Section	**Faculty**	**Page**
Table of Contents		*1*
Artificial Intelligence	[Unknown]	2
Artificial Intelligence solutions		6
Automata and Formal Languages	[Unknown]	9
Automata and Formal Languages solutions		12
Compilers	[Unknown]	15
Compilers solutions		20
Computer Architecture	[Unknown]	22
Logic solutions		28
Numerical Analysis	[Unknown]	29
Numerical Analysis solutions		31
Software Systems	[Unknown]	34
Software Systems solutions		36

Computer Science Department
Stanford University
Comprehensive Examination in Artificial Intelligence
Autumn 1998

October 27, 1998

PLEASE READ THIS FIRST

- You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.
- Be sure you have all the pages of this exam. There are three pages in addition to this.
- This exam is OPEN BOOK. You may use notes, articles, or books — but no help from people or computers.
- Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea does not work out. You can also get credit for realizing that certain approaches are incorrect. On a true/false question, you might get partial credit for explaining why you think something is true when we think it is false.
- Points in this exam add up to 60. Points are allocated according to the number of minutes we believe a student familiar with the material should take to answer the questions. **IF YOU ARE TAKING TOO LONG ON A QUESTION, WRITE DOWN WHATEVER YOU HAVE AND MOVE ON.**

1 Search (12 points)

The US Geological Survey asks you to write a program that colors any map with three colors in such a way that no two bordering countries have the same color. They give you access to a list C of countries, and to a predicate **neighbor(c, d)** that returns **true** iff countries **c** and **d** border each other.

a. (3 points) Formulate this problem as a blind tree search. In particular, state clearly what nodes and edges mean in your tree, how nodes are expanded, and which search algorithm is used. Just name the search algorithm, do not spell it out. Better solutions get more credit.

b. (3 points) When does your algorithm find out if no solution is possible?

c. (3 points) Would a heuristic-repair algorithm be appropriate for your map coloring problem? Why or why not?

d. (3 points) Independently of your solution above, does it make sense to use A^* to solve this problem? Why or why not?

2 Robotics (4 points)

a. (2 points) Draw or describe the configuration space of a screw in a bolt. Only the screw can move.

b. (2 points) Draw or describe the configuration space of a pendulum in which the bob is free to move in two dimensions (that is, *not* a planar pendulum). Make sure that the topology of your configuration space is correct, in the sense that nearby configurations are represented by nearby points.

3 Logic-Based Knowledge Representation (9 points)

a. (3 points) Represent in first-order logic the assertion **Women do not love men who hate all women.**

b. (3 points) Represent in the situation calculus the assertion **After cleaning the room, all the toys are off the floor.**

c. (3 points) Using the predicates **Dog(x)**, **Cat(x)**, and other predicate symbols of your choice, write a first-order theory (that is, a set of first-order sentences) in every model of which the number of cats and dogs is the same. [Hint: You may use the equality predicate =.]

4 Logic (12 points)

a. (8 points) What does a model of a first-order theory consist of?

b. (4 points) True or false:

(i) Every satisfiable formula has a model

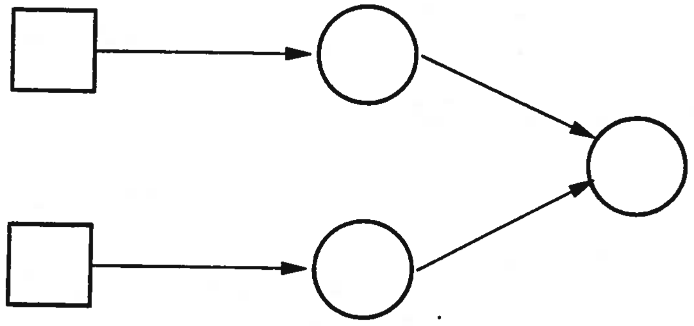
(ii) A formula is satisfiable if and only if it is not valid

(iii) Skolemizing can turn a satisfiable formula into an unsatisfiable one

(iv) If A entails B then every model of A is also a model of B

5 Learning (6 points)

Consider a neural network architecture with two input units and a single hidden layer with two units. We restrict each of the hidden units to have at most one input (but they can both have the same input). An example is given in the figure.



Assume that the inputs to all units can take on only binary values.

a. (3 points) Is the expressive power of such a unit equivalent to a single perceptron unit? Explain briefly.

b. (3 points) Is the expressive power of such a unit equivalent to a standard feed-forward neural network with two hidden units? Explain briefly.

6 Planning (17 points)

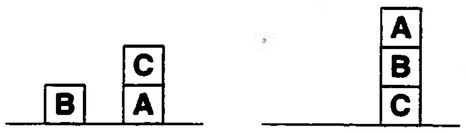
The operators used by STRIPS for standard block-stacking problems are

move(x, y, z): move block x from block y to block z

remove(x, y): move block x from block y to TABLE

stack(x, z): move block x from TABLE to block z

The Sussman anomaly, illustrated in the figure below, pointed out a basic problem with noninterleaved planners.



STRIPS will get stuck trying to either put B on top of C, or to clear A and put it on top of B. Either way, STRIPS will fail to find a plan.

a. (2 points) Using a STRIPS-like language, describe the initial state for the figure above. Your description should be detailed enough to let a complete planner find a solution.

b. (2 points) Do the same for the goal state.

c. (2 points) Write formal definitions in a STRIPS-like language for the three operators **move**, **remove**, **stack** mentioned above.

d. (3 points) Can STRIPS get lucky and find a plan for the Sussman anomaly problem above in some circumstances? If so, state which circumstances. If not, explain why not.

e. (3 points) Removing the **move** operator will force STRIPS to solve any stacking problem by first placing anything that needs to be moved onto the table, and then stacking everything back again in the required order. Does this solve the Sussman anomaly for block-stacking problems? Why or why not?

f. (5 points) Show how a partial-order planner would determine a plan for this problem. Draw a diagram illustrating a solution plan, including *all* ordering links and *all* causal links. Please use dashed lines for ordering links, and solid lines for causal links. Please also label with TR any ordering link that has been included for threat resolution. Also state a final plan as a sequence of operations.

Computer Science Department
Stanford University
Comprehensive Examination in Artificial Intelligence
Autumn 1998

Solution Samples

1 Search

a. Several answers are possible. In one, nodes are partial colorings of the map, and edges denote adding a color for a blank country. For each country in C, pick a color not yet used for that country, and use **neighbor** to check if the coloring is valid. Continue depth-first with the next country if the coloring is valid, and backtrack otherwise.

A better idea is to spend the n^2 time it takes to create a graph of the countries by using the **neighbor** predicate. One can then use a depth-first or breadth-first search on a spanning tree of the graph. Finding neighbors during search is a more expensive option, since work is done anew in case of backtracking.

The graph solution is better because it essentially conforms to the most-constrained-variable heuristic. Other solutions are possible.

b. Failure can be declared by the first algorithm above when backtracking from a country for which the last available color has been tried. The second algorithm may find out earlier if it checks the degree of every node in the graph.

c. Yes. Heuristic repair algorithms are specifically designed for constraint satisfaction problems, of which the map coloring problem is an instance. How well it works, however, depends on the initialization.

d. You may be tempted to answer "no," because no cost has been defined for a solution. The correct answer, however, is "yes," because appropriate costs can be defined. For instance, heuristic repair can be seen as a tree search where each node is a complete coloring, and improperly-colored nodes are selected for modification. Then, the cost of a solution could be the number of changes necessary to repair the map.

2 Robotics

a. A straight line. The parameter is the turning angle of the screw.

b. A torus. The parameters are polar angles of the bob in some reference system.

3 Logic-Based Knowledge Representation

a. $\forall xy\,(\text{Woman}(x) \land \text{Man}(y) \land \forall z(\text{Woman}(z) \Rightarrow \text{Hates}(y,z)) \Rightarrow \neg\text{Loves}(x,y)$

b. $\forall s\,\text{HOLD}(\text{Result}(\text{CleanRoom}, s), \forall x(\text{Toy}(x) \Rightarrow \text{OffFloor}(x)))$

c.

$$\begin{array}{l}
\forall x\,(\text{Dog}(x) \Rightarrow \exists y(\text{Cat}(y) \land \text{Corresponds}(x,y))) \\
\forall x\,(\text{Cat}(x) \Rightarrow \exists y(\text{Dog}(y) \land \text{Corresponds}(x,y))) \\
\forall xyz\,\text{Corresponds}(x,y) \land \text{Corresponds}(x,z) \Rightarrow y = z \\
\forall xyz\,\text{Corresponds}(x,y) \land \text{Corresponds}(z,y) \Rightarrow x = z
\end{array}$$

4 Logic

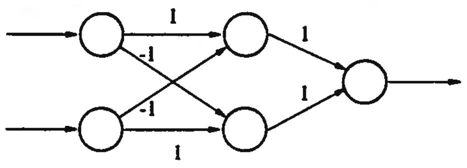
a. A FO model is a tuple (D,F,R,p,q) where the domain D is a set, F is a set of functions defined on D (each with its own arity), R is a set of relations defined on D (each with its own arity), and the function p (q) maps the function (predicate, resp.) symbols in the logic to F (R, resp.). The variable assignment, which maps variables to D, is also sometimes considered part of the model.

b. (i) yes, (ii) no, (iii) no, (iv) yes.

5 Learning

a. Yes. The additional units can only change the weights that inputs are multiplied by, so they can be subsumed by the two hidden units.

b. No. In a standard feed-forward architecture, each input unit can connect to both hidden units. This gives greater expressive power. For instance, XOR cannot be computed by the perceptron, or equivalently (from the previous answer) by the network in the question figure. On the other hand, XOR can be computed with a two-layer neural network:



Thresholds are zero in this network.

6 Planning

a. On(B, TABLE) ∧ Clear(B) ∧ On(A, TABLE) ∧ On(C, A) ∧ Clear(C).

b. On(C, TABLE) ∧ On(B, C) ∧ On(A, B) ∧ Clear(A). The last clause is optional.

c. Op(ACTION: Move(x, y, z),
    PRECONDITION: On(x, y) ∧ Clear(x) ∧ Clear(z),
    EFFECT: On(x, z) ∧ Clear(y) ∧ ¬ Clear(z))

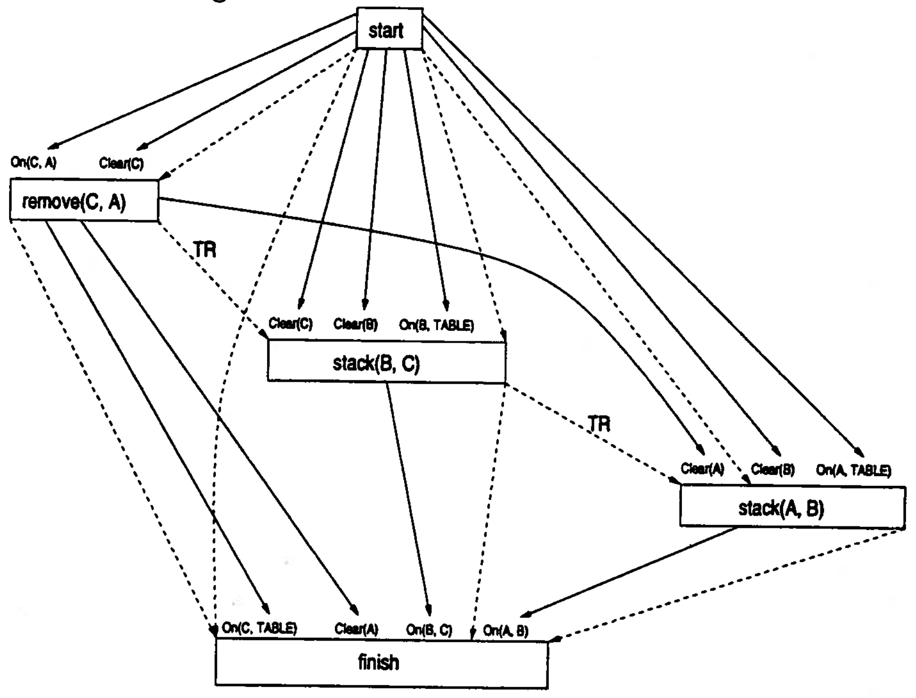
   Op(ACTION: Remove(x, y),
    PRECONDITION: On(x, y) ∧ Clear(x),
    EFFECT: On(x, TABLE) ∧ Clear(y))

   Op(ACTION: Stack(x, y),
    PRECONDITION: On(x, TABLE) ∧ Clear(x) ∧ Clear(y),
    EFFECT: On(x, y) ∧ ¬ Clear(y))

d. Yes, STRIPS can get lucky if the clauses in the goal staqte are given in the order On(C, TABLE), On(B, C), On(A, B), and if STRIPS creates subgoals top-down.

e. No. STRPIS will still attempt both bad moves mentioned in the description of the Sussman anomaly.

f. Here is a solution diagram.



The resulting plan is Remove(C, A), Stack(B, C), Stack(A, B).

Computer Science Department
Stanford University
Comprehensive Examination in Automata and Formal Languages
Autumn 1998

October 26, 1998

READ THIS FIRST!

1. You should write your answers for this part of the Comprehensive Examination in BLUE BOOKS. There are five problems in the exam. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

2. The number of POINTS for each problem indicates how elaborate an answer is expected. The exam takes 1 hour.

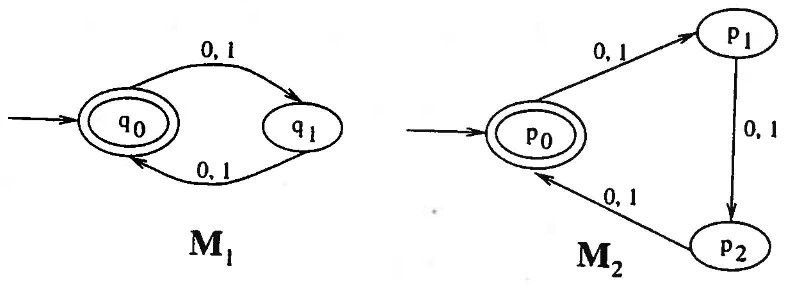
3. This exam is OPEN BOOK. You may use notes, articles, or books—but no help from other sentient agents such as other humans or robots.

4. Show your work, since PARTIAL CREDIT will be given for incomplete answers.

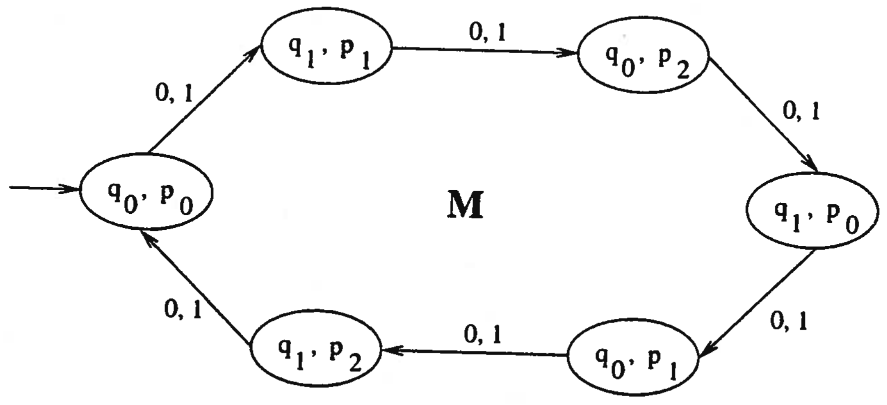
Comprehensive Exam: Autumn 1998-99

Automata and Formal Languages (60 points)

Problem 1. [10 points]
Consider the following DFAs (deterministic finite-state automata) called M_1 and M_2 over the alphabet $\{0, 1\}$. Let L_1 be the language of M_1 and L_2 be the language of M_2.



a). [4 points]
Give succinct descriptions of the languages L_1 and L_2.
b). [6 points]
Consider the machine M given below that is the cross-product of the two machines M_1 and M_2. (Running the machine M on an input string corresponds to running M_1 and M_2 together in parallel on that input string. Each state in the machine M corresponds to a pair of states, one each from M_1 and M_2. Each transition in M is the combination of the state transitions from M_1 and M_2.)



For each of the following languages, specify the choice of final states in M that would cause M to accept that language.
(i). $L_1 \cup L_2$
(ii). $L_1 - L_2$ (the set of strings in L_1 that do not belong to L_2)
(iii). $\overline{L_1}$

Problem 2. [10 points]
In the following, R denotes a *regular* language, and C, C' denote *context-free* languages. Classify each of the following statements as being TRUE or FALSE. *You will receive 2 points for each correct answer and −1 point for each incorrect answer.*
a). There must exist a *deterministic* push-down automata that accepts R.
b). There must exist a *deterministic* Turing machine that accepts $C \cap C'$.
c). $R \cap C$ must be regular.
d). $C \cup C'$ cannot be regular.
e). $\overline{C}$ must be recursive.

Problem 3. [20 points]
Classify each of the following languages as being in one of the following classes of languages: *empty, finite, regular, context-free, recursive, recursively enumerable.* You must give the *smallest* class that contains *every possible language* fitting the following definitions. For example, the language of a DFA M could be *empty* or *finite*, and must always be *context-free*, but the smallest class that is appropriate *regular.*
a). $L = \{a^ib^jc^kd^l \mid i = j \text{ or } k = l\}$.
b). $L = \{a^ib^jc^kd^l \mid i = j \text{ and } k = l\}$.
c). $L = \{a^ib^jc^kd^l \mid i = j \text{ and } j = k\}$.
c). $L = \{a^ib^jc^kd^l \mid i = l \text{ and } j = k\}$.
e). $L = \{a^ib^jc^kd^l \mid i \times j \times k \times l \text{ is divisible by } 5\}$.
f). The language of a push-down automaton with only one state.
g). The set of strings encoding all push-down automata that accept non-recursive languages.
h). The language of a PDA with two stacks.
i). The complement of a language in NP.
j). The intersection of a recursive language and a recursively enumerable language.

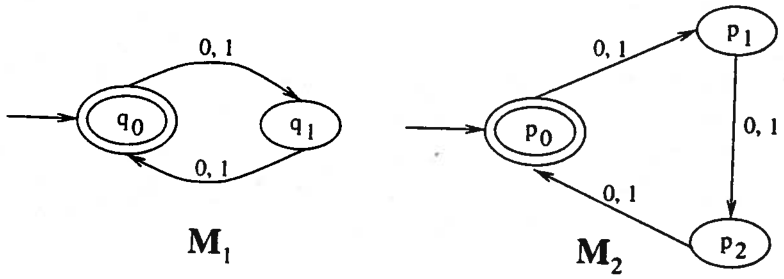
Problem 4. [10 points]
Suppose we have languages A and B such that A has a *polynomial-time reduction* to B. Classify each of the following statements as being TRUE or FALSE. *You will receive 2 points for each correct answer and −1 point for each incorrect answer.*
a). If B is recursively enumerable, then A must be recursively enumerable.
b). If A is recursive, then B must be recursive. ✗
c). If B is NP-hard, then A must be NP-hard.
d). If A is NP-complete, then B must be NP-complete.
e). It is possible that A is solvable in polynomial time but B is not even in NP.

Problem 5. [10 points]
For any language L define its reversal as follows:
$$L^R = \{w^R \mid w \in L\}.$$
Recall that for a string $w = w_1w_2 \ldots w_k$, we define its reversal as $w^R = w_n \ldots w_2w_1$.
Suppose that L is an NP-complete language. Then, is L^R also an NP-complete language? For full credit, you must sketch a proof for your answer.

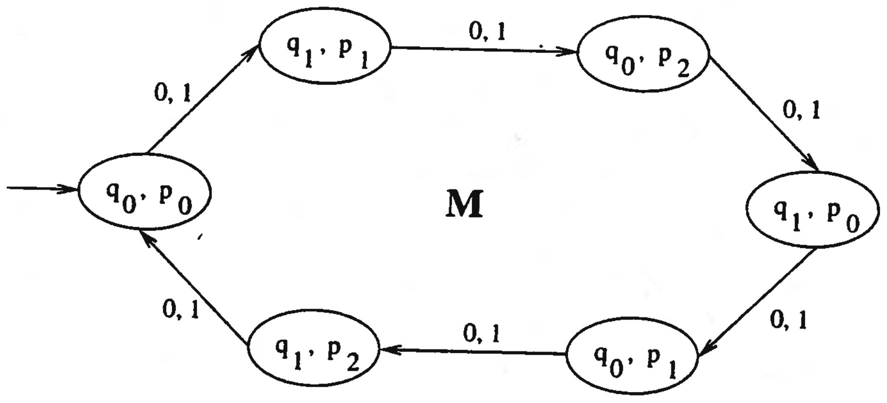
Comprehensive Exam: Autumn 1998-99

Automata and Formal Languages (60 points)
Sample Solutions

Problem 1. [10 points]
Consider the following DFAs (deterministic finite-state automata) called M_1 and M_2 over the alphabet $\{0, 1\}$. Let L_1 be the language of M_1 and L_2 be the language of M_2.



a). [4 points]
Give succinct descriptions of the languages L_1 and L_2.
b). [6 points]
Consider the machine M given below that is the cross-product of the two machines M_1 and M_2. (Running the machine M on an input string corresponds to running M_1 and M_2 together in parallel on that input string. Each state in the machine M corresponds to a pair of states, one each from M_1 and M_2. Each transition in M is the combination of the state transitions from M_1 and M_2.)



For each of the following languages, specify the choice of final states in M that would cause M to accept that language.
(i). $L_1 \cup L_2$
(ii). $L_1 - L_2$ (the set of strings in L_1 that do not belong to L_2)
(iii). $\overline{L_1}$

Solution:
The language L_1 contains all strings of even length from $\{0,1\}^*$. The language L_2 contains all strings of length divisible by 3.
(i). For $L_1 \cup L_2$, make (q_0, p_0), (q_0, p_2), (q_1, p_0) and (q_0, p_1) the final states.
(ii). For $L_1 - L_2$, make (q_0, p_2) and (q_0, p_1) the final states.
(iii). For $\overline{L_1}$, make (q_1, p_1), (q_1, p_0), and (q_1, p_2) the final states.

Problem 2. [10 points]
In the following, R denotes a *regular* language, and C, C' denote *context-free* languages. Classify each of the following statements as being TRUE or FALSE. *You will receive 2 points for each correct answer and −1 point for each incorrect answer.*
a). There must exist a *deterministic* push-down automata that accepts R.
b). There must exist a *deterministic* Turing machine that accepts $C \cap C'$.
c). $R \cap C$ must be regular.
d). $C \cup C'$ cannot be regular.
e). $\overline{C}$ must be recursive.

Solution:
a). TRUE
b). TRUE
c). FALSE
d). FALSE
e). TRUE

Problem 3. [20 points]
Classify each of the following languages as being in one of the following classes of languages: *empty, finite, regular, context-free, recursive, recursively enumerable.* You must give the *smallest* class that contains *every possible language* fitting the following definitions. For example, the language of a DFA M could be *empty* or *finite*, and must always be *context-free*, but the smallest class that is appropriate *regular.*
a). $L = \{a^i b^j c^k d^l \mid i = j \text{ or } k = l\}$.
b). $L = \{a^i b^j c^k d^l \mid i = j \text{ and } k = l\}$.
c). $L = \{a^i b^j c^k d^l \mid i = j \text{ and } j = k\}$.
c). $L = \{a^i b^j c^k d^l \mid i = l \text{ and } j = k\}$.
e). $L = \{a^i b^j c^k d^l \mid i \times j \times k \times l \text{ is divisible by } 5\}$.
f). The language of a push-down automaton with only one state.
g). The set of strings encoding all push-down automata that accept non-recursive languages.
h). The language of a PDA with two stacks.
i). The complement of a language in NP.
j). The intersection of a recursive language and a recursively enumerable language.

Solution:
a). context-free
b). context-free
c). recursive

d). context-free
e). regular
f). context-free
g). empty
h). recursively enumerable
i). recursive
j). recursively enumerable

Problem 4. [10 points]
Suppose we have languages A and B such that A has a *polynomial-time reduction* to B. Classify each of the following statements as being TRUE or FALSE. *You will receive 2 points for each correct answer and −1 point for each incorrect answer.*
a). If B is recursively enumerable, then A must be recursively enumerable.
b). If A is recursive, then B must be recursive.
c). If B is NP-hard, then A must be NP-hard.
d). If A is NP-complete, then B must be NP-complete.
e). It is possible that A is solvable in polynomial time but B is not even in NP.

Solution:
a). TRUE
b). FALSE
c). FALSE
d). FALSE
e). TRUE

Problem 5. [10 points]
For any language L define its reversal as follows:

$$L^R = \{w^R \mid w \in L\}.$$

Recall that for a string $w = w_1w_2 \ldots w_k$, we define its reversal as $w^R = w_n \ldots w_2w_1$.

Suppose that L is an NP-complete language. Then, is L^R also an NP-complete language? For full credit, you must sketch a proof for your answer.

Solution: Indeed, L^R is NP-complete. To prove this, we need to show two things: that L^R is in NP and that it is NP-hard. First observe that a Turing machine given an input w can reverse its input and obtain w^R in polynomial time. Thus, given any Turing machine for a language L we can construct another one for the language L^R with only a polynomial overhead in the running time. Since L must be in NP, there exists a nondeterministic Turing machine M for L that runs in polynomial time. If M is modified to first reverse its input (from w to W^R), then it accepts exactly the language L^R in polynomial time, implying that L^R is also in NP. To show NP-hardness, we give a reduction from the NP-hard language L to L^R. The reduction merely takes a string w and outputs w^R; clearly, this is a polynomial-time reduction. The correctness of the reduction follows from the definition: $w \in L$ if and only if $w^R \in L^R$.

Computer Science Department
Stanford University
Comprehensive Examination in Compilers
1998

October 29, 1998

READ THIS FIRST!

1. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

2. Be sure you have all the pages of this exam. There are 4 pages.

   The exam takes 1 hour.

3. This exam is CLOSED BOOK.

4. Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea doesn't work out; you can also get credit for realizing that certain approaches are incorrect. On a true/false question, you might get partial credit for explaining why you think something is true when it is actually false. But no partial credit can be given if you write nothing.

Compilers

This exam is *closed book.*

1. (20 points) For each of the statements below, circle "true" or "false", as appropriate. Unless there is an explicit statement to the contrary, assume that all context-free grammars have *no* useless symbols. Some of these questions are intentionally tricky!

[✓ **true**] **false** Every LR(2) grammar is also LR(1).

[✓ **true**] **false** If a CFG with no useless symbols has a single nonterminal with both left-recursive and right-recursive productions, the CFG is ambiguous.

[✓ **true**] **false** If a CFG is ambiguous, its grammar cannot be LR(1). and $\alpha \neq \beta$

[✓ **true**] **false** If a CFG has productions $A \to \alpha$ and $A \to \beta$ and both α and β are nullable (i.e., derive the empty string), the CFG is ambiguous.

[✓ **true**] **false** Some unambiguous context-free grammars can have more than one derivation for the same input string.

[✓ **true**] **false** The language of a CFG is empty if and only if the sentence symbol is useless.

[✓ **true**] **false** For every CFG with useless symbols, there is an equivalent CFG with no useless symbols, if the language of the original CFG is non-empty.

true **false** An LR(1) and SLR(1) parser for the same CFG will detect errors at exactly the same point in the parse for every input string.

true **false** An SLR(1) parser never does more reduce actions than shift actions during a parse.

true **false** Every context-free language has at least one CFG with no left recursion.

2. (7 points) Suppose that Lex or Flex were given regular expressions for different lexemes in the following order: a, ab^*a, a^+c, ca, and ba. Draw vertical lines to show the sequence of lexemes that would be recognized in the following input string using the longest lexeme rule as used in Lex or Flex (this rule says that, if several prefixes of the input string match patterns, the longest such prefix should be returned by the lexer).

$$a\ b\ a \mid a\ c \mid a\ b\ a \mid a\ a \mid b\ a \mid a$$

22

3. (15 points) The following CFG is ambiguous:

$$\begin{array}{ll} 0 & S \to E \\ 1 & E \to E\%E \\ 2 & E \to E\#E \\ 3 & E \to E@ \\ 4 & E \to id \end{array}$$

(a) The SLR(1) parse table below shows the multiple entries that would result. Write down the entries (row, column, and entry) that should be *deleted* to obtain a parser which gives @ the highest precedence, gives # the next highest precedence and makes it *left-associative* and gives % the lowest precedence and makes it right-associative.

	ACTION					GOTO
state	$	id	%	#	@	E
0		s6				1
1	acc		s2	s4	s7	
2		s6				3
3	r1		s2, r1	s4, r1	s7, r1	
4		s6				5
5	r2		s2, r2	s4, r2	s7, r2	
6	r4		r4	r4	r4	
7	r3		r3	r3	r3	

(b) Show the sequence of stacks and inputs that occurs when parsing the string "id#id%id@" using the resulting parse table. Your answer should be in the format shown below: each line should have the stack of states written horizontally with the top to the right, and the remaining input to be parsed.

Stack (top at right)	Input
0	id # id % id

4. (8 points) This question and the next are based on the stack machine instruction set summarized on the last page of the exam.

The following is a fragment of a context-free grammar for expressions in a programming language. Add actions in C for each production in the grammar to generate code that computes the value of the expression (just print the instructions to the standard output using `printf`). When this code is executed, it should have the effect of pushing the expression value on top of the stack without disturbing any of the old stack entries.

Code is generated by other parts of the grammar that aren't shown. In particular, the nonterminals **`varname`** and **`literal`** have productions, that, when reduced, generate code that leaves the value of the variable on the top of the stack. You may assume that all variables and values are one word long.

```
expr    :     expr '+' expr    { printf("add"); }

        :     expr '-' expr    { printf("sub"); }

        :     expr '*' expr    { printf("mul"); }

        :     varname

        :     '(' expr ')'

        :     literal
```

5. (10 points) The following is a fragment of a grammar for "while" loops. A while loop first tests whether its expression is true; if so, the loop executes its statement and repeats; otherwise, the loop exits. Rearrange the grammar and add reduce actions to generate code for the loop, using the stack machine instruction set described below.

You should assume that the grammar is being parsed using a LALR parser such as YACC. However, YACC has a useful feature where you can insert actions between symbols in the right-hand side of a production. *Do not do this feature.* We want you to add actions only at the ends of the productions, to be executed when the production is reduced by the parser. This requires *rewriting* the production into an equivalent form, so you can perform each action at the right time during the parse.

Assume that expressions are compiled as in the previous question. Assume you have a function **`newlabel`** which returns a unique integer. For example, to generate a new label and define it, you might say: **`printf("L%d: ", newlabel())`**.

Hints: The answer requires a small amount of code. You will need to create labels. Note that labels can appear as a target of a jump before the label is defined (the assembler for the stack machine deals with keeping track of the values of labels). Be careful about nested loops!

```
whileloop       :  WHILE expr DO stmt ;
```

Stack Machine Summary

Here is a summary of the relevant part of a stack machine instruction set, for use in the previous problems. All stack entries and addresses are one word long. This is assembly language, which is translated into binary machine code by an assembler.

Labels

Labels are of the form "L123" (the letter L followed by a number). A label is defined when it appears followed by a colon before another label or instruction, in which case it is bound to the address of the next instruction. A label may be used in any context where a number can appear, and it is interpreted as the address to which it is bound. A label can be used in the text before it is defined (this is useful for forward jumps).

Instructions

`add` pop two entries, push their sum

`sub` subtract the top element from the second-from-top element.

`mul` pop two entries, push their product

`jump <label>` unconditional jump to constant address

`branch_true <op>` Pop top of stack; if it is non-zero, jump to the location described by the operand.

`branch_false <op>` Same, but jump if stack value is 0.

Compilers Solutions

1. (20 points)

   false. Every LR(2) LANGUAGE is also LR(1), but it may be necessary to modify an LR(2) CFG to get an equivalent LR(1) CFG.

   true. This is easily proved by sketching parse trees.

   true.

   true. If a leftmost derivation has $A \rightarrow \alpha \overset{*}{\Longrightarrow} \epsilon$, that can be replaced by $A \rightarrow \beta \overset{*}{\Longrightarrow} \epsilon$ to yield another leftmost derivation for the same terminal string.

   true. An unambiguous grammar cannot have more than one *leftmost* derivation for the same string, but there may be several derivations for the string, at most one of which is *leftmost*.

   true.

   true. Just delete the useless symbols and all productions that include them.

   false. The SLR(1) parser may perform more reductions before it "notices" that the next input doesn't work.

   false. Suppose you have a CFG with a lot of ϵ productions.

   true. There is an algorithm for eliminating all left recursion in the text.

2. (7 points) $a\ b\ a\ |a\ c\ |a\ b\ a\ |a\ a\ |b\ a\ |a$

3. (15 points)

   (a) row 3, col %, delete entry r1
   row 3, col #, delete entry s4
   row 3, col @, delete entry s7
   row 5, col %, delete entry s2
   row 5, col #, delete entry r2
   row 5, col @, delete entry s7

   (b)

Stack	Input
0	id # id % id $
0 6	# id % id $
0 1	# id % id $
0 1 4	id % id $
0 1 4 6	% id $
0 1 4 5	% id $

Stack	Input
0 1	% id $
0 1 2	id $
0 1 2 6	$
0 1 2 5	$
0 1	$
accept	$

4. (8 points)

```
expr     :       expr '+' expr  { printf("add"); }

         |       expr '-' expr  { printf("sub"); }

         |       expr '*' expr  { printf("mul"); }

         |       varname      { /* do nothing */ }

         |       '(' expr ')'     { /* do nothing */ }

         |       literal      { /* do nothing */ }
         ;
```

5. (10 points)

Here is what YACC would do if you included actions before and after the expression:

```
whileloop       :  WHILE M1 expr M4  DO stmt
                   {
                     printf("L%d: ", $4); /* end label */
                     printf("jump L%d\n", $2);
                   }
                ;

M1              : /* empty */  { printf("L%d: ", $$=newlabel()); }

M2              : /* empty */  { printf("br_false L%d\n", $$=newlabel()); }
```

The grammar can also be broken up in different ways.

It is important to keep the labels on a stack, so that labels for inner loops do not over-write labels for enclosing loops. In the solution above, we are using the value stack that YACC maintains (where it keeps **$2**, etc.).

Computer Science Department
Stanford University
Comprehensive Examination in Computer Architecture
Autumn 1998

October 28 1998

READ THIS FIRST!

1. All work to be done on the test itself! Fill in the blanks, NO BLUE BOOKS. There are 5 pages.

2. This exam is CLOSED BOOK. You may not use notes, articles, or books.

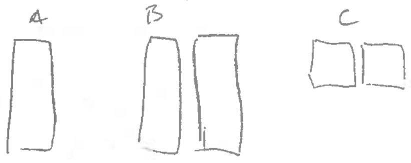
3. Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea doesn't work out; you can also get credit for realizing that certain approaches are incorrect. On a true/false question, you might get partial credit for explaining why you think something is true when it is actually false. But no partial credit can be given if you write nothing.

Computer Science Comprehensive Examination
Computer Architecture
[60 points]

Please do all of your work on these sheets. Do not do your work in a blue book.

Problem 1: Short Answer [2 points each, 12 points total] (Keep your answer to one line or less)

A. Suppose machine A has a 8K-byte direct mapped cache, machine B has a 16K-byte two-way set associative cache, and machine C has an 8K-byte two-way set associative cache. All caches have 16-byte lines. If the three machines process identical sequences of memory references, which of the following statements are true? Circle all that apply.



[✓ (a)] B's cache will always contain every line in A's cache
X(b) C's cache will always contain every line in A's cache
[✓ (c)] B's cache will always contain every line in C's cache
X(d) A's cache will always contain every line in C's cache

B. Which instruction set makes it easier to express instruction-level parallelism: a stack instruction set or a [✓ three-address general-register instruction set?]

C. A machine with register renaming is able to reorder instructions without regard to what type of dependencies? Circle all that apply.

(a) Data dependencies RAW
[✓ (b)] Output dependencies WAW
[✓ (c)] Anti-dependencies WAR
(d) Control dependencies br

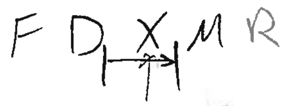
D. Adding an interleaved memory to a system changes which of the following memory parameters? Circle all that apply and denote the direction of change with an up arrow or a down arrow.

X(a) Memory latency
[✓ (b)] Memory bandwidth
X(c) Memory address space
X(d) Memory reliability

E. A RISC machine with a 5-stage pipeline uses the adder in the execution stage to perform branch address calculations. If we add a special branch address adder to the register read stage, what instructions will benefit? By how much?

unconditional branches, jumps

end of RR $\xrightarrow{\text{one cycle}}$ end of X



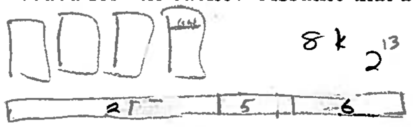
F. Which will have better throughput, a machine that uses scoreboarding (with no bypassing) to avoid read-after-write hazards, or a machine that uses bypassing to avoid these hazards?

bypassing

30

Problem 2: Cache Architecture [3 points each, 15 points total]

A. Suppose you have an 8Kbyte 4-way associative cache with a block size of 64 bytes. What size tag array (directory) is needed for this cache? Assume that addresses are 32-bits in length.

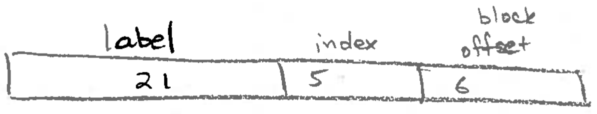


8 k 2^{13} 2^7 blocks → 2^5 lines

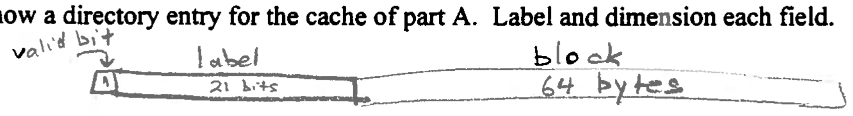
$2^7 \cdot 21$ bits

$\boxed{336 \text{ bytes}}$

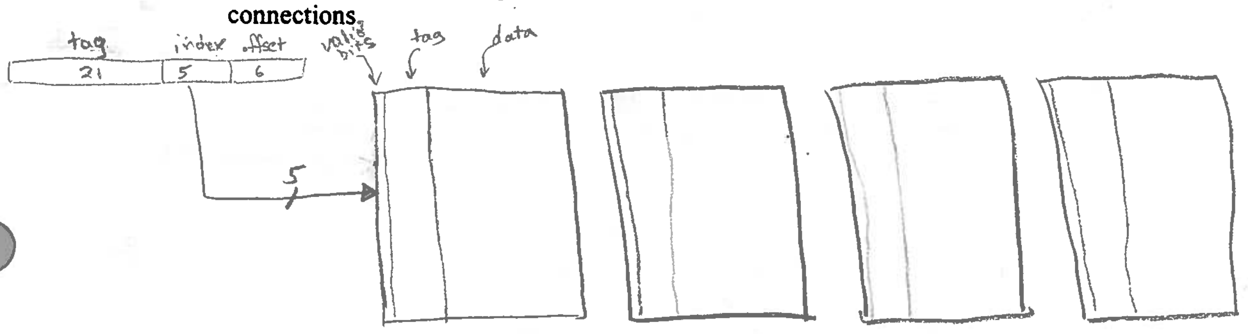
B. Show an address word divided into the fields used to access the cache of part A. Label and dimension each field.



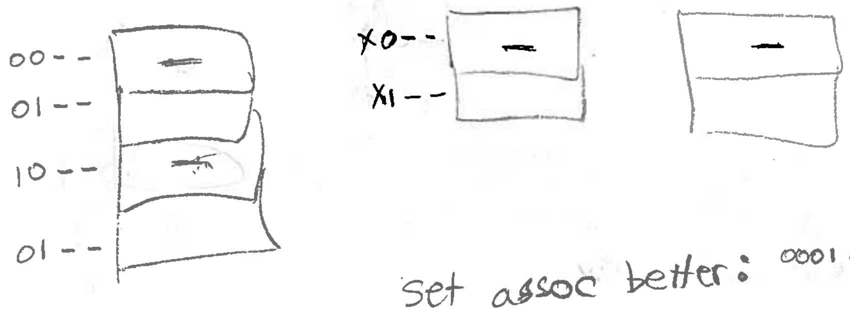
C. Show a directory entry for the cache of part A. Label and dimension each field.



D. Sketch a block diagram of the cache of part A. Show the tag array, data array, all multiplexers or tri-state drivers, and all comparators. Show, label, and dimension (with bit widths) all address and data connections.



E. Suppose you have two caches with a capacity of 4 4-byte blocks. One is direct mapped and the other is two-way set associative. Give an address sequence (byte addresses) for which the set associative cache outperforms the direct-mapped cache. Give a second sequence for which the direct-mapped cache outperforms the set associative cache.



set assoc better: 0001 0000, 0000 0000, 0001 0000 . .

dm better: 00 10 A00 B00 10 A00 B00 10

Read A0
Read A0
write B0
write B0

31

Problem 3: Pipeline Microarchitecture [10 points]

Suppose you have a CPU with a 6 stage pipeline containing the following stages:

- F: instruction fetch
- D: instruction decode
- R: register read (and branch address calculation)
- A: execute ALU operations (including determination of branch condition)
- M: memory load and store
- W: write back to register file

The register file cannot read a value that is being written in the same cycle. The machine has bypass paths from the output of the A, M, and W stages to both inputs of the ALU.

A. [2 points] What is the latency of an unconditional branch on this machine? Assume no prediction or speculation is employed. (Note: Latency is defined as the number of cycles from when this instruction is executed until the next instruction is executed. E.g., the latency of a simple instruction with no dependencies is one.)

FDRAMW
F

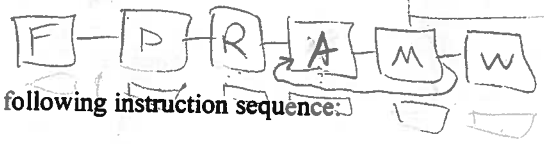
3 cycles

B. [2 points] What is the latency of a conditional branch? Again assume no speculation about branch direction or distance.

4 cycles

C. [3 points] The instruction register at each pipeline stage is denoted by IR_{stage}; for example IR_A is the instruction register at the ALU stage. The source and destination fields of the IR are denoted IR.A (source 1), IR.B (source 2), and IR.C (the destination). For example, the destination field of the IR at the M stage is IR_M.C. Each IR also has a valid field IR.V that indicates when it contains a valid instruction. Using this notation, write a logical expression that indicates when the bypass path from the output of the M stage to the second (B) input of the A stage should be activated.

$IR_W = \text{load} \wedge IR_A.B = IR_W.C \wedge IR_A.V \wedge IR_W.V$



D. [3 points] Consider the following instruction sequence:

```
Brz                         r5, FUBAR
Load                        r2 <- [r1 + 4]
Add                         r4 <- r2 + r3
```

Assuming that the branch is not taken and that no instructions are fetched speculatively, how many cycles does this take to execute from the time the IP points to the branch instruction until the result of the add is written to the register file? Explain your answer. You may want to draw a timeline.

12

1	2	3	4	5	6	7	8	9	10	11	12
F	D	R	A	M	W						
				F	D	R	A	M	W		
					F	D	R	–	A	M	W

32

Problem 4: Virtual Memory [12 points]

Consider a hypothetical machine with one-byte virtual addresses consisting of a 4-bit page field and a 4-bit offset field. The page field of a virtual address references a page table stored in page 0 of physical memory. Each entry of the page table is either the index of the page frame in physical memory that contains the page in question or the constant FF if the page is not in physical memory. At a given point in time the page 0 of physical memory has the following entries (all numbers are in hexadecimal):

page 0:

0: 07	8: 06
1: 01	9: 02
2: 03	A: 04
3: FF	B: 05
4: FF	C: FF
5: FF	D: FF
6: FF	E: FF
7: FF	F: 07

A. [3 points] What physical address, if any, corresponds to virtual address 84 (hex)?

64

B. [3 points] What virtual address, if any, corresponds to physical address 47 (hex)?

A7

C. [3 points] Is virtual address 47 (hex) in main memory?

no

D. [3 points] Is the mapping from virtual addresses to physical addresses one-to-one? Explain your answer?

no virtual addresses 00 and F0 refer to the same physical address

33

Problem 5: Performance [11 points]

Consider an unpipelined microprocessor that executes all instructions in a single 40ns clock cycle. The cycle is broken down into the following steps

```
10ns   Instruction fetch (read instruction referenced by IP)
5ns    Register read (fetch source operands of instruction)
5ns    ALU operations (execute ALU instructions, resolve
       branches)
15ns   Memory (read or write data memory)
5ns    Register write (write result to register file)
```

Suppose this machine executes an instruction mix that is 50% ALU operations, 20% loads, 10% stores, and 20% branches. Further assume that all instructions depend on the immediately preceding instruction and that the register file cannot be read and written simultaneously.

A. [3 points] What is the performance of this baseline machine in MIPS?

| 10 | 5 | 5 | 15 | 5 | 40×10^{-9} sec/cycle $\xrightarrow{1/x}$ $\boxed{25 \text{ MIPS}}$

B. [4 points] Suppose you are able to insert a single pipeline stage into this microprocessor without adding bypass paths or branch prediction. Where would you insert the pipeline stage (between which two of the above steps) to get the maximum performance benefit? What is the performance of the machine with this two-stage pipeline? Assume that adding a pipeline register adds no delay to the numbers above.

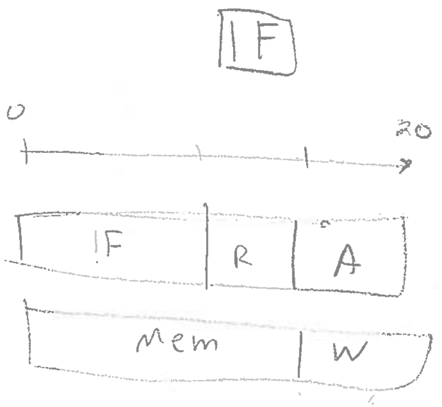
30 30

| IF | R, ALU | Mem, Write |

~~ALU → 30~~
LD → 30
ST → 30
Branch → 60

avg clock = 32 → 27.8 ~~MIPS~~

C. [4 points] Suppose you are able to insert an arbitrary number of pipeline stages into this microprocessor. How many stages would you insert? and where would you insert these stages (between which steps above) to get the maximum performance? What is the performance of this configuration?



34

SOLUTIONS
Comprehensive Examination in LOGIC
October 1998

	(A)	(B)	(C)	(D)	(E)
1.					X
2.		X			
3.				X	
4.			X		
5.		X			
6.	X				
7.				X	
8.			X		
9.		X			

	(A)	(B)	(C)	(D)	(E)
10.					X
11.		X			
12.		X			
13.			X		
14.				X	
15.				X	
16.			X		
17.			X		
18.				X	

Computer Science Department
Stanford University
Comprehensive Examination in Numerical Analysis
Autumn 1998

October 30, 1998

READ THIS FIRST!

1. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

2. This exam is CLOSED BOOK. You may not use notes, articles, or books.

3. Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea doesn't work out; you can also get credit for realizing that certain approaches are incorrect. On a true/false question, you might get partial credit for explaining why you think something is true when it is actually false. But no partial credit can be given if you write nothing.

Comprehensive Exam: Numerical Analysis (30 points). Fall 1998

(Solution I)

(16 points). **Rootfinding for Nonlinear Equations**

(a) Define the *order of convergence* of a sequence $\{x_n | n \geq 0\}$ to a point α. When is the convergence said to be *linear*? If the convergence is linear, define the *rate of linear convergence.*

(b) Given a continuous function $f : \mathbb{R} \to \mathbb{R}$ and two points $a, b : f(a)f(b) < 0$ define the *bisection method,* in algorithmic form, to find a root α in $[a, b]$ satisfying $f(\alpha) = 0$.

Let $c_1 = [a + b]/2$ and let $\{c_n | n \geq 1\}$ be the sequence of approximations to α generated by the method. Show that, for some root α,

$$|\alpha - c_n| \leq \frac{|b - a|}{2^n}.$$

What can be said about the rate of linear convergence in this case?

(c) Given a twice continuously differentiable function $f : \mathbb{R} \to \mathbb{R}$ define Newton iteration to generate a sequence $\{x_n | n \geq 0\}$ to locate a root α satisfying $f(\alpha) = 0$.

By Taylor expansion show that

$$\alpha = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{(\alpha - x_n)^2}{2} \frac{f''(\xi_n)}{f'(x_n)}$$

for some $\xi_n \in \mathbb{R}$. Assume that $|f''(x)| \leq 2$ and $|f'(x)| \geq 1$ for all real x. Prove that, if $|x_0 - \alpha| < 1$, then x_n converges to α with order 2.

(Problem II)

(14 points). **Iterative Solution of Linear Equations**

This question concerns the solution of systems of linear equations in the form

$$Ax = b,$$

where A is an $m \times m$ matrix and x and b are vectors of length m.

(a) Describe the simplest form of *iterative improvement* (also known as *residual correction* or *iterative refinement*) to solve the linear system. Describe, and briefly explain, the effect of machine precision on this algorithm.

(b) Given a matrix C which approximates the inverse of A, consider the following general residual correction method for the solution of the linear system:

$$r^m = b - Ax^m,$$

$$x^{m+1} = x^m + Cr^m.$$

State the precise condition under which this iteration converges; prove your assertion.

(c) Write the matrix A in the form $A = L + D + U$ where L and U are (strictly) lower and upper triangular respectively and D is diagonal, define the Jacobi and Gauss-Siedel iterations for the solution of the linear system.

Solutions to Comprehensive: Numerical Analysis (30 points). Fall 1998

(Solution I)
(16 points). **Rootfinding for Nonlinear Equations**

(a) Define the *order of convergence* of a sequence $\{x_n | n \geq 0\}$ to a point α. When is the convergence said to be *linear*? If the convergence is linear, define the *rate of linear convergence.*

SOLUTION: A sequence of iterates $\{x_n | n \geq 0\}$ is said to converge with *order* $p \geq 1$ to a point α if

$$|\alpha - x_{n+1}| \leq C|\alpha - x_n|^p, \quad n \geq 0$$

for some $C > 0$. The convergence is *linear* if $p = 1$. The *rate of convergence* is then given by C.

(b) Given a continuous function $f : \mathbb{R} \to \mathbb{R}$ and two points $a, b : f(a)f(b) < 0$ define the *bisection method*, in algorithmic form, to find a root α in $[a, b]$ satisfying $f(\alpha) = 0$.

Let $c_1 = [a + b]/2$ and let $\{c_n | n \geq 1\}$ be the sequence of approximations to α generated by the method. Show that, for some root α,

$$|\alpha - c_n| \leq \frac{|b - a|}{2^n}.$$

What can be said about the rate of linear convergence in this case?

SOLUTION: The bisection algorithm can be written as $Bisection(a, b, f(\bullet), \epsilon)$:
- (1) Define $c := (a + b)/2$;
- (2) If $(b - c) \leq \epsilon$ then accept the root c;
- (3) If $f(b)f(c) \leq 0$ then $a := c$; otherwise let $b := c$;
- (4) Return to (1).

Let L_n be the length of the interval in which a root α is guaranteed to lie at the n^{th} step of the algorithm. Clearly $L_1 = |b - a|$ since c_1 is the mid-point of $[a, b]$ and we know that the root α lies in $[a, b]$. By construction it is clear that

$$L_{n+1} = \frac{1}{2}L_n.$$

Thus

$$L_n = \frac{|b - a|}{2^{n-1}}.$$

Since c_n is the midpoint of I_n and since α is guaranteed to lie in I_n we deduce that

$$|c_n - \alpha| \leq \frac{1}{2}L_n = \frac{|b - a|}{2^n}.$$

The order of convergence is hence *linear* in this case and the rate is at least $\frac{1}{2}$.

(c) Given a twice continuously differentiable function $f : \mathbb{R} \to \mathbb{R}$ define Newton iteration to generate a sequence $\{x_n | n \geq 0\}$ to locate a root α satisfying $f(\alpha) = 0$.

By Taylor expansion show that

$$\alpha = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{(\alpha - x_n)^2}{2}\frac{f''(\xi_n)}{f'(x_n)}$$

for some $\xi_n \in \mathbb{R}$. Assume that $|f''(x)| \leq 2$ and $|f'(x)| \geq 1$ for all real x. Prove that, if $|x_0 - \alpha| < 1$, then x_n converges to α with order 2.

SOLUTION: The Newton algorithm is:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Now

$$0 = f(\alpha) = f(x_n + \alpha - x_n) = f(x_n) + (\alpha - x_n)f'(x_n) + \frac{(\alpha - x_n)^2}{2} f''(\xi_n)$$

for some real ξ_n. Solving for α gives

$$\alpha = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{(\alpha - x_n)^2}{2} \frac{f''(\xi_n)}{f'(x_n)}.$$

Hence, substracting from the Newton iteration scheme we get

$$|x_{n+1} - \alpha| = \frac{|x_n - \alpha|^2}{2} \frac{|f''(\xi_n)|}{|f'(x_n)|}.$$

Now, we have that

$$|f''(x)| \leq 2, \quad \frac{1}{|f'(x)|} \leq 1$$

for all real x. Thus

$$|x_{n+1} - \alpha| \leq |x_n - \alpha|^2.$$

Thus

$$|x_n - \alpha| \leq |x_0 - \alpha|^{2^n}.$$

Quadratic convergence of x_n to α follows if $|x_0 - \alpha| < 1$.

(Problem II)

(14 points). **Iterative Solution of Linear Equations**

This question concerns the solution of systems of linear equations in the form

$$Ax = b,$$

where A is an $m \times m$ matrix and x and b are vectors of length m.

(a) Describe the simplest form of *iterative improvement* (also known as *residual correction* or *iterative refinement*) to solve the linear system. Describe, and briefly explain, the effect of machine precision on this algorithm.

SOLUTION:

Given an approximation x^m to the solution x of the linear system, the residual r^m is defined by

$$r^m = b - Ax^m.$$

Iterative improvement generates the sequence of approximations

$$x^{m+1} = x^m + \hat{e}^m$$

where $\hat{e}$ is the computed solution of

$$Ae^m = r^m.$$

For such an iteration it is important to obtain accurate values for r^m relative to the precision used in the remainder of the calculation. The reason is simply that otherwise the errors in $\hat{e}^m$ may be comporable with the errors in the original calculation.

(b) Given a matrix C which approximates the inverse of A, consider the following general residual correction method for the solution of the linear system:

$$r^m = b - Ax^m,$$

$$x^{m+1} = x^m + Cr^m.$$

State the precise condition under which this iteration converges; prove your assertion.

SOLUTION: *The iteration converges provided that the spectral radius of the matrix $I - CA$ is less than one.* To prove this note that the iteration gives

$$x^{m+1} - x = x^m - x + C[b - Ax^m] = x^m - x + C[Ax - Ax^m].$$

Hence the error $\delta^m = x^m - x$ satisfies

$$\delta^{m+1} = [I - CA]\delta^m.$$

It is well-known that a necessary and sufficient condition for the existence of a matrix norm in which B is less then 1 is for the spectral radius of B to be less than one. Using this norm gives

$$\|\delta^{m+1}\| \leq \zeta\|\delta^m\|$$

for some $\zeta \in (0,1)$. Iterating thia gives

$$\|\delta^m\| \leq \zeta^m\|\delta^0\|$$

and hence proves convergence.

(c) Write the matrix A in the form $A = L + D + U$ where L and U are (strictly) lower and upper triangular respectively and D is diagonal, define the Jacobi and Gauss-Siedel iterations for the solution of the linear system.

SOLUTION: We have

$$(L + D + U)x = b.$$

The *Jacobi iteration* is to generate x^m according to

$$Dx^{m+1} = b - [L + U]x^m.$$

The *Gauss-Siedel iteration* is to generate x^m according to

$$(L + D)x^{m+1} = b - Ux^m.$$

Computer Science Department

Stanford University

Comprehensive Examination in Software Systems

Fall 1998

READ THIS FIRST!!!

1. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.
2. The number of POINTS for each problem indicates how elaborate an answer is expected. For example, a question worth 6 points or less doesn't deserve an extremely detailed answer, even if you feel you could expound at length upon it. Short, bulleted answers are encouraged.
3. The total number of points is 30, although you have one hour in which to take the exam.
4. This exam is CLOSED BOOK. You may NOT use notes, books, computers, other people, etc.
5. Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea doesn't work out. You can also get credit for realizing that certain approaches are incorrect.
6. If you are convinced you need to make an assumption to answer a question, state your assumptions(s) as well as your answer.
7. Be sure to provide justification for your answers.

Fall 1998 Comprehensive Exam: Software Systems (30 points total)

1. (10 points) In UNIX the file directory structure (including hard and soft links) may be an acyclic graph. In MS-DOS it may only be a tree structure.
   a. Compared to a tree structure, what are the advantages and disadvantages of the acyclic graph directory structure?
   b. Why is a general graph directory structure not usually used for a file system?
2. (10 points) This question concerns page replacement.
   a. When is a process considered to be thrashing?
   b. What is the difference between global and local page replacement algorithms?
   c. What are the advantages and disadvantages of the two types of page replacement algorithms?
3. (10 points) Let's assume that a new type of main memory is invented that is cheap and usually very fast -- almost as fast as hardware registers. The only problem is that 0.1% of the time you access this memory, it is actually very slow (as slow as a disk access). The problem is that you have no way of predicting ahead of time when these slow accesses will occur.
   a. Would you incorporate this new type of memory into your system? Why or why not?
   b. Assuming you must incorporate it, would you make any changes to the virtual memory system or the file system or the network software or the process scheduler or the structure of the kernel?

Fall 1998 Comprehensive Exam: Software Systems (30 points total)
SOLUTIONS

1. (10 points) In UNIX the file directory structure (including hard and soft links) may be an acyclic graph. In MS-DOS it may only be a tree structure.

   a. Compared to a tree structure, what are the advantages and disadvantages of the acyclic graph directory structure?

Advantages: Files may be shared by more than one name in the directory structure (using links, they may have more than one name). This has many uses. For instance, one can maintain compatibility between old and new software that expects certain files to be in different locations.

Disadvantages: Traversing up the directory structure is more of a problem, since a file may have more than one parent if you include links.

A file may have more than one pathname, so when traversing the directory structure, you cannot rely on a new pathname indicating that it points to a file you have not already traversed.

Hard links require reference counting, which adds complexity.

Software links may be broken (left pointing at nothing).

You must have some way to ensure that cycles are not created in the directory structure. (UNIX does not allow users to create hard links to directories.)

   b. Why is a general graph directory structure not usually used for a file system?

The problem is cycles in the graph structure. If you remove certain elements, you may leave some files and directories "stranded" since their reference count will be zero, but there is no pathname by which to refer to them. In this case you must traverse the whole structure and do garbage collection.

2. (10 points) This question concerns page replacement.

   a. When is a process considered to be thrashing?

A process is considered to be thrashing when it is spending more time paging than executing.

   b. What is the difference between global and local page replacement algorithms?

In global replacement, the replaced page may come from the address space of a process other than the one that took the page fault. In local replacement, the page fault may only grab a page frame belonging to another page in the same process's address space.

   c. What are the advantages and disadvantages of the two types of page replacement algorithms?

With global replacement, overall system performance may be better, since a process that needs more pages can take page frames from a dormant process that isn't really using them. Thus the physical memory is more efficiently utilized. The disadvantage is that the paging behavior of one process can affect the performance of another process.

With local replacement, a badly faulting process cannot kick out the pages of another process, so its paging behavior will not hurt other processes. The disadvantage is that physical memory may not be as efficiently utilized.

3. (10 points) Let's assume that a new type of main memory is invented that is cheap and usually very fast -- almost as fast as hardware registers. The only problem is that 0.1% of the time you access this memory, it is actually very slow (as slow as a disk access). The problem is that you have no way of predicting ahead of time when these slow accesses will occur.

   a. Would you incorporate this new type of memory into your system? Why or why not?

I'll accept both answers here, if they include a convincing explanation. This new type of memory is tricky, though, since it causes more variability in system performance and behavior.

   b. Assuming you must incorporate it, would you make any changes to the virtual memory system or the file system or the network software or the process scheduler or the structure of the kernel?

Assuming the hardware will allow it, you will want to be able to put processes asleep if they access memory and it turns out to be one of the very slow accesses. (You could have a timeout whose expiration indicates it's a slow access.) This would affect the VM system and process scheduler. The network software may also be affected, since touching network buffers may trigger a slow access in the middle of, say, sending an ack. This means that timeouts in protocols may need to be extended. The kernel may need to be multi-threaded to allow some threads to sleep when they have triggered a slow access. File system cache hits no longer necessarily indicate that the file block sought will be speedily accessible. Thus you may need to allow processes accessing the file cache to sleep even on a cache hit. You may even want to make block/page sizes smaller so that there are fewer word accesses that could trigger a slow access if only a portion of the block/page is actually needed. (Overall, though, for the same amount of data actually copied or accessed, you'll suffer the same number of slow references.)