Computer Science Department
Stanford University
Comprehensive Examination in Artificial Intelligence
97/98

January 21, 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.

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. 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 *we* think it is actually false. But no partial credit can be given if you write nothing.

This exam contains 60 points. We have allocated points according to the number of minutes that we believe a student familiar with the material should take to do the questions. If you are somewhat less familiar with the material, a question may well take you longer than the number of points that it's worth. **DON'T FALL INTO THIS TRAP.** If you are taking too long on a question, write down whatever you have and move on. You may well get substantial partial credit for a partial answer, but you will not get any partial credit for a blank answer!

1. **KNOWLEDGE REPRESENTATION [16 points]**

   (a) **[3 points]** Translate each of the following sentences into first-order logic. Use the following vocabulary: the predicate symbols Workstation(x), Monitor(x), Component(x,y) (x is a component of y), Belongs(x,p) (x belongs to p), and At(x,y) (x is at location y); the function symbol Office-of(p); and the constant symbols Tom and Sun.[1]

   i. **[1 points]** All workstations come with a monitor.

   ii. **[1 points]** Any workstation belonging to a person is in that person's office.

   iii. **[1 points]** The workstation Sun belongs to Tom.

   (b) **[2 points]** Can you prove from these axioms that there is a monitor in Tom's office? If not add any (consistent) axioms necessary for proving this fact.

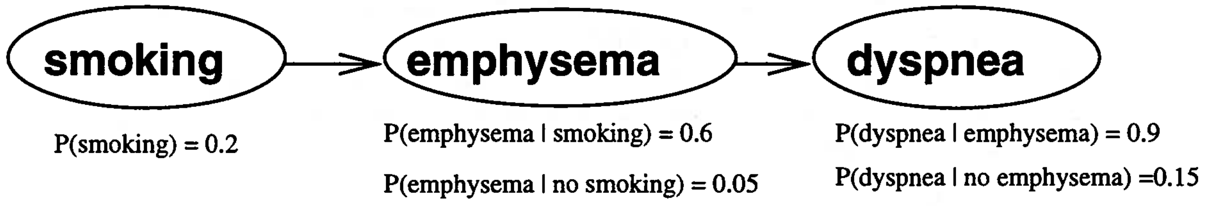
   (c) **[4 points]** Translate each of the above sentences, including the negated query, into clausal form. For brevity, you may use (here and below) the first initial of the various vocabulary symbols to represent them (e.g., you can use $C(x, y)$ instead of *Component(x,y)*). You may use either disjunctive normal form or implicational normal form.

   (d) **[7 points]** Using the clauses you have generated, use refutation resolution to prove that there exists a monitor in Tom's office.

   Answer format: number the clauses in your axioms, as well as any clauses generated during your proof; for each step in your proof, specify the numbers of the clauses resolved, any variable unifications needed, and the resulting clause. For brevity, you may resolve more than two clauses in one resolution step.

2. **PROBABILITY [8 points]**

   Consider the following Bayesian network (influence diagram) over three binary-valued variables:


smoking
emphysema
dyspnea
P(smoking) = 0.2
P(emphysema | smoking) = 0.6
P(emphysema | no smoking) = 0.05
P(dyspnea | emphysema) = 0.9
P(dyspnea | no emphysema) =0.15

   (a) **[2 points]** Name one conditional independence assumption which is encoded in the structure of this network.

   (b) **[6 points]** Show how you would compute *P(smoking | dyspnea)* in this network. For brevity, you may use the abbreviations s, e, and d for the events smoking, emphysema, and dyspnea, and the abbreviations $\neg s$, $\neg e$, and $\neg d$ for their negations. A formula for the answer is fine; you do not have to compute the final numerical answer.

[1]If we were being careful, you would also have a Person(x) predicate. However, for simplicity we'll ignore that.

3. **LEARNING [10 points]**

The marketing department of Microsquish Corporation is trying to construct a classifier which will tell them whether a customer will like a piece of software. From responses to survey forms, they have collected the following data:

Form #	customer is computer nerd	s/w has lots of features	customer liked s/w
1	yes	yes	yes
2	yes	yes	yes
3	yes	yes	yes
4	yes	yes	yes
5	yes	yes	yes
6	yes	yes	yes
7	yes	yes	yes
8	yes	no	no
9	yes	no	no
10	no	yes	yes
11	no	yes	no
12	no	yes	no
13	no	no	yes
14	no	no	yes
15	no	no	no

(a) **[2 points]** What Boolean function would do the best job of classifying these examples?

(b) **[5 points]** What decision-tree, including classifications, would be output by an ID3-style decision-tree learning algorithm? Explain or show your computations.

(c) **[3 points]** Is it possible to construct a neural network with a single thresholding element (i.e., a perceptron) which classifies these examples as well as the decision tree? If so, show the parameters of the thresholding unit. If not, explain why not.

4. **SEARCH [14 points]**

(a) **[9 points]** Consider the problem of heuristic search in a space where our heuristic function h' is almost, but not quite, admissible. More precisely, we have that for each node n, $h'(n) \leq h(n) + \epsilon$. In this case, we are not guaranteed that the first goal node returned by the A* algorithm will be the *optimal* goal node. Explain briefly how you could extend the A^* algorithm in order to find the optimal goal node n^*. (Hint: Consider the f-values of n and n^*.)

(b) **[5 points]** Consider the problem of search in the (familiar) blocks world domain. The domain consists of several square blocks of equal size on a (very large) table. Each block can be on the table or on top of exactly one other block. (Locations on the table are not labelled, so we don't care where on the table a block is.) A block can have at most one block on top of it. Operators in this space consist of taking one clear block (one which is not under any other block) and moving it to any other legal location (on top of a different clear block or to the table). Each application of an operator has cost 1.

Assuming your state space consists of complete blocks world configurations, and that the goal is a single fully specified state (e.g., a specific configuration of towers), define a nontrivial *admissible* heuristic function for this domain. Try to make your heuristic as powerful as possible while still making it admissible and relatively easy to compute. Explain why your heuristic is admissible.

5. **SHORT ANSWERS [12 points]** Each of the following questions requires at most one sentence in response. Do not write more.

   (a) **[1 points]** True or false: α-β pruning, although typically more efficient than minimax, can occasionally result in a less desirable move. (No explanation is required.)

   (b) **[3 points]** Consider the action of toggling a light switch, whose effect is to turn the light on if it's currently off, and to turn the light off if it's on. Can you provide a pure STRIPS description (as in Section 14.5 of Ginsberg's book) of the *toggle* action, assuming that the predicates in our language are *On(x)* and *Off(x)*? Either show the description, or explain (in one sentence) why one is impossible.

   (c) **[3 points]** R2D1 is an office cleaning robot which vacuums the floor as it moves around. The following is a STRIPS description of its *move(x,y)* action (where *at(x)* is true if the robot is at location x):

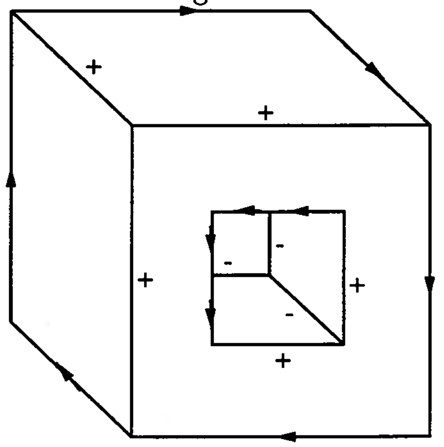
   Preconditions: *at(x)*, *adjacent(x,y)*.
   Add list: *at(y)*, *clean(x)*.
   Delete list: *at(x)*.[2]

   Is the following situation calculus axiom an equivalent description of the effects of this action (yes/no)?

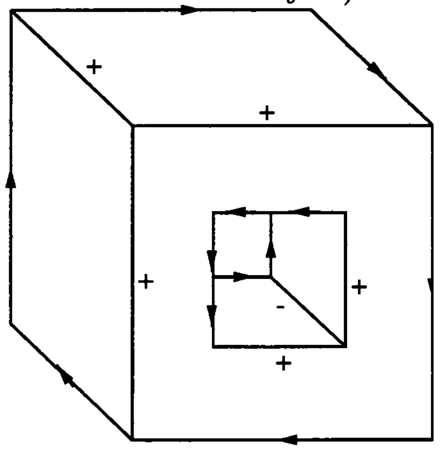
$$\forall s, x, y((at(x,s) \land adjacent(x,y)) \rightarrow \begin{array}{l}(at(y, result(move(x,y),s)) \land \\ clean(x, result(move(x,y),s)) \land \\ \neg at(x, result(move(x,y),s)))\end{array})$$

   If not, why not (one sentence)?

   (d) **[3 points]** The following figure contains two consistent Waltz labellings for the same line drawing. The first (a) is essentially the one given by Ginsberg, only slightly simplified. Is the second labelling (b) illegal (yes/no only)? If not, what differences in the physical object do the differences in the labellings represent (one sentence)? (Note: the differences are in the labelling of the construction at the center of the object.)



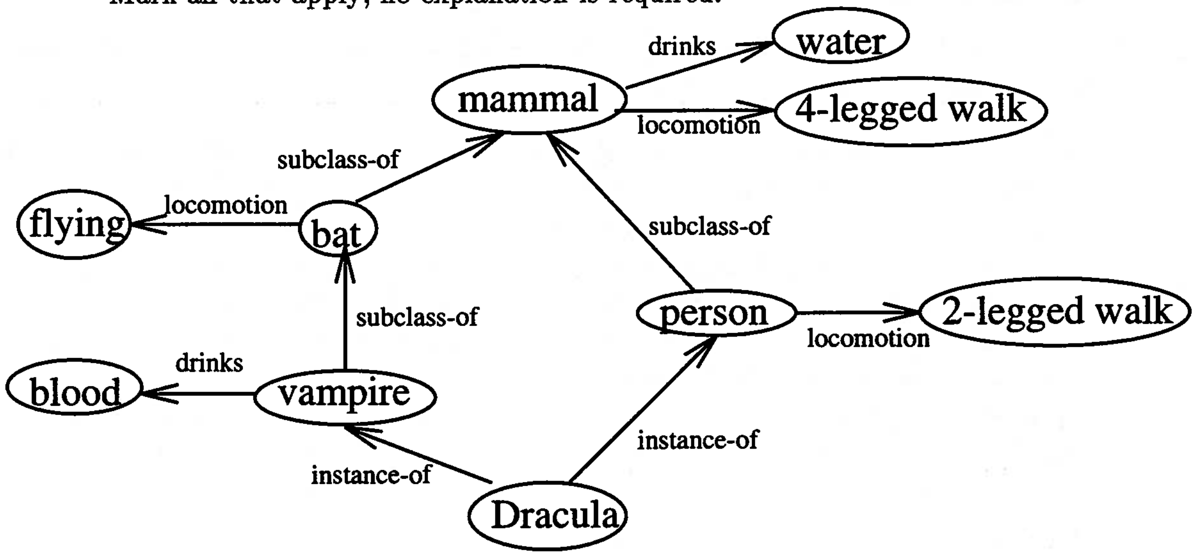
(a)



(b)

[2]In the notation of Russell & Norvig, the effects of this action are *at(y)*, *clean(x)*, *¬at(x)*.

(e) [**2 points**] In the following nonmonotonic semantic network, which of the following conclusions can we make (using brave extensions, as described in Ginbserg's book)? Mark all that apply; no explanation is required.



(a) Dracula's locomotion is by walking on two legs.
(b) Dracula's locomotion is by walking on four legs.
(c) Dracula's locomotion is by flying.
(d) Dracula drinks blood.
(e) Dracula drinks water.

AI COMP SOLUTIONS

1. KNOWLEDGE REPRESENTATION [16 points]

(a) **[3 points]** Translate each of the following sentences into first-order logic. Use the following vocabulary: the predicate symbols Workstation(x), Monitor(x), Component(x,y) (x is a component of y), Belongs(x,p) (x belongs to p), and At(x,y) (x is at location y); the function symbol Office-of(p); and the constant symbols Tom and Sun.[1]

i. **[1 points]** All workstations come with a monitor.
ii. **[1 points]** Any workstation belonging to a person is in that person's office.
iii. **[1 points]** The workstation Sun belongs to Tom.

i. $\forall x(Workstation(x) \rightarrow \exists y(Monitor(y) \wedge Component(y,x)))$
ii. $\forall x,p((Workstation(x) \wedge Belongs(x,p)) \rightarrow At(x, Office\text{-}of(p)))$
iii. $Workstation(Sun) \wedge Belongs(Sun, Tom)$

(b) **[2 points]** Can you prove from these axioms that there is a monitor in Tom's office? If not add any (consistent) axioms necessary for proving this fact.

No. We also need the following axiom:

$$\forall x,y,l((At(x,l) \wedge Component(y,x)) \rightarrow At(y,l))$$

(c) **[4 points]** Translate each of the above sentences, including the negated query, into clausal form. For brevity, you may use (here and below) the first initial of the various vocabulary symbols to represent them (e.g., you can use $C(x,y)$ instead of $Component(x,y)$). You may use either disjunctive normal form or implicational normal form.

The query is $\exists z(M(z) \wedge A(z,O(T)))$, so its negation is $forall z(\neg M(z) \vee \neg A(z,O(T)))$.

(1) $\neg W(x_1) \vee M(f(x_1))$
(2) $\neg W(x_2) \vee C(f(x_2),x_2)$
(3) $\neg W(x_3) \vee \neg B(x_3,p) \vee A(x_3,O(p))$
(4) $W(S)$
(5) $B(S,T)$
(6) $\neg A(x_4,l) \vee \neg C(y,x_4) \vee A(y,l)$
(7) $\neg M(z) \vee \neg A(z,O(T))$

(d) **[7 points]** Using the clauses you have generated, use refutation resolution to prove that there exists a monitor in Tom's office.

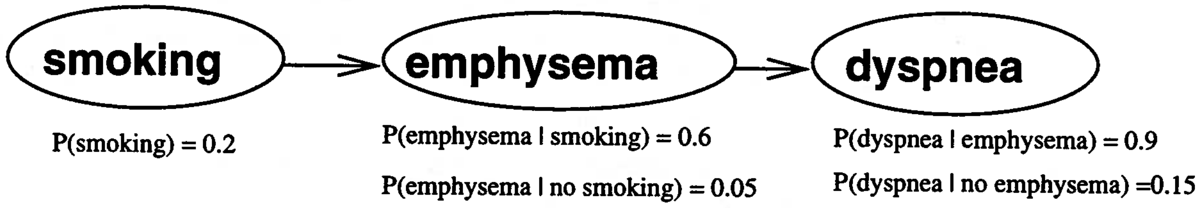
Answer format: number the clauses in your axioms, as well as any clauses generated during your proof; for each step in your proof, specify the numbers of the clauses resolved, any variable unifications needed, and the resulting clause. For brevity, you may resolve more than two clauses in one resolution step.

(9)	$A(S,O(T))$	(3),(4),(5)	$[x_3/S,\ p/T]$
(10)	$C(f(S),S)$	(2),(4)	$[x_2/S]$
(11)	$A(f(S),O(T))$	(6),(9),(10)	$[x_4/S,\ l/O(T),\ y/f(S)]$
(12)	$M(f(S))$	(1),(4)	$[x_1/S]$
(13)	false	(7),(11),(12)	$[z/f(S)]$

[1] If we were being careful, you would also have a Person(x) predicate. However, for simplicity we'll ignore that.

2. **PROBABILITY [8 points]**

Consider the following Bayesian network (influence diagram) over three binary-valued variables:


smoking
emphysema
dyspnea
P(smoking) = 0.2
P(emphysema I smoking) = 0.6
P(emphysema I no smoking) = 0.05
P(dyspnea I emphysema) = 0.9
P(dyspnea I no emphysema) =0.15

(a) **[2 points]** Name one conditional independence assumption which is encoded in the structure of this network.

Dyspnea is conditionally independent of smoking given emphysema.

(b) **[6 points]** Show how you would compute *P(smoking | dyspnea)* in this network. For brevity, you may use the abbreviations s, e, and d for the events smoking, emphysema, and dyspnea, and the abbreviations $\neg s$, $\neg e$, and $\neg d$ for their negations. A formula for the answer is fine; you do not have to compute the final numerical answer.

$$\begin{aligned}
P(d \mid s) &= P(d \mid e, s)P(e \mid s) + P(d \mid \neg e, s)P(\neg e \mid s) && (1)\\
&= P(d \mid e)P(e \mid s) + P(d \mid \neg e)P(\neg e \mid s) && (2)\\
&= 0.9 \cdot 0.6 + 0.15 \cdot 0.4 = 0.6 && (3)\\
P(d \mid \neg s) &= P(d \mid e)P(e \mid \neg s) + P(d \mid \neg e)P(\neg e \mid \neg s) && (4)\\
&= 0.9 \cdot 0.05 + 0.15 \cdot 0.95 = 0.1875 && (5)\\
P(d) &= P(d \mid s)P(s) + P(d \mid \neg s)P(\neg s) && (6)\\
&= 0.6 \cdot 0.2 + 0.1875 * 0.8 = 0.27 && (7)\\
P(s \mid d) &= \frac{P(d \mid s)P(s)}{P(d)} && (8)\\
&= \frac{0.6 \cdot 0.2}{0.27} = 0.0324 && (9)
\end{aligned}$$

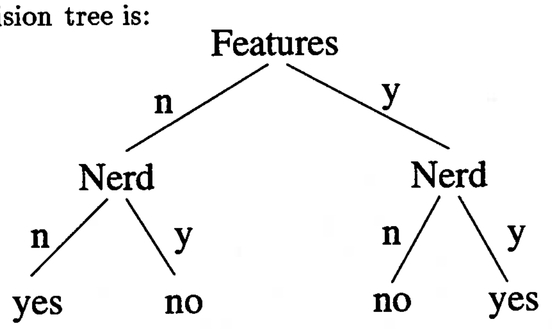
3. **LEARNING [10 points]**

The marketing department of Microsquish Corporation is trying to construct a classifier which will tell them whether a customer will like a piece of software. From responses to survey forms, they have collected the following data:

Form #	customer is computer nerd	s/w has lots of features	customer liked s/w
1	yes	yes	yes
2	yes	yes	yes
3	yes	yes	yes
4	yes	yes	yes
5	yes	yes	yes
6	yes	yes	yes
7	yes	yes	yes
8	yes	no	no
9	yes	no	no
10	no	yes	yes
11	no	yes	no
12	no	yes	no
13	no	no	yes
14	no	no	yes
15	no	no	no

(a) **[2 points] What Boolean function would do the best job of classifying these examples?**
The best Boolean function would be the negation of the exclusive or function, i.e., it would say "yes" when both variables are true or both variables are false, and "no" otherwise.

(b) **[5 points] What decision-tree, including classifications, would be output by an ID3-style decision-tree learning algorithm? Explain or show your computations.**
The optimal decision tree is:



The reason for splitting on the number of features first is as follows. If we split on the features (F), the distribution of positive and negative examples is 2:3 on the "no" branch and 8:2 on the "yes" branch. If we split on nerdiness (N), the distribution is 3:3 on the "no" branch and 7:2 on the "yes" branch. Even without going through the math, it's clear that the information gain is higher if we split on F. Formally, the entropy along each branch for the F split is lower than the entropy along the corresponding branch for the N split, and therefore the weighted average entropy is also lower, so that the information gain is higher.

(c) **[3 points] Is it possible to construct a neural network with a single thresholding element (i.e., a perceptron) which classifies these examples as well as the decision tree? If so, show the parameters of the thresholding unit. If not, explain why not.**
A single perceptron is limited to describing functions that are linearly separable. Since the NXOR function is not linearly separable, we cannot represent it using a perceptron, so we cannot achieve the optimal accuracy on this set of examples.

4. **SEARCH [14 points]**

(a) **[9 points]** Consider the problem of heuristic search in a space where our heuristic function h' is almost, but not quite, admissible. More precisely, we have that for each node n, $h'(n) \leq h(n) + \epsilon$. In this case, we are not guaranteed that the first goal node returned by the A* algorithm will be the *optimal* goal node. Explain briefly how you could extend the A^* algorithm in order to find the optimal goal node n^*. (Hint: Consider the f-value of n^*.)

We know from the assumptions and from the definitions that:

$$\begin{aligned} f(n^*) &= g(n^*) + h'(n^*) && (10)\\ h'(n^*) &\leq h(n^*) + \epsilon && (11)\\ h(n^*) &= 0 && (12) \end{aligned}$$

Therefore, we have that

$$f(n^*) \leq g(n^*) + \epsilon. \qquad (13)$$

Now, consider any other goal node n'. Since n^* is a better goal node than n', we also have that

$$g(n^*) \leq g(n') \qquad (14)$$

Putting (13) and (14) together, we get that

$$f(n^*) \leq g(n') + \epsilon \qquad (15)$$

Based on this equation, we can extend A* as follows:

- Run A* until the first goal node n is found.
- Compute $c = g(n)$.
- Continue expanding nodes in A* order until all nodes of f-cost at most $c + \epsilon$ have been expanded.
- Output the lowest cost goal node found.

In fact, since (15) holds for any goal node found, we can do even better by reducing our bound on f if a cheaper goal node is found after the first goal node. I.e., we have c be the cost of the cheapest goal node found so far, and stop the algorithm as soon as all unexpanded nodes have f-value greater than $c + \epsilon$. Under this modification the algorithm is guaranteed to expand only nodes whose f-value is $\leq c^* + \epsilon$, where c^* is the (true) cost of the optimal goal node. Intuitively, this performance is the best we can expect given our guarantees on the heuristic function.

(b) **[5 points]** Consider the problem of search in the (familiar) blocks world domain. The domain consists of several square blocks of equal size on a (very large) table. Each block can be on the table or on top of exactly one other block. (Locations on the table are not labelled, so we don't care where on the table a block is.) A block can have at most one block on top of it. Operators in this space consist of taking one clear block (one which is not under any other block) and moving it to any other legal location (on top of a different clear block or to the table). Each application of an operator has cost 1.

Assuming your state space consists of complete blocks world configurations, and that the goal is a single fully specified state (e.g., a specific configuration of towers), define a nontrivial *admissible* heuristic function for this domain. Try to make your heuristic as powerful as possible while still making it admissible and relatively easy to compute. Explain why your heuristic is admissible.

One simple and nontrivial heuristic function is the one that counts the number of blocks in the current state that are not in the desired location according to the goal state. It's admissible because we need at least one move to get each block into the right place, and therefore this heuristic underestimate the required number of moves.

5. **SHORT ANSWERS [12 points]** Each of the following questions requires at most one sentence in response. Do not write more.

   (a) **[1 points]** True or false: α-β pruning, although typically more efficient than minimax, can occasionally result in a less desirable move. (No explanation is required.)

   False; *alpha*-β pruning only eliminates nodes that are clearly suboptimal, and therefore it cannot result in a less desirable move.

   (b) **[3 points]** Consider the action of toggling a light switch, whose effect is to turn the light on if it's currently off, and to turn the light off if it's on. Can you provide a pure STRIPS description (as in Section 14.5 of Ginsberg's book) of the *toggle* action, assuming that the predicates in our language are *On(x)* and *Off(x)*? Either show the description, or explain (in one sentence) why one is impossible.

   No. Standard STRIPS does not allow the conditional effects, i.e., where the effects depend on the preconditions.

   (c) **[3 points]** R2D1 is an office cleaning robot which vacuums the floor as it moves around. The following is a STRIPS description of its *move(x,y)* action (where *at(x)* is true if the robot is at location x):

   Preconditions: *at(x), adjacent(x,y).*

   Add list: *at(y), clean(x).*

   Delete list: *at(x).*[2]

   Is the following situation calculus axiom an equivalent description of the effects of this action (yes/no)?

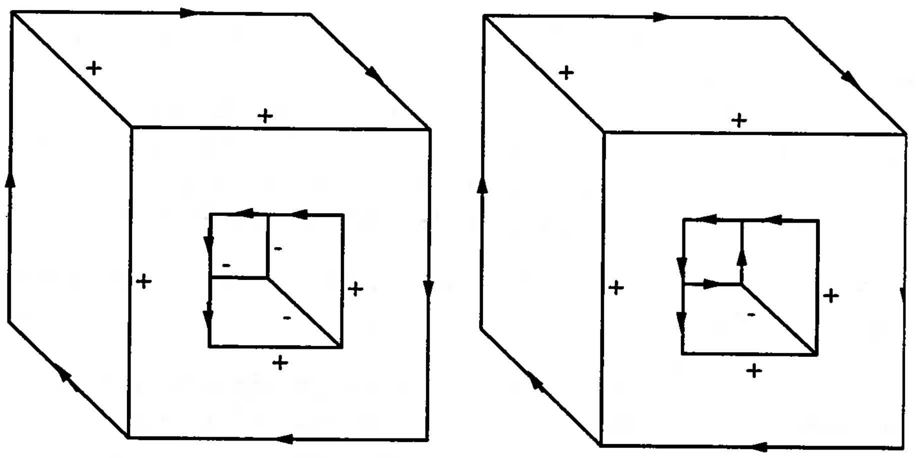
$$\forall s,x,y((at(x,s) \wedge adjacent(x,y)) \rightarrow \begin{array}{l}(at(y, result(move(x,y),s)) \wedge \\ clean(x, result(move(x,y),s)) \wedge \\ \neg at(x, result(move(x,y),s)))\end{array}$$

   If not, why not (one sentence)?

   No. In order to capture the STRIPS description completely, we must also specify frame axioms, i.e., the axiom must also state the properties of the world that *do not* change (e.g., the cleanliness of locations other than x).

   (d) **[3 points]** The following figure contains two consistent Waltz labellings for the same line drawing. The first (a) is essentially the one given by Ginsberg, only slightly simplified. Is the second labelling (b) illegal (yes/no only)? If not, what differences in the physical object do the differences in the labellings represent (one sentence)? (Note: the differences are in the labelling of the construction at the center of the object.)

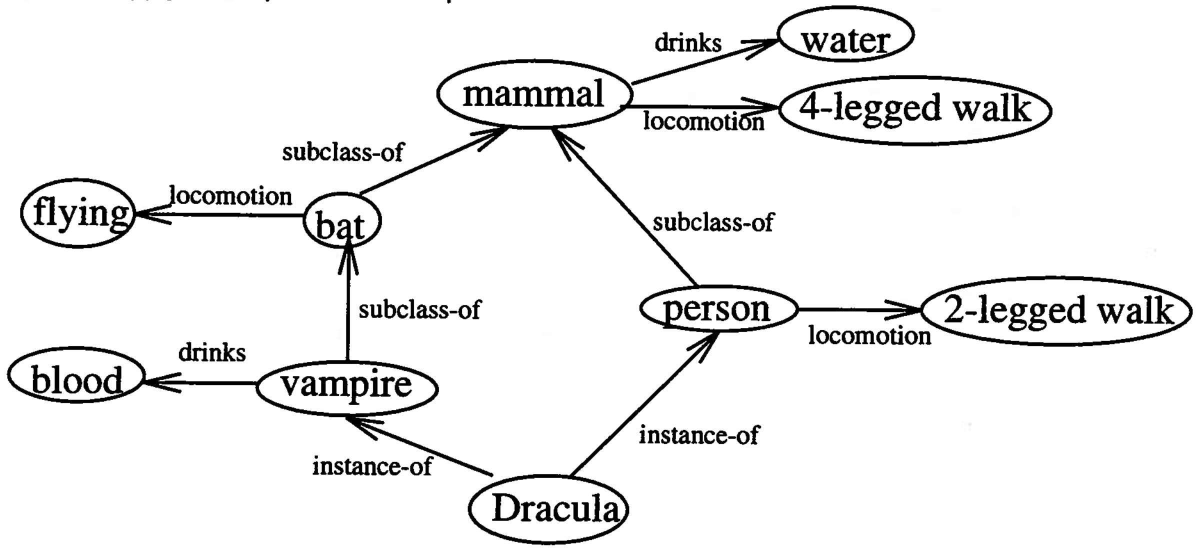
[2] In the notation of Russell & Norvig, the effects of this action are *at(y), clean(x), $\neg$at(x)*.



(a) (b)

Both figures describe to legal physical objects. (a) corresponds to a cube with a hole, where the hole only goes partway through the cube. (b) corresponds to a cube where the hold goes all the way through to the back.

(e) **[2 points]** In the following nonmonotonic semantic network, which of the following conclusions can we make (using brave extensions, as described in Ginbserg's book)? Mark all that apply; no explanation is required.



(a) Dracula's locomotion is by walking on two legs.
(b) Dracula's locomotion is by walking on four legs.
(c) Dracula's locomotion is by flying.
(d) Dracula drinks blood.
(e) Dracula drinks water.

We can reach all of these conclusions except for (b). The reason (b) is blocked is because, along any path, there is a more specific default for locomotion that takes precedence: Along the "vampire" path, locomotion by flying takes precedence, and along the "person" path, locomotion by two-legged walking takes precedence. By contrast, the drinking water default is blocked along the "vampire" path, but can be reached along the "person" path.

Automata and Formal Languages (30 points)

Winter 1997/98

Instructions.

1. Write your answers in a blue book. Put your magic number on the cover of every blue book you use.

2. Half an hour is allowed. The questions are worth a total of 30 points.

3. The exam is open book. Notes, articles, and books may be consulted but not people or computers.

4. Show all work. Incomplete answers may receive partial credit.

Automata and Formal Languages (30 points)—Winter 1997/98

1. [10] Prove that $\{a^{\lceil \log_x n \rceil} \mid n \geq 1\}$ is regular, where the base x of the logarithm is real and satisfies $x > 1$. (Here n is integer; when n is real you may find useful that the derivative of $\log_x n$ is $\frac{1}{n \ln x}$.)

2. [10] Assuming that nondeterministic pushdown automata accept exactly the context-free languages, show that nondeterministic *one-state* pushdown automata also accept exactly the context-free languages, provided they accept by empty stack.

3. [10] In this question all languages are over a fixed alphabet Σ. Define a language L to be complete for P (the class of polynomial-time recognizable languages) when every L' in P is polynomial-time reducible to L.

(i) Show that neither the empty language nor Σ^* is complete for P.

(ii) Identify all the other languages in P that are not complete for P.

Automata and Formal Languages (30 points)
Winter 1997/98
SOLUTIONS

1. [10] Prove that $\{a^{\lceil \log_x n \rceil} \mid n \geq 1\}$ is regular, where the base x of the logarithm is real and satisfies $x > 1$. (Here a is a letter of an alphabet and n is integer. When n is real you may find useful that the derivative of $\log_x n$ is $\frac{1}{n \ln x}$.)

Solution. For $n \geq \frac{1}{\ln x}$, $\log_x(n+1) - \log_x n \leq 1$ (consider the slope at n). Hence the language can only omit a^i for $i < \frac{1}{\ln x}$, and the complement of a finite set is regular.

2. [10] Assuming that nondeterministic pushdown automata accept exactly the context-free languages, show that nondeterministic *one-state* pushdown automata also accept exactly the context-free languages, provided they accept by empty stack.

Solution. One-state PDA's accept *at most* the context-free languages because they are a special case of PDA's. The usual proof that every context-free language is accepted by some pushdown automaton constructs a one-state pushdown automaton that accepts by empty stack, whence automata of this kind accept *all* context-free languages.

3. [10] In this question all languages are over a fixed alphabet Σ. Define a language L to be complete for P (the class of polynomial-time recognizable languages) when every L' in P is polynomial-time reducible to L.

(i) Show that neither the empty language nor Σ^* is complete for P.

Solution. There can be no function (polynomial-time computable or not) on Σ^* which sends every member of Σ^* to a member of $\emptyset$ because there is no member of the latter to which to send for example the empty string. The same holds when the two languages are interchanged and "member" is replaced by "nonmember."

(ii) Identify all the other languages in P that are not complete for P.

Solution. The languages in (i) are the only such. Let $u \in L$ and $v \notin L$. Given L' in P, the function $f : \Sigma^* \to \Sigma^*$ sending members of L' to u and nonmembers to v is computable in polynomial time because membership in L' is, and $w \in \Sigma^*$ is in L' if and only if $f(w)$ is in L. Hence L is complete for P.

(TEST) BUT NO solutions)

Computer Science Comprehensive Examination
Computer Architecture
[60 points]

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

A. Increasing the block size of a cache while holding total size and associativity constant will improve performance for programs that exhibit what type of locality? # of blocks fewer

spatial

B. List one advantage and one disadvantage of variable-length instruction sets.

C. Dynamic scheduling using register renaming allows instructions to run out-of-order without concern for which type or types of hazards? WAW - output
WAR - anti-dependencies

D. Suppose you run a program that has 50% vector operations and 50% scalar operations on a machine with 10MFLOPS scalar performance and 1GFLOPS vector performance. To first approximation what will the performance of your program be?

E. Does a pipelined processor with a single-cycle ALU and bypassing need interlocks to stall the pipeline on read-before-write hazards on arithmetic operations? Why?

F. Which will have better performance on conditional code (code with many 'IF' statements), a SIMD machine or a MIMD machine?

Problem 2: Cache Architecture [3 points each, 12 points total]

A. Suppose you have a 4Kbyte 2-way associative cache with a block size of 32 bytes. What size tag array (directory) is needed for this cache?

B. Suppose you take the cache from part (A) and divide each block into two 16-byte half-blocks, each with their own valid bit. (This is called a sectored cache). With the new organization, only a half-block is fetched on each cache miss but replacements are done a whole block at a time. What affect does this change have on miss rate? on miss penalty?

C. For what types of programs would the sectored cache of (B) outperform the cache of (A)?

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

Problem 3: Pipeline Microarchitecture [12 points]

A. [5 points] Consider the five-stage pipeline shown on the last page. Suppose it is intended to execute a fixed-format 32-bit instruction set with a 5-bit opcode, 3 5-bit register specifiers, and a 12-bit signed immediate. All data and address paths are 32-bits. Draw the missing lines connecting the boxes and label each with its width in bits. Insert any multiplexers that are needed for correct operation without bypassing. Do not add bypass multiplexers.

B. [5 points] Consider the instruction sequence

```
Load  r2 <- [r1 + 4]
Add   r4 <- r2 + r3
Store [r1 + 8] <- r4
```

How many cycles does this take to execute from the time the IP points to the load instruction until the store has committed? Explain your answer. You may want to draw a timeline.

C. [2 points] Suppose the `Store` instruction in the sequence above raises an exception in the fetch stage (perhaps a protection violation in accessing the I-cache). Explain briefly (3 lines or less) how your pipeline must handle this exception to ensure 'precise exceptions'.

Problem 4: Virtual Memory [12 points]

A. [2 points] Consider two machines. X supports just segmented memory, and Y supports just paged memory. List one advantage in favor of each machine.

B. [5 points] Virtual addresses are byte addresses and are 32-bits in length. Y's pages are 4K-bytes in length and are referenced through a direct-mapped translation-lookaside buffer (TLB) with 8 entries. Sketch a block diagram showing the translation process and label and dimension the fields of the virtual address and physical address.

C. [5 points] X supports arbitrary sized segments from 4-bytes up to 2^{32} bytes in length (in increments of 4-bytes) and can support up to 2^{30} 4-byte segments. X uses a fully associative TLB with address masking to perform the translation. Sketch a possible format for an entry from this TLB labeling and dimensioning each field.

Problem 5: Performance [12 points]

Suppose you have a machine with the following instruction mix:

ALU	60%
LOAD	20%
STORE	10%
BRANCH	10%

All ALU instructions complete in one cycle. Branch instructions complete in one cycle if correctly predicted but take 5 cycles if mispredicted. The machine has blocking LOAD and STORE instructions. That is, subsequent instructions are stalled until each access completes.

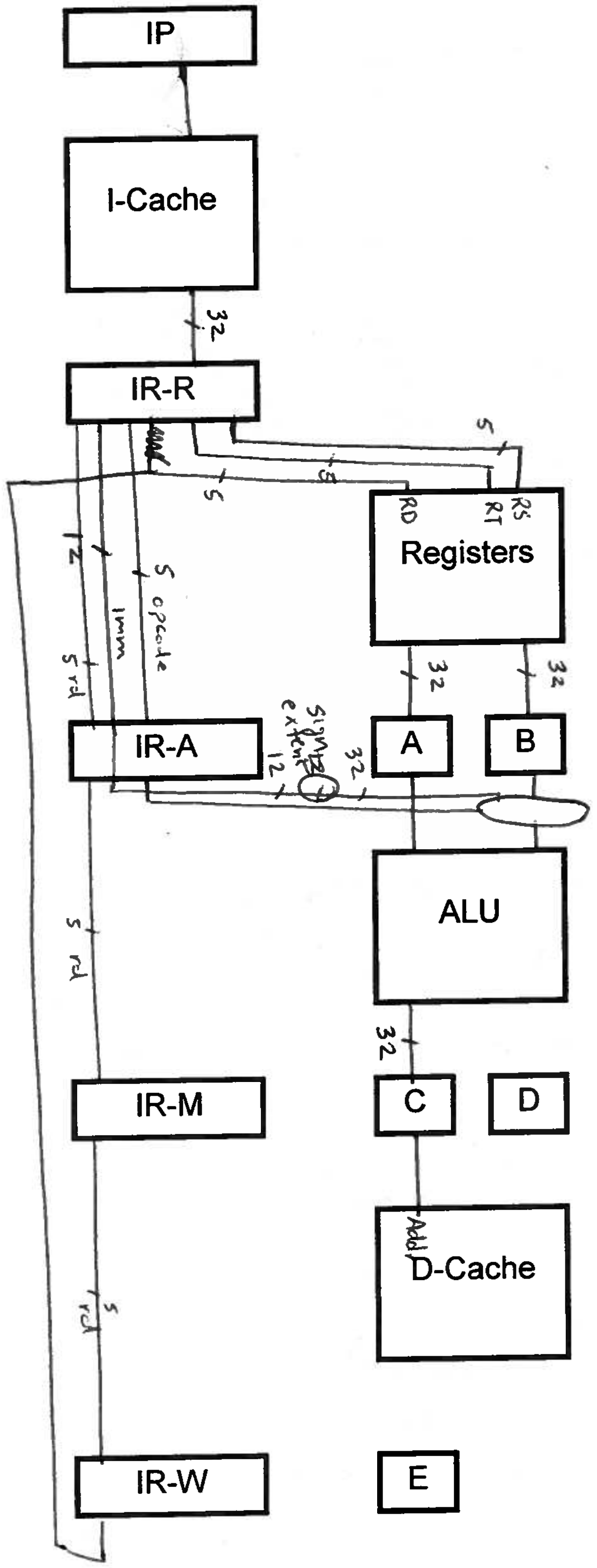
A. [2 points] If you had an ideal memory system (all accesses complete in one cycle) and the branch predictor is 80% accurate, what will the CPI of this machine be?

B. [10 points] Unfortunately ideal memory systems cannot be built. You are considering three alternatives for a memory system with parameters as shown below.

	L1		Main Mem	
Option	Miss	Access	Access	
A	10%	1	20	
B	5%	3	20	
C	10%	2	20	Non blocking

The first option (A) is to use a small cache that can be accessed in one cycle but has a high miss rate (10%). The second option is to use a larger cache that has a smaller miss rate but has a 3-cycle access time on a hit. The third option is to use the small cache but to make memory instructions non-blocking. That is, the pipeline is never stopped by a store (we'll assume an infinite write buffer for now) and is stopped by a load only when the result is needed. The complexity of the non-blocking memory hardware adds a cycle to the hit latency. Assume that on average execution can continue for 5 instructions before the load value is needed.

What is the average memory access time for each approach? What is the CPI for each approach? Which would you choose?


IP
I-Cache
32
IR-R
5
5
5
RD
RT
RS
Registers
12
Imm
5 opcode
5 rd
32
32
IR-A
A
B
sign extend
12
32
ALU
5 rd
32
IR-M
C
D
Add
D-Cache
5 rd
IR-W
E

Answers:

1997 Database Comprehensive Exam
30 minutes, January 19 1998

1. Design (7 minutes)
List the normalized (3NF) relations needed to represent the data corresponding to this E-R model.

Underline the key-fields.

Patients (idno, name, address)

Personnel (pidno, pname, paddress, type, works-at[Clinic])

/ works-at: Clinic has multiple personnel */*

Clinic (title, location)

/ has: Patient can have multiple diagnoses */*

/ is-of: Disease-class can have multiple Diagnoses */*

Diagnosis (idno[Patient], ICD, ~~idno~~, date, severity, is-of[Diseaseclass])

Diseaseclass (code, system)

sees (idno[Patient], pidno[Personnel]) / ER m-n relationship set requires distinct relation */*

certified (pidno[Personnel], code[Diseaseclass]) / ER m-n relationship set */*

serves (title[Clinic], code[Diseaseclass]) / ER m-n relationship set */*

Information in [ ] is redundant, since all attributes have unique names.

2. Design revision (5 minutes)
The committee decides on a refinement: for three values of type: { M.D., nurse, clerk },

distinct data are needed:

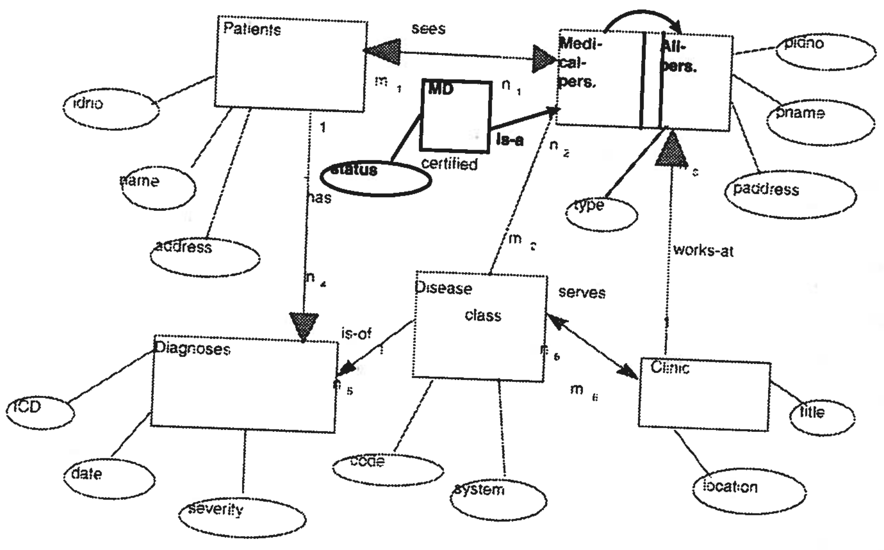
a. for the M.D.: Status {intern, resident, staff, community, consultant}
add MD ((pidno[(Medical-)Personnel], status)

b. for the clerk, no certification or patient seen data is needed.
split Personnel into two:
Medical-personnel(pidno[All-Personnel])
All-personnel(pidno, pname, paddress, type, works-at[Clinic]))
and change the association relations to
sees (idno[Patient], pidno[Medical-personnel])
certified (pidno[Medical-personnel], code[Diseaseclass])

An alternative is to create 3 subrelations of Personnel, one each for MD, nurse, and clerk, but then there will also be two `sees' and two `certified' relations, which have to managed distinctly.

Letting the certified and sees field for a clerk be NULL is a lazy (and common) solution, but will give problems when joining along these links, as in question 4.

Sketch the diagram revision (only the changes) and list the new relations needed.



3. SQL (5 minutes)
In the **paddress** text field appears a town name that can match the **location** of a **Clinic**.

Write the SQL query that lists **Personnel pnames** living in the same town as a **Clinic title**.

SELECT pname FROM Clinic, Personnel WHERE paddress CONTAINS location.
If the format of paddress is structured with town name in front then
SELECT pname FROM Clinic, Personnel WHERE location LIKE paddress
should work,

Note: the query was for all personnel, not just for WHERE works-at= title

4. Relational Algebra (8 minutes)
Write relational algebra statement that create a relation listing

Patients name, Diagnoses, and Personnel pname , where the personnel was NOT certified for the **Disease-class** for the **Diagnoses.**

Create two relations with matching attributes,
one with all Personnel seen and the Disease class for the Diagnoses, where the Disease code = is-of,
and one with the Personnel seen that is certified for the Disease code, then take the difference,
and project the code out of the final result. The Disease-class.code must be included in both intermediate relations to avoid subtracting out an uncertified personnel member for some code, who also seen the patient for a good certified code. (That would in fact the most common problem in practice)

Π name, ICD, pname (

Π name, ICD, pname, code (* $\bowtie$

Personnel {pidno |X| pidno}sees{idno |X| pidno}Patients{has |X| ICD}Diagnoses) -

Π name, ICD, pname, code ←is-of (*

Patient {idno |X| idno} sees {pidno |X| pidno}Personnel {pidno |X| pidno} certified))

*The indicated {join attributes} are optional. The * projections can also be omitted at a cost.*

---

5. Performance (5 minutes)
   A database optimization heuristic is to perform selections and projections prior to join operations.

What is the reason for this heuristic?
The heuristic is to reduce the sizes of the inputs to a join, since a join is O(n x m). Selection reduces the number of rows and projection the number of columns.

When is it invalid?

Some joins, as joins among sorted relations O(m+n) are linear, and even other joins, as hash-joins with small sizes are quite fast O(n(1+d)), while projection is typical O(n log n) and selection without an index is O(n).
Don't project out attributes used in subsequent join or selects, consider if doing two projections is worth it.
<< Don't project out potentially useful indexes. >>

---

6. Culture (1 minute)
   Who defined the relational model and when?
   E.F. *(Ted) Codd, of IBM San Jose Research, the prime paper appeared in Comm.ACM, 1970. There were some earlier IBM reports and later elaborations in ACM SIGFDT (predecessor of SIGMOD) Proceedings by Codd.*

   What were the prior alternatives in designing databases?
   Hierarchical, following systems specifications as IBM's IMS or others.
   Network, following the CODASYL model.
   The E-R model came later (Chen in ACM TODS 1975).

NUMERICAL ANALYSIS

1997-1998

Comprehensive Examination

Closed Book

1.(15 points) What is the condition number of a nonsingular matrix A? If we want to solve Ax = b and indeed solve Ax' = b' where b' is a perturbation of b, show how the condition number relates the relative error of the solution to the relative error in the data b.

2.(15 points) Consider the approximation

$$f'(x) = (f(x+h) - f(x-h))/2h + O(h^2)$$

and generate an $O(h^4)$ accurate approximation to f'(x) using Richardson extrapolation assuming f is as differentiable as you need.

TEST, BUT
NO solutions Available

1997-1998

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

1. (15 points) Some possible criteria for a process scheduler are the following:

   1. Fairness: make sure each process gets its fair share of the CPU.

   2. Efficiency: keep the CPU busy with useful work as much as possible.

   3. Response time: minimize response time for interactive users.

   4. Turnaround: minimize the time batch users must wait for output.

   5. Throughput: maximize the number of jobs processed per hour.

   Please answer the following questions given these criteria.

   a. Which of these criteria are contradictory with each other, and why?

   b. In a system with only small interactive processes, all of which are of equal priority, what would be a good scheduling algorithm? Why? (Pick as simple an algorithm as you think would be suitable.)

   c. Let's say we add processing and display of a real-time video feed to this system otherwise dominated by small interactive processes. What kinds of new scheduling problems or issues arise? (You do not need to come up with a scheduling algorithm to solve them.)

   d. Consider a system with more than just small interactive processes. Besides the interactivity, priority and real-time nature of a process, what other characteristics of a process might you take into account in the scheduler?

   e. What technology trends would have an effect on the importance of the characteristics you listed in question d?

2. (5 points) What is a race condition? Give an explicit example.

3. (10 points) Upon rebooting after a crash, a boot program checks for the consistency of the file system. Let's assume the system uses a standard UNIX file system. The program first checks for file block consistency. To do so, it builds a table with two counters per block, both initially 0. The first counter keeps track of how many times the block is present in a file; the second records how often it is present in the free list (or bit map of free blocks). The program then reads all the i-nodes. Starting from an i-node, it is possible to build a list of all the block numbers used in the corresponding file. As each block number is read, its counter in the first table is incremented. The program then examines the free list or bit map, to find all the blocks that are not in use. Each occurrence of a block in the free list results in its counter in the second table being incremented. (The system also checks for consistency of the directory structure, but you can ignore that.)

   a. What are all the types of inconsistencies that could be found, and how does the program detect them from these two tables? (One of these inconsistencies is impossible if the system uses a bit map rather than free list.)

   b. For each of these inconsistencies, how would the program fix the problem?

Software Systems Answers 1/98 Process scheduling, race, block consistency

1.a. There are lots of answers here. A few:

Minimizing response time and turnaround time are contradictory. The first favors small interactive processes while the second favors the batch processes.

Maximizing fairness and throughput are contradictory. Fairness means giving everyone some "fair" chunk of the cpu in a timely fashion, while maximizing throughput would favor short batch jobs over the others.

Maximizing efficiency and response time can be contradictory. Minimizing response time can mean context switching more often, and the CPU isn't during "useful" work while context switching.

1.b. Round robin, First Come First Server, and FIFO are all reasonable, since the processes are all of equal priority. ~~As a process returns from waiting on I/O, it gets put at the end~~ of the run queue. Time slices can be small, although the interactive processes are likely to block before the end of their time slice in any case.

1.c. The video feed needs continual, frequent attention from the CPU, or the video will be jerky or you'll have to drop frames to keep up, etc. Giving the video process higher priority may help, but that can hurt the performance of the small interactive processes. Some guarantee of some amount of CPU per unit of time would be good for the video.

1.d. You may want to consider process size. If memory is in high demand, then swapping out a large process may make room for several smaller processes to run. If you bring back in the large process, this takes time to bring its pages in, so you'll want to run it for longer to amortize this cost. (Fewer but longer time slices.)

1.e. Cheaper memory has meant more memory on computers, which makes it easier to avoid swapping and paging problems. There are still a few really huge processes, depending on your workload, that may need to be swapped or paged, but most processes are more likely to avoid this problem.

2. If more than one process or thread accesses and modifies a shared resource (variable, data structure, file, etc.) in a non-atomic way, then the interleaving of these processes' actions can leave data in an inconsistent state.

For example, if two processes are allowed to add blocks to a file at the same time, we could end up with a corrupted file.

As another example, consider the following code:

```
P1                                P2
---                               ---
R := x;                           R := x;
R ++;                             R ++;
x := R;                           x := R;
```

If the three lines of code are protected by a lock, then if both processes run, in either order, x will be incremented twice (the desired outcome).

But if the code sections are not protected, but interleaved, then the processes can interfere with each other, and the effect of one's increment can be lost. For example, assuming x = 1, P1 executes line 1 and line 2 and is then descheduled. P2 executes all three statements, incrementing x to 2. Then P1 continues running and also sets x to 2, since it is still using the old initial value of x.

3.a. The block can be neither in a file nor on the free list (both counters = 0)

The block may be in more than one file (file counter >= 1)

The block can be both in a file or files and on the free list (both counters >= 1)

The block may be on the free list more than one time (free list counter >= 1) Note that this can't happen if the free list is implemented as a bitmap.

3.b. To fix the first problem, assign the block to the free list since it's not in use.

To fix the second problem, assuming the block appears in N files, make N-1 copies of it in previously free blocks and assign these blocks to those N-1 files in place of the multiply referenced block.

To fix the third problem, remove the block from the free list. If the block also appears in multiple files, handle it as per the second problem.

To fix the fourth problem, remove the extra block references from the free list leaving just one reference to the block.