

EXAMINATIONS — 2005

END-YEAR

<p>COMP 307</p> <p>INTRODUCTION TO</p> <p>ARTIFICIAL INTELLIGENCE</p>
--

Time Allowed: 3 Hours

Instructions: There are a total of 180 marks on this exam.
Attempt all questions.
Calculators may be used.
Non-electronic foreign language translation dictionaries may be used.

Questions

- | | |
|--------------------------------|------|
| 1. Prolog | [17] |
| 2. Prolog: Lists | [10] |
| 3. Search | [27] |
| 4. Rule Based Systems | [22] |
| 5. Machine Learning | [44] |
| 6. Planning | [40] |
| 7. Natural Language Processing | [20] |

Question 1. Prolog

[17 marks]

Consider the following program.

```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

female(pam).
female(liz).
female(ann).
female(pat).

male(tom).
male(bob).
male(jim).

offspring(Y, X):- parent(X, Y).
mother(X, Y):- parent(X, Y), female(X).
grandparent(X, Z):- parent(X, Y), parent(Y, Z).

ancestor(X, Z):- parent(X, Z).
ancestor(X, Z):- parent(X, Y), ancestor(Y, Z).
```

(a) [8 marks] Write the output generated by the following queries. **Give all the possible answers in the order they are generated.**

```
|?- grandparent(pam, X).
```

```
| X = ann ? ;  
| X = pat ? ;  
| no
```

```
|?- ancestor(pam, X).
```

```
| Y = bob ? ;  
| Y = ann ? ;  
| Y = pat ? ;  
| Y = jim ? ;  
| no
```

(Question 1 continued on next page)

(Question 1 continued)

(b) [4 marks] Define the following relation:

`sister(X, Y)` where X is a sister of Y.

```
sister(X, Y):-  
parent(Z, X),  
parent(Z, Y),  
female(X),  
X≠Y.
```

(c) [5 marks] Define the following relation:

`relatives(X, Y)` where X is a relative of Y.

Two persons are relatives if

- one is a ancestor of the other, or
- they have a common ancestor, or
- they have a common descendant. (X is a descendant of Y if Y is an ancestor of X.)

Question 2. Prolog: Lists

[10 marks]

Define the following relation:

```
remove_duplicates(+List, ?Pruned)
```

Pruned is the result of removing all identical duplicate elements in List. For example:

```
| ?- remove_duplicates([1,2,3,2,3,1], P).
```

```
P = [1,2,3]
```

Do not use any predicates from the list library. For example, you must define your own versions of member, append, etc. if you want them.

Question 3. Search

[27 marks]

Parts (a) and (b) refer to following problem:

A farmer with his dog, rabbit and lettuce come to the east side of a river they wish to cross. There is a boat at the river's edge, but of course only the farmer can row. The boat can only hold two things (including the rower) at any one time. If the dog is ever left alone (without the farmer) with the rabbit, the dog will eat it. Similarly if the rabbit is ever left alone with the lettuce, the rabbit will eat it. How can the farmer get across the river so that the dog, rabbit and lettuce arrive safely on the other side?

(a) [7 marks] Formalise the above problem in terms of state-space search by

- Suggesting a suitable representation for the problem state.
- Expressing the initial and final states in this representation.

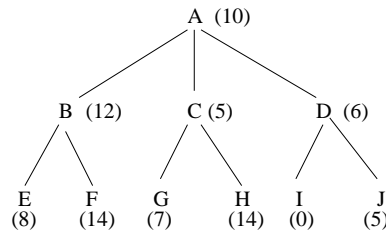
Student ID:

(b) [10 marks] Using your state representation from (a), write Prolog code to represent the available operators for getting from one state to the next, including any constraints to prevent anything being eaten. Your Prolog code should work with the depth-first search algorithm used in the 3P&3C 's program.

(Question 3 continued on next page)

(Question 3 continued)

(c) [10 marks] Given the following search tree, state the order in which the nodes will be expanded for the following search strategies. You may assume that the search finishes when a solution is reached. State any additional assumptions you made.



Breadth first search:

Depth first search:

Iterative deepening search:

Greedy best first search: Assume the numbers on the nodes indicate the estimated cost to solution.

Hill climbing search: Assume the numbers on the nodes indicate the values of the nodes, where smaller numbers are better.

Student ID:

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 4. Rule Based Systems

[22 marks]

Consider the following set of rules.

R1:: if Y and D then Z.

R2:: if X and B and E then Y.

R3:: if A then X.

R4:: if C then L.

R5:: if L and M then N.

Z is the goal to be established. A, B, C, D, E are the questions that can be asked of the user. Assume the rules are scanned top down, that is, a rule with a smaller number has higher priority.

(a) [5 marks] Backward chaining

Suppose a backward chaining interpreter will ask the user whenever it reaches a fact that can be asked, and will stop with the first goal that it discovers to be true. The answers for A, B, C, D, E are yes.

Show how the backward chaining interpreter will use the rules given above to reach a conclusion.

(b) [5 marks] Forward chaining

A, B, C, D, E are true and are saved in the working memory. Assume that every time the forward chaining interpreter adds a fact to the working memory, it scans the rules from the beginning again.

Show how a forward chaining interpreter will use the rules to reach a conclusion.

(c) [3 marks] To design an expert system for a specific domain, how do you choose between forward and backward chaining?

(d) [2 marks] State one way to combine forward and backward chaining.

(e) [7 marks] What are the advantages and disadvantages of rule-based expert systems?

Question 5. Machine Learning

[44 marks]

(a) [3 marks] The Naïve Bayes algorithm uses Bayes' rule to compute the probability of a class C given data D :

$$p(C|D) = \frac{p(D|C)p(C)}{p(D)}$$

If the data is described by a vector of n attributes (A_1, A_2, \dots, A_n) , state how Naïve Bayes approximates the value of $p(D|C)$, and state what assumption it is making about the data.

$$p(D|C) \approx p(A_1|C)p(A_2|C) \cdots p(A_n|C)$$

Assumption:

It assumes that the attributes are conditionally independent of each other, given the class

(b) [4 marks] The decision tree learning algorithm uses an impurity measure to choose between attributes. Explain why the $p(A)p(B)$ formula is a reasonable impurity measure for a set of instances belonging to two different classes A and B

$p(A)p(B)$ is 0 if all the instances are in one category, and is a maximum if the two categories are equal, which is when they are equally mixed, and it is smooth in between.

(c) [5 marks] An alternative impurity formula measures the difference between the number of instances in each category:

$$\frac{1}{(1 + (|A| - |B|))}$$

where $|A|$ and $|B|$ are the number of instances of category A and category B . Like the other formula, it is maximised when $|A| = |B|$ (equally mixed).

Explain why this is not a good impurity measure.

Although it has the property that it is a maximum (1) when $|A| = |B|$ (equally mixed), and is a minimum when $|A| = 0$ or $|B| = 0$ (pure), the minimum value also depends on the size of the set, so a small pure set (eg, 3:0) will have the same score as a large, almost equally mixed set (20:23). This measure will therefore not be good at comparing attributes that generate different sized sets of instances.

(Question 5 continued on next page)

(Question 5 continued)

(d) [10 marks] Consider the following data set describing 10 houses that a real estate agent has shown to a client. The client liked 4 of them but didn't like the other 6. The houses are described by three binary attributes.

Instance	Garden	Levels	Garage	Category
#1	grass	two	yes	Liked
#2	grass	one	yes	Liked
#3	flowers	one	yes	Liked
#4	flowers	two	yes	Liked
#5	flowers	one	no	disliked
#6	grass	one	yes	disliked
#7	grass	one	no	disliked
#8	grass	one	yes	disliked
#9	flowers	one	no	disliked
#10	flowers	two	no	disliked

The agent wants to build a decision tree to help identify other houses the client may like. Which attribute should the agent choose for the root of the decision tree if they use the impurity function $p(\text{Liked})p(\text{disliked})$. Show your working.

$$\text{garden: } 5/10 * (3/5 * 2/5) + 5/10 * (2/5 * 3/5) = 6/25 = 24\%$$

$$\text{levels: } 7/10 * (2/7 * 5/7) + 3/10 * (1/3 * 2/3) = 1/7 = 14\%$$

$$\text{garage: } 6/10 * (3/6 * 3/6) + 4/10 * (0/4 + 4/4) = 9/60 = 15\%$$

levels has the lowest score, therefore the algorithm will use

levels at the root

(e) [5 marks] Even if the data above were real data, the decision tree is unlikely to be useful in practice for predicting houses that the client likes. Justify this claim.

- (i) There are too few attributes: almost certainly there are other factors that are important to the client
- (ii) There are too few data points. With only 10 examples, the decision tree is likely to simply store the examples - ie, overfit the dataset

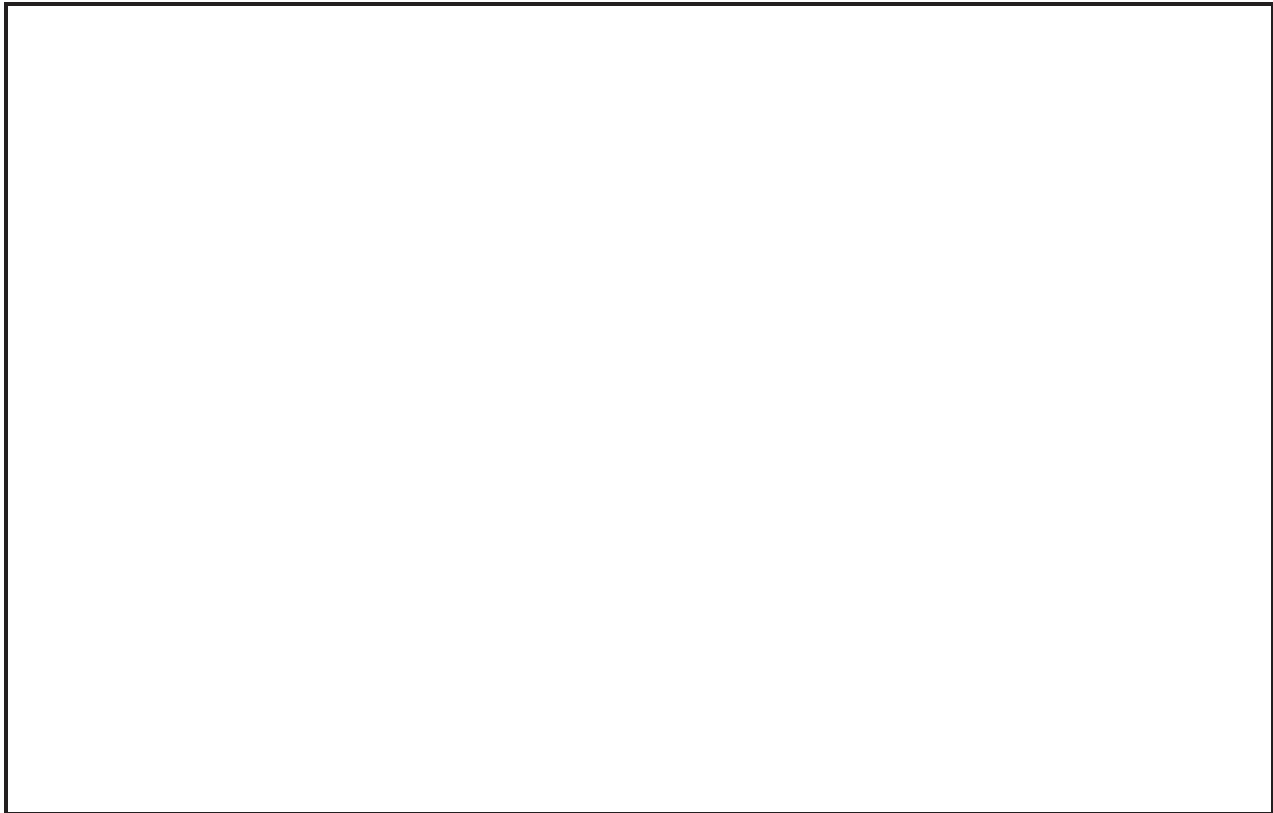
(Question 5 continued on next page)

(Question 5 continued)

(f) [8 marks] Show how a perceptron would learn a set of weights to distinguish the “Liked” houses from the “disliked” houses above. Starting with initial weights of all 0, show the weights it would learn if the following four instances were each presented once in the order: #1, #5, #4, #6.

For your convenience, the following table is a translation of the attributes of these instances into 0/1 feature values:

instance:	features	category
#1	1 0 1	+ve (Liked)
#5	0 1 0	-ve (disliked)
#2	0 0 1	+ve (Liked)
#6	1 1 1	-ve (disliked)



(g) [4 marks] Explain why the perceptron would never converge on the house data set above, no matter how many iterations were done.

The perceptron will only converge if the instances are linearly separable into the two categories
Since instances 2 and 6 have identical descriptions, but are in opposite categories, it is not possible to separate the instances (even by a non-linear) surface. Therefore, the perceptron will never converge.

(h) [5 marks] In the assignment to learn a perceptron to distinguish categories of simple images, we only had a training set of images, and we did not use a test set to evaluate the perceptron. Explain why you should use a test set, separate from the training set, to evaluate a learning algorithm. State what problems may arise if you evaluate the performance on the training set.

Question 6. Planning

[40 marks]

The following situation calculus rule describes part of the effect of pouring the water that is in one container into another container:

$$\text{container}(C1) \ \& \ \text{container}(C2) \ \& \ \text{water}(\text{Water}) \ \& \ \text{contains}(C1, \text{Water}, S) \ \& \ \text{empty}(C2, S) \\ \implies \text{contains}(C2, \text{Water}, \text{result}(\text{pour}(\text{Water}, C1, C2), S)).$$

As in prolog, the variables are capitalised.

(a) [3 marks] Explain the meaning of S and $\text{result}(\text{pour}(\text{Water}, C1, C2), S)$ in this rule.

S : | A variable representing a state

$\text{result}(\text{pour}(\text{Water}, C1, C2), S)$: | The new state after performing the action of
| pouring the water from the jug into the cup in state S

(b) [3 marks] The contains predicate is a ‘fluent’. Explain why it needs its third argument.

| Because it changes — $\text{contains}(A,B)$ might be true in one state and not
| in another. Therefore, it needs an argument specifying the state

(c) [4 marks] Suppose we assume that the `pour` action always transfers all the water from the first container to the second container. Write a second situation calculus rule that expresses this effect of the action.

| $\text{container}(C1) \ \& \ \text{container}(C2) \ \& \ \text{water}(\text{Water}) \ \& \ \text{contains}(C1, \text{Water}, S) \ \& \ \text{empty}(C2, S)$
| \implies
| $\neg \text{contains}(C1, \text{Water}, \text{result}(\text{pour}(\text{Water}, C1, C2), S)) \ \&$
| $\text{empty}(C1, \text{result}(\text{pour}(\text{Water}, C1, C2), S))$

(d) [4 marks] Express the Pour(Water, C1, C2) action as a STRIPS rule.

```
Pour(Water, C1, C2)
Preconditions: contains(C1, Water), empty(C2), container(C1), container(C2), wa-
ter(Water)
Deletes: contains(C1, Water), empty(C2)
Adds: contains(C2, Water), empty(C1)
```

(e) [5 marks] Suppose the only actions that can cause a container to contain something are the pour action above, and the following fillFromTap action:

```
fillFromTap(C, Water)
Preconditions: empty(C), inSink(C), container(C), water(Water)
Deletes: empty(C),
Adds: contains(C,Water)
```

Write a successor-state axiom for the contains fluent:

```
poss(A, S)  $\implies$ 
contains(C, W, result(A, S))  $\equiv$ 
A = pour(W, X, C) OR,
A = fillFromTap(C, W) OR,
contains(C, W, S) & A  $\neq$  pour(W, C, Y),
```

(Question 6 continued on next page)

(Question 6 continued)

Assume that a goal stack planner has the following STRIPS rules for actions to put shoes and socks on:

pullOn(Sock, Foot)

Preconditions: empty(Sock), bare(Foot), sock(Sock)

Deletes: empty(Sock), bare(Foot)

Adds: wearing(Sock, Foot)

putOn(Shoe, Foot)

Preconditions: empty(Shoe), noShoes(Foot), shoe(Shoe)

Deletes: empty(Shoe), noShoes(Foot), bare(Foot)

Adds: on(Shoe, Foot)

Suppose the planner is given the goal:

on(leftShoe, foot) & wearing(blackSock, foot)

and an initial state of

empty(blackSock), empty(leftShoe), bare(foot), noShoes(foot), sock(blackSock), shoe(leftShoe)

(f) [10 marks] Show the evolving state of the planner on the *facing page*, up to the point where the planner is stuck and has to backtrack. Assume that it will first consider subgoals in the order given. Make the order of the steps clear by numbering each change to the goalstack, the current state, and the current plan.

(g) [4 marks] Explain where the planner will backtrack to, and the next choice that it will consider. You do not need to show the plan that it constructs.

It will backtrack to the most recent choice point. It will backtrack to the most recent choice point.
The only choice point is the first step of choosing one of the subgoals. Therefore, it will undo everything and try to satisfy the wearing(blackSock, foot) subgoal first

Current Plan

Goal Stack

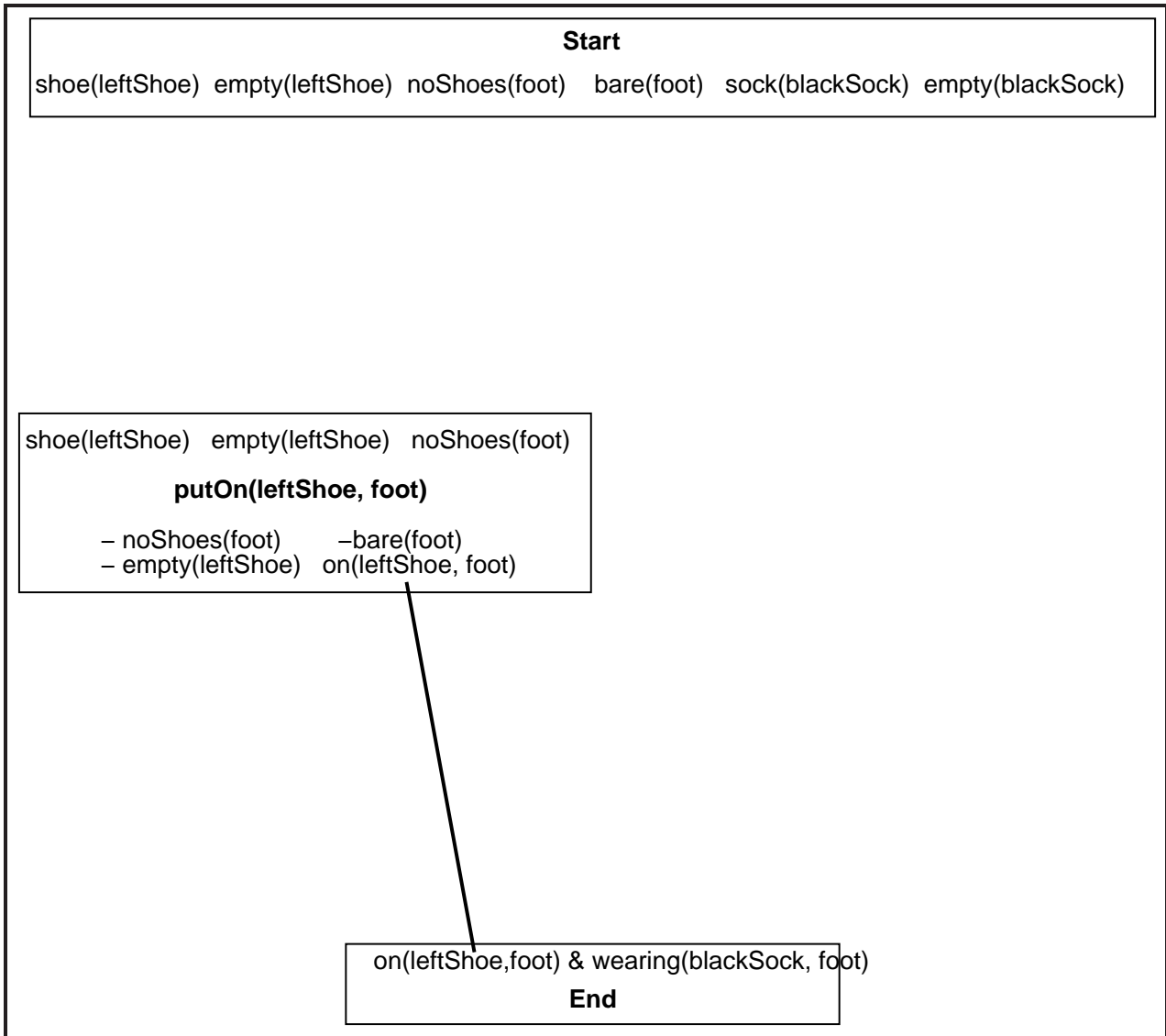
on(leftShoe,foot) & wearing(blackSock, foot)

Current state: empty(blackSock) & empty(leftShoe) & bare(foot) & noShoes(foot)
& sock(blackSock) & shoe(leftShoe)

(Question 6 continued on next page)

(Question 6 continued)

(h) [7 marks] Show how the partial order planner could construct a plan for the previous problem (putting sock and shoe on) without having to backtrack. The **Start** and **End** actions have been given, and the **putOn** action has been added; you need to show the links, any additional actions, the threats, and the ordering to resolve the threats.



Student ID:

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 7. Natural Language Processing

[20 marks]

Suppose you are writing a simple grammar checking system for a text editor. The DCG grammar and lexicon below cover some simple English sentences containing pronouns (“I/me”, “you”, “he/him”, *etc.*).

Grammar:

s --> np, vp, pps.

s --> np, vp.

vp --> verb, np.

vp --> verb, np, np.

pps --> pp, pps.

pps --> pp.

pp --> prep, np.

np --> det, noun.

np --> pronoun.

verb --> [V], {lexicon(V, verb)}.

noun --> [N], {lexicon(N, noun)}.

pronoun --> [PN], {lexicon(PN, pronoun, -, -, -)}.

prep --> [P], {lexicon(P, preposition)}.

det --> [D], {lexicon(D, det)}.

Lexicon:

lexicon(table, noun).

lexicon(present, noun).

lexicon(box, noun).

lexicon(cup, noun).

lexicon(shop, noun).

lexicon(gave, verb).

lexicon(opened, verb).

lexicon(put, verb).

lexicon(the, det).

lexicon(a, det).

lexicon(on, preposition).

lexicon(to, preposition).

lexicon(in, preposition).

lexicon(for, preposition).

⋮

lexicon(i, pronoun, subject, first, nonrefl).

lexicon(me, pronoun, object, first, nonrefl).

lexicon(myself, pronoun, object, first, reflex).

lexicon(you, pronoun, subject, second, nonrefl).

lexicon(you, pronoun, object, second, nonrefl).

lexicon(yourself, pronoun, object, second, reflex).

lexicon(he, pronoun, subject, thirdmale, nonrefl).

lexicon(him, pronoun, object, thirdmale, nonrefl).

lexicon(himself, pronoun, object, thirdmale, reflex).

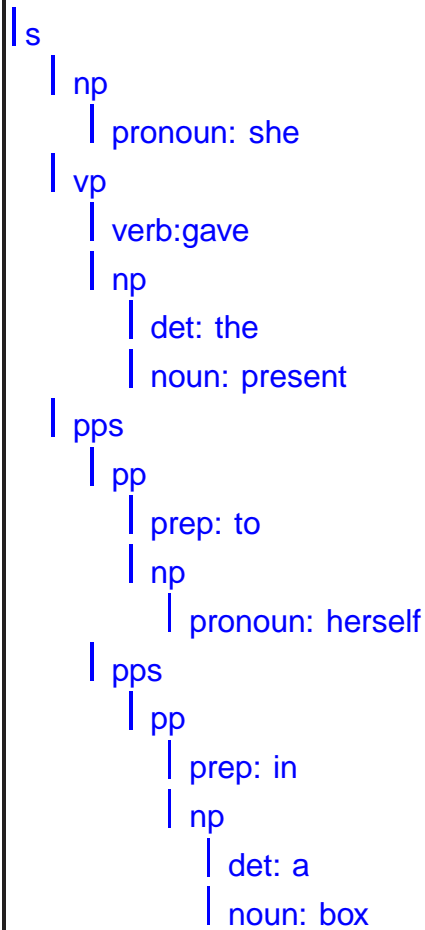
lexicon(she, pronoun, subject, thirdfem, nonrefl).

lexicon(her, pronoun, object, thirdfem, nonrefl).

lexicon(herself, pronoun, object, thirdfem, reflex).

(a) [5 marks] Show the parse tree for the following sentence:

“she gave the present to herself in a box”



she gave the present to herself in a box

(Question 7 continued on next page)

(Question 7 continued)

(b) [8 marks] The grammar above does not enforce the constraint that a pronoun in the subject of a sentence (the first np of a simple sentence) must be subjective (eg, “I”, “He”, “She”), but a pronoun in other parts of the sentence (in a vp or pp) must be objective (eg, “me”, “him”, “her”).

For example, the sentences on the left are valid, but the sentences on the right are invalid:

he gave me a present

I put him on the table

she opened the box for him

* him gave I a present

* me put he on the table

* her opened the box for he

Modify the grammar to enforce this constraint. Note that the lexicon already contains the information needed for these constraints.

You only need to show the changed rules.

```

|s --> np(subject), vp, pps.
|s --> np(subject), vp.
|vp --> verb, np(object).
|vp --> verb, np(object), np(object).
|np(_) --> det, noun.
|np(Case) --> pronoun(Case).
|pronoun(Case) --> [PN], {lexicon(PN, pronoun, Case, _)}.

```

(c) [7 marks] There is an additional constraint involving reflexive pronouns (eg, “myself”, “himself”, “herself”): a reflexive pronoun, whether in an np inside the vp or in a pp must match the subject of the sentence.

For example, the sentences on the left are valid, but the sentences on the right are invalid:

he gave himself a present	* he gave myself a present
she put the box on herself	* she put the box on myself
you gave a present to yourself	* you gave a present to himself
the shop gave her a present for me	* the shop gave herself a present for me

Modify the grammar to enforce this constraint also.

You only need to show the changed rules.

```
| s --> np(subject,Agent), vp(Agent), pps(Agent).
| s --> np(subject,Agent), vp(Agent).
| vp(Agent) --> verb, np(object, Agent).
| vp(Agent) --> verb, np(object, Agent), np(object, Agent).
| np(subject,thirdImp) --> det, noun.
| np(object,_) --> det, noun.
| np(Case,Agent) --> pronoun(Case,Agent).
| pronoun(subject,Agent) --> [PN], {lexicon(PN, pronoun, subject, Agent, nonrefl)}.
| pronoun(object,_) --> [PN], {lexicon(PN, pronoun, object, _, nonrefl)}.
| pronoun(object,Agent) --> [PN], {lexicon(PN, pronoun, object, Agent, reflex)}.

| a noun in a subject must be marked as thirdperson impersonal (or any other constant)
| so that reflexive personal pronouns in an object are prevented
| a pronoun in the subject must return the person (first, second, etc)
| a reflexive pronoun in an object must match the Agent
| a non-reflexive pronoun in an object does not need to match the Agent
```
