

For return on 24 January 2014 (late submission: 7 February 2014)

Electronic submission: pdf files only

1. (3%) Construct the truth-table for the Boolean function given by the Boolean formula

$$\neg(A \wedge (B \rightarrow C)) \wedge \neg B.$$

Use the truth-table to realise this function by a formula with the connectives \neg , \vee , \wedge only. Simplify the formula in such a way that the corresponding Boolean circuit contains a minimal number of gates. Show the Boolean circuit.

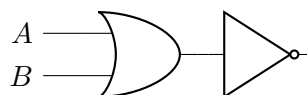
Truth Table

A	B	C	\neg	$(A \wedge (B \rightarrow C))$	\wedge	$\neg B$
0	0	0	1	0	1	1
0	0	1	1	0	1	1
0	1	0	1	0	0	0
0	1	1	1	0	0	0
1	0	0	0	1	0	1
1	0	1	0	1	0	1
1	1	0	1	0	0	0
1	1	1	0	1	0	0

Simplification

1. $(\neg A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge C)$
2. $(\neg(A \vee B) \wedge \neg C) \vee (\neg(A \vee B) \wedge C)$
3. $\neg(A \vee B) \vee (\neg C \wedge C)$
4. $\neg(A \vee B) \vee 0$
5. $\neg(A \vee B)$

Boolean Circuit



2. (3%) Are the following Boolean formulas equivalent? Explain your answer.

(a) $A \rightarrow (B \wedge C)$ and $(A \rightarrow B) \wedge (A \rightarrow C)$

A	B	C	A	\rightarrow	$(B \wedge C)$
0	0	0		1	0
0	0	1		1	0
0	1	0		1	0
0	1	1		1	1
1	0	0		0	0
1	0	1		0	0
1	1	0		0	0
1	1	1		1	1

A	B	C	$(A \rightarrow B)$	\wedge	$(A \rightarrow C)$
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	1	0	0
1	1	1	1	1	1

Yes they are equivalent

(b) $(A \wedge B) \rightarrow C$ and $(\neg C \rightarrow \neg A) \wedge (\neg C \rightarrow \neg B)$

A	B	C	$(A \wedge B)$	\rightarrow	C
0	0	0	0	1	
0	0	1	0	1	
0	1	0	0	1	
0	1	1	0	1	
1	0	0	0	1	
1	0	1	0	1	
1	1	0	1	0	
1	1	1	1	1	

A	B	C	$(\neg C \rightarrow \neg A)$	\wedge	$(\neg C \rightarrow \neg B)$
0	0	0	1	1	1
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	0	1	0

No they are not equivalent

(c) $(A \vee B) \rightarrow C$ and $(A \rightarrow B) \vee (A \rightarrow C)$

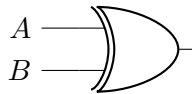
A	B	C	$(A \vee B)$	\rightarrow	C
0	0	0	0	1	
0	0	1	0	1	
0	1	0	1	0	
0	1	1	1	1	
1	0	0	1	0	
1	0	1	1	1	
1	1	0	1	0	
1	1	1	1	1	

A	B	C	$(A \rightarrow B)$	\vee	$(A \rightarrow C)$
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	1	1	1

No they are not equivalent

3. **(3%)** A parity function is a Boolean function whose value is 1 if the input has an odd number of ones. Design a Boolean circuit for the 2-bit parity function. Show your working. (Hint: you may find XOR gates useful.)

A	B	
0	0	0
0	1	1
1	0	1
1	1	0



4. **(3%)** Suppose $\alpha = a_{31}a_{30}\dots a_1a_0$ is a 32-bit binary word. Consider the 32-bit binary word $\beta = b_{31}b_{30}\dots b_1b_0$ computed by the following algorithm: scan α from right to left and copy its bits to β until the first 1 is found (which is also copied to β); after that, copy the Boolean negations of the bits in α . For example, $\alpha = 10100\dots 00$ is transformed to $\beta = 01100\dots 00$. Explain what this algorithm computes if α and β are interpreted as binary numbers.

Given a two's complement binary number, the algorithm computes the conversion into a binary number.

5. (6%) Given the machine 32-bit word

1100 0001 0011 0000 0000 0000 0000

find the decimal number represented by this word assuming that it is

(a) a two's complement integer;

1. Read the first character from the left of the word to get the sign of the integer it represents.
In this case it is a one so the number is negative.
2. Use the two's complement algorithm to convert it into a binary number. Starting from the right of the word, move to the first one :
* 1100 0001 0011 00...00
3. Move left, from the first on flipping the bit after the first one until you reach the far left:
* 1100 0001 0001 00...00
* 1100 0001 0101 00...00
* 1100 0001 1101 00...00
* 1100 0000 1101 00...00
* 1100 0010 1101 00...00
* 1100 0110 1101 00...00
* 1100 1110 1101 00...00
* 1101 1110 1101 00...00
* 1111 1110 1101 00...00
* 1011 1110 1101 00...00
* 0011 1110 1101 00...00
4. Then add all the base two 1's together to get the integer:
* $2^{29} + 2^{28} + 2^{27} + 2^{26} + 2^{25} + 2^{23} + 2^{22} + 2^{20} = 1053818880$
5. And multiply it by the sign, which is -1:
* $-1(1053818880) = -1053818880$

(b) an unsigned integer;

As an unsigned integer, add the base two ones together:
 $2^{31} + 2^{30} + 2^{24} + 2^{21} + 2^{20} = 3241148416$

(c) a single precision IEEE 754 floating-point number.

$$(-1)^S \times (1 + F) \times 2^E$$

S = Sign

F = fraction ($0 < F < 1$)

E = Exponent - Bias

Bias = 127 for single precision

If the first character of the word is a one then the sign is negative:

S = 1

The next eight bits make up the exponent:

$$E = 100\ 0001\ 0 = 2^7 + 2^1 = 128 + 2 = 130 - \text{Bias} = 130 - 127 = 3$$

F = 011 0000 0000 0000 0000 0000

Going from left to right, with left as the zero point and going into the negative:

$$2^{-2} + 2^{-3} = 0.25 + 0.125 = 0.375$$

$$(-1)^1 \times (1 + 0.375) \times 2^3$$

$$-1 \times 1.375 \times 8$$

The answer is -11

6. (6%) Find computer representations of the following numbers:

(a) -1022 as a two's complement 32-bit binary number;

First turn it into a positive binary number:

S = -1

N = 1022

$$1022/2 = 511 \rightarrow 0$$

$$511/2 = 255.5 \rightarrow 1$$

$$255/2 = 127.5 \rightarrow 1$$

$$127/2 = 63.5 \rightarrow 1$$

$$63/2 = 31.5 \rightarrow 1$$

$$31/2 = 15.5 \rightarrow 1$$

$$15/2 = 7.5 \rightarrow 1$$

$$7/2 = 3.5 \rightarrow 1$$

$$3/2 = 1.5 \rightarrow 1$$

$$1/2 = 0.5 \rightarrow 1$$

Now we have 0...0011 1111 1110

Since it needs to be a two's complement number we need to use the algorithm from before to flip the bits.

Which becomes: 1...1100 0000 0010 Since it is a negative number we leave the first character as a one:

Answer = 1111 1111 1111 1111 1111 1100 0000 0010

(b) -32.75 as an IEEE 754 32-bit floating-point number;

S	E (Exponent)	F (Fraction)
1bit	8 bits	23 bits

Bias = 127
 S = 1 since it is a negative number.

 32.75
 $32 = 10000$ in binary
 $0.75 \times 2 = 1.5 \rightarrow 1$
 $0.5 \times 2 = 1 \rightarrow 1$
 1.1

 100001.1

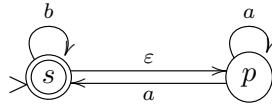
 $1.000011 \times 2^5 = 100001.1$
 $E = 5 + 127 = 132 = 10000100$
 $F = 000011$

1	100 0010 0	000 0110 0000 0000 0000 0000
S	E (Exponent)	F (Fraction)

(c) 77 as a two's complement 32-bit binary number.

$77/2 = 38.5 \rightarrow 1$ $38/2 = 19 \rightarrow 0$ $19/2 = 9.5 \rightarrow 1$ $9/2 = 4.5 \rightarrow 1$ $4/2 = 2 \rightarrow 0$ $2/2 = 1 \rightarrow 0$ $1/2 = 0.5 \rightarrow 1$ Answer = 0000 0000 0000 0000 0000 0000 0100 1101
--

7. (6%) Consider the following finite automaton:



- (a) Give all the computations of the automaton on the input strings aab , aba , and ε , and determine if the strings are accepted.

Input: aab

Computation:

- (s, aab), (p, aab), (p, ab), (s, b), (s, ε) Accepted
- (s, aab), (p, aab), (s, ab), (p, ab), (s, b), (s, ε) Accepted
- (s, aab), (p, aab), (p, ab), (p, b) stuck
- (s, aab), (p, aab), (p, ab), (s, b), (p, b) stuck
- (s, aab), (p, aab), (s, ab), (p, ab), (s, b), (s, ε) Accepted

word aab is Accepted.

Input: aba

Computation:

- (s, aba), (p, aba), (p, ba) stuck
- (s, aba), (p, aba), (s, ba), (s, a), (p, a), (p, ε) stuck
- (s, aba), (p, aba), (s, ba), (s, a), (p, a), (s, ε) Accepted
- (s, aba), (p, aba), (s, ba), (p, ba) stuck

word aba is Accepted.

Input: ε

Computation:

- (s, ε), (p, ε) stuck
- (s, ε) Accepted

word ε is Accepted.

- (b) Describe the language accepted by the automaton in English.

The language accepts any amount of a's and b's and the empty word.

- (c) Describe the language accepted by the automaton by means of a regular expression.

$(a \cup b)^*$

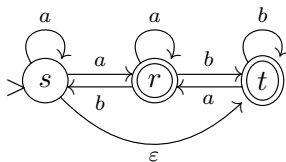
- (d) Describe the language accepted by the automaton by means of a context-free grammar.

$$S \rightarrow \varepsilon$$

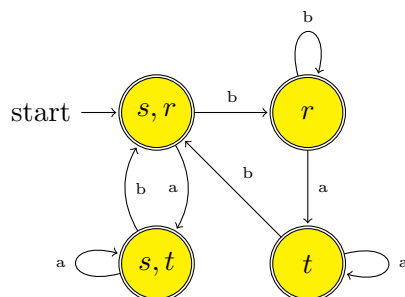
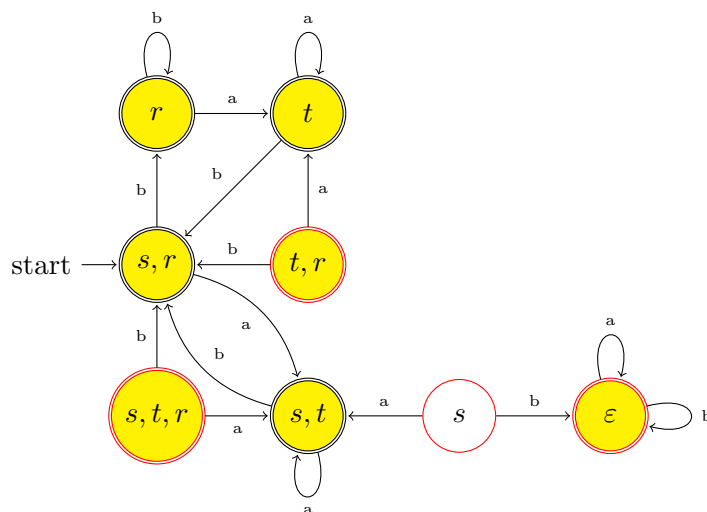
$$S \rightarrow aS$$

$$S \rightarrow bS$$

8. (10%) Transform, using the subset construction, the following nondeterministic finite automaton into an equivalent deterministic finite automaton. Show your working.



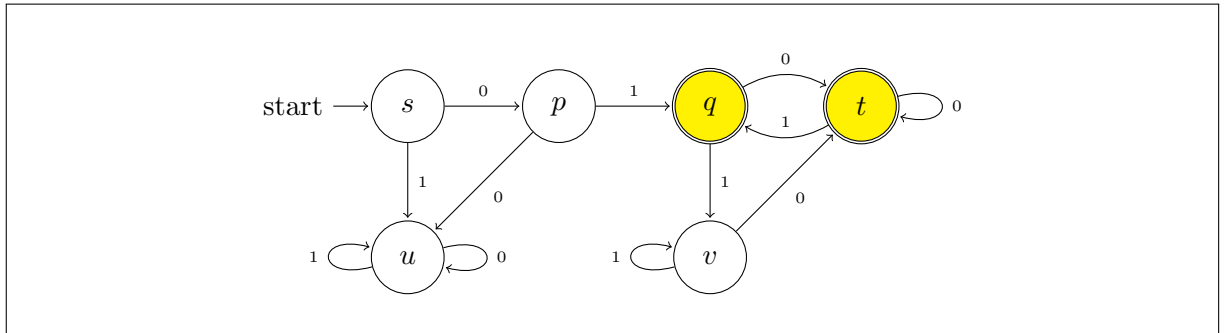
	A	B
s	s,t	ε
t	t	s,r
r	t	r
s,t	s,t	s,r
s,r	s,t	r
t,r	t	s,r
s,t,r	s,t	s,r
ε	ε	ε



What is the language of this automaton.

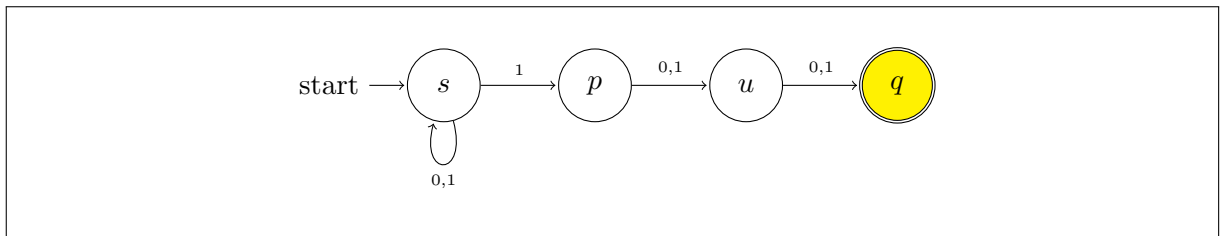
This language accepts any number of a's and b's.

9. (7%) Design a (deterministic or nondeterministic) finite automaton A such that $L(A)$ consists of all strings over the alphabet $\{0,1\}$ that begin with 01 and do not end with 11. Find a regular expression representing the language $L(A)$.



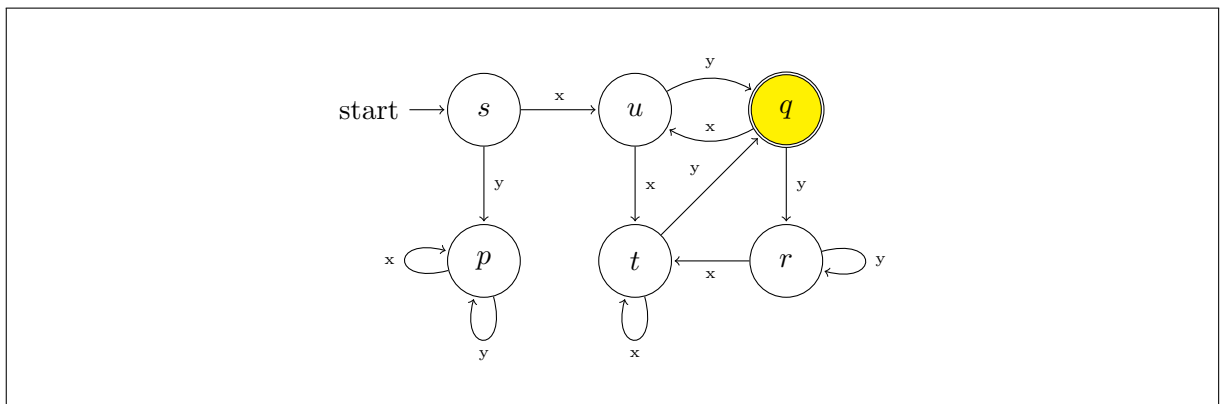
$01((0 \cup 1)^*(0 \cup 01))^*$

10. (6%) Design a (deterministic or nondeterministic) finite automaton A such that $L(A)$ consists of all strings over the alphabet $\{0,1\}$ whose third symbol from the right end is 1 (for example, 100101 is in $L(A)$, but 100011 is not). Find a regular expression representing $L(A)$.



$(0 \cup 1)^*1(0 \cup 1)(0 \cup 1)$

11. (8%) Convert the regular language $L[x((y \cup x)^*x)^*y]$ to a finite automaton accepting it.



12. (4%) Consider the following context free grammar:

$$S \rightarrow SS, \quad S \rightarrow L0L0L, \quad L \rightarrow \varepsilon, \quad L \rightarrow 1L, \quad L \rightarrow 0L.$$

- (a) Give a derivation for the string 101101.
- (b) Describe in English the language of this grammar.

13. (6%) Construct context free grammars for the following languages

- (a) $\{w \in \{0,1\}^* \mid w \text{ starts and ends with different symbols}\},$
- (b) $\{w \in \{0,1\}^* \mid \text{the length of } w \text{ is even}\}$

14. (15%) Construct a context free grammar and a pushdown automaton for the language of words over the alphabet $\{0,1\}$ that start and end with the same symbol and have the same number of 0s as 1s.

15. (5%) Consider the following transition table of a Turing machine:

s	0	s	\rightarrow
s	1	s	\rightarrow
s	\sqcup	p	\leftarrow
s	\triangleright	s	\rightarrow
p	0	h	1
p	1	p	\rightarrow
p	\sqcup	h	0
p	\triangleright	s	\rightarrow

(i) Give the computations of the machine starting with the configurations

- $(s, \triangleright \underline{0})$,
- $(s, \triangleright \underline{1}11)$,
- $(s, \triangleright \underline{1}00)$.

(ii) Describe in English what this Turing machine does.

16. (9%) Consider the following $\mathbb{N} \rightarrow \mathbb{N}$ function f :

$$f(n) = \begin{cases} 4n & \text{if } n \text{ is odd,} \\ n/2 & \text{if } n \text{ is even.} \end{cases}$$

(Don't forget that all numbers are represented in binary.)

- (i) Explain what it means to say that a Turing machine *computes* this function f .
- (ii) Give an implementation level description in English of a Turing machine that computes this f .
- (iii) Give the complete transition table of this Turing machine.
- (iv) Give the computations of your Turing machine on inputs 0, 11 and 100.