NAME: _____

COMPSCI 250
Introduction to Computation
Final Exam Spring 2019 – Solutions

M. Minea and S. Reddy 3 May 2019

DIRECTIONS:

- Answer the problems on the exam pages.
- There are four problems on pages 2-7, each with multiple parts, for 110 total points plus 10 extra credit. Probable scale is somewhere around A=100, C=70, but will be determined after we grade the exam.
- If you need extra space use the back of a page.
- No books, notes, calculators, or collaboration.

Question 1 (30): Last month Cardie and Duncan were invited to an Easter Egg Hunt with their neighbors Shelby, Vianna, and Whistle. The set of dogs in the hunt was $D = \{c, d, s, v, w\}$. The unary predicate $T(x)$ on D means "dog x is a terrier". Each dog x found a natural number of eggs $f(x)$, and the binary predicate $ME(x, y)$ on D means "dog x found strictly more eggs than dog y" or equivalently "$f(x) > f(y)$".

a. (10) Translate each of these four statements as indicated.

Statement I (to English): $T(d) \wedge ME(v, d)$

Duncan is a terrier and Vianna found more eggs than Duncan.

Statement II (to symbols): Each terrier found more eggs than each non-terrier.

$\forall x : \forall y : ((T(x) \wedge \neg T(y)) \rightarrow ME(x, y))$

Statement III (to English): $\exists x : \exists y : (x \neq y) \wedge \neg ME(x, y) \wedge \neg ME(y, x)$

There exist two different dogs such that neither found more eggs than the other.
(This implies: there exist two different dogs who found the same number of eggs).

Statement IV (to symbols):
Neither Cardie nor Whistle are terriers, and each found more eggs than did Shelby.

$\neg T(c) \wedge \neg T(w) \wedge ME(c, s) \wedge ME(w, s)$

b. (10) Prove that Vianna is a terrier who found at least three eggs.

From Statement II by Specification ($x = d$, $y = v$), we have $T(d) \wedge \neg T(v) \rightarrow ME(d, v)$. This is equivalent to $\neg T(d) \vee T(v) \vee ME(d, v)$, and further to $T(d) \rightarrow (\neg ME(d, v) \rightarrow T(v))$ (1). From (I), we have $T(d)$ (2) and $ME(v, d)$ (3) by Separation. Modus Ponens from (1) and (2) yields $\neg ME(d, v) \rightarrow T(v)$ (4). (3) is equivalent to $f(v) > f(d)$, which implies $\neg(f(d) > f(v))$, which is $\neg ME(d, v)$ (5). From (4) and (5) by Hypothetical Syllogism we have $T(v)$.

By Separation from IV we get $\neg T(c)$ and $ME(c, s)$, thus $f(c) > f(s)$. Specification of II ($x = d$, $y = c$) yields $T(d) \wedge \neg T(c) \rightarrow ME(d, c)$. By Conjunction, $T(d) \wedge \neg T(c)$ and then by Modus Ponens, $ME(d, c)$, thus $f(d) > f(c)$, so $f(d) \geq f(c) + 1$. $M(v, d)$ is $f(v) > f(d)$, so $f(v) \geq f(d) + 1 \geq f(c) + 2$. Since $f(c) > f(s) \geq 0$, we get $f(c) \geq 1$ and $f(v) \geq 3$.

c. (10) What is the smallest total number of eggs (found by all dogs) if Statement III is false, while the other statements are true? What if we know that Shelby found an egg?

The same arguments for Cardie apply to Whistle, so we get $f(d) > f(w) > f(s)$.

The negation of (III) can be rewritten as $\forall x : \forall y : (x \neq y \rightarrow (ME(x, y) \vee ME(y, x))$. Specifying $x = c$, $y = w$ yields $ME(c, w) \vee ME(w, c)$ (by Modus Ponens, as $c \neq w$). Asume $f(w) > f(c)$ (the other case is symmetric). We then have $f(v) > f(d) > f(w) > f(c) > f(s) \geq 0$. If S is the total number of eggs found by all dogs, this gives us $S \geq 5f(s)+4+3+2+1 = 5f(s)+10$. The smallest total number of eggs is 10. If $f(s) \geq 1$, the smallest total is 15.

Question 2 (30):

a. (10p) A sequence of integers a_n is defined recursively by $a_0 = 0$, $a_1 = 1$, $a_n = a_{n-1} - 2a_{n-2}$ for $n > 1$. Find and prove a theorem stating for which values of n we have $a_n \equiv 0 \pmod 3$.

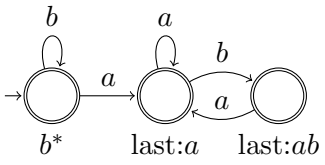
For $n \geq 4$, we can rewrite: $a_n = a_{n-1} - 2a_{n-2} = a_{n-2} - 2a_{n-3} - 2a_{n-2} = -a_{n-2} - 2a_{n-3} = -a_{n-3} + 2a_{n-4} - 2a_{n-3} = 2a_{n-4} - 3a_{n-3}$. Thus, $a_n \equiv 2a_{n-4} \pmod 3$. Since 2 and 3 are relatively prime, this implies $a_n \equiv 0 \pmod 3 \leftrightarrow a_{n-4} \equiv 0 \pmod 3$. We have $a_2 = 1$, $a_3 = -1$. We can now state that $a_n \equiv 0 \pmod 3 \leftrightarrow n \equiv 0 \pmod 4$. We prove this by strong induction. The statement is true for $n < 4$. Taking $n \geq 4$, and assuming it true for all $k < n$, we have $a_n \equiv 0 \pmod 3 \leftrightarrow a_{n-4} \equiv 0 \pmod 3 \leftrightarrow n - 4 \equiv 0 \pmod 4 \leftrightarrow n \equiv 0 \pmod 4$, q.e.d.

We could also do a proof by cases stating the value of $a_n \pmod 3$ for each of the four values of $n \pmod 4$, but avoiding case splitting is shorter. Note that we proved not just the implication $n \equiv 0 \pmod 4 \rightarrow a_n \equiv 0 \pmod 3$, but equivalence, these are the only values divisible by 3.

b. (10p) Show by induction that if A is a regular expression then $L(A)^R$, the set of reversals of strings in $L(A)$, is a regular language.

See proof in book (sec. 5.5.2)

c. (10p) No-abb is the language of strings over $\Sigma = \{a, b\}$ that have no *abb* substring. An NFA for No-abb with properties of strings reaching each state is shown. Prove by induction that the number of strings of length n in No-abb is $S_n = F_{n+3} - 1$, where F_n is the Fibonacci sequence ($F_0 = 0, F_1 = 1, F_{n+1} = F_n + F_{n-1}$ for $n > 1$).

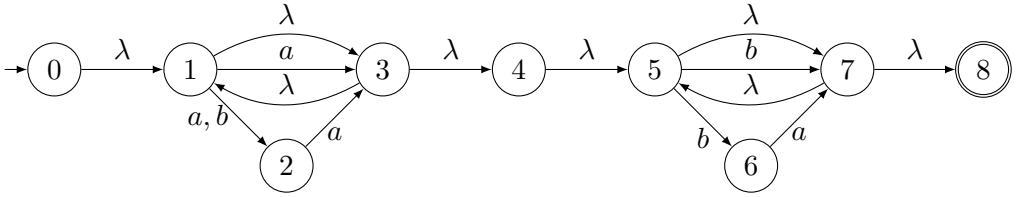


Hint: Consider the last letter(s) and write recurrences for the number of No-abb strings that end in a and ab respectively. They are also related to the Fibonacci sequence.

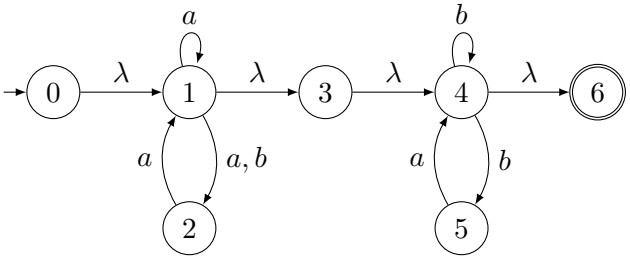
As the NFA shows, a No-abb string either has a last, or has a next-to-last (ends in ab), or contains no a (since if the last a is earlier, it would be followed by at least two b's). There is exactly one string of n b's, so we can write: $S_n = A_n + AB_n + 1$. No-abb strings can end in a or ab precisely if the remaining string is in No-abb. Thus, we have $A_n = S_{n-1}$, $AB_n = S_{n-2}$, and $S_n = S_{n-1} + S_{n-2} + 1$. We can now prove the statement by strong induction. We have $S_0 = 1 = F_3 - 1$, and $S_1 = 2 = F_4 - 1$ (all strings of length < 2 are in No-abb). Taking $n \geq 2$ and assuming $S_k = F_{k+3}$ for $k < n$, we get $S_n = S_{n-1} + S_{n-2} + 1 = F_{n+2} - 1 + F_{n+1} - 1 + 1 = F_{n+2} + F_{n+1} - 1 = F_{n+3} - 1$ and the statement is proved.

Question 3 (30): Do the following constructions for the language $L = (a + (a + b)a)^*(ba + b)^*$.

a. (5) Find a λ-NFA N whose language is given by the regular expression $(a+(a+b)a)^*(ba+b)^*$. For full credit, use the construction from the lecture and text *exactly*, without simplifications; you may use the alternate Kleene star construction given in the lecture.



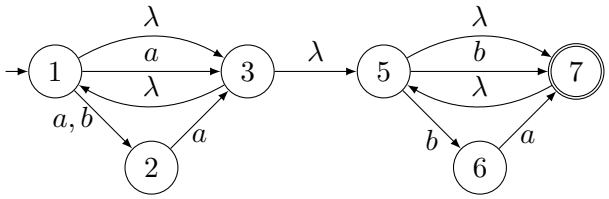
"Compressed" self-loop construction:



b. (10) Using the construction from the text on your λ-NFA N from part (a), build an ordinary NFA N' such that $L(N') = L(N)$.
You may make simplifications of N before doing this, as long as they do not change the language of the machine and you clearly argue that what you are doing is correct.

The two λ-transitions from the initial state and into the final state do not affect the language. The two central λ-moves can only be taken in sequence; we merge them, removing one state.

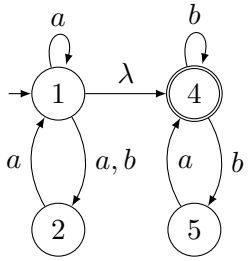
Full star-construction:



The transitive closure of λ-transitions is:
$1 \to 1,3,5,7 \quad 3 \to 1,3,5,7, \quad 5 \to 5,7 \quad 7 \to 5,7.$

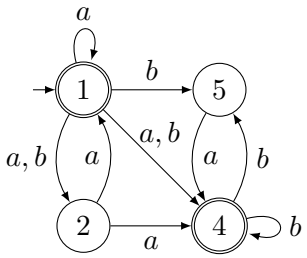
a - transitions	b - transitions
$(1,a,2)$: $1,3 \to 2$	$(1,b,2)$: $1,3 \to 2$
$(1,a,3)$: $1,3 \to 1,3,5,7$	$(5,b,6)$: $1,3,5,7 \to 6$
$(2,a,3)$: $2 \to 1,3,5,7$	$(5,b,7)$: $1,3,5,7 \to 5,7$
$(6,a,7)$: $6 \to 5,7$	(1) becomes final.

Compressed star-construction:



a - moves	b - moves
$1 \to 1,4$	$1 \to 2$
$1 \to 2$	$1,4 \to 4$
$2 \to 1,4$	$1,4 \to 5$
$5 \to 4$	(1) is final.

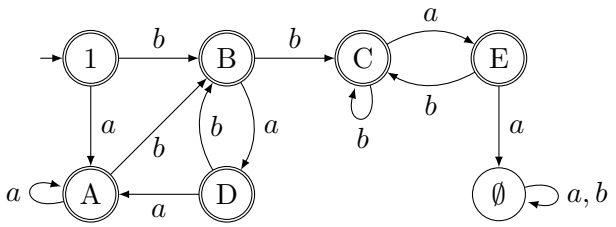
The NFA resulting from the "compressed" star construction is:



c. (5) Using the Subset Construction on N', find a DFA D such that $L(D) = L(N')$.

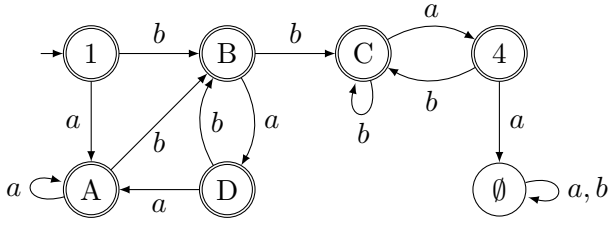
From "full" construction:

	a	b
1	12357	2567
A	12357	2567
B	2567	1357	567
C	567	57	567
D	1357	12357	2567
E	57	$\emptyset$	567



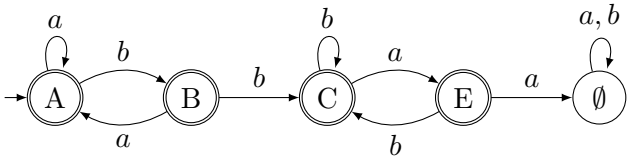
We get the same result from the "compressed" construction:

	a	b
1	124	245
A	124	124	245
B	245	14	45
C	45	4	45
D	14	124	245
4	$\emptyset$	45



d. (5) Find a minimal DFA D' with $L(D') = L(D)$. You may use the minimization construction, or prove directly that your D is already minimal.

States 1, A, and D are equivalent since they transition to A and B respectively. We merge 1 and D with A. The resulting automaton is:



We can also do the standard minimization construction, starting from two classes, f and n. At each step, the rightmost column shows the classes each state transitions to on a and b.

	a	b	
1	A	B	ff
A	A	B	ff
B	D	C	ff
C	E	C	ff
D	A	B	ff
E	∅	C	nf

	a	b	
1	A	B	ff
A	A	B	ff
B	D	C	ff
C	E	C	Ef
D	A	B	ff
E	∅	C	E

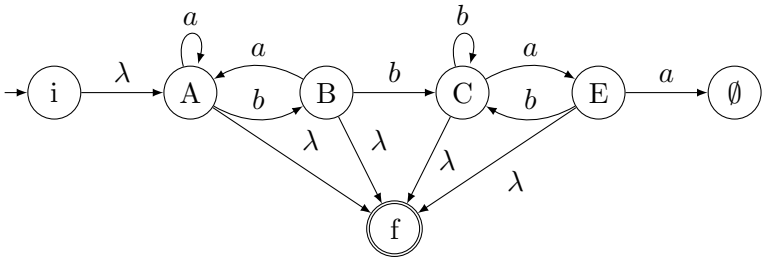
	a	b	
1	A	B	ff
A	A	B	ff
B	D	C	fC
C	E	C	C
D	A	B	ff
E	∅	C	E

	a	b	
1	A	B	ff
A	A	B	ff
B	D	C	B
C	E	C	C
D	A	B	ff
E	∅	C	E

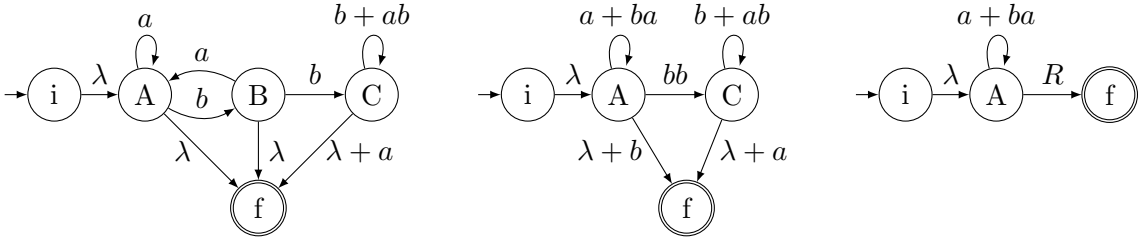
At this point, we can no longer separate states; the three states 1, A, and D are equivalent and we get the automaton depicted above.

e. (5) Using the state elimination construction on D or D', find a regular expression for $L(N)$. (The construction may produce a simpler or more complicated expression than the initial one).

We first insert a unique initial and a unique final state.



We eliminate ∅ (useless), then E and B (the nodes with no self-loops), and then C:



where $R = \lambda+b+bb(b+ab)^*(\lambda+a)$. The final language is $L = (a+ba)^*(\lambda+b+bb(b+ab)^*(\lambda+a))$.

You did not have to simplify this, but one can: we have repetitions of no-bb followed by no-aa. In part 2, we rewrite $b(b+ab)^*(\lambda+a) = b((\lambda+a)b)^*(\lambda+a) = (b(\lambda+a))^+ = (b+ba)^+$. We get $\lambda+b+b(b+ba)^+ = \lambda+b(\lambda+(b+ba)^+) = \lambda+b(b+ba)^*$. Then, $L = (a+ba)^*(\lambda+b(b+ba)^*) = (a+ba)^*(ba)^*(\lambda+b(b+ba)^*) = (a+ba)^*((ba)^* + (ba)^*b(b+ba)^*) = (a+ba)^*(b+ba)^*$.
We can see that in the initial regular expression, the $+aa$ in the first repetition is superfluous, being covered by $+a$, so $(a+aa+ba)^* = (a+ba)^*$.

Question 4 (30p)

The following are fifteen true/false questions, with no explanation needed or wanted, no partial credit for wrong answers, and no penalty for guessing.

a. In Question 1, neither the relation $ME(x, y)$ nor its complement $\neg ME(x, y)$ is a partial order.
FALSE, $\neg ME$ is the relation $\leq$ on naturals, which is a partial order.

b. Let $x > 1$ be an integer. Then propositions 2 and 3 are equivalent negations of proposition 1:
1. x is a composite number. 2. x is not a composite number. 3. x is a prime number.
TRUE, a number > 1 is defined as composite iff it is not prime.

c. If $P(0)$, $P(1)$ and $P(2)$ are true, and for all $n > 3$, $(P(n-4) \rightarrow P(n)) \vee (P(n-3) \rightarrow P(n))$ then $P(n)$ is true for all n.
FALSE, $P(3)$ could be false

d. If $a, b \in N, a > b$, and $a = bq + r$, then $\gcd(a, b) = \gcd(b, r)$.
TRUE, either if b is zero, or by Euclid's algorithm

e. The following is an equivalence relation over the set of all functions from $\mathbb{Z}$ to $\mathbb{Z}$:
$\{(f, g) \mid \exists c : \forall x : f(x) - g(x) = c\}$.
TRUE, $c = 0$ for reflexivity, $-c$ for symmetry, $c_1 + c_2$ for transitivity

f. Let $\mathcal{R}$ be the set of all regular languages and $\mathcal{D}$ the set of al DFAs, both with alphabet Σ. Then the function $L : \mathcal{D} \rightarrow \mathcal{R}$ defining the language of a DFA is onto, but not one-to-one.
TRUE, all regular languages are accepted by DFAs, but there are many DFAs for one language.

g. A regular expression for the set of strings containing a string of 1s so that the number of 1s equals 2 modulo 3, followed by an even number of 0s is: $11(111)^*(100)^*$.
FALSE, the answer is $11(111)^*(00)^*$

h. If L is a regular language and a is a symbol, then the language $\{w : wa \in L\}$ is regular.
TRUE, make any a-predecessor of an accepting state accepting.

i. Let $\Sigma = \{a, b\}$, $w \in \Sigma^*$, and L the language of all strings in Σ^* that end in w. Then the minimal DFA and the minimal NFA recognizing L may have the same number of states.
TRUE, if w is n a's or b's, we need $n + 1$ states (counting up to n for the DFA).

j. The language of descriptions of 2WDFAs that may loop on some input string is Turing decidable.
TRUE: If any string causes a loop, there is a short one (the number of configurations is bounded). We check all strings up to that length, and detect a loop by remembering configurations.

k. If X is Turing recognizable, and $X \setminus Y$ is not Turing recognizable, then Y is not Turing decidable.
TRUE, otherwise $\bar{Y}$ would be decidable, thus recognizable, and so would be $X \cap \bar{Y} = X \setminus Y$.

l. It may be that a language is not Turing recognizable, but its complement is Turing recognizable.
TRUE, we have seen this for the " Barber of Seville" language L_{BS}.

m. A Turing machine with tape symbols 0, 1, b (blank) and transition function $\delta(\iota, 0) = (h, 1, R)$, $\delta(\iota, 1) = (\iota, 1, R)$, $\delta(\iota, b) = (\iota, b, R)$ will replace the first 0 with a 1 and will not change any of the other symbols on the tape.
TRUE, this skips blanks and 1s to the first 0, writes a 1 and halts.

n. If we keep merging any DFA states p and q for which $\delta(p, x) = \delta(q, x)$ for any letter $x \in \Sigma$, we will obtain an equivalent minimal DFA.
FALSE, the automaton will be equivalent but we can't simplify a 3-cycle in this way.

o. Let L be the language of strings for which in any prefix, the count of a's and the count of b's are at most 2 apart. Then the L-equivalence relation has an infinite number of equivalence classes.
FALSE, it has six equivalence classes, for the integers from -2 to 2, and the rejected strings.