Name: ___________________________________

CS188 Intro. to AI
Fall, 2004 R. Wilensky

Midterm

- This is an open-book, open-notes, no-electronic-devices 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:	__________	(25)
Problem 2:	__________	(20)
Problem 3:	__________	(20)
Problem 4:	__________	(35)
Total :	__________	(100)

Name: ___________________________________

Problem 1 --- True or False (20 points, 4 each)

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

A) Depth-first search with iterative deepening uses less storage than breadth-first search, but runs asymptotically as fast.

T. DFID, like DF, uses O(d) storage, as opposed to O(b^d) for bf; since most of the nodes searched are on the frontier of the tree, the run time is about the same.

B) Alpha-beta search generally enables searching to a greater depth because it has the effect of reducing the branching factor.

T. Pruning effectively reduces the branching factor (ideally to b^1/2), so the number of nodes that need to be explored at a given depth will be smaller, and hence one can search further (ideally, twice as far) in the same time.

C) Hierarchical planning is generally more efficient than non-hierarchical planning because the former eliminates any need to back up.

F. Back up may still occur, both proposing steps in a plan and in proposing expansions of abstract operators.

D) The Manhattan distance is an admissible heuristic for finding the minimal walking distance between points in Manhattan, where some streets cut across on a diagonal.

F. The distance along a diagonal will be shorter than the Manhattan distance, so this heuristic would not always be optimistic.

E) In general, constraint satisfaction algorithms will be a better choice than path finding algorithms for problems in which the order of operations is not important.

T. Path finding algorithms may try all alternative orders of possibilities from a situation doomed to failure, which CS algorithms can often avoid.

Name: ______________________________________

Problem 2 --- Predicate Calculus (25 points)

These sentences are meant to describe the “n queens” problem, i.e., the problem of placing queens on each row of an $n \times n$ board so that no two can capture each other. (Recall that queens can capture each other if they are on the same row, column, or diagonal.)

(i) $\forall\, i\ (i \geq 1 \wedge i \leq N) \rightarrow \exists j\ S(i,j)$

(ii) $\forall\, i,j\ (i \leq 0 \vee j \leq 0 \vee i > N \vee j > N) \rightarrow \neg S(i,j)$

(iii) $\forall\, i,j,k\ \ S(i,j) \wedge S(i,k) \rightarrow j=k$

(iv) $\forall\, i,j,k\ \ S(j,i) \wedge S(k,i) \rightarrow j=k$

(v) $\forall\, i,j,k\ (S(i,j) \wedge (S(i+k,j+k) \vee S(i-k,j-k) \vee S(i+k,j-k) \vee S(i-k,j+k)) \rightarrow k=0)$

(a) (10 points) In English, state what the predicate **S** must mean, and what the sentences i-v each state, if indeed these sentences are to describe the n queens problem.

S(i,j) means that the square at row i, column j, has a queen in it.

(i) Every row has a queen on it, between columns 1 and n.
(ii) There aren’t any queens outside the bounds of the “board”.
(iii) Only one queen in each row.
(iv) Only one queen in each column.
(v) Only one queen on the diagonal.

(b) (10 points) State whether this set of sentences is either *valid*, *satisfiable but not valid*, or *unsatisfiable*, and explain why.

Since a model can supply any interpretation for any of the symbols, not to mention any domain, the set of sentences clearly isn’t valid. (As an example, even with the domain of objects being the natural numbers, the sentence wouldn’t be true for a board with no queens on it, or for any board in the case of N being 2, as we can see below.)

On the other hand, we know there are solutions to the 8 queens problem, so the sentences aren’t unsatisfiable. (Note also that one queen on a 1x1 board is also a model.)

Therefore, the set of sentences is satisfiable, but not valid.

Name: ______________________________________

Problem 3 --- Propositional Calculus (20 points)

Consider the “2 queens” problem, i.e., putting a queen in each row of a 2x2 board so that no two queens can attack each other.

A) (10 points) Represent the 2 queens problem in propositional calculus, using the propositional symbols **IJ** to represent the proposition “A queen is on row i, column j”. (E.g., ¬12 would mean that there is no queen at row 1, column 2.) I.e., write a sentence, in conjunctive normal form, which expresses the goal of the problem along with all the constraints.

```
  11 ∨ 12              ; a queen on row 1
∧ 21 ∨ 22              ; a queen on row 2
∧ ¬11 ∨ ¬12      ; only one queen on row 1
∧ ¬21 ∨ ¬22      ; only one queen on row 2
∧ ¬11 ∨ ¬21      ; only one queen on column 1
∧ ¬12 ∨ ¬22      ; only one queen on column 2
∧ (11 (  (22      ; only one queen on one diagonal
(  (21 (  (12     ; only one queen on the other diagonal
```

(Note: You might say ((11 (12), etc., to intuitively express a constraint, but must convert to CNF.)

□B) (10 points) Use resolution to show that there cannot be a queen at row 1, column 1. That is, starting from the clauses in A, use resolution to prove (11.

$$\begin{tikzcd}
11 \lor 12 \arrow[dr] & & \neg 21 \lor \neg 12 \arrow[dl] & & & \\
& 11 \lor \neg 21 \arrow[dr] & & \neg 11 \lor \neg 21 \arrow[dl] & & \\
& & \neg 21 \arrow[dr] & & 21 \lor 22 \arrow[dl] & \\
& & & 22 \arrow[dr] & & \neg 11 \lor \neg 22 \arrow[dl] \\
& & & & \neg 11 &
\end{tikzcd}$$

Or, if you assumed 11:

$$\begin{tikzcd}
11 \arrow[dr] & & \neg 11 \lor \neg 21 \arrow[dl] & & & \\
& \neg 21 \arrow[dr] & & 21 \lor 22 \arrow[dl] & & \\
& & 22 \arrow[dr] & & \neg 11 \lor \neg 22 \arrow[dl] & \\
& & & \neg 11 \arrow[dr] & & 11 \arrow[dl] \\
& & & & \square &
\end{tikzcd}$$

Name: ______________________________________

Problem 4 --- Planning (35 points)

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

Operator:	**Pick-up(x,y)**	**Put-down(x,y)**
Add:	**Have(Robot,x)**	**In(x,y)**
Del:	**In(x,y)**	**Have(Robot,x)**
Pre:	**At(Robot,y) ∧ In(x,y)**	**At(Robot,y) ∧ Have(Robot,x)**

Operator:	**Move(x,y)**	**Make-widget(x,y)**
Add:	**At(Robot,y)**	**Widget(x)**
Del:	**At(Robot,x)**	**Protow(x)**
Pre:	**At(Robot,x)**	**At(Robot,y) ∧ In(x,y) ∧ Protow(x) ∧ Worktray(y)**

A) (10 points) Write a *successor-state axiom* for the predicate **Have**, assuming that there are no other relevant operators. In formulating your answer, you may either give each predicate an additional argument for a situation, or use the function **Holds**, and make **Have**, etc., functions.

A satisfactory solution is the following:

$$\forall\, x,s,a \;\; \mathbf{Have(Robot,x,Result(a,s))} \equiv \mathbf{Have(Robot,x,s)} \wedge \neg \exists\, z\, \mathbf{(a = Put\text{-}down(x,z))}$$
$$\vee\; \exists\, y\, \mathbf{(At(Robot,y,s)} \wedge \mathbf{In(x,y,s)} \wedge \mathbf{a = Pick\text{-}up(x,y))}$$

(Technically, one should add the preconditions of Put-down to the first disjunct; I haven't been doing this in my examples of successor-state axioms, because they aren't necessary with the simple knowledge bases I was using, so we won't require them here either.)

The need for the existential quantifier in the second disjunct IS a bit tricky. We need this here because the operator introduces a parameter not in the predicate, and we need to state that the item was somewhere. Since this was tricky, we didn't take off much it you got the quantification wrong,

B) (5 points) A planner using the action descriptions above might end up exploring some patently silly plan possibilities for some goals, in some situations. How might such fruitless alternatives arrive, and how might we modify our action descriptions (and planner) to prevent such silliness?

The precondition of **Make-Widget** uses the predicates **Protow** and **Worktray**, neither of which can be effected by any plans. Therefore, the planner may set these up as subgoals, and then fail to achieve them. Making the conjuncts using these predicates *filter conditions* will avoid this problem.

(Continued on next page.)

Name: ______________________________________

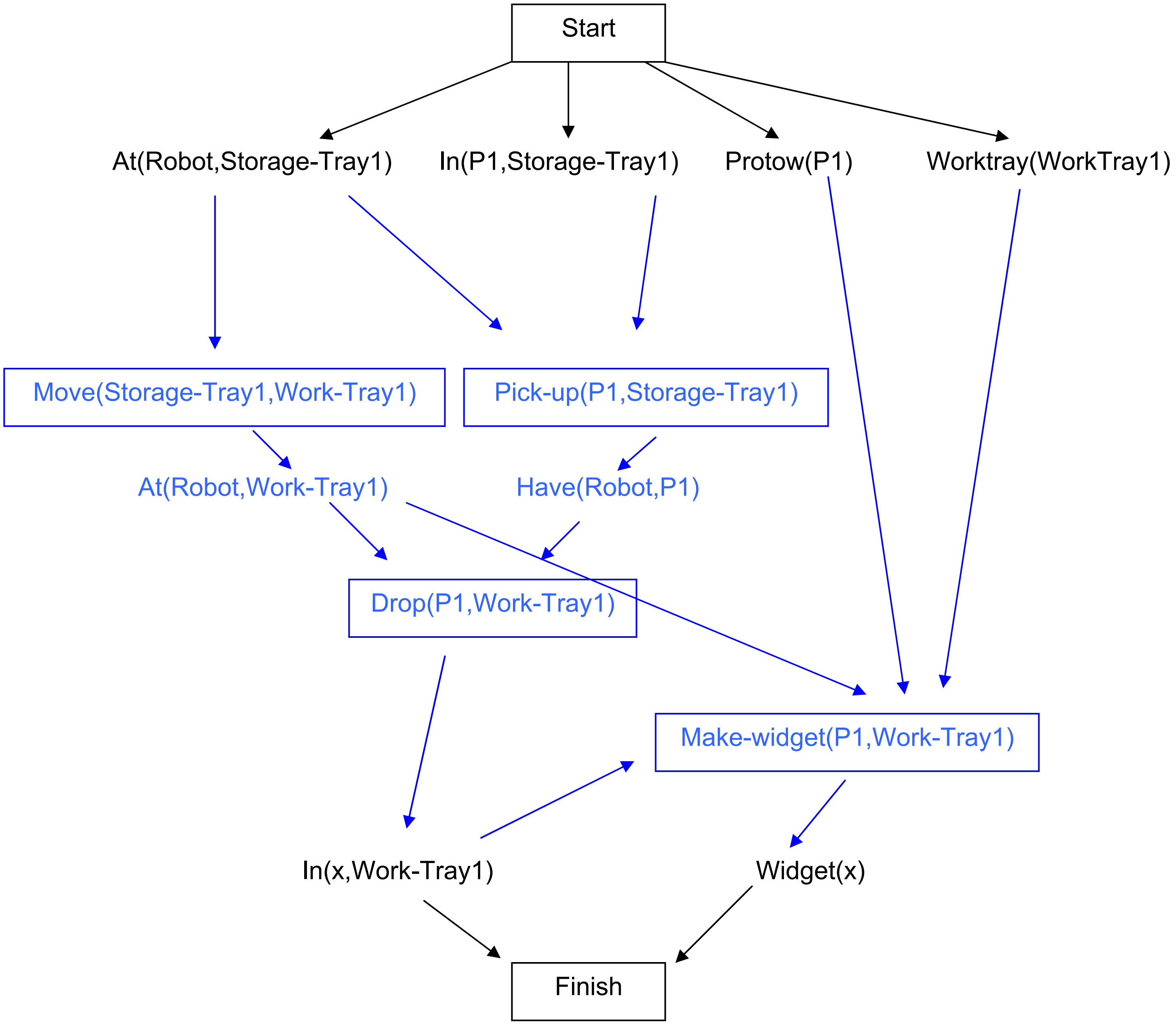
C) (15 points) Assume that the initial state of the world is

Protow(P1) ∧ In(P1,StorageTray1) ∧ At(Robot, StorageTray1) ∧ Worktray(WorkTray1)

and that our goal is

In(x,WorkTray1) ∧ Widget(x)

Below is some of a diagram for the plan for this goal and initial state that a partial-order planning algorithm would produce. (It uses plain arrows both to show which actions result in which states, and which states meet preconditions for which actions.) Add to this diagram all the steps needed for this plan, along with their results and preconditions).



Name: ___________________________________

(d) (5 points) Describe, in English, *one* conflict present in your plan, and how it can be resolved (if indeed it can be).

Moving from the storage tray to the work tray conflicts with picking up the proto-widget from the storage tray. The conflict can be resolved by promotion, i.e., by requiring that Move step comes after the Pick-up step.

This is in fact the only conflict in the plan. Note that the order in which preconditions are fulfilled makes the rest of the plan completely linear.