

Exam, quiz, and group-work questions

The multiple-choice questions below are intended as a study guide to the factual content of the course. By sampling the questions, students may learn where they need to focus their study. Facts are not worth learning in isolation from concepts and problem-solving methods, but knowledge of facts is necessary to understand the concepts in the course. Knowledge of facts will emerge from working with problems and concepts.

Introduction and Background

See questions on paper, on the following topics:

- Logic
 - Sets
 - Relations
 - Functions
 - Inductive proofs
1. A model abstracts from (a) foundation concepts; (b) details; (c) utility; (d) simplicity; (e) none of these
 2. Computing is traditionally viewed as (a) interaction; (b) algebra; (c) intelligence; (d) the execution of algorithms; (e) what the Internet does
 3. Top-down theoretical results in computer science use (a) empirical data; (b) testing benchmarks; (c) conjecture and proof; (d) democratic vote; (e) none of these
2. A proof that proceeds by showing the existence of something desired is by (a) induction; (b) construction; (c) contradiction; (d) prevarication; (e) none of these
 3. A proof that shows that a certain property holds for all natural numbers is by (a) induction; (b) construction; (c) contradiction; (d) prevarication; (e) none of these

Data structures

1. One member of a linked-list node must be a(n) (a) character; (b) integer; (c) node; (d) reference; (e) none of these
2. To access a certain node in a singly-linked list (a) takes one step; (b) takes two steps; (c) requires that all its predecessors be visited; (d) requires that all its successors be visited
3. To add to a stack, we carry out a(n) _____ operation. (a) dequeue; (b) enqueue; (c) pop; (d) push; (e) traversal
4. The first-in, first-out structure is the (a) list; (b) array; (c) queue; (d) stack; (e) tree
5. A tree has no (a) edges; (b) vertices; (c) paths; (d) cycles; (e) connectivity
6. To model a hierarchy, it is most convenient to use a(n) (a) simple type; (b) array; (c) linked list; (d) binary search tree; (e) tree
7. A graph is (a) a set of integers; (b) a set of vertices; (c) a set of vertices and a set of edges; (d) a set of edges; (e) a set of paths

Logic

Sets

1. $\{1,2,3\} \cup \{2,4,5\} =$ (a) $\{\}$; (b) $\{1,2\}$; (c) 2 ; (d) $\{2\}$; (e) $\{1,2,3,4,5\}$

Relations

1. A *relation* on set A is (a) an element of A ; (b) a subset of A ; (c) an element of $A \times A$; (d) a subset of $A \times A$; (e) none of these

Functions

1. A function $f: \{1,2,3\} \rightarrow \{0,1\}$ is a set of (a) integers; (b) ordered pairs; (c) sets; (d) relations; (e) none of these

Proofs

1. A proof that begins by asserting a claim and proceeds to show that the claim cannot be true is by (a) induction; (b) construction; (c) contradiction; (d) prevarication; (e) none of these

Topic 1: Cardinalities, incompleteness, coinduction

Cardinalities

See printed sheet with several questions.

1. What proof method was used by Cantor to show that the reals are uncountable? (a) inductive; (b) diagonal; (c) constructive; (d) immediate; (e) none of these
2. Which of these sets is countable? i. strings
ii. streams iii natural numbers iv real numbers.
(a) i and ii; (b) i and iii; (c) ii and iii; (d) ii and iv;
(e) none of these
3. The set of strings of length k , over a finite alphabet, for given constant k , is (a) countably infinite; (b) finite; (c) uncountable; (d) undecidable; (e) none of these
4. Two sets have the same cardinality if (a) they are both finite and have the same element count; (b) neither is strictly included in the other; (c) a bijection exists between them; (d) they are both infinite; (e) none of these
5. Cantor showed that the reals are (a) infinite; (b) countable; (c) uncountable; (d) dense; (e) none of these
6. The natural numbers (a) can be paired up with the reals; (b) are countable; (c) are uncountable; (d) are as numerous as any set; (e) none of these
7. Cantor's proof about the cardinalities of real and natural numbers was by (a) induction; (b) diagonalization; (c) construction; (d) statistical methods; (e) none of these
8. A diagonal proof is by (a) induction; (b) contradiction; (c) construction; (d) statistical methods; (e) none of these
9. Σ is by convention (a) finite; (b) countable; (c) uncountable; (d) a sequence; (e) none of these
10. Σ^* is (a) finite; (b) countable; (c) uncountable; (d) an alphabet; (e) none of these
11. Σ^∞ is (a) finite; (b) countable; (c) uncountable; (d) an alphabet; (e) none of these

12. Σ^* is (a) a number; (b) a symbol; (c) an alphabet; (d) a language; (e) none of these

Incompleteness

1. Gödel's theorem was proven in the ___ century
(a) 18th; (b) 19th; (c) 20th; (d) 21st; (e) none of these
2. A logical system in which no false assertion can be proven is (a) consistent; (b) complete; (c) ambiguous; (d) paradoxical; (e) none of these
3. A logical system in which every true assertion can be proven is (a) consistent; (b) complete; (c) ambiguous; (d) paradoxical; (e) none of these
4. Gödel numbers (a) are cardinalities; (b) are reals; (c) encode assertions; (d) encode programs; (e) none of these
5. Gödel's incompleteness theorem was proven by (a) induction; (b) diagonalization; (c) construction; (d) statistical methods; (e) none of these

Coinduction

1. Induction is used to define (a) branch control structures; (b) finite objects; (c) infinite objects; (d) finite sets; (e) none of these
2. Coinduction is used to define sets of (a) branch control structures; (b) finite objects; (c) infinite objects; (d) finite sets; (e) none of these
3. A coinductive definition has no (a) base case; (b) inductive case; (c) endpoint; (d) purpose; (e) none of these
4. Coinduction may define sets of (a) numbers; (b) programs; (c) proofs; (d) strings; (e) streams
5. The wellfoundedness axiom states that the notion of a set belonging to itself (a) is well-founded; (b) is meaningless; (c) is doubtful; (d) is mandatory; (e) none of these
6. Σ^∞ is a set of (a) numbers; (b) symbols; (c) strings; (d) streams; (e) none of these

Answers

Cardinalities

1. b
2. b
3. b
4. c
5. c
6. b
7. b
8. b
9. a
10. b
11. c
12. d

Incompleteness

1. c
2. a
3. b
4. c
5. b

Coinduction

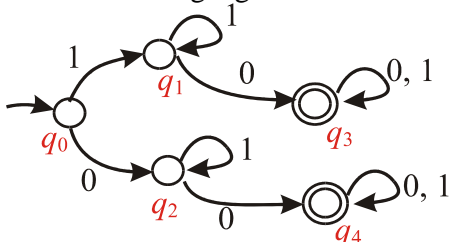
1. b
2. c
3. a
4. e
5. b
6. d

Topic 2: DFAs and regular languages

See also printed sheet with several questions.

1. Deterministic finite automata

- In our discussion of DFAs, δ is (a) a function; (b) an alphabet; (c) a symbol; (d) a string; (e) none of these
- In our discussion of languages, λ is (a) a function; (b) an alphabet; (c) a symbol; (d) a string; (e) none of these
- In our discussion of languages, Σ is (a) a function; (b) an alphabet; (c) a symbol; (d) a string; (e) none of these
- The reflexive transitive closure of δ maps from (a) states to states; (b) states and symbols to states; (c) states and strings to states; (d) states and symbols to symbols; (e) none of these
- All ___ languages are regular (a) known; (b) finite; (c) infinite; (d) recursively definable; (e) none of these
- What is the language of this?



- (a) CF but not regular; (b) 0^*1^* ; (c) 1^*0^* ;
 (d) $(1+0)^*1^*(0+1)^*$; (e) $(1+0)1^*0(0+1)^*$

2. Regular languages and regular expressions

- A DFA may be converted to a regular expression using (a) induction; (b) a construction; (c) contradiction; (d) compilation; (e) none of these
- The operations that form regular expressions are concatenation, iteration, and (a) selection; (b) inversion; (c) transposing; (d) negation; (e) none of these
- The operations that form regular expressions are iteration, selection, and (a) concatenation; (b) inversion; (c) transposition; (d) negation; (e) none of these
- The operations that form regular expressions are concatenation, selection, and (a) iteration; (b) inversion; (c) transposing; (d) negation; (e) none of these

- A regular expression may be converted to a DFA using (a) induction; (b) a construction; (c) contradiction; (d) compilation; (e) none of these
- A regular expression (a) is a language; (b) is an alphabet; (c) defines a language; (d) defines an alphabet; (e) none of these
- The regular-expression meta-symbol '+' stands for (a) concatenation; (b) selection; (c) iteration; (d) addition; (e) reversal
- The regular-expression meta-symbol '*' stands for (a) concatenation; (b) selection; (c) iteration; (d) addition; (e) reversal
- The string "1100" is an element of the regular language defined by (a) $0^*1^*0^*$; (b) $0(0+1)^*$; (c) 1^*0^*1 ; (d) 1^*010 ; (e) none of these

3. Nondeterministic finite automata

- NFAs accept (a) not as many languages as DFAs; (b) more languages than DFAs; (c) the same languages as DFAs; (d) CFLs; (e) none of these
- An NFA may be converted to a DFA using (a) induction; (b) a construction; (c) contradiction; (d) compilation; (e) none of these
- NFAs accept what kind of transition not accepted by DFAs? (a) α ; (b) δ ; (c) λ ; (d) π ; (e) θ
- A NFA's transition function returns (a) a Boolean; (b) a state; (c) a set of states; (d) an edge; (e) a number
- The proof that the concatenation of regular languages is regular is by construction of a(n) (a) alphabet; (b) DFA; (c) language; (d) NFA; (e) program
- DFAs recognize ___ languages compared to NFAs (a) fewer; (b) more; (c) the same; (d) no; (e) all
- The proof that for every NFA there is a DFA that recognizes the same language is by (a) contradiction; (b) induction; (c) construction; (d) diagonalization; (e) none of these
- The subset construction creates a DFA whose states are (a) Booleans; (b) numbers; (c) strings; (d) sets of NFA states; (e) none of these
- The subset construction shows that every NFA accepts a (a) string; (b) function; (c) regular language; (d) context-free language; (e) decidable set

4. Proving a language is not regular

1. The Pumping Lemma (a) holds that certain languages are not regular; (b) is used in proofs that certain languages are not regular; (c) is a corollary to theorems about regular languages; (d) is a hypothesis; (e) is an open research question
2. The Pumping Lemma is used to show that a certain language is (a) regular; (b) finite; (c) infinite; (d) not regular; (e) none of these
3. We “pump out” of the possibility that a language is regular using (a) a regular expression; (b) the idea of a loop on a DFA; (c) the fact that the language is finite; (d) the idea of a branch; (e) none of these
4. To show that a language is not regular we may use (a) the Pumping Lemma; (b) diagonalization; (c) induction; (d) construction; (e) NFAs
5. The Pumping Lemma uses repetition of (a) while loops; (b) inputs; (c) substrings that return a DFA to the same state; (d) numbers; (e) none of these
6. Regular languages are closed under (a) union and intersection; (b) union but not concatenation; (c) concatenation but not intersection; (d) concatenation but not reversal; (e) complement but not union
7. For any DFA (a) a minimal version does not exist; (b) a minimal version exists but cannot be computed; (c) an algorithm exists to derive the minimal version; (d) a variety of designs for a minimal version exist; (e) minimality is a meaningless concept

8. DFA minimalization uses ___ states
(a) unreachable; (b) duplicate; (c) distinguishable; (d) indistinguishable; (e) unique
9. For any regular language there exists (a) an optimal DFA; (b) a unique minimal DFA; (c) a unique maximal DFA; (d) no unique DFA; (e) none of these

5. A foundation for lexical analysis

1. DFAs are used to construct (a) symbol tables; (b) lexical analyzers; (c) parsers; (d) optimizers; (e) code generators
2. A lexical analyzer recognizes (a) grammar rules; (b) numbers; (c) tokens; (d) correctness; (e) type errors
3. An identifier is recognized by a (a) parser; (b) lexical analyzer; (c) minimizer; (d) subset construction; (e) pumping lemma
4. Lexical analysis is done by (a) table lookup; (b) parsing; (c) a PDA; (d) a DFA; (e) a Turing machine

Answers

1 . Deterministic finite automata

1. A
2. D
3. B
4. C
5. B
6. E

2 Regular languages and regular expressions

1. B
2. A
3. A
4. A
5. B
6. C
7. B
8. C
9. A

3.Nondeterministic finite automata

1. C
2. B
3. C
4. C
5. D
6. C
7. C
8. D
9. C

4. Proving a language is not regular

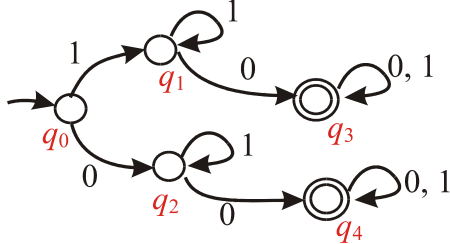
1. B
2. D
3. B
4. A
5. C
6. A
7. C
8. D
9. B

5. A foundation for lexical analysis

1. B
2. C
3. B
4. D

Short answer

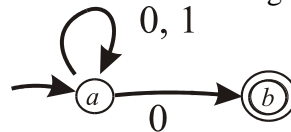
1. (a) What is this a diagram of? (b) What language does it accept?.



2. What are nondeterministic automata good for?
3. Design a finite automaton that recognizes the language $(01 \mid 100)^*(0 \mid 1)^*$.
4. Use the Pumping Lemma to show that the language consisting of strings in which the number of 0s is greater than the number of 1s is not regular.
5. Using diagrams, construct DFAs that recognize the following languages. Supply also in proper notation, one element of δ for each.
 - (a) Binary numerals divisible by 4
 - (b) Real numbers (i.e., with an optional decimal point)
6. What proof method is used when Pumping Lemma is used?
7. Defend or refute: "Nondeterministic automata are of great practical use." Give examples
8. Design a finite automaton that recognizes the language $(01 \mid 100)^*(0 \mid 1)^*$
9. Use the Pumping Lemma to show that the language consisting of strings in which the number of 0s is greater than the number of 1s is not regular.
10. Using both a tuple and a diagram, define a finite automaton that recognizes the language $(01)^*$.
11. Explain why a finite automaton does or does not correspond to a graph.
12. Design a finite automaton that recognizes the language $(0^* + 1)0$.
13. Use the Pumping Lemma to show that the language consisting of strings that are palindromes is not regular.
14. How are nondeterministic finite automata useful?
15. Diagram an NFA that recognizes $(01)^*(0 + 11)$

16. Using both a tuple and a diagram, define a finite automaton that recognizes the language
 - a. $(00 + 1)^*2$
 - b. $(0^* + 1)0$
 - c. 01^*0
 - d. of Java relational operators
 $(==, >=, <=, <, >, !=)$

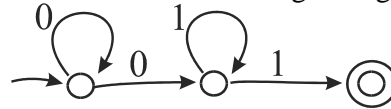
17. Given δ of $\{ \dots \}$, what state does DFA $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ go into on input 101?
18. What is the *union* of the following languages?
 $L_1 = \{0, 00, 000\}, L_2 = \{00, 01, 10, 11\}$
19. What is the *intersection* of the languages listed in the previous problem?
20. What is the intersection of the language whose elements begin with 0 and the language $(0+1)1^*$?
21. What does an *edge* in a DFA's diagram represent? Explain in one sentence.
22. Draw a DFA or NFA that recognizes $(1 + 00^*)(01^*)^*$
23. Using *both* a tuple and a diagram, define an automaton that recognizes the language $(0^* + 1)0$
24. Write a regular expression for strings that have an odd number of zeroes.
25. Design a finite automaton that accepts words that do not end with "01".
26. Design a finite automaton that recognizes the language $(01 \mid 10)(0 \mid 1)^*$, showing your work.
27. Use the Pumping Lemma to show that the language 0^n1^{2n} (strings with a certain number of 0's followed by twice as many 1's) is not regular. Include a restatement of the Pumping Lemma.
28. Convert the following NFA to a DFA.



29. Use the Pumping Lemma to show that the language consisting of strings that are palindromes is not regular.
30. What is an application of DFAs in program compilation?

31. Cohen:
pp. 49-50, #2-18
pp. 71-75, #1-19
p. 145, #14
pp. 185-186, #1-19
p. 203, #1-19
32. (a) Construct an NFA that recognizes strings that start and end with 1 and have an even number of bits.
(b) Write a regular expression for this language.
33. Design a nondeterministic finite automaton that recognizes the language $(0^* + 1)^* 0$.
34. Write a regular expression and design a corresponding NFA that accepts strings whose third-last symbol is 1.

35. Show a regular expression *and* an NFA for the language of all strings over $\{0,1\}$ that contain a 00.
36. Describe in plain English the language associated with the regular expression $(0 + 11)^*$
37. Convert to a DFA and give regular expression:



38. Show by the Pumping Lemma that $0^n 10^n$ is not regular.
39. Let A be a DFA and q a particular state of A , s.t. $\delta(q, a) = q$ for all inputs a . Prove by induction on the length of the input that for all input strings w , $\delta^*(q, w) = q$. (Hopcroft et al., p. 54)

Topic 3: PDAs and context-free languages

See printed sheet with several questions.

1. Pushdown automata

1. Pushdown automata (a) accept the decidable languages; (b) accept the context-free languages; (c) have finite storage; (d) are less expressive than DFAs; (e) none of these
2. The context-free languages are recognized by (a) their alphabets; (b) finite automata; (c) pushdown automata; (d) a Turing machines; (e) random-access machines
3. A pushdown automaton may have (a) more states than a finite automaton; (b) random-access memory; (c) an infinite alphabet; (d) a stack; (e) faster transitions
4. A pushdown automaton has (a) a finite number of configurations; (b) fewer configurations than a DFA; (c) an infinite number of configurations; (d) a queue; (e) a tape
5. In PDAs, δ is (a) a state; (b) a set of states; (c) a relation; (d) a function; (e) a stack alphabet
6. In PDAs, Σ is (a) a state; (b) a set of states; (c) a relation; (d) a function; (e) an alphabet
7. In PDAs, Q is (a) a state; (b) a set of states; (c) a relation; (d) a function; (e) a stack alphabet
8. In PDAs, Γ is (a) a state; (b) a set of states; (c) a relation; (d) a function; (e) a stack alphabet
9. In PDAs, q_0 is (a) a state; (b) a set of states; (c) a relation; (d) a function; (e) a stack alphabet
10. DFAs are _____ PDAs (a) as expressive as; (b) less expressive than; (c) more expressive than; (d) more complex than; (e) less complex than
11. What does a PDA have that a DFA does not? (a) states; (b) input alphabet; (c) tape; (d) stack alphabet; (e) all of these
12. The standard proof that all RLs are CF is by (a) induction; (b) construction; (c) contradiction; (d) diagonalization; (e) introspection
13. All regular languages are (a) finite; (b) context free; (c) infinite; (d) all of these; (e) none of these
14. The greater expressive power of PDAs compared to DFAs is due to (a) the set of states; (b) recursive productions; (c) the stack; (d) nondeterminism; (e) none of these

2. Context free languages and context-free grammars

1. Context-free languages are (a) a subset of regular languages; (b) a superset of regular languages; (c) a superset of decidable languages; (d) accepted by DFAs; (e) none of these
2. A CF grammar generates a language (a) requiring a Turing machine to accept it; (b) accepted by a DFA; (c) accepted by a PDA; (d) that is uncountable; (e) none of these
3. The set of all nested sequences of balanced parentheses is (a) a string; (b) a regular expression; (c) a regular language; (d) a context-free language; (e) a finite language
4. A context-free language is characterized by a set of _____ rules (a) regular; (b) regular and irregular; (c) an infinite number of; (d) alphabets as; (e) terminal and nonterminal
5. An element of a context-free grammar that requires definition in terms of other elements is (a) a leaf; (b) a terminal; (c) a regular expression; (d) a nonterminal; (e) a production
6. Recursiveness is often found in (a) productions in a context-free grammar; (b) regular expressions; (c) pushdown automata; (d) DFAs; (e) none of these
7. PDAs _____ CFGs (a) can be constructed from; (b) are the same thing as; (c) have little relationship with; (d) generate; (e) accept
8. PDAs _____ CFLs (a) can be constructed from; (b) are the same thing as; (c) have little relationship with; (d) generate; (e) accept

3. Derivations and parsing

1. Derivations apply (a) regular expressions; (b) CFG production rules; (c) proof rules; (d) definitions of DFAs; (e) none of these
2. The standard proof that all CFGs generate CFLs is by (a) induction; (b) construction; (c) contradiction; (d) diagonalization; (e) introspection
3. The standard proof that any CFL can be generated by some CFG is by (a) induction; (b) construction; (c) contradiction; (d) diagonalization; (e) introspection
4. The start symbol in a CFG (a) is a terminal; (b) is on the left side of a production rule; (c) is on the right side of a production rule; (d) is a production rule; (e) none of these

5. A parse tree is yielded by (a) a grammar; (b) an input string; (c) a PDA; (d) a CF grammar and an input string; (e) a regular expression and an input string
6. A derivation begins with (a) a parse tree; (b) a string; (c) a production rule; (d) a terminal; (e) a PDA
7. A derivation yields (a) a parse tree; (b) a grammar; (c) a production rule; (d) a terminal; (e) a PDA

4. Properties of PDAs, CFLs, and CFGs

1. A proof that a language is not context free is often by (a) induction; (b) construction; (c) contradiction; (d) diagonalization; (e) the Pumping Lemma for CFLs
2. If Σ is an alphabet and S is a start symbol, then $\{ w \in \Sigma^* \mid S \Rightarrow_G^* w \}$ is (a) a DFA; (b) a PDA; (c) the language generated by a CF grammar; (d) the language generated by a regular expression; (e) undecidable
3. The set of languages defined by CF grammars is (a) the set of regular languages; (b) finite; (c) the set of PDA-recognized languages; (d) the set of decidable languages; (e) null
4. A ___ may be constructed from any PDA (a) DFA; (b) NFA; (c) CFG; (d) regular expression; (e) none of these

Answers to T3 multiple-choice questions

1. Pushdown automata

1. B
2. C
3. D
4. C
5. C
6. E
7. B
8. E
9. A
10. B
11. D
12. B
13. B
14. C

1. Context free languages and context-free grammars

b

2. C
3. D
4. E
5. D
6. A
7. A
8. E

2. Derivations and parsing

1. B
2. B
3. B
4. B
5. B
6. C
7. A

3. Properties of PDAs, CFLs, and CFGs

1. E
2. C
3. C
4. C

Short answer

9. Explain the relationship between a language and a pushdown automaton.
10. Explain why all finite languages are context free.
11. Explain what is meant by
 $(\forall L \subseteq \Sigma^*) \text{Regular}(L) \Rightarrow \text{Context-Free}(L)$,
 and why it is true.
12. What operations can a PDA perform that a DFA cannot?
13. Write a derivation tree for
 $S \rightarrow Xa \mid aX \mid SX$
 $X \rightarrow aY \mid Ya$
 $Y \rightarrow ab \mid ba \mid \lambda \mid YY$
 and each of the following strings:
 (a) abbb; (b) aaba; (c) abab; (d) baab; (e) baaab
14. List differences and similarities between the DFA and the PDA models of computation. Give an example of how a stack is needed to recognize a non-regular language such as the one in HW3, #3.
15. (a) In plain terms, state what language is generated by the CFG
 $S \rightarrow 0X0 \mid 1X1$
 $X \rightarrow 0 \mid XX$
 (b) Perform a derivation using a 7- or 8-character string that you know is in the language and that is formed using all the productions.
16. Write a context-free grammar to specify the language consisting of all bit strings that have twice as many 0's as 1's.
17. Write a derivation tree for the string 10001, given the following:
 $S \rightarrow 0X0 \mid 1X1$
 $X \rightarrow 0 \mid 1 \mid \lambda$
18. Define a CF grammar for the following, and explain your design briefly:
 (a) $\{ x \mid n_0(x) = n_1(x) \}$ (equal 0's and 1's)
 (b) Palindromes
 (c) Balanced parentheses
 (not only $((()))$ but also $()((()))$ etc)
 (d) $0^n 1^{2n}$
 (e) $\{ xx \mid x \in \Sigma^* \}$
19. Given the following:
 $S \rightarrow 1X \mid 1S$
 $X \rightarrow 0 \mid 0X$
 (a) What is the above 2-line entity called?
 (b) What do these entities define in general?
 (c) What are the components of the above entity called and how many are there?
 (d) What particular set of (b) does the entity define, precisely?
 (e) Write a derivation tree to show that 1100 is in the set defined by the entity.
20. Prove by construction of a CFG that if $L_1, L_2 \in \mathcal{CFL}$, then $(L_1 \cup L_2)$ is CF. Discuss.
21. *RPN challenges* :
 a. Write a CF grammar for the set of all Reverse Polish Notation expressions.
 b. Define a PDA that accepts RPN expressions.
 c. Is RPN a regular language? If so, write a regular expression for it; otherwise show that it is not regular.
 d. Design a finite or pushdown transducer (PDT) that evaluates RPN expressions (OK for output to be in unary notation). A PDT is a PDA that also outputs a string at each transition:
 $\delta : Q \times \Sigma_{in} \times \Gamma \cup \{\lambda\} \rightarrow Q \times \Gamma \cup \{\lambda\} \times \Sigma_{out}$.

Topic 4: Turing machines and decidable problems

1. Introduction

2. TMs and TM computations

1. The Turing machine model is said to capture (a) regular languages; (b) interaction; (c) efficient computation; (d) algorithmic computation; (e) all of these
2. A TM is distinguished from DFAs and PDAs by (a) a set of states; (b) a stack; (c) RAM; (d) a tape; (e) a simpler transition function
3. A TM has ___ memory (a) random-access; (b) limited; (c) unbounded; (d) stack; (e) queue
4. A TM (a) lacks an alphabet; (b) has tape instead of states; (c) can compute any mathematical function; (d) stores data on a tape; (e) none of these
5. Each computation step on a TM may (a) read many symbols; (b) make several transitions; (c) move the tape head left or right; (d) push the stack; (e) none of these
6. When does a TM halt? (a) at end of input; (b) when it enters an accept or reject state; (c) in finite time; (d) never; (e) none of these
7. A TM configuration is determined by (a) most recent input; (b) most recent state; (c) tape contents; (d) state and tape symbol under head; (e) state and tape contents
8. A TM computation is a sequence of (a) tape symbols; (b) states; (c) configurations; (d) transition function values; (e) none of these
9. A TM that halts on all inputs (a) does not exist; (b) computes a partial function; (c) computes a total function; (d) terminates early; (e) none of these
10. A nonstandard variant of the TM has ___ tape(s) (a) no; (b) one; (c) two; (d) infinite; (e) none of these
11. A ___ TM simulates any TM (a) universal; (b) halting; (c) PDA; (d) one-tape; (e) none of these
12. The universal TM corresponds to (a) a PDA; (b) a stored-program computer; (c) a general-purpose DFA; (d) all TMs; (e) none of these

3. Turing decidability

1. Any computable function can be computed on some (a) DFA; (b) NFA; (c) PDA; (d) Turing machine; (e) none of these
2. Decision problems are equivalent to functions that return (a) natural numbers; (b) strings; (c) truth values; (d) Turing machines; (e) none of these
3. Every regular language is (a) finite; (b) uncountable; (c) TM-decidable; (d) undecidable; (e) none of these
4. Every context-free language is (a) finite; (b) uncountable; (c) TM-decidable; (d) undecidable; (e) none of these

4. Undecidability

1. The Halting Problem involves (a) testing a TM to see if it halts; (b) determining from the structure of the TM whether it halts; (c) determining how to change the transition function of a TM to cause it to halt; (d) determining what inputs halt a TM; (e) none of these
2. The standard proof that the Halting Problem is undecidable is by (a) induction; (b) indirection; (c) contradiction; (d) indirection; (e) none of these
3. Turing reducibility means (a) two models of computation are equivalent; (b) one model of computation is more expressive; (c) problem B is undecidable if problem A is undecidable and reducible to B ; (d) problem B is undecidable if problem A is undecidable and B is reducible to A ; (e) none of these
4. The Halting Problem is (a) decidable; (b) an example of a language that no TM accepts; (c) a μ -recursive function; (d) a machine; (e) none of these
5. Which is decidable? (a) DFA A accepts the language generated by regular expression E ; (b) TM M halts on blank input; (c) TM M accepts the language generated by regular expression E ; (d) program P in the S language says "arf" on input "meow"; (e) all of these

5. Semidecidability

1. A recursively enumerable set is (a) a μ -recursive function; (b) DFA recognizable; (c) TM recognizable, but no TM necessarily halts to reject non-members of the r.e. set; (d) accepted by a PDA; (e) none of these
2. A semi-decidable language is (a) regular; (b) context-free; (c) recursive; (d) recursively enumerable; (e) none of these
3. Some undecidable languages are (a) regular; (b) context-free; (c) recursive; (d) recursively enumerable; (e) none of these
4. Not all non-elements of a semi-decidable language are ___ by a recognizer TM
(a) accepted; (b) rejected; (c) generated; (d) output; (e) none of these
5. If both a language and its complement are recursively enumerable, then the language is
(a) regular; (b) decidable; (c) context-free; (d) empty; (e) universal
6. Unrestricted grammars generate languages accepted by (a) Turing machines; (b) linear bounded automata; (c) DFAs; (d) PDAs; (e) random access machines

6. Church-Turing Thesis

1. The Church-Turing thesis associates the TM with
(a) regular languages; (b) parsing; (c) lexical analysis; (d) algorithms; (e) interaction
2. The Church-Turing Thesis refers to
(a) a formalization of an intuitive notion; (b) a theorem; (c) provable; (d) disprovable; (e) a paper written at Harvard
3. According to the Chomsky hierarchy
(a) $\mathcal{RL} \subset \mathcal{CFL}$; (b) $\mathcal{CFL} \subset \mathcal{RL}$; (c) HALT is decidable; (d) DFAs accept all recursive sets; (e) PDAs can decide the Halting Problem
4. TMs (a) are interactive; (b) always halt; (c) perform input alternating with output; (d) mimic Internet servers; (e) mimic command-line environments

Answers to T4 multiple-choice questions

2. TMs and TM computations

1. D
2. D
3. C
4. D
5. C
6. B
7. D
8. C
9. C
10. D
11. A
12. B
- 13.

3. Turing decidability

5. D
6. C
7. C
8. C

5. Undecidability

1. B
2. C
3. C
4. B
5. A

5. Semidecidability

1. C
2. D
3. D
4. B
5. B
6. B

6. Church-Turing Thesis

1. D
2. A
3. A
4. E

Short answer

1. How does a TM differ from a PDA in its input, output, and storage?
2. What three actions are specified by the TM's transition function?
3. Explain why any language recognized by a DFA must also be recognized by some TM.
4. Compare and contrast the diagonal proofs by Cantor (cardinality of reals is greater than that of natural numbers) and Turing (some problems are uncomputable). What basic proof method do they share?
5. What is the term for the relationship of problem A to problem B , when we can show that a solution to problem B would provide us with a way to solve problem A ?
6. What is the term for the relationship of problem A to $HALT$, when we can show that a solution to problem A would provide us with a way to solve $HALT$? What can we say about A ?
7. Suppose it were proven that problem P_1 is undecidable. How might this fact be used to show that problem P_2 is undecidable?
8. Write (as a diagram or table) the definition of a Turing machine that will add a binary numeral to a unary numeral (tally, i.e., $11=2$, $111=3$, etc.), yielding binary output. Document by describing the purpose of each loop. As test results, include JFLAP traces of a few test inputs.
9. Briefly explain why no TM can solve the halting problem
10. Briefly explain why no TM can solve the problem of whether a given TM halts on blank input
11. Briefly explain why no TM can solve the problem of whether a given TM halts on blank input

12. Suppose that if you had a solution to problem P , you could use it to construct a solution to problem Q . Also suppose that you know that problem Q is undecidable.
 - (a) What can you say about problem P ?
 - (b) What is the relationship between P and Q called?
 - (c) Give an example of the above as discussed in this course.
 - (d) Briefly explain why no TM can solve the problem of whether a given TM recognizes the language _____
 - (e) What TMs solve semidecidable problems?

Longer answer

1. TM Construction

Write (as a diagram or table) the definition of the following Turing machine corresponding to your group number. Document by describing the purpose of each loop. If you use JFLAP, include as test results JFLAP traces of a few test inputs.

1. add a binary numeral to a unary numeral (tally, i.e., $11=2$, $111=3$, etc.), yielding binary output. Use '+' to separate the two parts of the input. For input $100+11$ ($4+3$), output should be 111 (7).
 2. Binary to twos-complement converter
 3. unary to binary converter
 4. Unary to unary adder
 5. unary to unary subtracter
 6. logical OR for two 1-bit inputs
 7. logical AND for two 1-bit inputs
 8. logical OR for multiple-bit input
 9. logical AND for multiple-bit input
 10. count 0's in input, display as unary numeral
2. Discuss how Rice's theorem is valid with respect to two models of algorithmic computation. Why is it valid under both models?
 3. Prove, by referring to a well-known theorem, that
 $\{P \mid P \text{ is the code of a program whose first output is "meow" on input "arf"}\}$
is undecidable.

Topic 5: Random-access machines and μ -recursive functions

1. Random-access machines and the \mathcal{S} language

1. TM-computable functions are the same as those computable on a (a) DFA; (b) PDA; (c) random-access machine; (d) control device; (e) none of these
2. The \mathcal{S} language implements which model? (a) DFA; (b) PDA; (c) RAM; (d) TM; (e) none of these
3. Unlike TMs, random-access machines have (a) tape; (b) stack; (c) queue; (d) addressable storage; (e) hard disk
4. The \mathcal{S} language outputs (a) natural numbers; (b) real numbers; (c) symbols; (d) strings; (e) none of these
5. In the \mathcal{S} language, input (a) occurs many times during a computation; (b) is the values of X variables at the start of computation; (c) is of string form; (d) may follow output; (e) none of these
6. The RAM model enables use of strings as input via (a) a Java method; (b) a data type; (c) Gödelization; (d) binary encoding; (e) encryption
7. A universal program in the \mathcal{S} language (a) compiles to byte code; (b) is an interpreter as seen on a browser; (c) accepts encodings of other \mathcal{S} language programs as inputs; (d) is a Turing machine; (e) solves even undecidable problems

8. μ -recursive functions

1. The set of TM-computable functions is the same as the set of (a) those computable on a DFA; (b) those

computable on a PDA; (c) μ -recursive functions; (d) control device; (e) none of these

2. Recursively definable functions are equivalent to ___ as a model of computation (a) DFAs; (b) PDAs; (c) TMs; (d) all of these; (e) none of these
3. The ___ function is *not* a basic primitive recursive function (a) factorial; (b) zero; (c) successor; (d) projection; (e) predecessor
4. One computable operation on primitive recursive functions is (a) inverse; (b) search; (c) composition; (d) integration; (e) none of these
5. Obtaining a function by primitive recursion is a way to show that the function is (a) continuous; (b) a predicate; (c) computable; (d) uncomputable; (e) none of these
6. The result of minimalization of a primitive recursive function is (a) primitive recursive; (b) computable; (c) minimal; (d) undecidable; (e) none of these

3. The Church-Turing thesis

1. The observation that Turing machines and \mathcal{S} language programs compute the same functions corresponds most strongly to (a) Cantor's proof; (b) Gödel's theorem; (c) the Church-Turing thesis; (d) Rice's theorem; (e) none of these
2. The proof that any \mathcal{S} language program computes a Turing-computable function is (a) diagonal; (b) by construction; (c) by contradiction; (d) inductive; (e) none of these
3. The proof that any Turing machine computes an \mathcal{S} language computable function is (a) diagonal; (b) by construction; (c) by contradiction; (d) inductive; (e) none of these

Answers to T5 multiple-choice questions

1. Random-access machines and the \mathcal{S} language

1. C
2. C
3. D
4. A
5. B
6. C
7. C

2. μ -recursive functions

1. C
2. C
3. A
4. C
5. C
6. B

3. The Church-Turing thesis

1. C
2. B
3. B

Short answer

1. Write a program in the \mathcal{S} language that takes two inputs and outputs the difference; if the second number is greater, output zero.
2. How is the \mathcal{S} language similar to two languages, very different from each other, that you have studied in computer science?
3. Write an \mathcal{S} -language program that takes three inputs, a , b , c , and outputs $a \times b \div c$. You may use macros for variable assignment, unconditional jump, jump on zero, addition, and subtraction.
4. State the relationship between primitive recursion, algorithmic computability, and the RAM model of computation.
5. Use the computability of addition to show that multiplication is \mathcal{S} -computable.
6. Use the computability of decrementing to show that subtraction is \mathcal{S} -computable.
7. Use the computability of multiplication to show that finding the smallest x such that x^3 is odd is \mathcal{S} -computable.
8. Show by construction, with respect to the RAM model of computation, that if f and g are computable functions, and $(\forall x) (h(x) = g(f(x)))$, then h is computable.
9. Explain why we can say the same as the previous question says with respect to the model of computation provided by recursive function theory.
10. Distinguish primitive recursion from μ -recursion.
11. Distinguish primitive recursion from composition.
12. What statement in the \mathcal{S} language enables looping and how?
13. Explain the notion of *composition* in recursive function theory.

Longer answer

1. Write a program in the \mathcal{S} language that
 - a) takes the input x and outputs 1 if $x = 0$, otherwise outputs 0.
 - b) takes the input x and outputs 1 if $x > 0$, otherwise outputs 0.
 - c) takes inputs x_1, x_2 , and outputs 1 if $x_1 \geq x_2$, otherwise outputs 0.
 - d) takes two inputs (x_1, x_2) and outputs $(x_1 - x_2)$.
 - e) takes two inputs (x_1, x_2) and outputs $(x_1 \times x_2)$ (multiplication).
 - f) takes two inputs (x_1, x_2) and outputs (x_1 / x_2) (division). Very briefly distinguish the RAM model of computation from the transition system model, with examples and implementations. Option: discuss other models of your choice.

Topic 6 (Interaction and concurrency)

1. Algorithmic and serial computation

1. The Von Neumann architecture is associated with (a) algorithms; (b) interaction; (c) parallelism; (d) serial computing; (e) multi-core computing
2. Algorithms (a) compute functions; (b) provide services; (c) accomplish missions in multi-agent systems; (d) may execute indefinitely; (e) none of these

2. Sequential interaction

1. Persistent Turing machines are said to model (a) sequential interaction; (b) multi-stream interaction; (c) parallel computation; (d) distributed computing; (e) none of these
2. Sequential interaction is associated with (a) providing a service; (b) non-termination; (c) dynamic input during computation; (d) persistence of state; (e) all of these
3. Finite transducers extend (a) DFAs; (b) PDAs; (c) TMs; (d) RAMs; (e) none of these
4. A Mealy machine has a (a) regular language; (b) stream language; (c) finite language; (d) context-free language; (e) decidable language
5. Interactive observable behavior is associated with (a) natural numbers; (b) strings; (c) sets of strings; (d) sets of streams; (e) none of these
6. The persistent Turing machine is a model of (a) table lookup; (b) algorithmic computing; (c) sequential interaction; (d) multi-stream interaction; (e) adaptation
7. It is claimed in the course material that the persistent Turing machine is ____ the Turing machine (a) less expressive than; (b) as expressive as; (c) more expressive than; (d) more efficient than; (e) less efficient than
8. It is claimed in the course material that the persistent Turing machine is ____ the random-access machine (a) less expressive than; (b) as expressive as; (c) more expressive than; (d) more efficient than; (e) less efficient than
9. The persistent Turing machine augments the Turing machine with (a) algorithmic computational power; (b) storage that lasts between I/O steps; (c) more states; (d) more operations; (e) none of these
10. Finite transducers (a) compute the same functions as DFAs; (b) compute the same functions as TMs; (c) have stream I/O behavior; (d) have finite input and output; (e) none of these

11. A feature of algorithmic computation is (a) alternation of input and output; (b) processing before input; (c) output before processing; (d) input, then processing, then output; (e) none of these
12. A feature of algorithmic computation is finite (a) input; (b) output; (c) processing; (d) input, output, and processing; (e) none of these
13. Reactive systems (a) compute functions; (b) provide services; (c) accomplish multi-agent missions; (d) execute only finitely; (e) none of these
14. I/O in reactive systems is (a) static; (b) dynamic; (c) finite; (d) constrained; (e) none of these
15. Interaction is distinguished from algorithmic computation by the presence of (a) finite input; (b) persistent state; (c) input; (d) processing; (e) none of these
16. A mutual causal effect between two agents occurs in all (a) interaction; (b) algorithms; (c) communication; (d) computing; (e) none of these
17. Synchrony entails (a) communication; (b) taking turns; (c) input; (d) autonomy; (e) none of these
18. Stream I/O characterizes (a) interaction; (b) algorithms; (c) functions; (d) $O(1)$ processes; (e) none of these
19. Persistent state stores (a) parameters; (b) loop invariants; (c) memory of past interactions; (d) algorithmic computation; (e) none of these
20. The form of input/output in interactive computation is (a) static; (b) finite; (c) streams; (d) strings; (e) none of these
21. A *service* is characteristic of (a) an algorithm; (b) an interactive process; (c) a multi-agent system; (d) a parallel system; (e) none of these
22. UML is required to specify (a) functional problems; (b) services; (c) algorithms; (d) algorithmic inputs; (e) none of these
23. Mutual causality characterizes (a) algorithms; (b) only directly-interacting entities; (c) all interacting entities; (d) functions; (e) none of these

3. Multi-stream and indirect interaction

1. Interaction may be sequential or (a) algorithmic; (b) $O(n)$; (c) multi-stream; (d) data-driven; (e) none of these
2. A *mission* is characteristic of (a) an algorithm; (b) an interactive process; (c) a multi-agent system; (d) a parallel system; (e) none of these
3. Indirect interaction requires (a) mutual causality between entities that do not interact directly;

- (b) message passing; (c) synchrony; (d) static I/O; (e) none of these
4. Stigmergy is (a) algorithmic computing; (b) direct interaction; (c) indirect interaction; (d) multi-stream interaction; (e) none of these
- 4. Models of parallel and distributed computing**
1. Serial computing is the same as (a) concurrency; (b) parallelism; (c) distributed computing; (d) single-core; (e) none of these
 2. Parallel computing is a kind of (a) concurrency; (b) Von Neumann architecture; (c) serial paradigm; (d) single-core computing; (e) none of these
 3. A thread simulates (a) ownership of all resources; (b) ownership of a CPU; (c) ownership of data; (d) an array; (e) none of these
 4. PRAM is (a) a model of serial computing; (b) a kind of memory; (c) a model of parallel computing; (d) a greedy algorithm; (e) none of these
 5. Process algebras assume communication by (a) packet; (b) indirect interaction; (c) bus; (d) message passing; (e) none of these
 6. PRAM assumes that memory is (a) all distributed; (b) shared; (c) infinite; (d) magnetic; (e) none of these
 7. Concurrent write is as opposed to ___ write (a) serial; (b) automatic; (c) shared; (d) exclusive; (e) none of these
 8. The standard model for parallel computing is (a) CRCW; (b) RAM; (c) PRAM; (d) Von Neumann; (e) none of these
 9. SIMD assumes that all CPUs execute ___ instructions (a) the same; (b) different; (c) pipelined; (d) cached; (e) none of these
 10. SIMD assumes that all CPUs operate on ___ data (a) the same; (b) different; (c) pipelined; (d) cached; (e) none of these
 11. A distributed algorithm runs on (a) a single processor; (b) a parallel-processing system; (c) multiple independent processors; (d) web pages; (e) none of these
- 5. Models of learning agents**
- 6. Partially observable Markov decision problems**

Answers to T6 multiple-choice questions

1. Algorithmic and serial computation

1. D
2. A

2. Sequential interaction

1. A
2. E
3. A
4. B
5. D
6. C
7. C
8. C
9. B
10. C
11. D
12. D
13. B
14. B
15. B
16. A
17. B
18. A
19. C
20. C
21. B
22. B
23. C

3. Multi-stream and indirect interaction

1. C
2. C
3. A
4. C

4. Models of parallel and distributed computing

1. D
2. A
3. B
4. C
5. D
6. B
7. D
8. C
9. A
10. B
11. C

Short answer

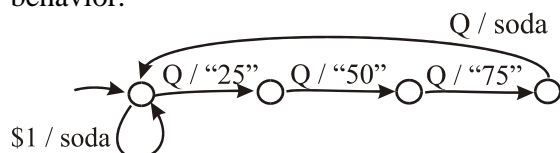
1. Is interaction more powerful than algorithms?
Give reasons why and why not.
2. What is a cellular automaton? Relate this to human biology.
3. Describe the inputs, outputs, and connectivity of a neuron. How does a neural net learn?
4. Describe a controversy relating to the modeling of interactive computation.
5. Describe the case presented in class that the Persistent Turing machine is a more expressive model of computation than the Turing machine.
6. Describe an interactive model of computation (better, two such models).
7. What is a transducer, and what sort of behavior is associated with one?
8. Draw a transducer that always outputs a 1 on input 0, and a 0 on input 1.
9. Define the following and relate to parallel or distributed computing:
 - a. Java threads
 - b. CRCW
 - c. SIMD
 - d. Cellular automata
 - e. Neural networks
 - f. Sensor networks
 - g. Mesh architecture
 - h. PRAM
 - i. Cache coherency
10. For the terms below, (a) give a definition; (b) give examples of models; (c) distinguish from the other terms in the list; (d) describe the role of synchrony/asynchrony if it applies
 - a. concurrency
 - b. parallelism
 - c. distributed processing
 - d. sequential interaction
 - e. multi-stream interaction
 - f. algorithm execution
 - g. coordination

Longer answer

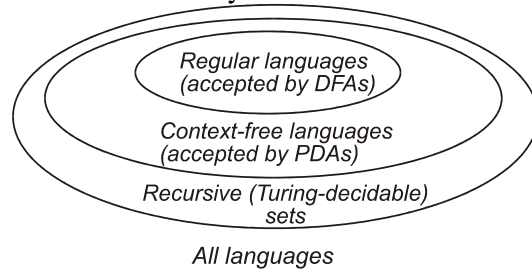
1. What is the term for computation in which more than one computing entity executes at the same time? Give some general subcategories of this kind of computation.
2. Distinguish the Persistent Turing Machine from the Turing Machine and from finite transducers.

Multiple topics

- What is the connection among languages, models of computation, and decision problems?
- Why are Cantor's, Godel's, and Turing's proofs similar? Begin by stating briefly the assertions proven by these proofs. Refer to proof methods.
- Give at least two examples from this course of proofs by
 - diagonalization
 - construction
- What were some of the different meanings of δ in our classroom?
- In the language theory we used, as applied to different automata models, distinguish \emptyset , λ , and \emptyset (a.k.a. '#').
- Describe and compare the following languages, with reference to automata discussed in this course
 - $\{0^m 1^n \mid m \neq n\}$
 - $\{P \mid P \text{ is the code for an } \mathcal{S}\text{-language program that halts}\}$
 - $11^*(0|11)^*$
 - $\{xx \mid x \in \Sigma^*\}$
- Explain how your research project will relate to one or more of the core topics of the course, including specifics.
- Describe the model of computation that would correspond best to the work you did in your research project. If your research project was itself about a model of computation, say so and relate it to at least one other model.
- What are the common elements among the transition-system-based models that we discussed?
- Suppose you wished to show that there is *no* one-to-one correspondence between two sets of sequences observable in some computational or mathematical context. What approach would you use? Give an example.
- (a) What is this? (b) Characterize its observable behavior.



- What does this say?



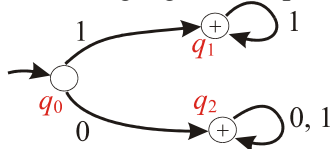
- Distinguish among λ , \emptyset , and \emptyset , as used in the course.
- Distinguish among Σ , Σ^* , and Σ^∞ as used in the course, with examples.
- What computational issues arise in speech recognition?
- Would you use *formal proof*, *simple counterexample*, or *evidence* to argue for the following? Explain.
 - the Church-Turing thesis
 - cellular automata are computationally equivalent to Turing machines
 - a given language is not regular
 - Kleene's theorem
 - a given DFA does not accept the language defined by a given regular expression
 - neural nets are computationally equivalent to Persistent Turing machines
- Which of the following are *sets*, *sequences*, *languages*, or *sets of languages*? (The categories may be overlapping) Give examples of each.
 - all CFLs
 - an alphabet
 - all Z accepted by a particular DFA M
 - a single item accepted by a particular DFA M
- Give an examples of each of the following and relate it to some model of computation:
 - a finite sequence of symbols
 - a finite sequence of (input, output) pairs
 - a finite set of finite sequences
 - a set of sets of finite sequences
 - an infinite set of finite sequences
 - an infinite sequence
 - an infinite set of infinite sequences

14. The following list of concepts relate to three major topics discussed in the course. Arrange the terms into a three-column table and explain it:
- regular expression*
pushdown automaton
Turing machine
context-free language
recursive set
deterministic finite automaton
computable problem
regular language
context-free grammar
15. Which is, or is representable by, a (set, graph, language, function, sequence)? Explain in one or two sentences.
- a DFA
 - a finite transducer
 - what is computed by a DFA
 - what is accepted by a DFA
 - a PDA
 - what is computed by a PDA
 - what is accepted by a PDA
 - what is computed by a TM
 - what is accepted by a DFA
16. Which of the following halt (a)lways, (s)ometimes, or (n)ever? For each category (*a*, *s*, *n*) state the general reason for halting, sometimes halting, or never halting.
- ___ C++ programs
 - ___ \mathcal{L} -language programs
 - ___ DFAs
 - ___ PDAs
 - ___ TMs
 - ___ Transducers
 - ___ Window 2000 GUI
17. State and explain the relationship among transducers, Turing machines, and at least one of the following:
- Expert systems
 - Cellular automata
 - Video games
 - Neural nets
 - Voice-recognition systems
 - Learning systems
 - Compilers
18. Relate the computational power of the \mathcal{L} language to the power of DFAs and of Turing machines.
19. Name two ways in which a finite transducer differs from a DFA, PDA, or TM.
20. What is the *range* of δ , for Turing machines? What does this allow TMs to do that PDAs can't do?
21. Does a transducer compute an algorithm? Why or why not?
22. What models of computation that we have discussed would be model a learning agent and why?
23. What data structure does a PDA have that DFA lacks? Briefly explain.
24. In the context of this course, what does $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \cup \{L, R\}$ mean and what model uses it?
25. In the context of this course, what does $\delta : Q \times \Sigma \rightarrow Q$ mean and what model uses it?
26. Why do you think that the following would not be considered sufficient as an entry in a bibliographic reference list following a journal paper?
<http://www.cs.bu.edu/~anson/res/rl/nsf2002.pdf>
27. What is this?
 $S \rightarrow SS \mid a \mid \lambda$
28. What sort of device accepts a language that can be described by a grammar that consists of *productions*?
29. What sort of device accepts a language that can be described by an expression that uses the +, *, and concatenation operators?
30. What is a regular expression and what sort of model of computation is associated with it?
31. In a cellular automaton, what sort of rules apply?
32. What model of computation is considered equivalent in expressiveness to a programming language such as C++?

33. In terms of discrete-mathematics categories such as *sets*, *sequences*, *functions*, and *languages*, discuss the formal aspect of transition-system-based models of computation (the classes of automata discussed in our seminar). You may do this by specifying an instance of each model as a tuple, $M = \langle \dots \rangle$, saying what *set*, *sequence*, *function*, or *language* each member of the tuple stands for, and saying what sort of *set*, *sequence*, *function*, or *language* is accepted or computed by the automaton.

34. Categorize the model below and specify what happens at each step of the two computations that occur on inputs of *baaa* and *aaba*, along with the results of these computations.

35. What language is accepted by this:



36. What is this and what model of computation corresponds to it?

$$S \rightarrow SS$$

$$S \rightarrow a$$

$$S \rightarrow \lambda$$

37. In the context of this course, what does

$$\delta : Q \times \Sigma \rightarrow Q$$

mean and what model uses it?

38. What two models that we discussed correspond to the notion of *algorithmic computability*? Describe them in one or two sentences.

39. What famous open question is equivalent to the question of whether the Satisfiability problem is tractable? What does “tractable” mean? What topic in the course plan are we talking about here?

40. Give language-related examples showing contexts in which λ (empty string), $\{\}$ (null set), and \flat or $\#$ (blank symbol) are used in this course, showing that they differ.

41. Explain what is meant by
 $(\forall L \subseteq \Sigma^*) \text{Regular}(L) \Rightarrow \text{Context-free}(L)$,
 and why it is true.

42. If language L is accepted by a Turing machine, then is it necessarily context free? Explain the relationship between CFLs and recursive sets, referring to the respective models of computation.

43. Relate the notions of interaction, Turing machines, and parallel computation.

44. How is artificial intelligence said to relate to models of computation?

45. Comment on the assertion that intelligence is a process of search.

46. In each category below, (a)ll halt, (s)ome, or (n)one. For each, choose *a*, *s*, or *n*. State the general reason why all halt, some halt, or none halt; this may relate to input, output, or computation.

- | | |
|-------------------------------------|----------------|
| a. C++ programs | d. PDAs |
| b. \mathcal{L} -language programs | e. TMs |
| c. DFAs | f. Transducers |

47. In terms of discrete-mathematics categories such as *sets*, *sequences*, *functions*, and *languages*, discuss the formal aspect of transition-system-based models of computation (the main classes of automata discussed in our seminar). You may do this by specifying an instance of each model as a tuple, $M = \langle \dots \rangle$, saying what *set*, *sequence*, *function*, or *language* each member of the tuple stands for, and saying what sort of *set*, *sequence*, *function*, or *language* is accepted or computed by the automaton. Use one of the machines from your answers in part A as an example, if you wish.

48. Discuss at least one of the following from a *computational* point of view and explain its relationship to DFAs, PDAs, transducers, or Turing machines:

1. Expert systems
2. Cellular automata
3. Video games
4. Neural nets
5. Voice-recognition systems
6. Learning systems
7. Perceptrons
8. Persistent Turing machines
9. Compilers