CS 188	Introduction to AI	
Fall 2002	Stuart Russell	Midterm

You have 1 hour and 20 minutes. The exam is open-book, open-notes. 100 points total.

You will not necessarily finish all questions, so do your best ones first.
Write your answers in blue books. Check you haven't skipped any by accident. Hand them all in. Panic not.

1. (18 pts.) True/False

(a) (3) True. What counts for breadth-first is the depth of the goal, not cost.

(b) (3) True.

(c) (3) True (in fact, equivalent—check via truth tables or convert to CNF and compare).

(d) (3) True—illustrated in lecture slides; *-alpha-beta algorithm.

(e) (3) False—rook can cross the board in one move, so Manhattan is pessimistic hence not admissible.

(f) (3) False—a unary constraint might eliminate all possible values for a variable.

2. (20+5 pts.) Constraint satisfaction
The key aspect of any correct formulation is that a complete assignment satisfying the constraints must be a real solution to the real problem.

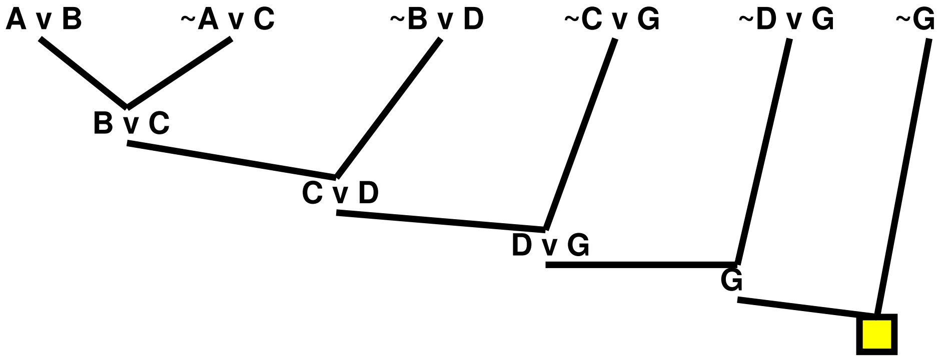
(a) (8) Variable domains: all pairs of adjacent squares. You might want to avoid having the same pair of squares appearing twice in different orders.
Constraints: every pair of variables is connected by a constraint stating that their values may not overlap (i.e., they cannot share any square).

(b) (8) Variable domains: the set of (up to four) adjacent squares. The idea is that the domino covering this square can choose exactly one of the adjacent squares to cover too.
Constraints: between every pair of adjacent squares A and B. A can have value B iff B has value A.

(c) (4) The constraints in (b) are binary constraints relating adjacent squares, so they will form a loop whenever the squares form a loop (e.g., any 2 by 2 block). So any problem where the "width" of the shape is 1 is OK, e.g., 6 in a row, and there are no loops.

3. (24 pts.) Propositional Logic

(a) (8) Here's the "standard" resolution tree obtained by negating the goal:



It's also OK simply to derive G from the premises.

(b) (8) This can be answered with or without $True$ and $False$ symbols. We'll omit them for simplicity.
First, each 2-CNF clause has two places to put literals. There are $2n$ distinct literals, so there are $(2n)^2$

syntactically distinct clauses. We gave 6 pts for any answer that was quadratic in n.
Now, many of these clauses are semantically identical. Let us handle them in groups. There are $C(2n,2) = (2n)(2n-1)/2 = 2n^2 - n$ clauses with two different literals, if we ignore ordering. All these clauses are semantically distinct except those that are equivalent to $True$ (e.g., $(A \vee \neg A)$), of which there are n, so that makes $2n^2 - 2n + 1$ clauses with distinct literals. There are $2n$ clauses with repeated literals, all distinct. So there are $2n^2 + 1$ distinct clauses in all.

(c) (6) Resolving two 2-CNF clauses cannot increase the clause size; therefore, resolution can generate only $2n^2 - n$ distinct clauses before it must terminate.

(d) (2) First, note that the number of 3-CNF clauses is $O(n^3)$, so we cannot argue for nonpolynomial complexity on the basis of the number of different clauses! The key observation is that resolving two 3-CNF clauses can *increase* the clause size to 4, and so on, so clause size can grow to $O(n)$, giving $O(2^n)$ possible clauses.

4. (20 pts.) First-order logic

(a) (7) "No two people have the same social security number."

$$\neg\exists\, x,y,n \;\; Person(x) \wedge Person(y) \;\Rightarrow\; [HasSS\#(x,n) \wedge HasSS\#(y,n)]$$

This uses $\Rightarrow$ with $\exists$. It also says that no person has a social security number because it doesn't restrict itself to the cases where x and y are not equal. Correct version:

$$\neg\exists\, x,y,n \;\; Person(x) \wedge Person(y) \wedge \neg(x = y) \wedge [HasSS\#(x,n) \wedge HasSS\#(y,n)]$$

(b) (2) "John's social security number is the same as Mary's."

$$\exists\, n \;\; HasSS\#(John,n) \wedge HasSS\#(Mary,n)$$

This is OK.

(c) (5) "Everyone's social security number has nine digits."

$$\forall\, x,n \;\; Person(x) \;\Rightarrow\; [HasSS\#(x,n) \wedge Digits(n,9)]$$

This says that everyone has every number. $HasSS\#(x,n)$ should be in the premise:

$$\forall\, x,n \;\; Person(x) \wedge HasSS\#(x,n) \;\Rightarrow\; Digits(n,9)$$

(d) (6) Here $SS\#(x)$ denotes the social security number of x. Using a function enforces the rule that everyone has just one.

$$\neg\exists\, x,y \;\; Person(x) \wedge Person(y) \;\Rightarrow\; [SS\#(x) = SS\#(y)]$$
$$SS\#(John) = SS\#(Mary)$$
$$\forall\, x \;\; Person(x) \;\Rightarrow\; Digits(SS\#(x),9)$$

5. (18 pts.) Planning

We will use $Peg(x)$ to say there is a peg in location x and $Empty(x)$ to say there isn't. (We need both to keep within STRIPS rules, just as in the blocks world.)

(a) (4) *Start* has as postconditions $Peg(x)$ for every square except C, which has $Empty(C)$.
Finish has as preconditions $Empty(x)$ for every square except C, which has $Peg(C)$.

(b) (8)

$$\begin{array}{l} Action(Move(u,w), \\ \quad \text{PRECOND:} Peg(u) \wedge Line(u,v,w) \wedge Peg(v) \wedge Empty(w) \\ \quad \text{EFFECT:} \neg Peg(u) \wedge Empty(u) \wedge \neg Peg(v) \wedge Empty(v) \wedge \neg Empty(w) \wedge Peg(w)) \end{array}$$

(c) (6) A conflict occurs when an action in the plan threatens a *causal link* (not just a precondition or postcondition, but a link from an action postcondition to another action's precondition). (Therefore, it's not enough just to say that, e.g., one move might fill a hole that another move requires to be empty.) Instead, name a precondition (e.g., a specific square X is empty), an action that achieves it (e.g., moving a peg from X to Y), and an action that threatens it (e.g., moving a peg from Z to X).