

York University
CSE 2001 Fall 2017 – Midterm E
Instructor: Jeff Edmonds

1. Consider the alphabet $\Sigma = \{a, b\}$. Describe in English a property defining which strings are in the language Σ^*a .

- Answer: The language consists of all strings of a and b that end in an a .

2. Short Answers: What does the following mean, is it true, and very quickly why?

- “The regular languages are closed under intersection”
- “Every subset of a regular language is regular.”

- Answer:

(a) If L_1 and L_2 are regular then so is $L_1 \cap L_2$. True. Proof: Given DFA M_1 and M_2 we construct a DFA such that $L(M) = L(M_1) \cap L(M_2)$.

(b) If L_1 is regular and $L_2 \subseteq L_1$, then L_2 is regular. False: $0^n 1^n \subseteq \{0, 1\}^*$, the second is regular and the first is not.

3. First Order Logic: We state as follows that a Universal TM is a TM $M_{universal}$ is given a description of a TM M and its input I and simulates this computation.

\exists a TM $M_{universal}$, \forall inputs $I_{universal} = \langle M, I \rangle$, $M_{universal}(M, I) = M(I)$.

How is the *First Order Logic Game* game played for this statement?

Explain how this game states whether $M_{universal}$ is likely to have more states or fewer than M .

- Answer: First Order Logic Game:

We construct $M_{universal}$

The adversary gives us an arbitrary input $I_{universal} = \langle M, I \rangle$

(Knowing $M_{universal}$ and hence M can be much bigger than $M_{universal}$)

Prove $M_{universal}(M, I) = M(I)$.

4. **Turing Machine:** (25 marks) Write all the transition rules for a Turing Machine that solves the computational problem that takes as input a string of characters from $\{0, 1, \dots, 9\}$ and replaces any occurrence of 666 with 667. Longer blocks of 6 are not changed.

Input: 943666982796666293666

Output: 943667982796666293667

Pseudo Code:

0: Put head on first character.

$c = \text{getCharAtHead}()$

$\#6 = 0$

1: loop ***loop-invariant***: The variable $\#6$ holds the number of 6 in the continuous block₁ to the left of the head and that we are looking at a new character c . Previous changes have been made.

if($\#6 = 3$ and $c \neq 6$) then we have see 666

 Go back to change 666 to 667

 i.e. move left back to char 6 and go to line 2

elseif($c = \text{blank}$)

 Halt

elseif($c = 6$) Current block of 6 has another

$\#6 = \min(\#6 + 1, 4)$

```

else then current block of 6 has ended
    #6 = 0
    Move right to next character
    c = getCharAtHead()
end loop
2: Subroutine: Change 666 to 667
    writing a 7
    move right back to character we were just on
    #6 = 0
    goto line 1

```

- (a) Translate this code into Turing Machine Transitions.
 (b) How many states does your TM have?

- Answer: The first thing to know is the form of the transition function is $\delta(q, c) = \langle q', c', direction \rangle$, i.e. when in current state is q and the character on the tape being seen by the head is c , then transition to state q' , write c' onto the tape, and move the head in the direction indicated.

The second thing to know is that the names of the states should indicate which line the pseudo code is in and what values are being saved on pooh bear's black board.

- (a) Turing Machine Transitions Start State: $q_{\langle line=1, \#6=0 \rangle}$
 $i = 3$ and $\forall c \in \{0, \dots, 9\} - 6 + blank$
 $\delta(q_{\langle line=1, \#6=3 \rangle}, c) = \langle q_{\langle line=2 \rangle}, c, left \rangle$
 $\forall i \in \{0, 1, 2, 4\}$ and $c = blank$
 $\delta(q_{\langle line=1, \#6=i \rangle}, blank) = \langle q_{\langle line=halt \rangle}, blank, stay \rangle$
 $\forall i \in \{0, 1, 2, 3, 4\}$ and $c = 6$
 $\delta(q_{\langle line=1, \#6=i \rangle}, 6) = \langle q_{\langle line=1, \#6=\min(i+1, 4) \rangle}, 6, right \rangle$
 $\forall i \in \{0, 1, 2, 4\}$ and $\forall c \in \{0, \dots, 9\} - 6$
 $\delta(q_{\langle line=1, \#6=i \rangle}, c) = \langle q_{\langle line=1, \#6=0 \rangle}, c, right \rangle$
 Have gone back to write the 7.
 $\delta(q_{\langle line=2 \rangle}, 6) = \langle q_{\langle line=1, \#6=0 \rangle}, 7, right \rangle$
- (b) It has $5 + 1 + 1 = 7$ states counting the halt state.

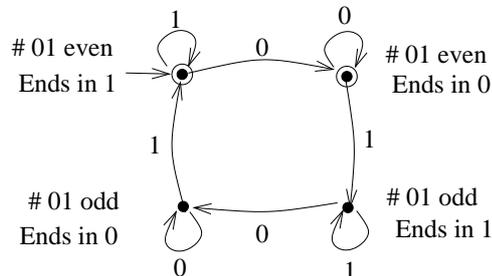
5. DFA: Draw a DFA for the following language.

$L = \{ \alpha \in \{0, 1\}^* \mid \alpha \text{ contains the substring } 01 \text{ an even number of times} \}$.

Example: 1100110001100 contains 01 twice, two is even, and hence the string is in the language. 1100 does not contain it, zero is even, and hence the string is in the language.

Be sure to label the states of your DFA with names indicating what the DFA "knows" when in that state.

- Answer:



6. Bumping Lemma is stated as follows:

$[\exists \text{ infinite set } S, \forall \alpha, \beta \in S \text{ with } \alpha \neq \beta, \exists \zeta, L(\alpha\zeta) \neq L(\beta\zeta)] \Rightarrow [\forall \text{ DFA } M, \exists \text{ an input } I, M(I) \neq L(I)].$

(a) No need for a full proof, but give reasonable intuition to why it is true.

• Answer:

Answer from 2017: The conclusion $[\forall \text{ DFA } M, \exists \text{ an input } I, M(I) \neq L(I)]$ is that there is no DFA computing L .

Break each string into a first and a last name. Here S is a set of first names with α and β being examples, while ζ is a last name. After a DFA M reads one of the first names in S , it must remember precisely which of these it has read. It does this by going into a different state for each one. Hence, the number of states M has must be at least the size of $|S|$, which in this case is infinite.

If, on the other hand, M were confused to whether the first name it read was α or β , then it would also be confused as to whether the final string is $\alpha\zeta$ or $\beta\zeta$, where ζ is some last name. The statement of the claim is that the language L distinguishes between every pair $\alpha, \beta \in S$ of first names, namely that there is such a ζ such that the language L gives different answers to $\alpha\zeta$ and $\beta\zeta$. Hence, M will give the wrong answer to one of these.

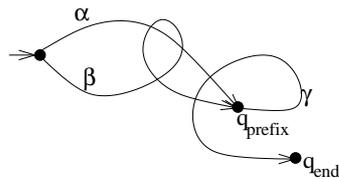
Answer from 1999:

Theorem 1 *If language L distinguishes the prefixes S , then any DFA that decides L requires at least $|S|$ states.*

The intuition is that after the DFA has read in some prefix in S , it needs enough memory to remember which one it has seen or equivalently it needs enough states to distinguish between them.

Corollary 2 *If language L distinguishes the prefixes S , where S contains an infinite number of prefixes, then L is not a regular language, because any DFA deciding it requires an infinite number of states.*

Proof of Theorem ??:



By way of contradiction, assume that the DFA M decides L and has fewer than $|S|$ states. For each prefix $\alpha \in S$, run M on it and place it on the state of M at which the machine ends up. The *pigeon hole principle* states that you can't put $n + 1$ pigeons into n pigeon holes without putting more than one pigeon in the same hole. By this principle, there is a state of M which we will denote q_{prefix} that receives two prefixes $\alpha, \beta \in S$. Because the prefixes α and β can be distinguished by L , there exists a postfix ζ such that $\alpha\zeta$ and $\beta\zeta$ are given different answers by L . I claim that the state q_{end} the M ends on when run on $\alpha\zeta$ is the same as when run on $\beta\zeta$. The reason is that after reading α and β , M is in the same state q_{prefix} and then it continues in the same way from there when reading ζ . This last state q_{end} is either an accept state or is not. Hence, it cannot give the correct answer on both input instances $\alpha\zeta$ and $\beta\zeta$, which have different answers. ■

(b) For each of these languages, either prove that it is not regular or prove that it is. Give the intuition.

- i. $L = \{a, b\}^n \# \{u, v\}^{(n \text{ or } (n+1))}$
- ii. $L = \{a, b\}^m \# \{u, v\}^{(n \text{ or } (n+1))}$

• Answer: The first is not regular because a DFA would need to count and remember the number of a or b in order to match it up with the number of u or v .

We have to design the infinite set S of first names. An obvious set of first names would be $S = \{a, b\}^n$. But the problem with this is that $\alpha = a^3$ and $\beta = b^3$ are definitely not distinguished. It is better to have a smaller set of first names, say $S = a^n$.

Concern: If $\alpha = a^3$, $\beta = a^4$, and $\zeta = u^4$, then both strings are yeses, namely $L(\alpha\zeta) = L(a^3u^{3+1}) = \text{yes}$ and $L(\beta\zeta) = L(a^4u^4) = \text{yes}$.

Avoiding Concern:

Let $S = \{a^{3n} \mid n \geq 0\}$.

Note then α and β can't be one apart like in the concerning example.

Let $\alpha = a^{3i}$ and $\beta = b^{3j}$ be arbitrary strings in S with $i \neq j$.

Let $\zeta = \#u^{3i}$.

Note $L(\alpha\zeta) = L(a^{3i}\#u^{3i}) = \text{yes}$, because we designed it this way.

We want to prove that $L(\beta\zeta) = L(a^{3j}\#u^{3i}) = \text{no}$.

In order for this to be a yes, because the number of a 's is $3j$, the number of b 's needed would be $3j$ or $3j + 1$. But the number of b 's we have is $3i$. Note that neither $3j$ or $3j + 1$ is equal to $3i$ because i and j are different and are integers.

Hence, $L(\beta\zeta) = L(a^{3j}\#u^{3i}) = \text{no}$.

Hence S is an infinite set of strings distinguished by the language.

Hence no DFA computes L .

Address the Concern:

Let $S = \{a^n \mid n \geq 0\}$.

Let $\alpha = a^i$ and $\beta = b^j$ be arbitrary strings in S with $i \neq j$.

Assume without loss of generality that $i > j$. Let $\zeta = \#u^{i+1}$.

Note $L(\alpha\zeta) = L(a^i\#u^{i+1}) = \text{yes}$, because we designed it this way.

We want to prove that $L(\beta\zeta) = L(a^j\#u^{i+1}) = \text{no}$.

In order for this to be a yes, because the number of a 's is j , the number of b 's needed would be j or $j + 1$. But the number of b 's we have is $i + 1$. Note that neither j or $j + 1$ is equal to $i + 1$ because $i > j$.

Hence, $L(\beta\zeta) = L(a^j\#u^{i+1}) = \text{no}$.

Hence S is an infinite set of strings distinguished by the language.

Hence no DFA computes L .

The second language $L = \{a, b\}^m \# \{u, v\}^n$ or $(n+1)$ is regular because the two blocks are not linked. A regular expression for it is $\{a, b\}^* \# \{u, v\}^*$.

7. For each of these languages, either give a context free grammar for it or argue that it can't be done.

(a) $L = 0^{2m+7} a^n 1^{2m} \{x, y\}^s \{u, v\}^{3n+6}$

(b) $L = 0^{2n+7} a^m 1^{2m} \{x, y\}^s \{u, v\}^{3n+6}$

- Answer: The following is a CFG for the second.

We give non-terminal names to the blocks as follows.

$$S = 0^{2n+7} a^m 1^{2m} \{x, y\}^s \{u, v\}^{3n+6}$$

$$U = \{u, v\}$$

$$T = a^m 1^{2m}$$

$$X = \{x, y\}^s.$$

The grammar rules are then as follows.

$$S \Rightarrow 00SUUU$$

$$S \Rightarrow 000000TXUUUUUU$$

$$U \Rightarrow u|v$$

$$T \Rightarrow aT11$$

$$T \Rightarrow \epsilon$$

$$X \Rightarrow xX|yX$$

$$X \Rightarrow \epsilon$$

The first cannot be expressed by a CFG because the blocks with parameters $mnmsn$ have the $m - m$ link crossing over the $n - n$ link.