CS188 Intro. to AI
Fall, 2003 R. Wilensky

Midterm

- This is an open-book, open-notes exam.
- Write your name in the space below; space is provided for your name on the top of each page as well.
- Provide your answers in the space provided on the exam.
- You have the usual 80-minute class period to work on the exam.
- There are 100 points total.
- Questions vary in difficulty; do what you know first.

YOUR NAME: ______________________________ and SID: ______________________________

(*Space below for official use only.*)

Problem 1:	__________	(15)
Problem 2:	__________	(20)
Problem 3:	__________	(25)
Problem 4:	__________	(15)
Problem 5:	__________	(25)
Total :	___________	(100)

Name: ________________________________

Problem 1 --- True or False -- 15 points (3 each)

State whether each of the following statements is true or false. Give one or two sentences explaining your answer.

A) Constraint satisfaction algorithms are preferred for problems like the 8 Queens Problem because path finding algorithms cannot be applied to problems of this sort.

False: Path finding algorithms will work, but will generally be much more inefficient.

B) To help guarantee that A* search will find an optimal solution, we must play it safe and use a heuristic that doesn't overstate how close a node is to the goal.

False: A* requires an optimistic heuristic to guarantee an optimal solution.

C) In a search space with local maxima, it may be a better strategy to sometimes choose a move that appears to make things worse, rather than to always pick a move that will make things better.

True: This is just what simulated annealing does.

D) In checkers, the Manhattan distance is an admissible heuristic for the problem of moving a king from one square to another.

False: The Manhattan distance for moving one square on a diagonal is 2, while a king can accomplish this in one step.

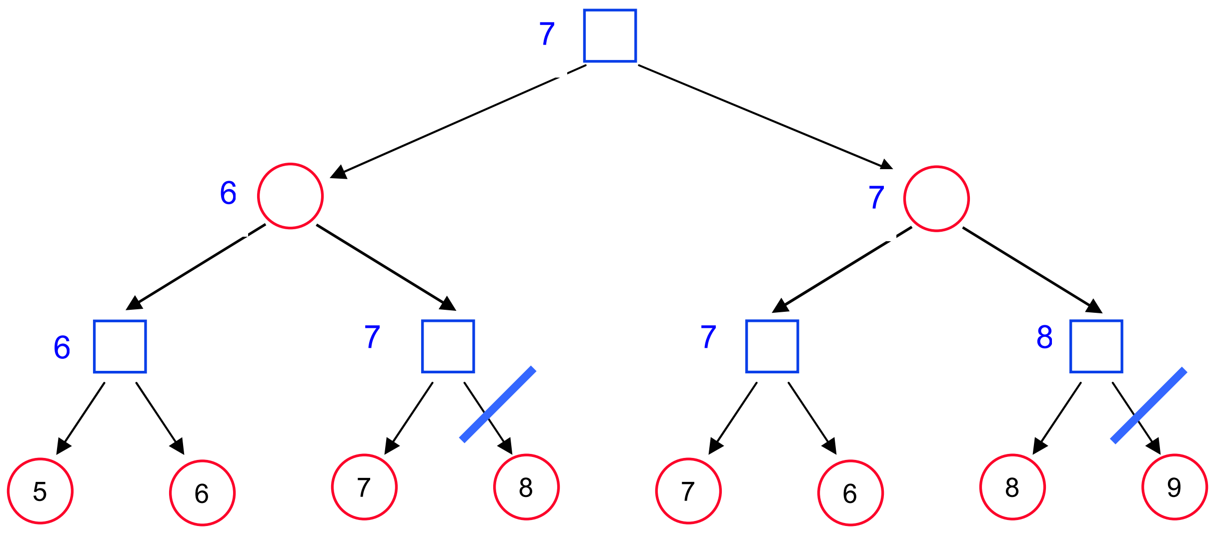
E) In an expert system that performs diagnosis, backward chaining would be an efficient strategy if there were potentially many possible causes of observations.

False: Backward chaining would tend to consider all possible causes without considering any observations first.

Name: ______________________________________

Problem 2 --- Game Playing (20 points)

Below is a game tree for a two player game, in which choices available to our agent emanate from squares, and choices available to the opponent are shown emanating from circles. The values in the circles at the bottom of the diagram show the values that a heuristic, or static evaluation function, would produce if applied to that game position.



Assuming descendents of a node are explored from left to right, write next to each node the value that *minimax* with *alpha-beta pruning* would propagate to that node. (If alpha-beta allows a subtree to be pruned, assume the value of the parent is computed from the remaining subtree.) *Draw a line* across those arrows representing branches that alpha-beta will allow minimax to prune, i.e., that do not have to be evaluated.

Name: ______________________________________

Problem 3 --- Predicate calculus -- 25 points

A) Using the predicates **Economic**, **SUV, Car** and **>**, and the function **MPG** (miles per gallon), represent in FOPC the following sentence: (10 points)

All SUVs get worse mileage than all economy cars.

(Note that, following our government's classification, SUV's are not cars.)

∀s,e (SUV(s) → (Economic(e) ∧ Car(e) → MPG(e)>MPG(s)))
or
∀s,e (SUV(s) ∧ Economic(e) ∧ Car(e) → MPG(e)>MPG(s))

B) Using the additional function **Quality**, we might represent the sentence "Some economy car is better than all SUVs." in one of the following ways:

∃e ∀s (SUV(s) → (Economic(e) ∧ Car(e) → Quality(e)>Quality(s)))

∃e (Economic(e) ∧ Car(e) ∧ ∀s (SUV(s) → Quality(e)>Quality(s)))

Determine whether these in fact are logically equivalent by turning each of them into clausal form, and comparing the results. Be sure to show all the steps of the conversions. (15 points)

(1) **∃e ∀s (SUV(s) → (Economic(e) ∧ Car(e) → Quality(e)>Quality(s)))**
Skolemize away existential variable:
∀s (SUV(s) → (Economic(E) ∧ Car(E) → Quality(E)>Quality(s)))
Drop quantifiers:
SUV(s) → (Economic(E) ∧ Car(E) → Quality(E)>Quality(s))
Get rid of implications:
¬SUV(s) ∨ (¬(Economic(E) ∧ Car(E)) ∨ Quality(E)>Quality(s))
Simplifying the result by distributing negations:
¬SUV(s) ∨ ¬Economic(E) ∨ ¬Car(E) ∨ Quality(E)>Quality(s))
Rename variables; turn into clauses:
{¬SUV(s'), ¬Economic(E), ¬Car(E), Quality(E)>Quality(s'))}

(2) **∃e (Economic(e) ∧ Car(e) ∧ ∀s (SUV(s) → Quality(e)>Quality(s)))**

Skolemize, drop universal:
Economic(E) ∧ Car(E) ∧ (SUV(s) → Quality(E)>Quality(s))
Get rid of implications:
Economic(E) ∧ Car(E) ∧ (¬SUV(s) ∨ Quality(E)>Quality(s))
Rename vars; turn into causes:
(i) **{Economic(E)}**
(ii) **{Car(E)}**
(iii) **{¬SUV(s'), Quality(E)>Quality(s')}**

So, no, they are not identical. In particular, the first form doesn't entail the existence of an economy car, which the second one does.

Name: ___________________________________

Problem 4 --- Situation calculus & STRIPS -- 15 points

Consider the following action, represented in the "STRIPS" style:

Operator: **Sign(d,p)**
Add: **Signed(d)**
Del: **Unsigned(d)**
Pre: **Document(d) $\wedge$ Unsigned(d) $\wedge$ Pen(p) $\wedge$ Have(p)**

That is, we can sign a document if it is previously unsigned, and if we have a pen.

A) Using the *situation calculus* style of representation, encode the above information using *successor-state axioms*. Assume there are no other relevant operators. (Hint: You'll need two axioms.) In formulating your answer, you may either give each predicate an additional argument for a situation, or use the function **Holds** to relate an action type function to a situation. (10 points)

$$\forall a,d,p,s\ \text{Signed}(d,\text{Result}(a,s)) \equiv (\text{Signed}(d,s) \vee (\text{Unsigned}(d,s) \wedge \text{Document}(d,s) \wedge \text{Pen}(p,s) \wedge \text{Have}(p,s) \wedge a{=}\text{Sign}(d,p)))$$

$$\forall a,d,p,s\ \text{Unsigned}(d,\text{Result}(a,s)) \equiv (\text{Unsigned}(d,s) \wedge \neg(a{=}\text{Sign}(d,p) \wedge \text{Document}(d,s) \wedge \text{Pen}(p,s) \wedge \text{Have}(p,s)))$$

Note that, while there are two statements we have to make, both are simple, because once something is signed, it stays signed, and there is no way to make something unsigned.

B) Assuming that the planner starts with a goal and regresses backward, as per the planner in our assignment, the use of the above operator might lead to the generation of foolish subgoals. What subgoals might our planner propose that seem foolish? How might the planner, and this particular plan, be altered so as to avoid generating such goals? (5 points)

There is nothing to prevent **Document(d)** or **Unsigned(d)** or **Pen(d)** from becoming a subgoal, but we are unlikely to have plans for such goals. (Indeed, we *know* we don't have plans for Unsigned, since we explicitly stated there were no other relevant operators.) Instead, we should make such predications *filter* conditions, and have our planner check for these as per the assignment.

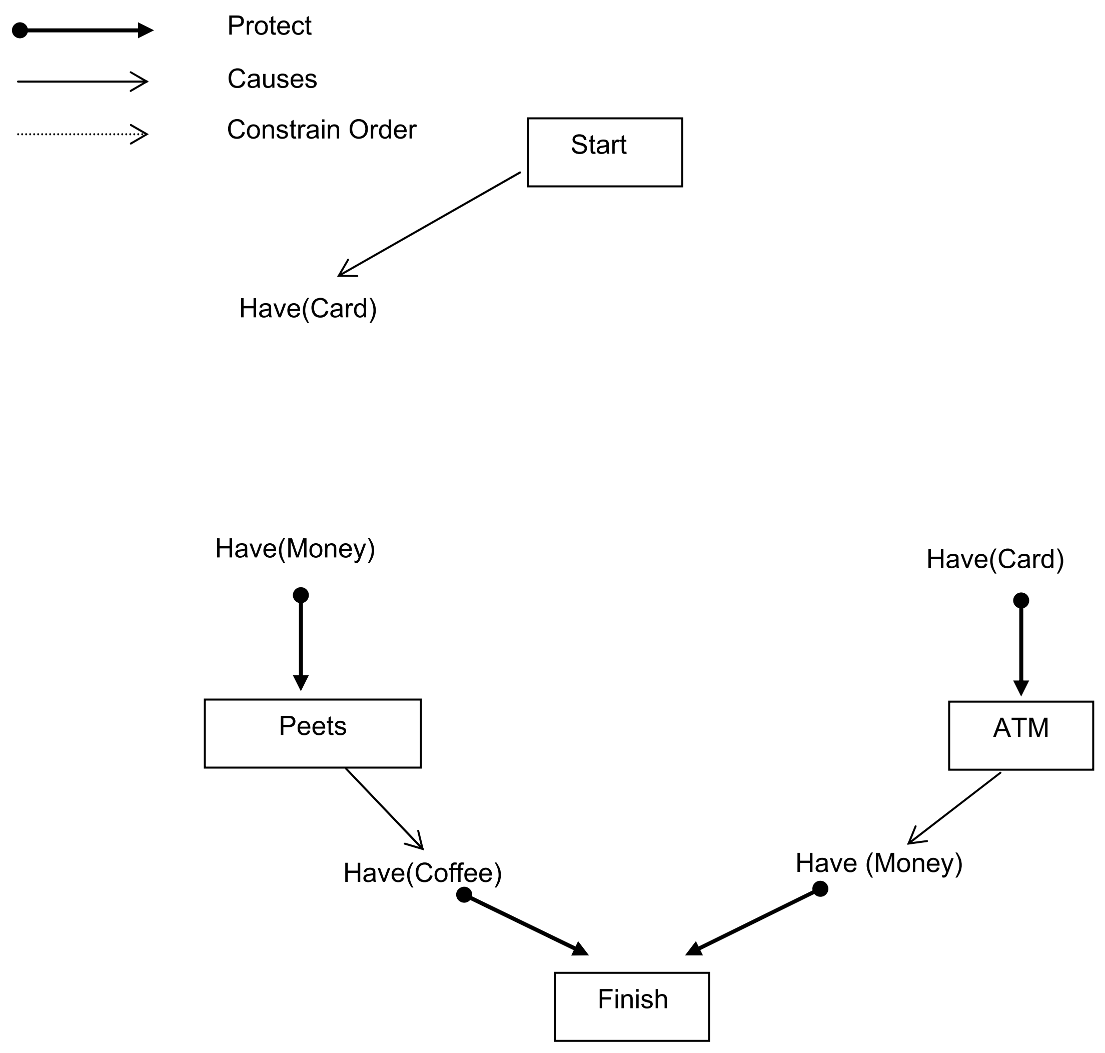
Name: ____________________________________

Problem 5 --- Partial-order Planning -- 25 points

Suppose we have the operators **ATM** and **Peets**. The **ATM** operator requires that you have your debit card, and gives you money (but, unfortunately, just enough money for one purchase at Peets). The operator **Peets** requires that you have money, takes that, and gives you some coffee.

Our goal is to have some coffee, and have some money.

Now, suppose we had constructed the following partial plan:



Name: ______________________________________

A) Using STRIPS-style notation, define the operator schemata **Peets** and **ATM**. (5 points)

Operator: **Peets()**

Add:	**Have(Coffee)**
Del:	**Have(Money)**
Pre:	**Have(Money)**

Operator: **ATM()**

Add:	**Have(Money)**
Del:	
Pre:	**Have(Card)**

B) Are there any preconditions in the partial plan that have yet to be planned for? If so, which are they? (5 points)

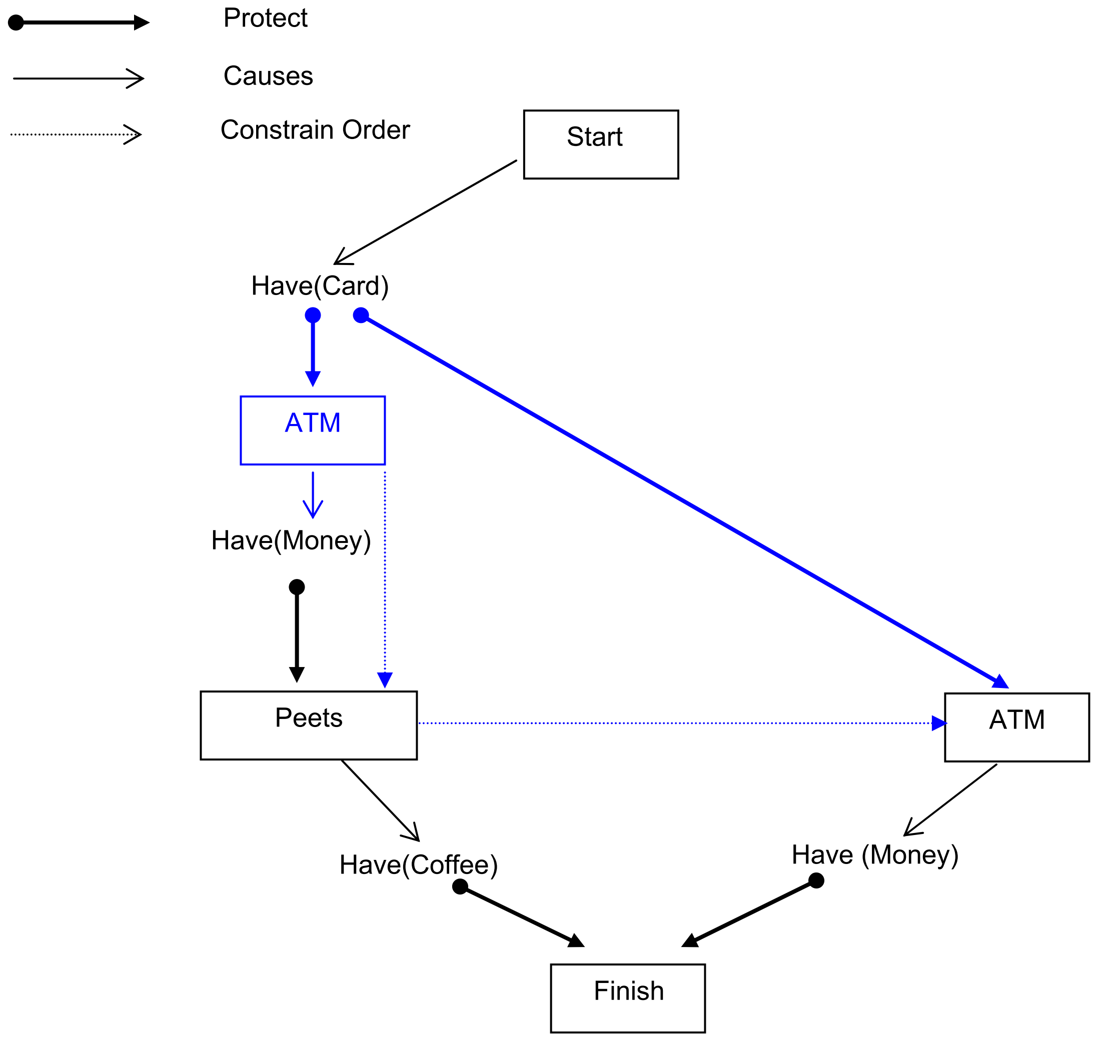
No plan steps are yet specified for the **Have(Money)** precondition of the Peets step, or the **Have(Card)** precondition of the ATM step.

C) Are there any unresolved potential conflicts in the partial plan? If so, what are they, and how, specifically, can they be resolved? (5 points)

There is nothing in the plan to prevent the **Peets** step from clobbering the **Have(Money)** state achieved by the **ATM** step. We can't *promote* **Peets** to after the **Finish**, but we can *demote* it to happen before **ATM**. So we constrain **Peets** to happen before **ATM**.

Name: ______________________________________

D) Adding whatever steps, causal and protection links, and temporal orderings you need to complete the plan, draw the complete plan below. (10 points)



We add another **ATM** step, which depends on having a card, established by **Start**, and which achieves the **Peets** step precondition of having money. The new **ATM** step should happen before the **Peets** step, so we also add a link constraining the order of these steps. We also make the precondition of the original **ATM** step be achieved by **Start**.

END OF EXAM