6.034 Final Examination Solutions
December 15, 2003

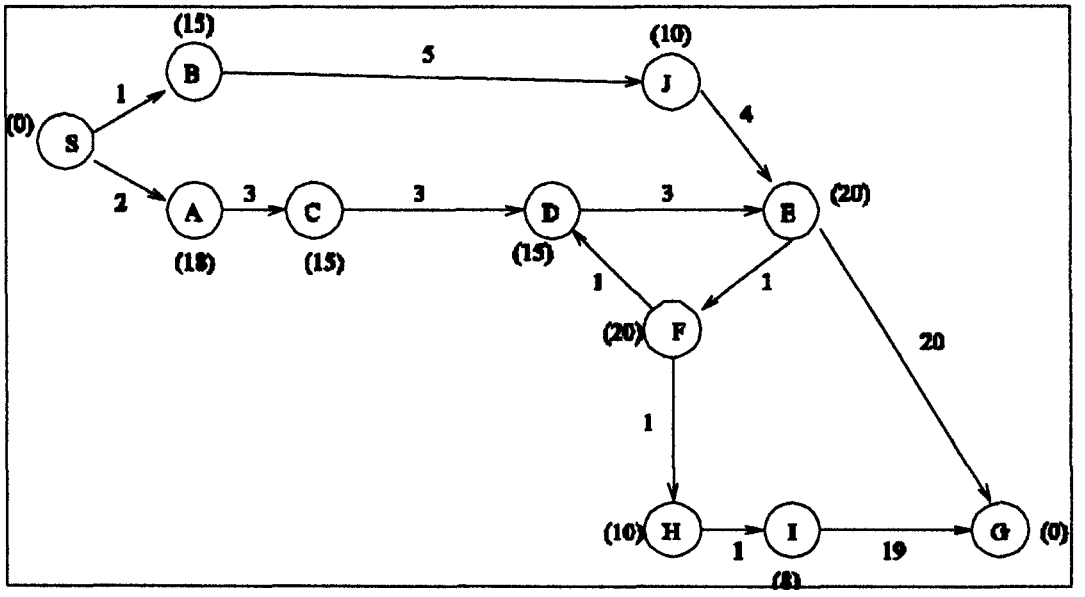
Name	
EMail	

Problems are arranged in the order the corresponding topics were discussed. In each problem, easy parts are in front. **Be sure to pace yourself so that you do all the easy parts! Note tear off sheet on back, which repeats helpful drawings and text.**

Problem number	Maximum	Score	Grader
1	17		
2	17		
3	19		
4	20		
5	15		
6	12		
Total	100		

Question 1: Search (17 points)

Suppose you are trying to find a path from S to G in the directed graph shown below. You will use several search techniques to find this path.



Make the following assumptions:

- The length of each link between nodes is given along the link.
- The heuristic value for each node is given in parentheses next to the node.
- All heuristics are admissible.
- Nodes are said to be "expanded" when a node is the final node in a path popped off the search queue.
- G will be discovered when a path to it is removed from the queue, not added to the queue.
- In the case of a "tie," nodes are expanded in alphabetical order.

Part 1.1: Different Algorithms (14 points)

1.1.1: Depth First Search (2 points)

List in order the nodes expanded by **depth-first search (no enqueued list)**:

SACDEG

1.1.2: Hill-Climbing (3 points)

List in order the nodes expanded by **hill-climbing search (with backup; no enqueued list)**:

S B J E G

1.1.3: A* (9 points)

List in order the nodes expanded by **A* search (with expanded list)**: (4)

S B J A C D E G

What is the path found by this A* search? (3)

S B J E G

When the A* search concludes, how many paths are left on the search queue? (2)

2

Part 1.2: Bidirectional Search (3 points)

Now, instead of doing a simple search from S to G, we choose to implement a **bidirectional search**. In the bidirectional search, a search from S to G is interdigitated with a search from G to S. In other words, one search starts from S and produces all paths of depth 1; then, another search starts from G, goes against the arrows, and produces all paths of depth 1. The searches then alternate, adding additional layers to their search queues, until an expansion produces a path from S that shares a node with a path from G.

Suppose we implement **bidirectional breadth-first search without an enqueued list**. That is, each of the two searches proceeds in a breadth-first manner. In the search from S, **nodes at each level are expanded in alphabetical order**. In the search from G, nodes at each level are expanded in **reverse alphabetical order**. What is the path found (from S to G) by such a search?

S B J E G

Question 2: Rules (17 points)

Note: the two parts of this problem are independent!

Part 2.1: Backward chaining (10 points)

2.1.1: Backward Chaining Simulation (9 points)

As anyone with a strong enough eyeglasses prescription knows, a key problem in wearing glasses is finding them again once you've set them down. In this problem, you will work with a backwards chaining rules system used for finding a pair of glasses. Conflict resolution is by rule order.

The rules, in order, are:

```
(R1 (IF (?x has bad breath)
        (?x is stiff)
     THEN (?x was sleeping)))
(R2 (IF (?x has back crick)
     THEN (?x was on couch)))
(R3 (IF (?x has bed-head)
     THEN (?x was in bed)))
(R4 (IF (?x has blanket-fuzz on face)
     THEN (?x was in bed)))
(R5 (IF (?x has wet hair)
     THEN (?x was showering)))
(R6 (IF (?x was sleeping)
        (?x was on couch)
     THEN (?x glasses are on coffeetable)))
(R7 (IF (?x was sleeping)
        (?x was in bed)
     THEN (?x glasses are on night-stand)))
(R8 (IF (?x was showering)
     THEN (?x glasses are on sink)))
```

You are to determine how the backward chainer works given **(Jake glasses are on ?x)** using the following assertions database.

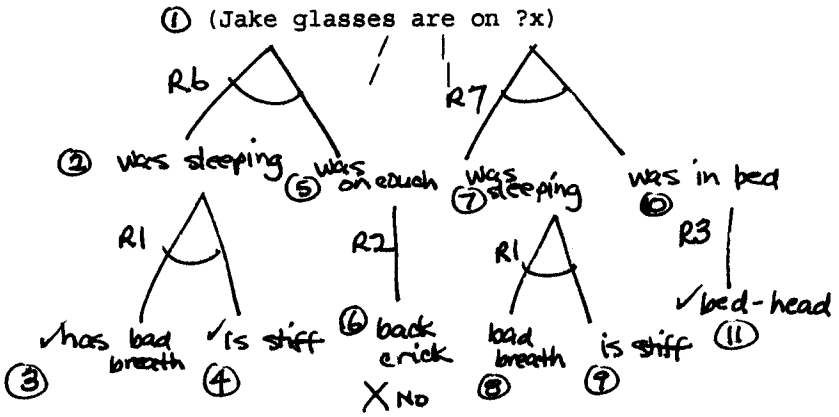
```
(Abi has back crick)
(Abi has wet hair)
(Jake has bad breath)
(Jake has bed-head)
(Jake is stiff)
```

You are to show how backward chaining works. In particular, you are to indicate the patterns the system tries to match against the database by filling out the following table, indicating the order in which the backward chainer tries to match pattern against the

database. To get you started, we have shown **(Jake glasses are on ?x)** as the first pattern matched against the database. You have enough room or more than enough room in the table.

Note that if the system matches an instantiated pattern against the database, you are to show in the table the pattern with each instantiated variable replaced by its binding. Assume that the user answers no to all questions. Assume that the backward chainer places no assertions into the database, so it may derive the same result more than once.

You may find it useful to complete the tree we have started for you, but this is for your own convenience. The only thing that will be graded is your answer in the table.



1	(Jake glasses are on ?x)
2	(Jake was sleeping)
3	(Jake has bad breath)
4	(Jake is stiff)
5	(Jake was on couch)
6	(Jake has back crick)
7	(Jake was sleeping)
8	(Jake has bad breath)
9	(Jake is stiff)
10	(Jake was in bed)
11	(Jake has bed-head)
12	
13	

2.1.2: Backward Chaining result (1 points)

Where are Jake's glasses?

on night-stand

Part 2.2: Rule Ordering (7 points)

You have these rules, but they are not necessarily shown in the order that they are provided in the database of rules:

```
(R1 (IF (baz ?x ?y)
     THEN (qux ?y ?x)))
(R2 (IF (?x ?y d)
     AND-IF (or (eq? '?x 'qux) (eq? '?x baz))
     THEN (qux a a)))
(R3 (IF (foo ?x)
         (bar ?y)
     THEN (baz ?x ?y)))
(R4 (IF (qux ?x ?y)
     THEN STOP))
(R5 (IF (foo ?x)
     THEN (bar ?x)))
```

You have these initial assertions, which **are** in the order shown. Assume new assertions are added to the end of the list.

```
(bar a)
(foo b)
(bar d)
(foo c)
(baz a d)
(bar b)
```

We run a forward chainer using "least recently fired" conflict resolution (i.e. given two triggered instances, the one from the rule fired longer ago will be chosen). If the least-recently-fired strategy produces a tie, then the rule earlier in the list has precedence. Running the forward chainer produces the following new assertions:

```
(bar c)
(baz b a)
(qux d a)
(qux a a)
STOP
```

Show the order the rules are provided in the database of rules. If the order cannot be determined, explain why.

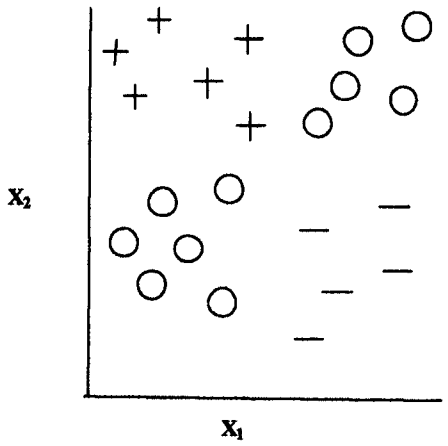
R5 R3 R1 R2 R4

Question 3: Perceptrons and Neural Nets (19 points)

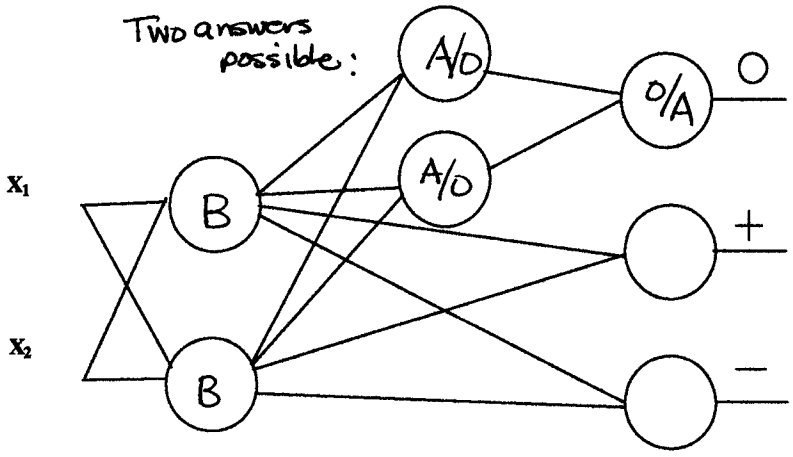
The two parts of this problem are independent.

Part 3.1: Perceptrons (7 points)

You are given the following data and perceptron network that classifies the data. Assume that perceptrons output 0 or 1, and that +, -, o are classes.

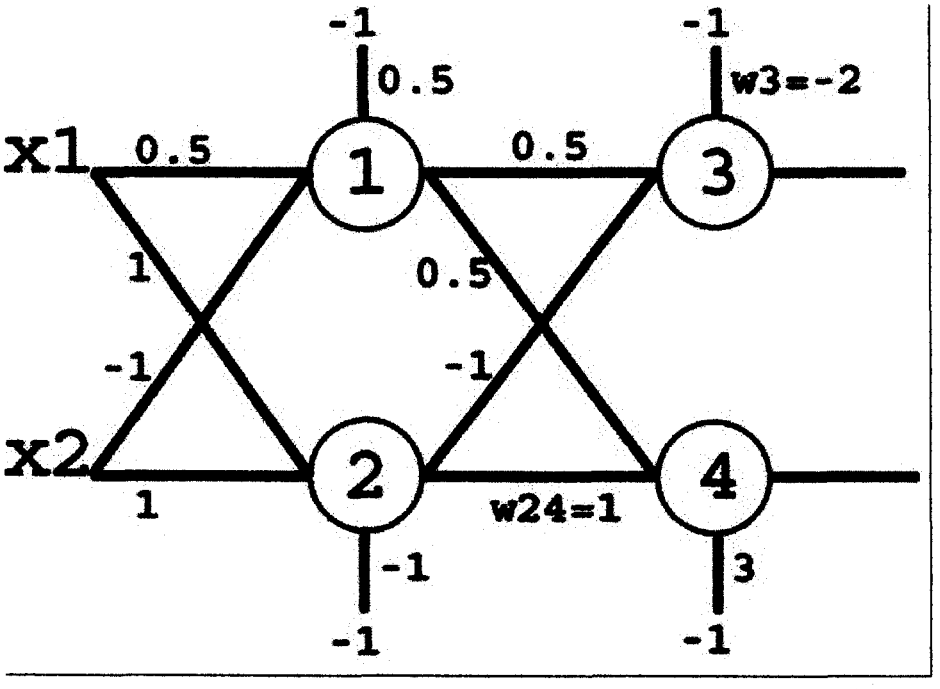


Label the nodes in the network according to their types: **B** (boundary line node), **A** (AND logic node), **O** (OR logic node), **N** (NOT logic node). Label a node by writing the appropriate letter (B, A, O, or N) inside the node.



Part 3.2: Neural Nets (12 points)

The following neural network has 4 neurons labeled 1, 2, 3 and 4. All neurons behave as sigmoid units except that they compute an output value using the function $y = 2z + 1$ (where z is the unit's input value) instead of the regular sigmoid function. Also, this network uses a **non-standard error function** $E = (1/3)(y^* - y)^3$. This network is to be used for the next two questions.



3.2.1: Forward Propagation (2 points)

Using the initial weights in the diagram and the input vector $[x1, x2] = [2, -1]$, compute the output of each specified neuron during forward propagation.

y1 (output from neuron 1)

$2(.5x_1 + -1x_2 - .5) + 1 = 4$

y2 (output from neuron 2)

$2(x_1 + x_2 + 1) = 5$

3.2.2: Backward Propagation (10 points)

Using a **learning rate of 1** and a desired ouput vector of *[y3, y4] = [0, 1]* backpropagate the network computing the δ values for units 1, 3 and 4 and new values for weights in the table below. Assume initial values for the weights are as specified in the diagram and assume the following values for neuron outputs:

- y1 (output at node 1) = -1.0
- y2 (output at node 2) = 5.0
- y3 (output at node 3) = 0.2
- y4 (output at node 4) = 0.9

Helpful backward propagation formulas appear on the tear-off sheet at the end of the examination

Express all δ s in terms of derivative free expressions.

δ_3

$-2(y_3^* - y_3)^2 = -2(-.02)^2 = -.08$

δ_4

$-2(y_4^* - y_4)^2 = -2(.1)^2 = -.02$

δ_1

$2[(\delta_3 w_{13}) + (\delta_4 w_{14})] = 2[(-.08)(.5) + (-.02)(.5)] = -.1$

Express the weights in terms of ds and numbers.

Old weight	New weight
$w_{24} = 1$	$1 - \delta_4 y_2 = 1.1$
$w_3 = -2$	$-1 - \delta_3 y_1 = -2.08$

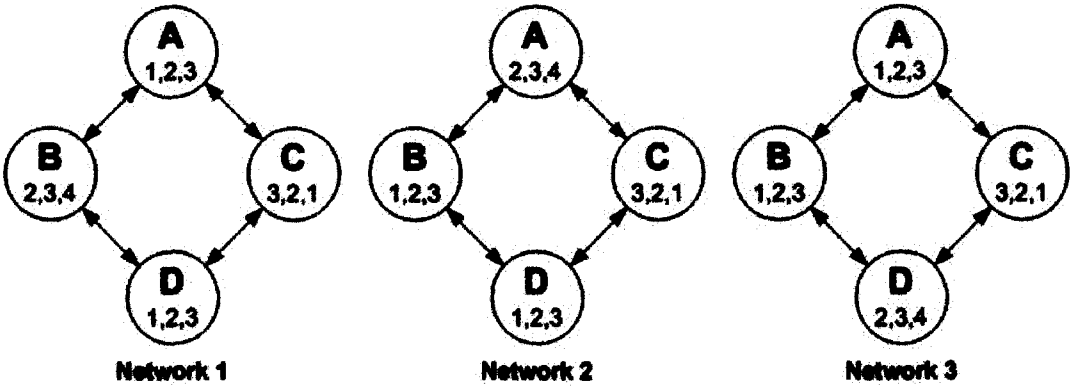
Question 4: Constraints (20 points)

Note: the parts of this problem are independent!

Part 4.1: Constrained search (14 points)

4.1.1: Pure Constraint Propagation (6 points)

Consider the following three constraint networks:



In each of these networks, the nodes represent variables A, B, C, and D. The numbers in each node are the domain of that variable. A link between a node represents a symmetric constraint, enforcing the condition that the two variables, when sorted in alphabetical order, will have their values in numerical order. In other words, the constraints enforce the conditions $A<B$, $A<C$, $B<D$, and $C<D$.

Solve each constraint network using **Pure Constraint Propagation,** which eliminates values from domains using constraints, but does not do any form of search. For each network:

a) Conclude that no solution exists
b) Find a solution that makes a unique assignment to each variable
c) Terminate inconclusively, without providing an answer as to whether a solution exists.

Then circle the appropriate conclusion for each network:

Network 1

- No solution
- ⊙ Returns solution
- Inconclusive

Network 2

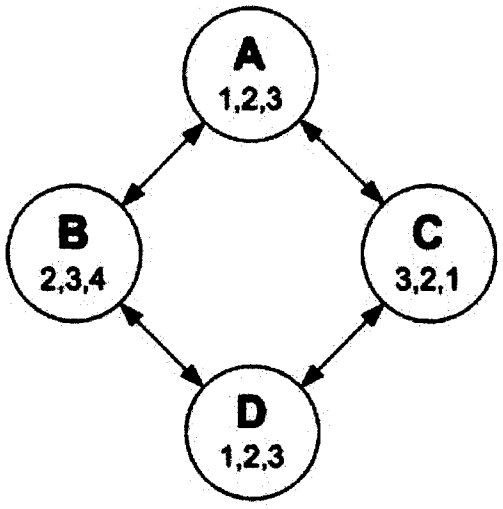
- ⊙ No solution
- • Returns solution
- • Inconclusive

Network 3

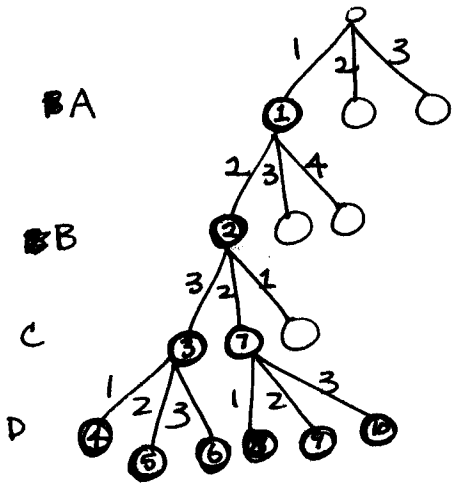
- • No solution
- • Returns solution
- ⊙ Inconclusive

4.1.2: Simple Backtracking Search (4 points)

Consider the following network, which is Network 1 from the previous part.



Network 1



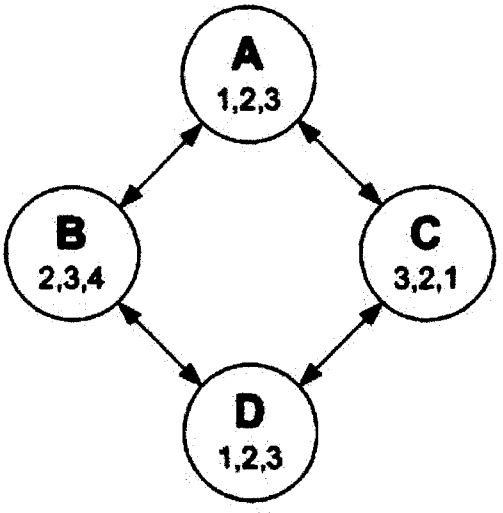
Find **one** valid solution for this problem, if it exists, using **backtracking (without forward checking). Do not assume that any constraint propagation has been done. Start from the initial domain values shown for each variable in the above network.** Examine variables in alphabetical order and values in left to right order as given. Assume that the search algorithm assigns a variable value, then checks constraints. Use the space above to show your work, but be sure to write your answers in the box.

How many variable assignments were made?

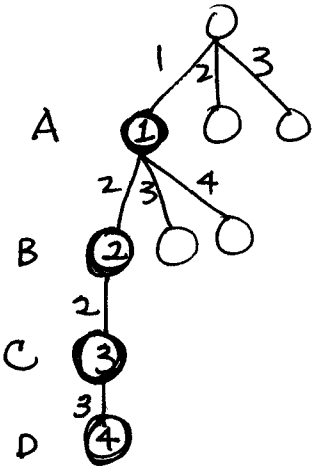
10

4.1.3: Backtracking Search with Forward Checking (4 points)

Consider the same network as the previous part, Network 1, which is shown again here for your convenience.



Network 1



var | new domain

A=1 C{3 2}

B=2 D{3}
C{2}

C=2

D=3

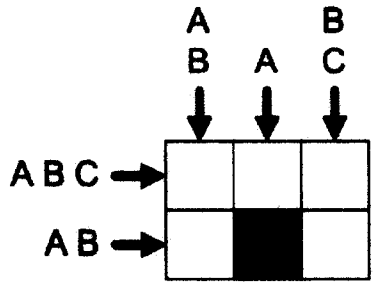
Find **one** valid solution for this problem, if it exists, using **backtracking with forward checking and propagation through domains of one. Do not assume that any constraint propagation has been done. Start from the initial domain values shown for each variable in the above network.** Examine variables in alphabetical order and values in left to right order as given. Assume that the search algorithm assigns a variable value, then checks constraints. Use the space above to show your work, but be sure to write your answers in the box.

How many variable assignments were made?

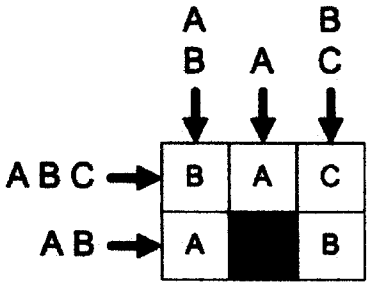
4

Part 4.2: Representing in CSP (6 points)

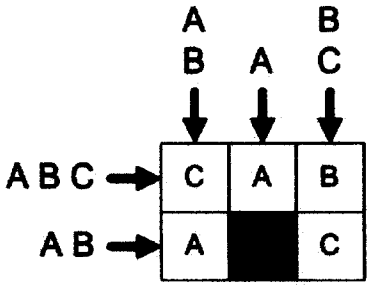
IAP is nearly here, which means that Mystery Hunt, MIT's annual puzzle competition, is right around the corner. In preparation, you decide to build some puzzle-solving tools.



Unsolved Puzzle



CORRECTLY Solved Puzzle



INCORRECTLY Solved Puzzle

One type of puzzle you might see looks like a crossword puzzle, but with different rules. When the puzzle is correctly solved, each non-black square in the grid has a letter assigned to it. Above each column of the grid is a set of letters, each of which must appear in exactly one of the cells in that column. Similary, to the left of each row is a set of letters, each of which must appear in exactly one of the cells in that row. Below the unsolved puzzles are two attempted solutions, one correct and one incorrect solution. The incorrect solution is faulty because cell (3,2) contains a C, but row 2 should only contain A and B. Similary, cell (1,1) contains a C, but column 1 should only contain A and B.

This class of puzzles can be represented using as a CSP problem with an appropriate choice of variables, values, domains, and constraints. Actually, there are at least three logical ways to do so. Below, we provide you with several options for variables, values, domains, and constraints. Your job is to find three different combinations of these options, each of which ensures that a CSP solver will be able to correctly solve the puzzle. **Note that you only need one type of constraint, and each pair of variables will be linked by at most one constraint.**

Variable Options:

Number	Your answer	Interpretation
1	cells	One variable for each *non-black* cell
2	rows	One variable for each row
3	columns	One variable for each column
4	rows+columns	One variable for each row, plus one variable for each column
5	letter types	One variable for each different letter (eg, 3 variables, A, B, and C)
6	letter instances	One variable for each instance of a letter above a column or beside a row

Value Options:

Number	Your answer	Interpretation
1	letters	individual letters
2	letters+empty	individual letters, plus the value "empty"
3	coordinates	cell coordinates
4	vectors	vectors of letters, optionally with dashes standing in for black cells.

Domain Options:

Number	Your answer	Interpretation
1	uniform	All variables initialized with the same domains
2	unique	Not all variables have the same domain

Constraint Options : Note that you only need one type of constraint, and each pair of variables will be linked by at most one constraint.

Number	Your Answer	Interpretation
1	equal value	each constraint requires that the values are equal
2	not equal value	each constraint requires that the values are not equal
3	specific target	each constraint requires that the values both equal a specific value, or neither equal that value
4	equal components	each constraint requires that specific components of the values are equal
5	not equal components	each constraint requires that specific components of the values are not equal

4.2.1: Representation Option 1 (2 points)

Variables:	1
Values:	1
Domain:	2
Constraints:	2

4.2.2: Representation Option 2 (2 points)

Variables:	4
Values:	4
Domain:	2
Constraints:	4

4.2.3: Representation Option 3 (2 points)

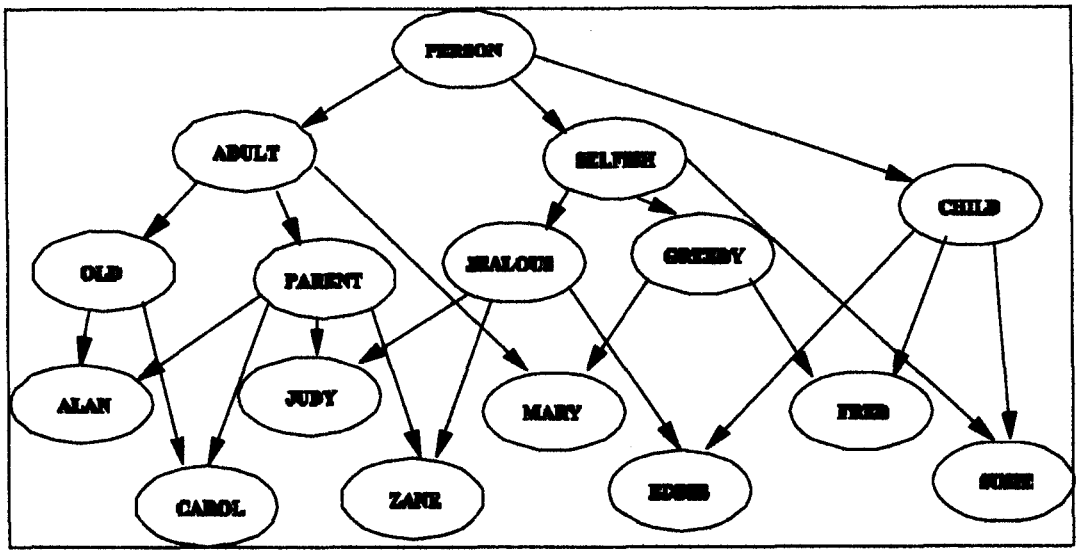
Variables:	6
Values:	3
Domain:	2
Constraints:	3

2 or 3
4
2
5

Question 5: Trees & Learning (15 points)

Part 5.1: Topological Sort (4 points)

For this question, use this inheritance tree when doing CLIMB-TREE operations. Resolve questions of precedence using the linearization produced by topological sort (using alphabetical order to break ties).



What is the most specific shared superclass for the following pairs? If there is no consistent answer, write "None." Two pairs have straightforward answers; one is a tricky case. **Note that you do not need to linearize the entire tree to work this problem, just those parts of the tree above the nodes specified in the pairs.**

Alan and Zane

Parent

Eddie and Susie

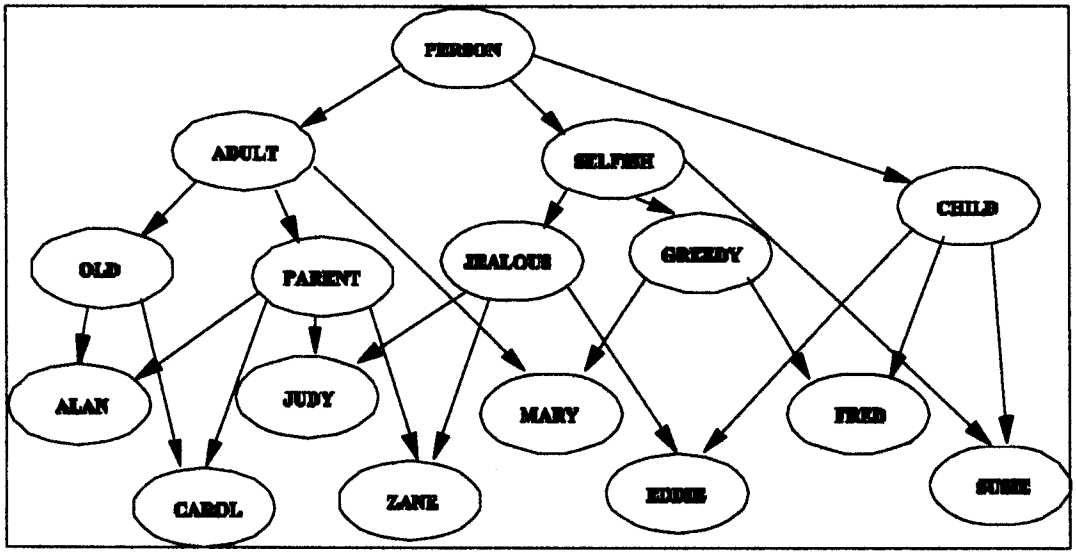
None

Jealous and Fred

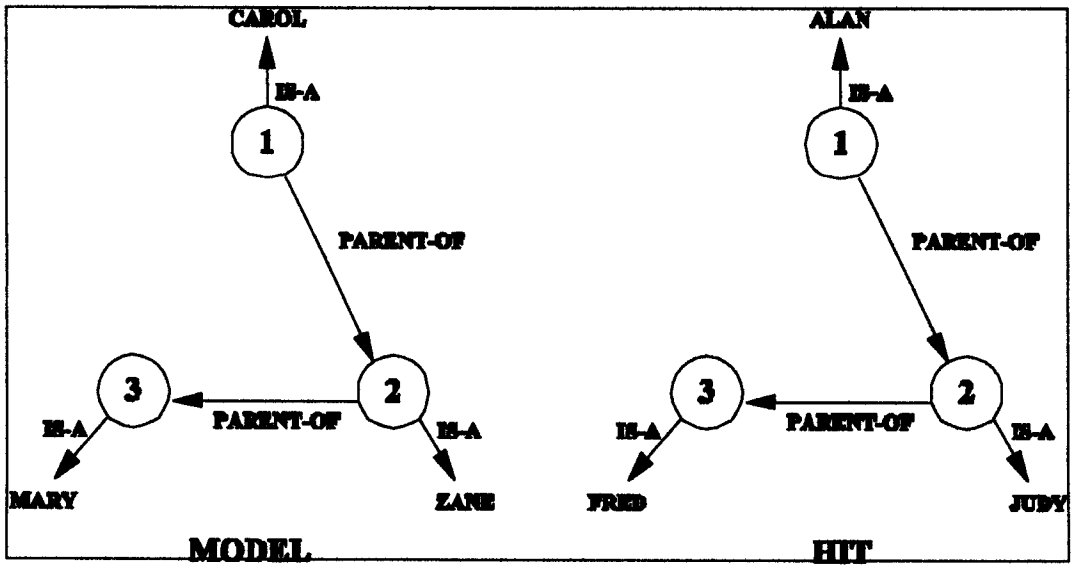
Selfish

Part 5.2: Near-Miss: Updating the Model (3 point points)

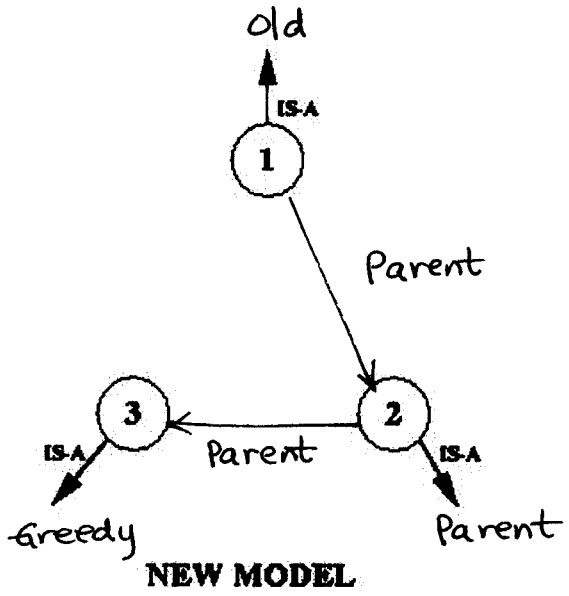
For this question, again use the inheritance tree provided in the previous part and resolve questions of precedence using the linearization produced by topological sort (using alphabetical order to break ties). **You need not have solved the previous part to do this part correctly. Also, there are no tricky cases.**



Your near-miss system currently has the model of "grandparent" shown on the left and is given the **HIT** training example shown on the right.



Show the model produced by applying near-miss learning:

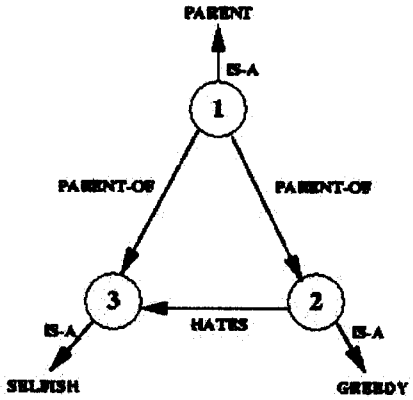


NEW MODEL

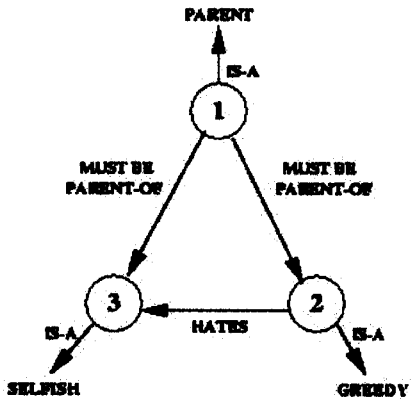
Part 5.3: Near-Miss: Teaching The System (5 points)

Your near-miss system currently has the model of sibling rivalry shown on the left and you wish to teach it the new model on the right by means of a single example.

In general, a near miss requires a single difference, but recall from the arch example in the near-miss lecture, two differences are allowed, and both are handled simultaneously, if they are the same kind of difference and they involve the same relations.



CURRENT MODEL



DESIRED MODEL

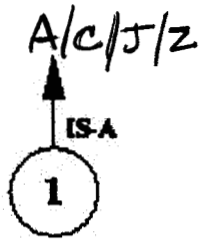
Your example should be a:

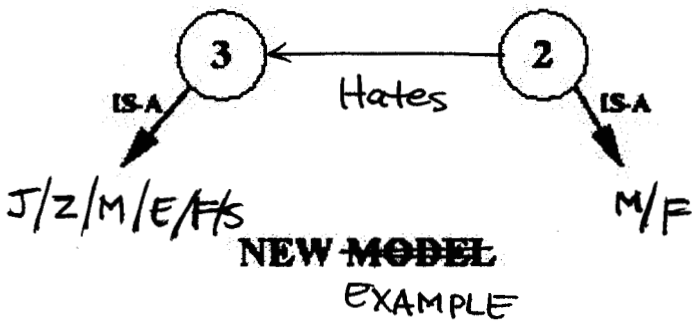
- Hit
- (•) Near-Miss

The type of learning done will be:

- (•) Specialization
- Generalization
- Both
- Neither

Show an example that works here:

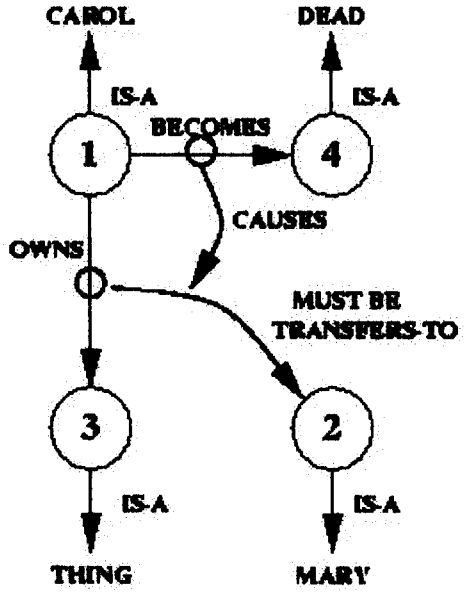




Note: Examples must be written in terms of instances, e.g. a Fred in the class of Freds.

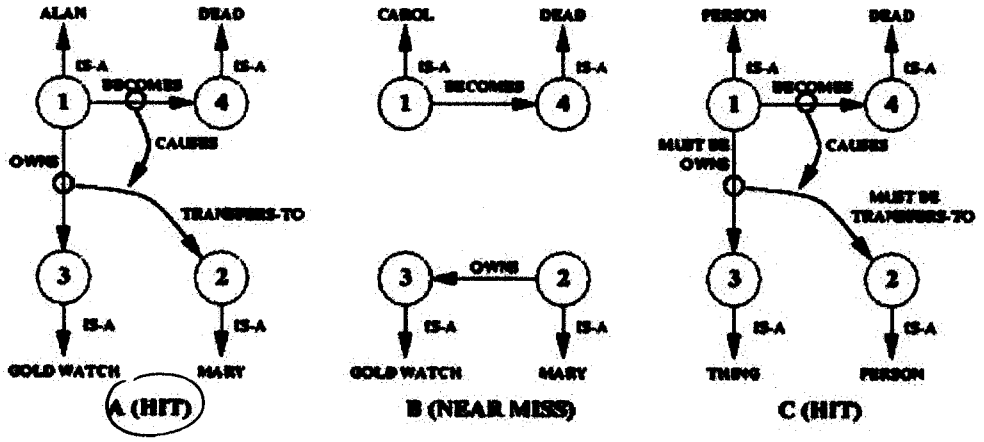
Part 5.4: Near-Miss: Quality of an Example (3 points)

Your near-miss learner currently has the following model of inheritance (the kind that occurs when you die).



MODEL

You decide your model needs work. Circle your choice of the following three examples for use as the next training example.



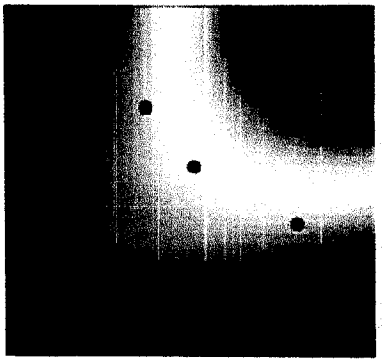
Question 6: Support Vectors (12 points)

The following diagrams represent graphs of support vector machines trained to separate pluses (+) from minuses (-). The origin is at the lower left corner in all diagrams. **See the separate color sheet for a clearer view of these diagrams.**

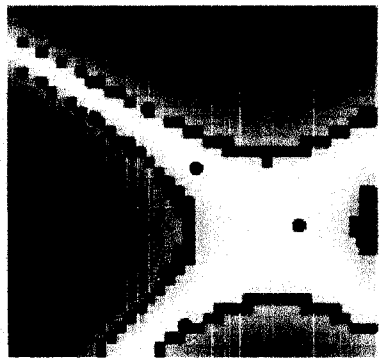
Part 6.1: Classifier quality (2 points)

Consider the following diagrams for SVMs, **all trained on the same data set**:

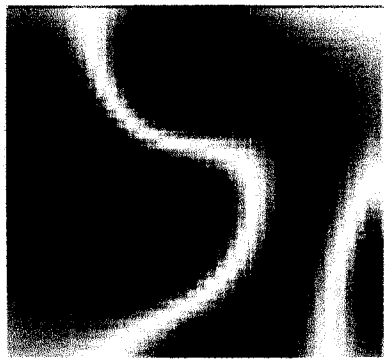
A



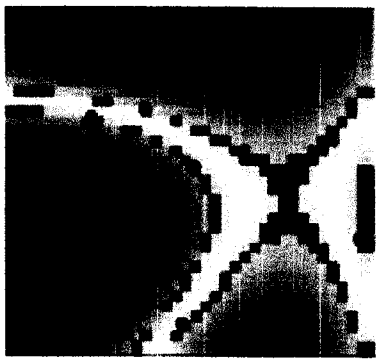
B



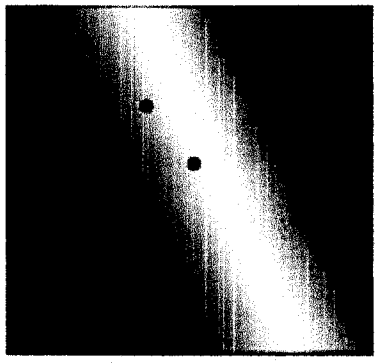
C



D



E



Which diagram represents the best classifier for the training data? Indicate your choice here:

D

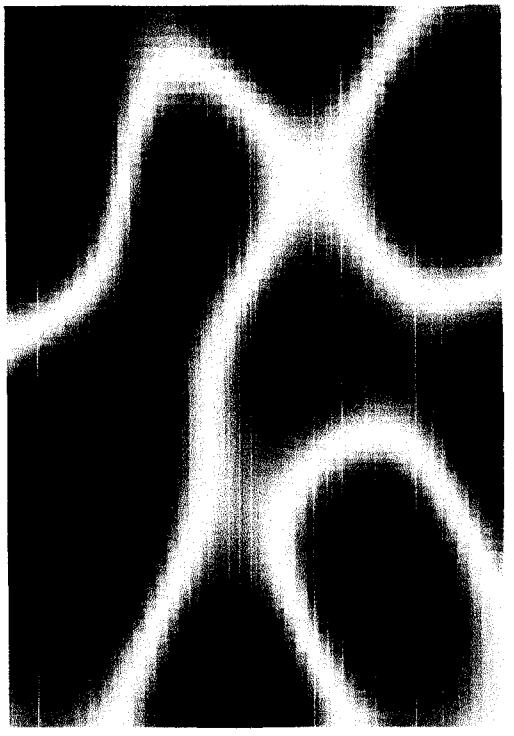
Part 6.2: Reasons for quality (3 points)

Circle **all** true statements.

The classifier claimed to be best...

- Has a linear kernel.
- Has a polynomial kernel.
- ⊙ Has fewer support vectors than some or all of the others.
- ⊙ Has fewer misclassified points than some or all of the others.
- Has fewer classified points than some or all of the others.
- ⊙ Has fewer non-zero weights than some or all of the others.

Part 6.3: Characteristics (3 points)

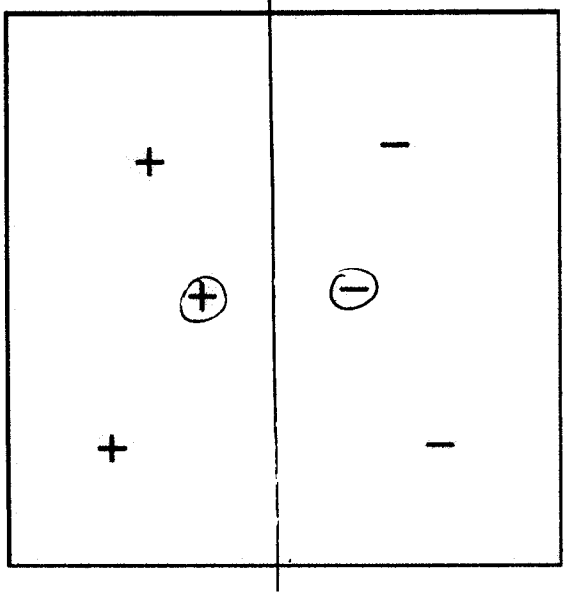


Given the diagram,what can we say about the SVM that produced it? Circle **all** true statements.

- It has a linear kernel.
- It has a polynomial kernel.
- ⊙ It has a radial basis kernel.
- ⊙ It has classified all points correctly.
- ⊙ It has overfit the data.

Part 6.4: Learning (1 points)

Assume that a support vector machine is to learn to separate all + from - in the following diagram. Sketch the **decision boundary** (not the gutters) and circle the points corresponding to support vectors assuming a **linear kernel**.



Part 6.5: Characteristics (3 points)

What can we say about SVMs produced with the training data shown in the diagram in the previous part? Circle **all** true statements.

- ⊙ A radial basis kernel finds the same decision boundary.
- ⊙ A radial basis kernel with a very small sigma finds more support vectors.
- • A radial basis kernel with a very small sigma finds fewer support vectors.
- • A radial basis kernel with a very large sigma finds more support vectors.
- • A radial basis kernel with a very large sigma finds fewer support vectors.

Tear-off sheet

Useful equations and definitions:

$z = \sum w_i x_i$ Neuron input function; w_i = weight, x_i = input value

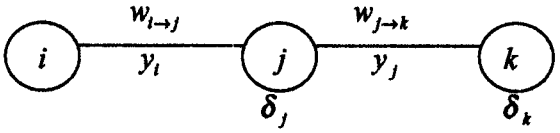
$y = f(z)$ Neuron output function; for sigmoid neural net, $y = s(z) = \frac{1}{1+e^{-z}}$

E Error function; standard error function; $E = \frac{1}{2}\sum (y^* - y)^2$

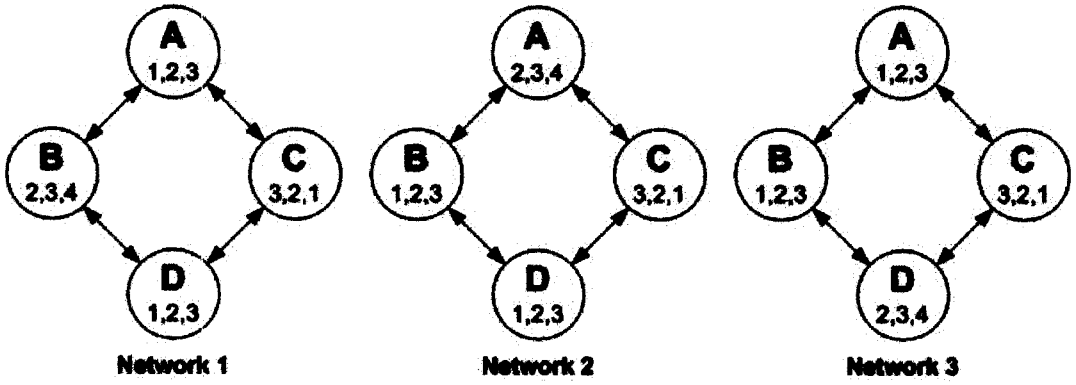
$\delta = \frac{\partial E}{\partial z}$ "Blame" for output neurons, derivative of error with respect to total input

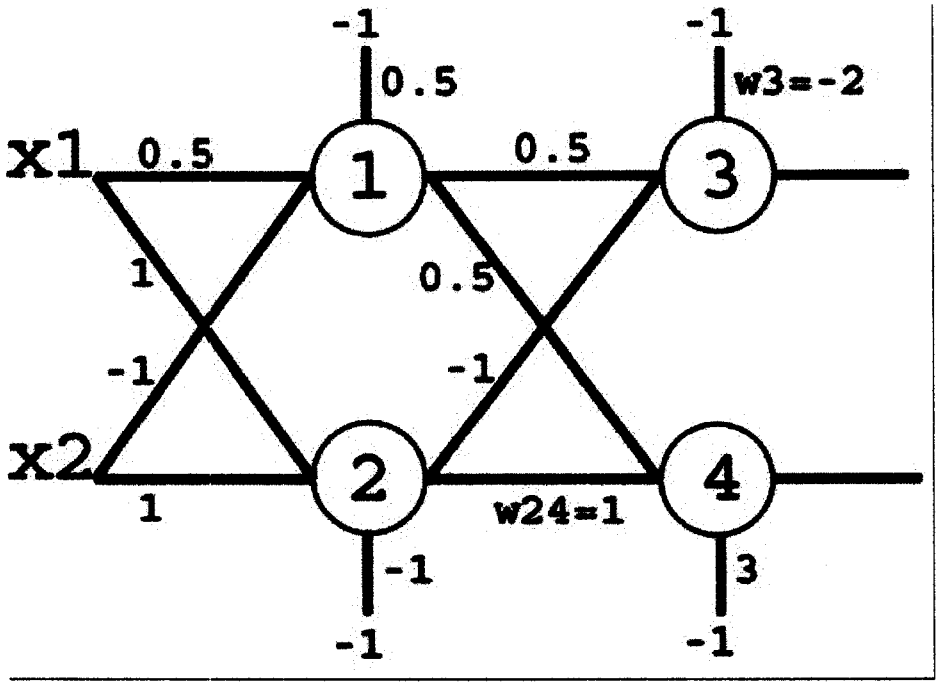
$\delta_j = \frac{dy}{dz_j} \sum_k \delta_k w_{j \to k}$ "Blame" for inner neurons

$w_i' = w_i - r\delta x_i$ Descent rule



y_i is x_i for input layer





The rules:

```
(R1 (IF (?x has bad breath)
        (?x is stiff)
     THEN (?x was sleeping)))
(R2 (IF (?x has back crick)
     THEN (?x was on couch)))
(R3 (IF (?x has bed-head)
     THEN (?x was in bed)))
(R4 (IF (?x has blanket-fuzz on face)
     THEN (?x was in bed)))
(R5 (IF (?x has wet hair)
     THEN (?x was showering)))
(R6 (IF (?x was sleeping)
        (?x was on couch)
     THEN (?x glasses are on coffeetable)))
(R7 (IF (?x was sleeping)
        (?x was in bed)
     THEN (?x glasses are on night-stand)))
(R8 (IF (?x was showering)
     THEN (?x glasses are on sink)))
```

The assertions database:

```
(Abi has back crick)
(Abi has wet hair)
(Jake has bad breath)
(Jake has bed-head)
(Jake is stiff)
```