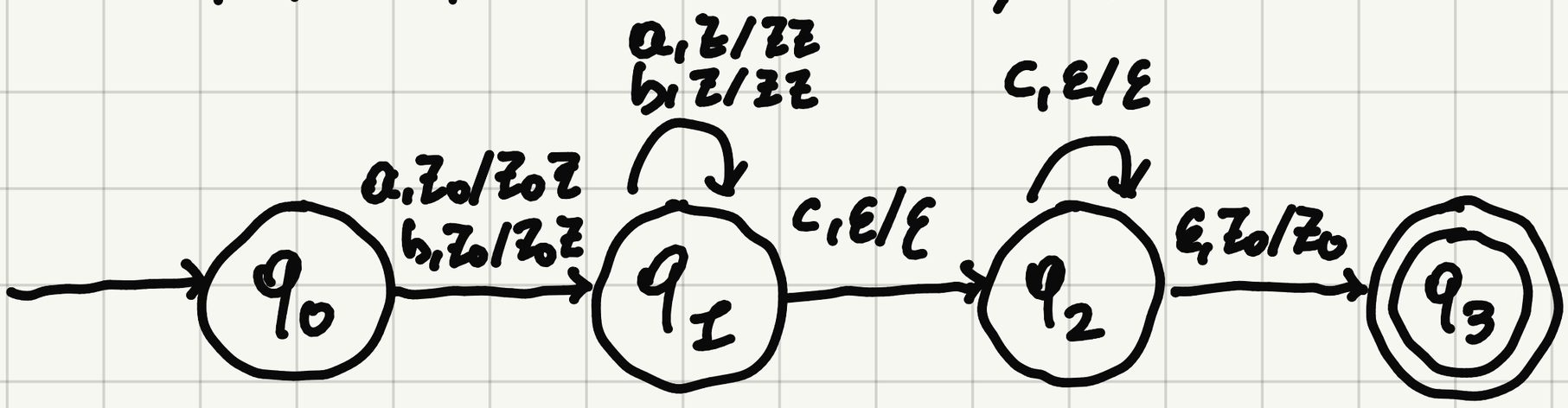
The language strings are alternated sequences of letters ***a*** and ***b***; the number of sequences is positive and even; each sequence of equal letters (be it made of ***a***'s or ***b***'s) is shorter than all the preceding ones and longer than all the following ones. Thus, for instance, $aaaaabbbaab \in L$, $aaaabbb \in L$, while $\varepsilon \notin L$, $aaabbbb \notin L$, $aaaabba \notin L$, $aaaabbbbaaab \notin L$.

1. Design, preferably in full detail, or at least describe in a clear and sufficiently detailed way, a 1-tape Turing machine (i.e., having one memory tape) that accepts language L and determine its time and space complexity class. Also determine the worst case and best case space complexity function in case the input string belongs to the language L. Provide suitable justifications for your answers.
2. Describe, in a clear and and sufficiently detailed way, a RAM machine accepting the same language L, and determine its time and space complexity class, both adopting the uniform and the logarithmic cost criterion. Provide suitable justifications for your answers.

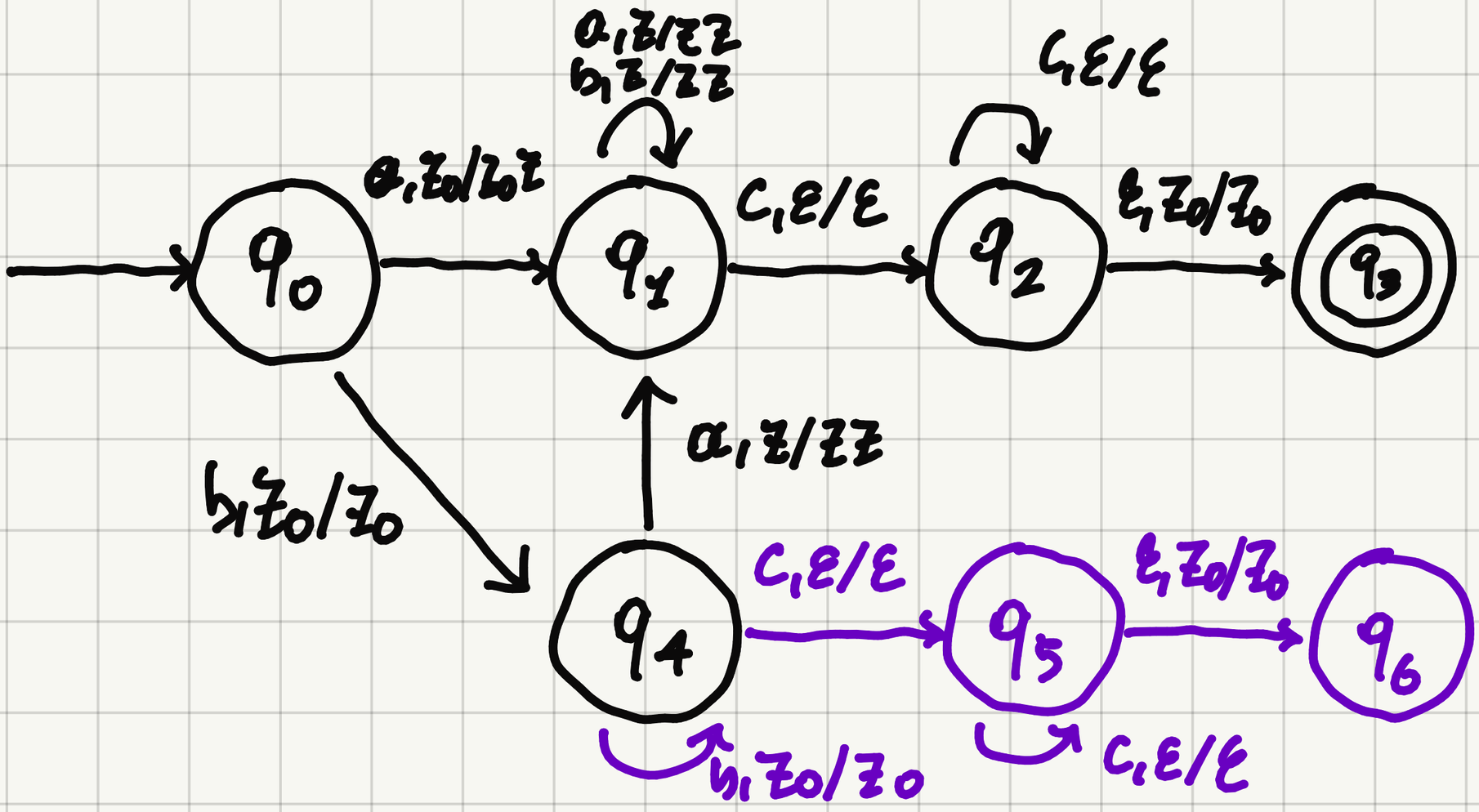
1

a DEFINIAMO L'' COME IL LINGUAGGIO(REGOLARE) DI TUTTE LE STRINGHE CON ALMENO UNA a. ESSENDO I LINGUAGGI REGOLARI CHIUSI RISPETTO ALL'INTERSEZIONE. L' = L ∩ L'' È REGOLARE

b
- SI NOTI LA SCRITTA "a|b", OSSIA a OR b



- UN LINGUAGGIO REGOLARE È RICONOSCIUTO DA UN AUTOMA A STATI FINITI. ANCHE CON LA CONDIZIONE IN a, OCCORRE COMUNQUE CONOSCERE IL NUMERO DI a E b, E QUINDI UN FA NON BASTA.



CASO IN CUI NESSUNA a VIENE LETTA

- L_i È UN LINGUAGGIO LIBERO DAL CONTESTO. L'AUTOMA A PILA CHE LO ACCETTA DI L_c ED L_r PUÒ ESSERE OTTENUTO DA L_c CON UN PRODOTTO CARTESIANO TRA GLI STATI

2 INPUT:

- in ARRAY DI DIMENSIONE nIn DI ELEMENTI POSITIVI
- nIn E nOut DIMENSIONI DEI VETTORI (UGUALI)

OUTPUT:

- OUT ARRAY DI DIMENSIONE nOUT > 2

a

PRECONDIZIONI] $(nIn>2)\wedge(nOUT=nIn)\wedge(\forall i(0\leq i<n\rightarrow in[i]>0)$

POSTCONDIZIONI] DEFINIAMO LE FUNZIONI $even(i)\leftrightarrow\exists k(i=2k)$ E

$odd(i)\leftrightarrow\exists k(i=2k+1)$

$\forall i(0\leq i<nIn\wedge even(in[i])\rightarrow\exists j(0\leq j<nOut\wedge odd[j]$

$\wedge in[i]=out[j])\wedge(0\leq i<nIn\wedge odd(in[i])\rightarrow$

$\rightarrow\exists j(0\leq j<nOut\wedge even(j)\wedge in[i]=out[j]))$

b LE POSTCONDIZIONI NON SONO COMPLETE IN QUANTO NON TENGONO CONTO DEL FATTO CHE:

- nIn PUÒ ESSERE DISPARI
- POSSONO ESSERCI PIÙ/MENO NUMERI PARI

c UNA POSSIBILE MODIFICA PUÒ ESSERE FATTA NELLE PRECONDIZIONI:

$nOut \geq 2\cdot nIn$

3 IL PROBLEMA CONSIDERATO È **DECIDIBILE** POICHÈ È POSSIBILE DEFINIRE UN ALGORITMO CHE, DATA UNA FRASE IN M.F.O., PUÒ COSTRUIRE UN EQUIVALENTE AUTOMA A PILA.

4 a UNA MACCHINA DI TURING LEGGE LE PRIME α E SALVA LA SUA LUNGHEZZA COMPIENDO h_1 PASSI A DESTRA. SUCCESSIVAMENTE, LEGGE LA SEQUENZA h SUCCESSIVA E SPOSTA IL NASTO VERSO SINISTRA.

IN QUESTO MODO, TUTTE LE SUCCESSIVE SEQUENZE RISPETTANO LE CONDIZIONI IMPOSTE SUGLI ESPONENTI.

SIA n LA LUNGHEZZA DELLA STRINGA:

$T(n) = \Theta(n)$ E $S(n) = \Theta(n)$. PER LA COMPLESSITÀ SPAZIALE.

$S(n) = n-1$ NEL CASO PESSIMO E $S(n) = \sqrt{n}$ NEL CASO OTTIMO

b UNA MACCHINA RAM LEGGE IN INPUT LA SEQUENZA E USA DEI CONTATORI PER EFFETTUARE I CONTROLLI OPPORTUNI.

COSTO UNIFORME:

$T(n) = \Theta(n)$

$S(n) = \Theta(1)$

COSTO LOGARITMICO:

$T(n) = \Theta(n \log n)$

$S(n) = \Theta(\log n)$