Computer Science Department
Stanford University
Comprehensive Examination in Artificial Intelligence
Autumn 1995

October 17, 1995

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 2 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.

1. Frame Representations [18 points]

Consider the knowledge base below represented in a frame language. The frame language represents monotonic knowledge by describing classes, members of classes, subclass relations, slot values that apply to all members of a class, and value restrictions on slots that constrain values of the slot to be members of a given class.

```
CompanyVehicle                          ; The class of vehicles owned by
    SubclassOf: Vehicle                 ; a corporation.
    Owner
        Value_Restriction: Corporation

CompanyCar                              ; The class of company vehicles
    SubclassOf: CompanyVehicle          ; that are sedans.
    Style: Sedan

Envig's_Car                             ; Envig's car.
    MemberOf: CompanyCar
    Owner: Envig
```

(a) What is the style of Envig's car? [2 points]

(b) What class is Envig a member of? [2 points]

(c) Represent in predicate logic the information in these frames using the following relations: [4 points]

```
(Subclass <class1> <class2>)        ; <class1> is a subclass of <class2>.
(Member <object> <class>)           ; <object> is a member of <class>.
(MemberValue <value> <slot> <class>)
        ; <value> is a value of <slot> in <class>.
(OwnValue <value> <slot> <object>)
        ; <value> is a value of <slot> in <object>.
(ValueRestriction <class1> <slot> <class2>)
        ; <class1> is a value restriction for <slot> in <class2>.
```

(d) State any additional axioms describing properties of the relations listed above that would be needed in order to derive the answers to questions (a) and (b). [10 points]

2. Adversary Search [22 points]

Suppose that you are using alpha-beta pruning to determine the value of a move in a game tree, and that you have decided that a particular node n and its children can be pruned. If the game tree is in fact a graph and there is another path to the node n, are you still justified in pruning this node? Either prove that you are or construct an example that shows that you cannot.

(7)

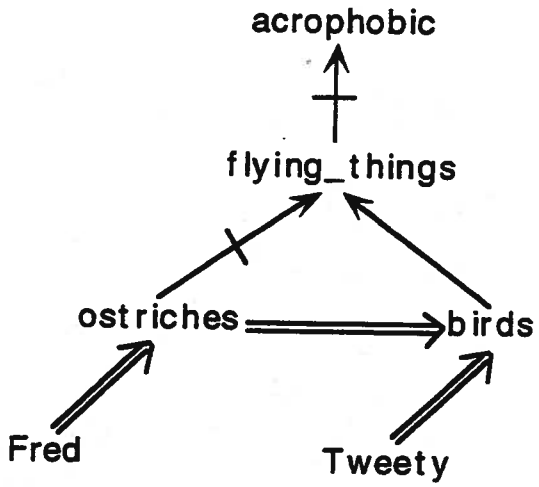
3. First-Order Logic [18 points]

Express the following sentences in clause normal form:

You can fool some of the people all of the time, but you can't fool all of the people all of the time.

4. Nonmonotonic Reasoning [22 points]

Consider the default theory below represented as a diagram. The notation in the diagram is as follows. Arrows indicate default implications (e.g., Birds are by default flying things), arrows with a line through them indicate default implications in which the conclusion has been negated (e.g., Ostriches are by default not flying things), and double lines indicate nondefault implications (e.g., Fred is an ostrich).



(a) Is the statement that Tweety is not acrophobic a brave consequence of this default theory? Is it a cautious consequence? [6 points]

(b) Is the statement that Fred is not acrophobic a consequence of this theory? [6 points]

(c) Is the statement that Fred is acrophobic a consequence of this theory? [6 points]

(d) Is it correct to say that the brave but not cautious consequences of a default theory are those for which there are arguments both for and against? What would be a more appropriate description? [4 points]

5. Probability [20 points]

Prove that $pr(p \mid q) \leq pr(q \rightarrow p)$.

(8)

SOLUTIONS: ARTIFICIAL INTELLIGENCE

1. Frame Representations

(a) Sedan

(b) Corporation

(c)

```
(Subclass CompanyVehicle Vehicle)
(ValueRestriction Corporation Owner CompanyVehicle)
(Subclass CompanyCar CompanyVehicle)
(MemberValue Sedan Style CompanyCar)
(Member Envig'sCar CompanyCar)
(OwnValue Envig Owner Envig'sCar)
```

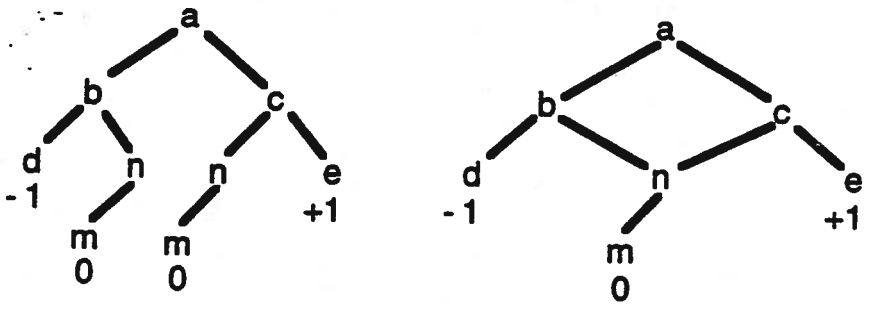
(d)

```
(forall (c s v x)
        (implies (and (MemberValue v s c) (Member x c)) (OwnValue v s x)))

(forall (c s v vr x)
        (implies
            (and (ValueRestriction vr s c) (OwnValue v s x) (Member x c))
            (Member v vr)))

(forall (c1 c2 x)
        (implies (and (Subclass c1 c2) (Member x c1)) (Member x c2)))
```

2. Adversary Search

We cannot necessarily prune nodes in game trees which are graphs. Let n be a node whose first child node evaluates to k. If n has a sibling with value less than or equal to k, then n would be pruned. However, if all of the values of n's siblings are greater than k, then n would not be pruned. In a graph, n could have two sets of siblings such that it would be correct to prune n, with respect to one set, but incorrect to prune it with respect to the other.

(9)



For example, consider the tree and graph above. First consider the example as a tree with n occurring twice. As a child of b, n is pruned, but as a child of c, it is not pruned. In this example, the value of b is -1 and c is 0, so a becomes 0.

Now consider the example as a graph. If we prune n from the graph during the evaluation of b, the value of c becomes 1 instead of 0, so the value of a also becomes 1. This change of value could easily affect the outcome of a game (e.g., if there were another possible move from a with value 0.99).

3. First-Order Logic

$$\neg\text{person(Sk-x1)} \vee \neg\text{time}(t_1) \vee \text{can-fool-at(SK-x1,}t_1\text{)}$$
$$\text{person(Sk-x3)}$$
$$\text{time(Sk-t3)}$$
$$\neg\text{can-fool-at(Sk-x3,Sk-t3)}$$

4. Nonmonotonic Reasoning

(a) The statement that Tweety is not acrophobic is a consequence of both extensions of this default theory. Therefore, ¬acrophobic(Tweety) is a cautious consequence of this theory.

(b) One extension of this theory has ¬acrophobic(Fred) as a consequence, but the other does not. Therefore, ¬acrophobic(Fred) is a brave consequence of this theory.

(c) acrophobic(Fred) is not a consequence of either extension of this theory. In fact, there is no way to derive acrophobic(x) for any x given the diagram we have.

(d) No. There is an argument for ¬acrophobic(Fred) in (b), but none against it. Brave consequences are those that are possibly, but not necessarily, true.

10

5. Probability

Consider the cases where $Pr(q \rightarrow p)$ holds:

$$Pr(q \rightarrow p) = Pr(q \wedge p) + Pr(p \wedge \neg q) + Pr(\neg p \wedge \neg q)$$

If we apply the definition of conditional probability, we get:

$$Pr(q \rightarrow p) = Pr(p \mid q)Pr(q) + Pr(p \mid \neg q)Pr(\neg q) + Pr(\neg p \mid \neg q)Pr(\neg q)$$

The latter two terms combine to produce:

$$Pr(q \rightarrow p) = Pr(p \mid q)Pr(q) + (Pr(p \mid \neg q) + Pr(\neg p \mid \neg q))Pr(\neg q)$$

and reduces to:

$$Pr(q \rightarrow p) = Pr(p \mid q)Pr(q) + Pr(\neg q)$$

Since $Pr(p \mid q) \leq 1$ and since $Pr(q) + Pr(\neg q) = 1$, we get $Pr(q \rightarrow p) \geq Pr(p \mid q)$.

(11)

SOLUTIONS: AUTOMATA AND FORMAL LANGUAGES

Instructions: You are expected to *sketch* the main ideas in your solutions, but be very brief and avoid unnecessary detail. You are permitted to invoke any result proved in the Hopcroft-Ullman book provided you include the appropriate citation.

1. (9 points) Consider the following context-free grammar G over the alphabet $\Sigma = \{a, b\}$.

$$\begin{aligned} S &\rightarrow aB \mid Ab \mid ab \\ A &\rightarrow aS \\ B &\rightarrow Sb \end{aligned}$$

(a) [3 points] Give a succinct description of $L(G)$.

(b) [4 points] Show that G is an ambiguous grammar.

(c) [2 points] What productions should you *delete* from G to render it unambiguous *without changing its language*?

Solution:

(a) $L(G) = \{a^n b^n \mid n \geq 1\}$.

(b) The following are two distinct leftmost derivations for the string $aabb$, implying that the grammar must be ambiguous.

$$S \stackrel{lm}{\Rightarrow} aB \stackrel{lm}{\Rightarrow} aSb \stackrel{lm}{\Rightarrow} aabb$$

$$S \stackrel{lm}{\Rightarrow} Ab \stackrel{lm}{\Rightarrow} aSb \stackrel{lm}{\Rightarrow} aabb$$

(c) Deleting the variable A or the variable B, as well as all related productions, gives an unambiguous grammar with the same language.

2. (12 points) Let $<M>$ denote any natural encoding of a Turing machine M. Consider the following decision problem:

$L_O = \{<M> \mid \exists$ odd numbers p, q such that $L(M)$ has strings of length both p and $q\}$

Prove that L_O is undecidable by using a reduction from the halting problem (which is known to be undecidable). Recall that the halting problem corresponds to the language $L_H = \{<M, w> \mid$ Turing machine M halts on input $w\}$.

Solution: We use the following reduction from L_H to L_O to obtain the proof of undecidability. Given a Turing machine M and input w, construct a Turing machine $\widehat{M}$ which behaves as follows on being given input $\widehat{w}$.

(a) $\widehat{M}$ simulates the behavior of M on input w, and

(b) $\widehat{M}$ accepts its input $\widehat{w}$ if M halts on w.

To show that the above reduction works, we need to prove that:

(a) There is an algorithm that computes $\widehat{M}$ given M, w.

(b) $<M, w> \in L_H$ if and only if $<\widehat{M}> \in L_O$.

The proof proceeds as follows.

(a) Given w and a description of M, it is easy to construct a Turing machine $\widehat{M}$ that simulates the computation of M on input w, using the same ideas used in the construction of the universal Turing machine (see Theorem 8.4 in the textbook).

(b) Let $\widehat{w}$ be any input string. Then,

$\widehat{M}$ accepts $\widehat{w}$
$\Leftrightarrow$ the simulation of M on w halts
$\Leftrightarrow <M, w> \in L_H$.

Consequently, if $<M, w> \in L_H$ then $\widehat{M}$ accepts *all* strings and so must belong to L_O. Conversely, if $<M, w> \notin L_H$ then $\widehat{M}$ does not accept *any* string at all and so cannot belong to L_O.

3. (9 points) Prove that the following decision version of the following 3-PATH problem is NP-complete.

INSTANCE: An undirected graph $G(V, E)$.
QUESTION: Is there a collection of 3 vertex-disjoint paths in G that cover all the vertices?

Informally, the answer is YES if there exist 3 paths in the graph such that each vertex is contained in *exactly* one of these paths.

(Hint: You may assume that the Hamiltonian Path problem is NP-complete. This is the problem of deciding whether a given graph has a path containing each vertex exactly once.)

Solution: We first verify that the 3-PATH problem is in NP. A polynomial time algorithm can easily "guess" three disjoint sequences of vertex labels, and then check that the following two conditions are satisfied: each vertex appears in exactly one of these three sequences; and, each sequence corresponds to a path in the graph G.

To establish the NP-hardness of 3-PATH, we provide a polynomial time reduction from the HAMILTONIAN PATH problem, which is known to be NP-complete. An instance of the HAMILTONIAN PATH problem is some graph H and the goal is to check whether it has a path containing all the vertices. We transform this to the 3-PATH problem by producing a graph G which consists of 3 disjoint and disconnected copies of the graph H. We now need to show that G has a 3-path cover if and only if H is hamiltonian.

If H is hamiltonian, then the three hamiltonian paths in the three copies of H in G will constitute a 3-path cover for G. Conversely, suppose that G has a 3-path cover. Since G consists of 3 disconnected copies of H, the 3 paths must lie in distinct copies. Clearly, each path must be a 1-path cover, or a hamiltonian path, for the copy of H in which it lies. Thus, H must be hamiltonian.

Computer Science Department
Stanford University
Comprehensive Examination in Compilers
Autumn 1995

October 19, 1995

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 is 1 page.

   The exam takes 30 minutes.

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

Compilers

There are a number of essay questions on this exam. You should, of course, keep your answers clear and concise.

1. (5 minutes) Answer true or false. In all questions, assume that the context-free grammar has no useless symbols.

   F (a) There exists a context-free grammar with that is LL(1) but not LR(1).

   T (b) There exists an context-free language that is LR(1) but not LL(1).

   T (c) YACC can correctly parse all ambiguous context-free grammars by using precedence and associativity hints.

   (d) There exists an unambiguous grammar that is not LR(1).

   F (e) LALR(1) parser generation (*not parsing*) is linear in the size of the input grammar.

2. (10 minutes) Consider the following grammar:

   $$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow aA \\ A &\rightarrow \epsilon \\ B &\rightarrow b \\ B &\rightarrow SC \\ C &\rightarrow c \end{aligned}$$

   (a) Compute the FIRST and FOLLOW sets for the nonterminals of the grammar.

   (b) Based on the FIRST and FOLLOW sets, is it LL(1)? Explain your answer.

   (c) What is the language of this grammar? From inspecting the language, is it obvious whether there exists an LL(1) answer (if you say it's obvious, explain [obviously]).

3. (10 minutes) These questions are about the handling of variables by a compiler for a simple language like C. In your answers, address only the compiler behavior that is *necessary for code generation*. Do not address type checking or other aspects of semantic analysis are not strictly necessary to emit code for correct programs.

   (a) Describe the compiler's processing of a global variable g of type `int`, both at the point of *declaration* and at the point of *use*.

   (b) How is the handling of a local variable declaration of type `int` different?

   (c) What information does the compiler have to maintain in the symbol table to generate code for `A[i].f[j]`?

4. (5 minutes) Imagine a language in which the types of expressions can be determined in a single bottom-up traversal of the syntax tree of a program. The language also has implicit type conversions (also called *coercions*), such as `int` to `float`.

   How could a compiler

   - recognize when coercions need to occur, and
   - generate code to perform them

   in a single bottom-up traversal of an expression?

(18)

Compilers Solutions

There are a number of essay questions on this exam. You should, of course, keep your answers clear and concise.

1. (5 minutes) Answer true or false. In all questions, assume that the context-free grammar has no useless symbols.

   (a) There exists a context-free grammar with that is LL(1) but not LR(1).
   false

   (b) There exists an context-free language that is LR(1) but not LL(1).
   true, e.g. $\{a^i b^j \mid i > j\}$

   (c) YACC can correctly parse all ambiguous context-free grammars by using precedence and associativity hints.
   false, e.g. $\{a^i b^j c^k \mid i = j \text{ or } j = k\}$

   (d) There exists an unambiguous grammar that is not LR(1).
   true, e.g. $S \rightarrow Aaa|Bab; A \rightarrow \epsilon; B \rightarrow \epsilon$

   (e) LALR(1) parser generation (*not parsing*) is linear in the size of the input grammar.
   false - it's exponential in the worst case. Parsing is linear in the input size, though.

2. (10 minutes) Consider the following grammar:

$$
\begin{aligned}
S &\rightarrow AB \\
A &\rightarrow aA \\
A &\rightarrow \epsilon \\
B &\rightarrow b \\
B &\rightarrow SC \\
C &\rightarrow c
\end{aligned}
$$

   (a) Compute the FIRST and FOLLOW sets for the nonterminals of the grammar.

   First and Follow sets

Symbol	First	Follow
S	a, b	$c, \$$
A	ϵ, a	a, b
B	a, b	$c, \$$
C	c	$c, \$$

   (b) Based on the FIRST and FOLLOW sets, is it LL(1)? Explain your answer.

   The grammar is not LL(1), because the right-hand-sides of productions $B \rightarrow b$ *and* $B \rightarrow SC$ *both have* b *in their FIRST sets. Also, the* a *is in FIRST of* $A \rightarrow aA$ *and FOLLOW of* A *(relevant because of* $A \rightarrow \epsilon$ *production).*

   (c) What is the language of this grammar? From inspecting the language, is it obvious whether there exists an LL(1) answer (if you say it's obvious, explain [obviously]).

   The language of the grammar is regular (a^*bc^*)*, so there is an LL(1) grammar.*

(19)

3. (10 minutes) These questions are about the handling of variables by a compiler for a simple language like C. In your answers, address only the compiler behavior that is *necessary for code generation.* Do not address type checking or other aspects of semantic analysis are not strictly necessary to emit code for correct programs.

   (a) Describe the compiler's processing of a global variable g of type `int`, both at the point of *declaration* and at the point of *use.*

   There is a large amount of latitude in the answers, depending on what assumptions one makes about the language and compiler. Here is an example:
   At the point of declaration, the compiler needs to enter info about g into the symbol table, including its type, etc. but especially, its location. The offset is established when the variable is allocated *within the global data section. The offset is a compile-time constant. The actual address of the global data section of memory is often not established until link time, but it is a constant when the program is actually executed.*
   At the point of use, the compiler needs to compute the address of the variable. If no array indexing is required (as when it is of type `int`*, the address is a known constant, so no code needs to be generated. However, code is needed to fetch the value of the variable from the memory location.*

   (b) How is the handling of a local variable declaration of type `int` different?

   Obviously, the variable is tagged as a local, not global variable when it goes in the symbol table. The offset is relative to a stack frame pointer of some kind, which is usually stored in a machine register. Usually, all the locals for the procedure are allocated in a single stack frame for the procedure. The stack frame pointer is set up when a function is called, and restored when the function returns.
   When the variable is accessed, code is needed to add the contents of the frame pointer to the offset for the local variable. This is almost exactly the same code that would be required for an expression with + in it. Once the location of the variable has been computed, code must be generated to fetch the value.

   (c) What information does the compiler have to maintain in the symbol table to generate code for `A[i].f[j]`?

   It needs to keep track of whether the variable `A` *is local or global, and what its offset is (as above). It also needs remember the sizes of the array elements (to do array indexing) and the offsets of fields within structures. The generated code computes (frame pointer) + (offset of A) + i * (size of A elements) + (offset of f) + j * (size of A[i].f elements)*

4. (5 minutes) Imagine a language in which the types of expressions can be determined in a single bottom-up traversal of the syntax tree of a program. The language also has implicit type conversions (also called *coercions*), such as `int` to `float`.

   How could a compiler

   - recognize when coercions need to occur, and
   - generate code to perform them

   in a single bottom-up traversal of an expression?

20

A coercion is necessary when there is a type mismatch between operands and operator, which can be detected during a bottom-up pass when the types of the operands have been determined and the operator is examined. For example, in the expression $i+x$, if i is an integer and x is a float, there is a type mismatch, because $+$ requires two integers or two floats.

Machines that support floating point have conversion instructions. The compiler acts as though it had inserted a conversion operation into the syntax tree at the point of the type mismatch, which it then compiles just like any other operator. For example, $i + x$ would become $flt(i) + x$ (which provides two float operands to $+$), and the appropriate machine instructions would be generated to compute the result of $flt(i)$, which is then added to x.

21

Computer Architecture
Comprehensive Examination
Fall 1995-96

SOLUTIONS: COMPUTER ARCHITECTURE

This exam is closed book. There are 55 points in total, and 60 minutes for the exam. Write your answers in the spaces provided. Where possible show all intermediate steps in computing your answers; that will allow us to give partial credit for incorrect answers.

1	15	
2	5	
3	5	
4	5	
5	5	
6	20	
TOTAL	55	

28

Question 1 - Memory System Design (15 points)

Imagine you have a cache-coherent multiprocessor system in which a cache-miss causes the hardware to place not only the memory block that you missed into the cache, but also the following memory block. Furthermore, whenever a cache line is accessed by the processor for the first time, the following cache line is also brought in.

(a) (8 points) What are the advantages and disadvantages of the above approach as compared to simply doubling the cache line size in the base system?

Answer:
Advantages: (1) Reduced false sharing in MPs (i.e., reduced bouncing of lines between one processor and another even though they are writing to distinct portions of the line); (2) Can totally eliminate cache misses in case of unit-stride data accesses; (iii) Lower miss penalty as cache-line-size is smaller.

Disadvantages: (1) More complex hardware; (2) More tag overhead with smaller lines; (3) Makes less efficient use of the bus, as several smaller transfers rather than a single large transfer is being done (can reduce number of processors that can be supported on a given bus); (4) may fetch more unnecessary data than the scheme with double cache line size.

(b) (4 points) What information would you need to quantitatively evaluate whether to implement the suggested approach on a particular machine?

Answer:
- effective miss rate of the two schemes (impacts stall time)
- average miss penalty under the two schemes (impacts stall time)
- bus bandwidth required per processor under the two schemes (determines when contention will kick in and limit number of processors that can be supported)

(c) (3 points) Would you expect the above approach to be more effective for instruction cache or data cache?

Answer: The prefetching approach should be quite effective for both instruction and data caches. The utility for data caches is a function of the spatial locality in the application (e.g., it may not work too well for programs that make heavy use of pointers, or where the arrays are being accessed in a non-unit-stride manner). The instruction reference stream of programs usually has more spatial locality, and therefore the benefits should spread across a larger segment of programs.

29

Question 2 - Processor Pipelines (5 points)

The standard DLX pipeline is quite short; only 5 stages deep. In contrast, the recently announced intel P6 processor has a 13-deep pipeline. Discuss, in general, the advantages and disadvantages of long versus short pipelines.

Answer:
Long Pipeline Advantages: It allows for higher clock rate (as work done in each stage is smaller), potentially offering higher performance.

Long Pipeline Disadvantages: (1) Requires extra concurrency, which may or may not be available in the program; said another way, there will be more load-delay slots and branch-delay slots that will need to be filled, which may not always be possible; (2) if branch-prediction is used, there will be higher penalty for mis-predicted branches; (3) tougher to balance the work between the various pipe stages; (4) more extensive and complex bypassing logic is needed, implying more complex hardware.

Question 3 - Precise vs. Imprecise Exceptions (5 points)

The DLX architecture presents a precise exception model.

(a) (2 points) Explain briefly what the term "precise exception model" means.

Answer: If the pipeline can be stopped so that the instructions just before the faulting instruction are completed, and those after it can be restarted from scratch, the pipeline is said to have precise exceptions.

(b) (3 points) Give a example of how the precise exception model could be violated in the DLX pipeline if the exceptions were signalled in the order in which they occurred rather than by waiting until the excepting instruction reaches a particular point in the pipeline. State any assumptions that you make.

Answer:

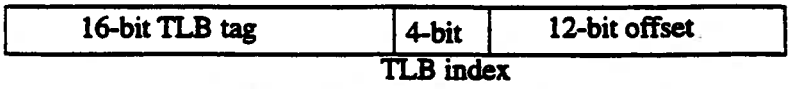
```
LW   R4, 256(R2)      ==> this instr has a protection fault

ADD  R7, R4, R5       ==> this instr lies on a different page
                          than the previous one, and the IFETCH gets
                          a page fault.
```

30

Question 4 - Virtual Memory Systems (5 points)

Assume a machine has a virtual address size of 32 bits, a page size of 4KB, and a 4-way associative TLB with 64 total entries. Indicate how a 32-bit virtual address is broken down into page offset, TLB index, and TLB tag fields. Show the width (in bits) and position of each of these fields.

Answer:



Question 5 - Cache Organization (5 points)

Consider 2 caches of the same total size but different organizations.

Cache-Organization-1:
- Total Size: 8 bytes
- Block Size: 1 byte
- Associativity: direct mapped
- Replacement policy: LRU

Cache-Organization-2:
- Total Size: 8 bytes
- Block Size: 1 byte
- Associativity: 2-way set associative
- Replacement policy: LRU

Assume both caches are initially empty (i.e., contain no valid data).

Provide a reference stream of no more than 4 references that exhibits a "higher" miss rate on cache-2 than it does on cache-1. Each element of the reference stream should be a byte address. Indicate next to each reference whether it misses or hots in each cache.

Answer:

	Org-1	Org-2
LD 0	miss	miss
LD 4	miss	miss
LD 12	miss	miss
LD 0	hit	miss

(31)

Question 6 - System Performance (20 points)

Consider the following 2 systems.

System-1:

200 MHz processor
Two load delay slots (first filled 75% and second filled 20% of the time)
256 KB data cache (miss rate 5%, miss penalty 25 cycles)

System-2:

150 MHz processor
One load delay slot (filled 75% of the time)
8KB first-level data cache (miss rate 10%; miss penalty 7 cycles if data in level-2 cache and 32 cycles if data must be fetched from main memory)
1 MB second-level data cache (global miss rate 2%)

Now consider the following workload running on both systems:

Loads: 25%
Stores: 10%
Other: 65%

(a) (14 points) Ignoring instruction-cache effects, determine which system is faster for the given workload. Assume that neither system has any write buffers. State any other assumptions that you make.

Answer:

```
System-1 CPI  = 0.65 + 0.25 (1 + .05*25 + .25 + .80) + 0.1 (1 +
                   .05 * 25)
              = 1.7
              This implies performance = 200/1.7 = 118 MIPS

System-2 CPI  = 0.65 + 0.25 (1 + 0.1 (.98*7 + .02*32) + 0.25) +
                  0.1 (1 + 0.1(.98*7 + .02*32))
              = 1.325
              This implies performance = 150/1.325 = 113 MIPS

Thus system-1 is faster in this case.
```

(b) (3 points) Assume system-2's clock rate is increased to 200 MHz, but all other parameters remain the same. Now which system is faster?

Answer: System-2's performance is now 200/1.325 = 151 MIPS. Thus system-2 will be faster if the same clock rate as system-1 can be achieved.

3.

(c) (3 points) Assume an infinitely-deep write buffer is added to system-2 inbetween the processor and the first-level cache, but all other parameters remain the same (system-2's clock rate is 150 MHz). Now which system is faster?

Answer: The implication is that all write-stalls will now be hidden by the write buffer. Thus:

System-2 CPI = 0.65 + 0.25 (1 + 0.1 (.98*7 + .02*32) + 0.25)
= 1.15
This implies performance = 150/1.15 = 130 MIPS,
which is faster than system-1 at 200 MHz.

Computer Science Department
Stanford University
Comprehensive Examination in Numerical Analysis
Autumn 1995

October 20, 1995

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: Numerical Analysis (30 points). Fall 1995

(Problem I)

(15 points). **Linear Algebra**

(a) Let A be a real $n \times n$ symmetric matrix with n distinct real eigenvalues. Show that the eigenvectors of A are orthogonal to one another. If A is also positive definite show that the eigenvalues are positive.

(b) Define the Euclidean norm of a vector in R^n.

(c) Let A be a real, positive-definite, $n \times n$ symmetric matrix with eigenvalues $0 < \lambda_1 < \ldots < \lambda_n$. Let

$$\|x\|_A^2 = x^T A x.$$

Prove that $\|\cdot\|_A$ as defined is indeed a norm on R^n.

(Problem II)

(15 points). **Differential Equations and Quadrature**

(a) Define the composite trapezoidal rule for the approximate integration of

$$I := \int_a^b f(x)dx$$

over n intervals of equal length $h = (b-a)/n$. State the order of accuracy of the method in terms of h, under an assumption on the smoothness of f which you should state.

(b) Use the quadrature rule derived in (a) to derive a numerical approximation to the differential equation

$$\frac{du}{dt} = f(t), \quad u(\tau) = u_0,$$

at time $t = \tau + 1$, by partitioning the interval $[\tau, \tau + 1]$ into n equal subintervals.

(c) How big (in terms of h) would you expect the error to be if you applied the method in (b) to the equation

$$\frac{du}{dt} = t^{3/2}, \quad u(\tau) = u_0,$$

with $\tau \geq 0$. What are the practical implications of using the method on this problem if τ is very small?

57

Solutions to Comprehensive: Numerical Analysis (30 points). Fall 1995

(Problem I)
(15 points). **Linear Algebra**

(a) Let A be a real $n \times n$ symmetric matrix with n distinct real eigenvalues. Show that the eigenvectors of A are orthogonal to one another. If A is also positive definite show that the eigenvalues are positive.

SOLUTION: Let

$$Ax = \lambda x, \quad Ay = \omega y.$$

Then

$$\lambda x^T y = (Ax)^T y = x^T (A^T y) = x^T (Ay) = \omega x^T y.$$

Thus

$$(\lambda - \omega) x^T y = 0.$$

Since $\lambda \neq \omega$ we have $x^T y = 0$ as required. Since A is positive definite:

$$0 < x^T Ax = \lambda x^T x$$

and $\lambda > 0$ follows.

(b) Define the Euclidean norm of a vector in R^n.

SOLUTION: If $v = (v_1, \ldots, v_n)$ then

$$\|v\|^2 := \sum_{j=1}^{n} v_j^2.$$

Thus $\|v\|^2 = v^T v$.

(c) Let A be a real, positive-definite, $n \times n$ symmetric matrix with eigenvalues $0 < \lambda_1 < \ldots < \lambda_n$. Let

$$\|x\|_A^2 = x^T Ax.$$

Prove that $\|\cdot\|_A$ as defined is indeed a norm on R^n.

SOLUTION: We need to show the following three things:

- $\|x\|_A \geq 0$ and $\|x\|_A = 0$ if and only if $x = 0$. *This follows from the fact that A is positive definite;*
- $\|\alpha x\|_A = |\alpha| \|x\|_A$ for any scalar α. *But*

$$\|\alpha x\|_A^2 = (\alpha x)^T A(\alpha x) = \alpha^2 x^T Ax = \alpha^2 \|x\|_A^2$$

and the result follows.

- $\|x + y\|_A \leq \|x\|_A + \|y\|_A$. *Now:*

$$\|x + y\|_A^2 = x^T Ax + y^T Ay + x^T Ay + y^T Ax$$

Now let $C = \sqrt{A}$ so that $C^2 = A$. Then

$$\|x + y\|_A^2 = x^T Ax + y^T Ay + (Cx)^T Cy + (Cy)^T Cx$$

and by the Cauchy-Schwarz inequality

$$\|x + y\|_A^2 \leq x^T Ax + y^T Ay + 2\|Cx\|\|Cy\|.$$

58

But $\|Cx\|^2 = x^T C^2 x = x^T Ax = \|x\|_A$. *Hence*

$$\|x+y\|_A^2 \le x^T Ax + y^T Ay + 2\|x\|_A \|y\|_A$$

and taking square-roots yileds the required result.

(Problem II)

(15 points). **Differential Equations and Quadrature**

(a) Define the composite trapezoidal rule for the approximate integration of

$$I := \int_a^b f(x)dx$$

over n intervals of equal length $h = (b-a)/n$. State the order of accuracy of the methods in terms of h, under an assumption on the smoothness of f which you should state.

SOLUTION: The rule is

$$I \approx \frac{h}{2}[f_0 + f_n] + h[f_2 + f_3 + \ldots + f_{n-1}].$$

Here $f_j = f(x_j)$ and $x_j = a + jh$. The error is $\mathcal{O}(h^2)$ provided $f \in C^2([a,b], R)$.

(b) Use the quadrature rule derived in (a) to derive a numerical approximation to the differential equation

$$\frac{du}{dt} = f(t), \quad u(\tau) = u_0,$$

at time $t = \tau + 1$, by partitioning the interval $[\tau, \tau+1]$ into n equal subintervals.

SOLUTION: The differential equation can be integrated to give

$$u(t) = u_0 + \int_\tau^t f(s)ds.$$

Thus

$$u(\tau+1) = u_0 + \int_1^{\tau+1} f(s)ds.$$

We introduce the mesh points $t_n = \tau + nh$ where $Nh = 1$ and let $f_n = f(t_n)$. Applying the quadrature rule to the integral shows that

$$u(\tau+1) \approx u_0 + \frac{1}{2}[f_0 + f_N] + [f_2 + f_3 + \ldots + f_{N-1}].$$

(c) How big (in terms of h) would you expect the error to be if you applied the method in (b) to the equation

$$\frac{du}{dt} = t^{3/2}, \quad u(\tau) = u_0,$$

with $\tau \ge 0$. What are the practical implications of using the method on this problem if τ is very small?

SOLUTION If $\tau > 0$ then $f(t) = t^{3/2} \in C^2([\tau, \tau+1], R)$ and the error will be $\mathcal{O}(h^2)$. If $\tau = 0$ then $f(t)$ is not a C^2 function on the interval in question and the error will be smaller (in fact $\mathcal{O}(h)$.) In practice, if τ is small, the method will require very small h before second order convergence is observed.

59

Computer Science Department
Stanford University
Comprehensive Examination in Software Systems
Autumn 1995

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, an essay-type question worth 6 points or less doesn't deserve an extremely detailed answer, even though a person can expound at length on just about any topic in computer science.
3) The total number of points is 60, and the exam takes 60 minutes. This "coincidence" can help you plan your time.
4) This exam is CLOSED BOOK. You may NOT use notes, articles, books, computer, 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; your 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 assumption(s) as well as the answer.
7) Be sure to provide justification for your answers.

66

Comprehensive Exam: Software Systems (60 points)

1) (10 points) Describe the purpose of device drivers in modern operating systems. (Hint: I'm looking for something more than "they drive devices.")

2) (18 points) The algorithms and data structures used in operating systems are frequently a function of the hardware technology. As hardware technology has changed so have operating systems. For the following hardware technology changes, describe the operating system algorithms and data structures that might need to be changed.
   (a) A very large (multiple order-of-magnitude) increase in physical memory size.
   (b) A large increase in the number of bits in a virtual address.
   (c) An increase in the number of CPUs in the system from one to many tens of CPUs.

3) (12 points) The UNIX operating system has a file buffer cache that uses a writeback policy with a 30 second timeout. This means that changes to file blocks can sit in the buffer cache for up to 30 seconds before being written back to the disk. What are motivations for and drawbacks of this scheme?

4) (8 points) Describe what an atomic operation is. Give examples and describe why they are useful.

5) (12 points) Describe the necessary and sufficient conditions for deadlock. For each condition describe a practical deadlock prevention technique that works by preventing the condition from arising.

67

Comprehensive Exam: Software Systems Solutions (60 points)

1) (10 points) Describe the purpose of device drivers in modern operating systems. (Hint: I'm looking for something more than "they drive devices.")

The key purpose of device drivers is to keep device-specific algorithms and data structures contained to a relatively small portion of the kernel. The operating system defines a standard interface to devices and it is the job of the device drive to convert this standard interface to the needed device-specific operations.

2) (18 points) The algorithms and data structures used in operating systems are frequently a function of the hardware technology. As hardware technology has changed so have operating systems. For the following hardware technology changes, describe the operating system algorithms and data structures that might need to be changed.
(a) A very large (multiple order-of-magnitude) increase in physical memory size.
(b) A large increase in the number of bits in a virtual address.
(c) An increase in the number of CPUs in the system from one to many tens of CPUs.

(a) A large increase in physical memory will stress the physical memory management data structures including the memory free list and the page replacement algorithm. Since memory is a less precious resource, we might want to modify the OS to spend less resources tracking it. For example, switching to a simple page replacement algorithm rather than LRU. Other algorithms that would probably need to be changed including the management of swap space (physical memory might be larger than the swap disks). Additional caching of files, etc. could be done with the extra memory.
(b) A large increase in virtual address space will stress the virtual to physical mapping data structures (i.e. page tables) maintained by the kernel. The OS will have to go to a multi-level or invented page tables. It can also cause problems for swap space management.
(c) Large changes will be needed in the synchronization used as well as the scheduling and many other parts of the system.

3) (12 points) The UNIX operating system has a file buffer cache that uses a writeback policy with a 30 second timeout. This means that changes to file blocks can sit in the buffer cache for up to 30 seconds before being written back to the disk. What are motivations for and drawbacks of this scheme?

The motivations include decoupling of file write speed from that of the disk, the ability to coalesce multiple writes to the same block into a single write to disk, and the ability to generate multiple block writes so the disk scheduler can optimize disk access patterns. The chief drawback is data can be lost if the system crashes.

4) (8 points) Describe what an atomic operation is. Give examples and describe why they are useful.

An atomic operation is an operation that, from some point of view, appears indivisible. Partial results of an atomic operation are never visible. From another entity viewing the operation, the changes made will happen instantly so the viewer will see either all or none of the changes. Examples of atomic operations are operations on a data structure such that all operations are performed while holding a lock. Atomic operations make it easier to reason and build systems because within atomic operations the presents of other processes can be ignored.

68

5) (12 points) Describe the necessary and sufficient conditions for deadlock. For each condition describe a practical deadlock prevention technique that works by preventing the condition from arising.

There are four necessary and sufficient conditions for deadlock:
(a) Limited access: resources cannot be shared.
(b) No preemption. Once given, a resource cannot be taken away.
(c) Multiple independent requests: processes don't ask for resources all at once.
(d) There is a circularity in the graph of who has what and who wants what.

Preventing the condition:

Violating (a) is pretty tough to do in a general-purpose system. Having enough resources so that they don't need to be shared is done is some special-purpose operating systems.

Violating (b) is again pretty difficult in a general-purpose system. Although it is possible to construct a system such that all resources are preemptable, resources attached to the outside world (e.g. terminals, printers) are difficult to preempt.

(c) can be violated by requiring processes to request all resources at once and having the system wait until all the resources are available before continuing.

(d) can be violated by forcing an order to resource requests such that cycles in the graph are avoided.

69