CS322 Fall 1997
Practice Final Questions

Here are some questions for the final exam.

The following chapters/sections will be covered: Chapter 1, Chapter 2, Chapter 3 (except Section 3.7), Chapter 4, Chapter 5, Chapter 6 (except 6.7 and 6.8), Sections 7.2, 7.3, 7.4, Chapter 8 (excluding the event calculus and partial-order planning), Sections 11.1 and 11.2.

This practice exam is designed to give you some impression of the types of questions that may be asked. You should also expect some questions based on the assignments. **You should also reread the practice midterm. The final exam covers the whole course, but this practice exam emphasizes the latter part of the course.**

Note that you can bring in one sheet of $8\frac{1}{2} \times 11$ paper, with anything you like written on it. **You must bring your student card to the final examination.** Your student card will be checked.

Question 1

Given the object-level knowledge base *KB* containing the clauses:

$$
\begin{aligned}
&\mathit{false} \Leftarrow a \& b.\\
&\mathit{false} \Leftarrow c \& d.\\
&a \Leftarrow p.\\
&b \Leftarrow r.\\
&b \Leftarrow e.\\
&e \Leftarrow p.\\
&c \Leftarrow f \& r.\\
&f \Leftarrow g \& p.\\
&f \Leftarrow q \& s.\\
&f \Leftarrow u.
\end{aligned}
$$

Suppose that the assumables are p, q, r, s, t and u.

1. What are all of the minimal conflicts? You will lose marks for omitting minimal conflicts and for giving things that are not minimal conflicts.
2. Give an SLD-resolution showing how how a delaying meta-interpreter can be used to find one of these minimal conflicts.

Extra question: suppose d were added as an object level fact. Show how this can be done. What are all of the minimal conflicts after d is added?

Answer to Question 1, part 1

What are all of the minimal conflicts?

There is one minimal conflict: $\{p\}$.

Note there are other conflicts, such as $\{p, r\}$ or even $\{p, s, t, u\}$, but these are not **minimal** conflicts.

Answer to Question 1, part 2

Give an SLD-resolution showing how how a delaying meta-interpreter can be used to find one of these minimal conflicts.

Consider the delaying meta-interpreter where $dprove(G, D_0, D_1)$ is true if D_0 is an ending of D_1 and G logically follows from the conjunction of the assumable atoms in D_1.

$dprove(true, D, D).$
$dprove((A\&B), D_1, D_3) \leftarrow$
$\quad dprove(A, D_1, D_2) \wedge$
$\quad dprove(B, D_2, D_3).$
$dprove(G, D, [G|D]) \leftarrow$
$\quad assumable(G) \wedge$
$\quad notin(G, D).$
$dprove(G, D, D) \leftarrow$
$\quad assumable(G) \wedge$
$\quad member(G, D).$
$dprove(H, D_1, D_2) \leftarrow$
$\quad (H \Leftarrow B) \wedge$
$\quad dprove(B, D_1, D_2).$

The following shows an SLD resolution to find conflicts:

$yes(D) \leftarrow dprove(false, [], D).$
$yes(D) \leftarrow (false \Leftarrow B_1) \wedge dprove(B_1, [], D).$
$yes(D) \leftarrow dprove((a\&b), [], D).$
$yes(D) \leftarrow dprove(a, [], D_1) \wedge dprove(b, D_1, D).$
$yes(D) \leftarrow (a \Leftarrow B_2) \wedge dprove(B_2, [], D_1) \wedge dprove(b, D_1, D).$
$yes(D) \leftarrow dprove(p, [], D_1) \wedge dprove(b, D_1, D).$
$yes(D) \leftarrow dprove(p, [], D_1) \wedge dprove(b, D_1, D).$
$yes(D) \leftarrow assumable(p) \wedge notin(p, []) \wedge dprove(b, [p], D).$
$yes(D) \leftarrow notin(p, []) \wedge dprove(b, [p], D).$

$yes(D) \leftarrow dprove(b, [p], D).$
$yes(D) \leftarrow (b \Leftarrow B_3) \wedge dprove(B_3, [p], D).$
$yes(D) \leftarrow dprove(e, [p], D).$
$yes(D) \leftarrow (e \Leftarrow B_4) \wedge dprove(B_4, [p], D).$
$yes(D) \leftarrow dprove(p, [p], D).$
$yes([p]) \leftarrow assumable(p) \wedge member(p, [p]).$
$yes([p]) \leftarrow member(p, [p]).$
$yes([p]) \leftarrow .$

Answer to question 1, extra question

Extra question: suppose d were added as an object level fact. Show how this can be done. What are all of the minimal conflicts after d is added?

d is added as an object-level fact by adding:

$d \Leftarrow true.$

With this clause added, there are 3 minimal conflicts: $\{p\}$, $\{q, s, r\}$, and $\{u, r\}$.

Note again that there are other conflicts, such as $\{g, p, r\}$, but these are not minimal, as there is a subset that is also a conflict.

Question 2

Consider the data on 4 Boolean attributes a, b, c, and d, where d is the target classification.

	a	b	c	d
e_1	true	true	false	false
e_2	false	true	false	true
e_3	false	true	true	true
e_4	false	false	true	false
e_5	true	false	false	false

In this question we will consider decision-tree learning based on this data.

1. What is a good attribute to split on first? Explain why.
2. Draw a decision tree that the top-down myopic decision tree learning algorithm could build. For each node (including the leaves) show which examples are used to determine the classification at that node. (The root note of the tree will be labelled with the list of all of the examples).
3. Explain how the learning bias inherent in learning decision-trees can be used to classify unseen instances. Give an instance that is not in the training data, show how the above tree classifies that instance. Justify why this is an appropriate classification.

Answer to Question 2, part 1

What is a good attribute to split on first? Explain why.

a is a good attribute to split on, as when a is true, all of the examples agree on the value of attribute d, and when a is false the examples are skewed.

b is also a good attribute to split on, as it divides the examples into the same ratios as a.

c is a worse attribute to split on, as knowing the value of c doesn't help us much in determining the value of d.

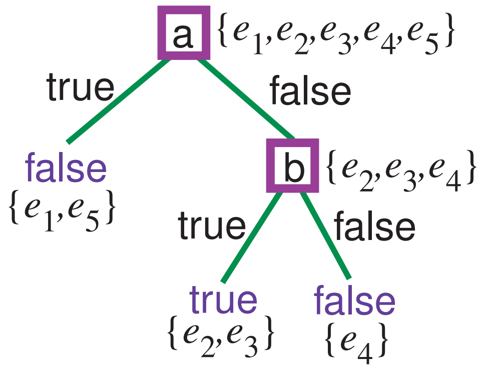
Solution to Question 2, part 2

Draw a decision tree that the top-down myopic decision tree learning algorithm could build. For each node (including the leaves) show which examples are used to determine the classification at that node.

Suppose you split on attribute a first. There are 2 examples with attribute a true, namely e_1 and e_5, and these examples agree on the value of d. There are 3 examples with attribute a false, namely e_2, e_3, and e_4, these don't agree on the value of d.

Next you have to choose an attribute to split the examples $\{e_2, e_3, e_4\}$. b is a good choice as the examples with b true all agree on the value of d and the examples with b false all agree on the value of d.

Here is the resultant decision tree showing which examples are used at each node:



Solution to Question 2, part 3

Explain how the learning bias inherent in learning decision-trees can be used to classify unseen instances. Give an instance that is not in the training data, show how the above tree classifies that instance. Justify why this is an appropriate classification.

From the 5 assignments of values to a, b and c in the training set, out of the 8 possible assignments, the decision tree makes predictions in all 8 cases. The ability to make predictions in unseen cases is the bias of the learning algorithm. The unseen cases are:

a	b	c	prediction for d
true	true	true	false
true	false	true	false
false	false	false	false

The bias is that c is always irrelevant to the classification, and that when a is true, b is also irrelevant. This is because c always seemed to be irrelevant in the training example. Similarly b was irrelevant when a was true.

Question 3

Suppose that you are new in a job, and that your boss thinks that, because you did so well in cs322, you are an expert in AI. You want to impress your boss (and don't want to say "That course was really dumb. We didn't learn anything."). The boss has heard of neural networks and decision trees, but doesn't know anything about them, and thinks that neural nets seem so great and decision trees so stupid (or the other way around). The boss wants your informed opinion. (Your informed opinion should be based on facts, which you should make explicit). You need to write a coherent, well written half a page executive summary (using proper English) to impress the boss.

Solution to Question 3

There is no correct answer to such a question. You have to give a sensible answer that is arguably reasonable.

The sort of answer I would like is that neither is more stupid than the other. They each embody a different learning bias. Which is a better bias depends on the domain. There are domains where decision trees work better and domains where neural networks work better. Tell your boss that blindly believing one is better will lead to some situations where you learn a representation than is much worse than possible.

Question 4

Suppose that someone has suggested using using A^* search for neural network learning. Is this feasible? Either say how it could work, or give three reasons why A^* search is not suitable for neural network learning. You need to use full sentences in your answer.

Solution to Question 3

A^* is used for path planning problems where the aim is to find the shortest path to a goal (the length of a path is the sum of the lengths of the arcs) and where there is heuristic information in terms of an estimate of the distance from a node to a goal node.

In neural network learning, the aim is to assign values to parameters that minimize the error. To use $A*$ this problem has to be converted to a path planning problem.

The way presented in the text to convert a constraint satisfaction problem to a search problem, namely where nodes correspond to assignments of values to (some of the) variables, and the neighbors correspond to an assignment of values to another variable, doesn't work because

- There is no value until all of the variables have been assigned values. (I.e., there are no arc costs except for the last arc).
- There is no heuristic function.
- Even if there was, the search space is too big for systematic search.

Question 5

Suppose that you have the action for the robot to spray everything at its location with paint of a certain color, with the result that everything at its location becomes that color.

1. How can such an action be defined in the situation calculus?
2. How can such an action be defined in STRIPS?

Answer to Question 5 part 1

Let's assume that the action $paint\_everything(Robot, Color)$ is always possible.

$$
\begin{array}{l}
color(Obj, Color, do(paint\_everything(Robot, Color), S)) \leftarrow \\
\quad at(Robot, Loc, S) \wedge \\
\quad at(Obj, Loc, S).color(Obj, C1, do(paint\_everything(Robot, Color), S)) \leftarrow \\
\quad at(Robot, Loc, S) \wedge \\
\quad at(Obj, Loc1, S) \wedge \\
\quad Loc \neq Loc1 \wedge \\
\quad color(Obj, C1, S). \\
color(Obj, C, do(A, S)) \leftarrow \\
\quad poss(A, S) \wedge \\
\quad \sim paint\_everything\_action(A) \wedge \\
\quad color(Obj, C, S). \\
paint\_everything\_action(paint\_everything(Robot, Color)). \\
poss(paint\_everything(Robot, Color), S).
\end{array}
$$

Answer to Question 5 part 1

This action cannot be defined in strips. This is because there isn't an appropriate addlist. What you would need is something like:

The addlist of $paint\_everything(Robot, Color)$ is

$$\{color(Obj, Color) : \exists P\ at(Robot, P) \wedge at(Obj, P)\}.$$

This is more complicated than is allowed in STRIPS (we need the addlist of a state to depend on the state).

Question 6

Suppose you have a base-level knowledge base defined in terms of

- the meta-level predicate symbol $\Leftarrow$, such that $H \Leftarrow B$ is true if $H \leftarrow B$ is in the base-level knowledge base, and

- the relation $primitive(A)$ that is true if atom A is primitive.

An example knowledge base is

$$
\begin{aligned}
&at(Ag, Pos) \Leftarrow sitting\_at(Ag, Pos).\\
&adjacent(P_1, P_2) \Leftarrow\\
&\quad between(Door, P_1, P_2)\&\\
&\quad unlocked(Door).\\
&between(door1, o103, lab2) \Leftarrow true.\\
&primitive(unlocked(D)).\\
&primitive(sitting\_at(A, P)).\\
&primitive(carrying(A, O)).
\end{aligned}
$$

Write a meta-interpreter $reduce\_prove(L0, L1)$ that, given a list $L0$ of atoms, reduces them to a list $L1$ of primitive atoms that, given the knowledge base, implies the elements of $L0$.

For example, the query

ask $reduce\_prove([at(rob, P), adjacent(P, lab2), carrying(rob, parcel)], L)$.

returns with one answer with $L = [carrying(rob, parcel), unlocked(door1), sitting_at(rob, o103)]$. Note that this operation was assumed in the regression planner.

Solution to Question 6

$$
\begin{aligned}
&reduce\_prove(L_0, L_1) \leftarrow\\
&\quad reduce\_prove\_acc(L_0, [], L_1).
\end{aligned}
$$

$reduce\_prove\_acc(L_0, A_0, A_1)$ is true if starting with primitives A_0 also proving the elements of L_0 results in primitives A_1.

$$
\begin{aligned}
&reduce\_prove\_acc([], L, L).\\
&reduce\_prove\_acc([true|R], A_0, A_1) \leftarrow\\
&\quad reduce\_prove\_acc(R, A_0, A_1).\\
&reduce\_prove\_acc([P|R], A_0, A_1) \leftarrow\\
&\quad primitive(P) \wedge\\
&\quad reduce\_prove\_acc(R, [P|A_0], A_1).\\
&reduce\_prove\_acc([A\&B|R], A_0, A_1) \leftarrow\\
&\quad reduce\_prove\_acc([A, B|R], A_0, A_1).\\
&reduce\_prove\_acc([H|R], A_0, A_1) \leftarrow\\
&\quad (H \Leftarrow B) \wedge\\
&\quad reduce\_prove\_acc([B|R], A_0, A_1).
\end{aligned}
$$

Question 7

Suppose a regression planner was used with the goal:

$$[\mathit{sitting\_at}(\mathit{rob}, o109), \mathit{sitting\_at}(\mathit{parcel}, \mathit{lab}2)]$$

- Draw the first-level of the search tree showing the possible actions and the weakest preconditions to achieve this goal.
- Show one element on the next level of the search tree. (That is one action to achieve one of the goals you constructed in the preceding part, and the corresponding weakest precondition.)
- What happens when the naive STRIPS planner is used with this goal? (A qualitative description of what will happen is what is needed).
- Describe qualitatively what happens when STRIPS is used so that it protects subgoals.

Solution to Question 7, part 1

The neighbors of $[\mathit{sitting\_at}(\mathit{rob}, o109), \mathit{sitting\_at}(\mathit{parcel}, \mathit{lab}2)]$ in a regression search tree are (assuming that we are not reducing to primitive goals):

- $[\mathit{autonomous}(\mathit{rob}), \mathit{sitting\_at}(\mathit{rob}, L), \mathit{adjacent}(L, o109), \mathit{sitting\_at}(\mathit{parcel}, \mathit{lab}2)]$ using the action $\mathit{move}(\mathit{rob}, L, o109)$.
- $[\mathit{sitting\_at}(\mathit{rob}, o109), \mathit{autonomous}(\mathit{parcel}), \mathit{sitting\_at}(\mathit{parcel}, L), \mathit{adjacent}(L, \mathit{lab}2)]$ using the action $\mathit{move}(\mathit{parcel}, L, o1$
- $[\mathit{sitting\_at}(\mathit{Ag}, o109), \mathit{autonomous}(\mathit{Ag}), \mathit{carrying}(\mathit{Ag}, \mathit{rob}), \mathit{Ag} \neq \mathit{rob}, \mathit{sitting\_at}(\mathit{parcel}, \mathit{lab}2)]$ for the action $\mathit{putdown}(\mathit{Ag}, \mathit{rob})$.
- $[\mathit{sitting\_at}(\mathit{rob}, o109), \mathit{sitting\_at}(\mathit{Ag}, \mathit{lab}2), \mathit{autonomous}(\mathit{Ag}), \mathit{carrying}(\mathit{Ag}, \mathit{parcel}), \mathit{Ag} \neq \mathit{parcel}]$ for the action $\mathit{putdown}(\mathit{Ag}, \mathit{parcel})$.

If we reduce to primitive goals, we get the neighbors as:

- $[\mathit{sitting\_at}(\mathit{rob}, o103), \mathit{sitting\_at}(\mathit{parcel}, \mathit{lab}2)]$ using the action $\mathit{move}(\mathit{rob}, o103, o109)$.
- $[\mathit{sitting\_at}(\mathit{rob}, \mathit{storage}), \mathit{sitting\_at}(\mathit{parcel}, \mathit{lab}2)]$ using the action $\mathit{move}(\mathit{rob}, \mathit{storage}, o109)$.
- $[\mathit{sitting\_at}(\mathit{rob}, o109), \mathit{sitting\_at}(\mathit{rob}, \mathit{lab}2), \mathit{carrying}(\mathit{rob}, \mathit{parcel})]$ for the action $\mathit{putdown}(\mathit{rob}, \mathit{parcel})$.

(The last one could be omitted as it is impossible).

Solution to Question 7, part 2

A neighbor of $[\mathit{sitting\_at}(\mathit{rob}, o103), \mathit{sitting\_at}(\mathit{parcel}, \mathit{lab}2)]$ is $[\mathit{sitting\_at}(\mathit{rob}, \mathit{lab}2), \mathit{sitting\_at}(\mathit{parcel}, \mathit{lab}2)]$ using the action $\mathit{move}(\mathit{rob}, \mathit{lab}2, o103)$.

Solution to Question 7, part 3

What happens when the naive STRIPS planner is used with this goal?

It will try to solve $sitting\_at(rob, o109)$, achieve this immediately, it will get the parcel into lab2, and then return, but the goal will not be satisfied.

Solution to Question 7, part 4

Describe qualitatively what happens when STRIPS is used so that it protects subgoals.

If it protects subgoals, it will choose a subgoal to achieve first, say $sitting\_at(rob, o109)$, solve it trivially, and try to solve $sitting\_at(parcel, lab2)$ (while protecting the achieved subgoal). It will find quickly that it cannot solve this subgoal without undoing the protected subgoal. So then it will choose another subgoal to achieve first, and try to achieve $sitting\_at(parcel, lab2)$, then protect this subgoal and try to achieve $sitting\_at(rob, o109)$, and find a solution.