

ARTIFICIAL INTELLIGENCE CSE352 HOMEWORK 1 SOLUTIONS

Book page8, Problem 1

Here is what one of the previous students wrote.

I think computer troubleshooting and searching for ore deposits are two expert tasks that might be suitable for an expert system.

In computer troubleshooting, computer users are stuck with a problem that prevents them from doing what they want to do with their computers. Sometimes, the problem is common and fairly easier to fix. Other times, it can be a very obscure issue that has a complicated solution. Either way, many people eventually need some help in diagnosing and fixing a computer problem. It would be very useful to automate the task of computer troubleshooting. An expert system would be very methodological in questioning the user about the symptoms of the problem. This would probably lead to a better identification of what the problem really is and the correct solution. Unlike a human expert, an expert system is available all the time. Hence, if a computer problem arises during odd times, help can easily be obtained from the expert system. An expert system for computer troubleshooting would require the key symptoms of many computer problems and the solutions to fix each of these problems. It would also need rules to correctly identify the problem from the given symptoms.

In the search for ore deposits, geological data of a certain area is collected to determine if the area is likely to have specific mineral deposits. Automating this task is useful because a lot of analysis is required to process the data to determine the likelihood of ore deposits. A human expert might make mistakes or even show bias to certain areas. However, an expert system is objective. Another reason is the expensive costs of mining operations. Mining the wrong areas would waste a lot of money and resources. This is why it's important to be objective and meticulous in locating potential ore deposits. Only an expert system can guarantee such qualities. An expert system for searching ore deposits would require the rules on how to interpret geological data and how each value of the data affects the probability of a deposit in a certain area.

Here is another student solution.

2. One task that would be well suited for an expert system would be city planning. Given the planning disaster that is Long Island, I would like to see how a computer would place the roads and buildings in the area to maximize population growth and ease traffic. The system would need complete knowledge of the current infrastructure of whatever city it was implemented in, including population statistics, business stats, and traffic flow stats as well as a set of rules for how to place new buildings and roads. Another task I think could be made into an expert system would be building design. Using an expert system for this could lead to greater energy efficiency and greater damage resiliency. This system would need to be given information such as the size of the plot which is being built on, an estimate of how much room is needed (floor and

ceiling space), what the temperature of the building needs to be and what kind of items the building will carry (people, merchandise, etc.).

4. I think that a good type of question to ask in the Loebner contest would be a series of “either-or” questions with one absurd choice, such as, “Would you like to go sky diving or play a baseball game on railroad tracks?” Unless the program is superbly written, it would probably recognize this as an “either-or” question and give back one of the two responses. A cleverly written program might answer that it doesn’t like either of the two choices. However, if it continued to give that answer I would be highly suspicious of it.

Book page 38, Problem 2

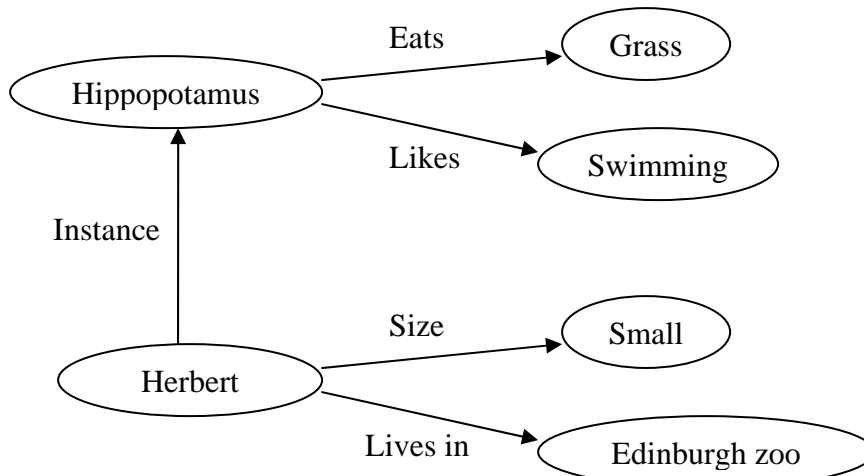
First, we have to define the predicates (or function expressions as called in the book) that we are going to use to represent the given facts. Note that “Apple(x)” is the intended interpretation used in AI and “A(x)” is the logic symbol used in logic for predicates.

Here are the representations of the given facts in predicate logic.

- $\forall x(A(x) \Rightarrow (G(x) \vee Y(x)))$
- In AI, under intended interpretation, we represent it as $\forall x(Apple(x) \Rightarrow (Green(x) \vee Yellow(x)))$
- $\neg \exists x(A(x) \wedge B(x))$
- $\neg \exists x(Apple(x) \wedge Blue(x))$
- $\forall x((A(x) \wedge G(x)) \Rightarrow T(x))$
- $\forall x((Apple(x) \wedge Green(x)) \Rightarrow Tasty(x))$
- $\forall x(M(x) \Rightarrow \exists y(A(y) \wedge T(y) \wedge L(x, y)))$
- $\forall x(Man(x) \Rightarrow \exists y(Apple(y) \wedge Tasty(y) \wedge Likes(x, y)))$

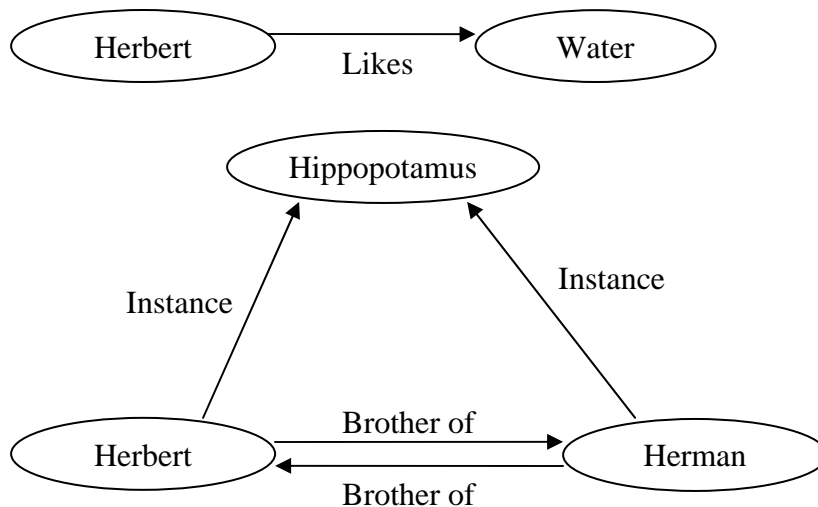
Book page 38, Problem 3

a) Here’s a semantic network to represent the given data.



b) In order to represent the given data in predicate logic, we must first define the predicates (function expressions) that we are going to use. We use INTENDED INTERPRETATION notation for constants, general notation for predicates. Here's the representation of the given data in predicate logic (under intended interpretation – we use names for CONSTANTS). Notice that the last two representations, $\forall x(H(x) \Rightarrow E(x, Grass))$ and $\forall x(H(x) \Rightarrow L(x, Swimming))$, may be replaced by the equivalent expression

a) Here are two new facts about Herbert that are easier to represent in a semantic network than in predicate logic.



b) Here are two new facts about Herbert that are easier to represent in logic than in a semantic network.

- Define a new predicate: $Feels(x, y) = F(x, y) = x \text{ feels } y$
- $F(Herbert, sad) \Rightarrow E(Herbert, ice\ cream)$
- $L(Herbert, Sleeping) \cup L(Herbert, Swimming)$

Book page 38, Problem 5

In order to express the rules in propositional logic, we must first define the propositions.

Here are the rules expressed in propositional logic.

In order to express the rules in predicate logic, we must first define the predicates (the textbook calls them function expressions) that we are going to use. Notice that we are using intended interpretation names for our predicates and constants

Here are the rules expressed in predicate logic (intended interpretation)

Here is a table that shows how our database looks like:

record	Savings	Income	Children	Partner	PartnerJob	InvestSavings	InvestStocks
X1	not adeq	Adequate	Yes	No	No	Yes	No
x2	adequate	not adeq	no	yes	yes	no	yes

a) This part will be solved using the representation in propositional logic. The database is $\{C, P, J\}$. We start off with the set of goals (or hypotheses) as $\{K\}$. K is not in the database. So, we must search for the sub-goals of K . Searching through the list of rules, we find that only R2 matches. So, we apply R2.

○ Apply R3. The set of goals is now $\{\neg C, I\}$. Notice that C is in the database.

So, $\neg C$ is unobtainable. The backwards search in this direction is rejected.

All searches for $\{K\}$ failed. So, $\{K\}$ is unobtainable.

Thus, $\{K\}$ is not supported by the database $\{C, P, J\}$.

b) This part will be solved using the representation in predicate logic. Here are additional predicates (function expressions) that we are going to use. Notice that we use intended interpretation names for our predicates instead of logic symbols ($A(x)$, $B(x)$, etc.).

Here are the extensions to the rule set in predicate logic.

Book page 47, Example

a) In order to express the rules in propositional logic, we must first define the propositions.

Here are the rules represented in propositional logic.

b) In order to express the rules in predicate logic, we must first define the predicates (function expressions) that we are going to use.

Here are the rules represented in predicate logic.

- $\text{Apple}(x) = A(x) = x$ is an apple
- $\text{Green}(x) = G(x) = x$ is green
- $\text{Yellow}(x) = Y(x) = x$ is yellow
- $\text{Blue}(x) = B(x) = x$ is blue
- $\text{Tasty}(x) = T(x) = x$ is tasty

- $\text{Man}(x) = \text{M}(x) = x$ is a man
- $\text{Likes}(x, y) = \text{L}(x, y) = x$ likes y
- Every apple is either green or yellow
- No apple is blue.
- If an apple is green then it is tasty.
- Every man likes a tasty apple.
- $\text{Hippopotamus}(x) = \text{H}(x) = x$ is a hippopotamus
- $\text{Eats}(x, y) = \text{E}(x, y) = x$ eats y
- $\text{Likes}(x, y) = \text{L}(x, y) = x$ likes y
- $\text{Small}(x) = \text{S}(x) = x$ is small
- $\text{LivesIn}(x, y) = \text{LI}(x, y) = x$ lives in y
- $S(\text{Herbert})$
- $LI(\text{Herbert}, \text{Edinburgh zoo})$
- $\forall x(H(x) \Rightarrow E(x, \text{Grass}))$
- $\forall x(H(x) \Rightarrow L(x, \text{Swimming}))$
- $\forall x(H(x) \Rightarrow (E(x, \text{Grass}) \cup L(x, \text{Swimming})))$
- Herbert likes water.
- Herbert has a brother named Herman (who is also a hippopotamus).
- If Herbert is sad, then he eats ice cream.
- Herbert likes sleeping or swimming.
- $S = \text{savings_adequate}$
- $V = \text{invest_savings}$
- $I = \text{income_adequate}$
- $K = \text{invest_stocks}$
- $C = \text{has_children}$
- $P = \text{has_partner}$
- $J = \text{partner_has_job}$
- R1: $\neg S \Rightarrow V$
- R2: $(S \wedge I) \Rightarrow K$
- R3: $\neg C \Rightarrow S$
- R4: $(P \wedge J) \Rightarrow I$
- $\text{Savings}(x, y) =$ the savings of x is considered to be y , values of y are adequate, not adequate
- $\text{InvestSavings}(x, y) = y$ indicates if x should invest savings, values of y are yes, no
- $\text{Income}(x, y) =$ the income of x is considered to be y , values of y are adequate, not adequate
- $\text{InvestStocks}(x, y) = y$ indicates if x should invest in stocks, values of y are yes, no
- $\text{HasChildren}(x, y) = y$ indicates if x has children, values of y are yes, no
- $\text{HasPartner}(x, y) = y$ indicates if x has a partner, values of y are yes, no
- $\text{PartnerHasJob}(x, y) = y$ indicates if the partner of x has a job, values of y are yes, no
-
- R1: $\text{Savings}(x, \text{notadequate}) \Rightarrow \text{InvestSavings}(x, \text{yes})$

- R2: $(Savings(x,adequate) \wedge Income(x,adequate)) \Rightarrow InvestStocks(x,yes)$
- R3: $HasChildren(x,no) \Rightarrow Savings(x,adequate)$
- R4: $(HasPartner(x,yes) \wedge PartnerHasJob(x,yes)) \Rightarrow Income(x,adequate)$
- Apply R1. The set of goals is now $\{S,I\}$. S is not in the database. So, we must search for sub-goals of S . Searching through the list of rules, we find that only R3 matches. So, we apply R3.
- PartnerWorksIn(x, y) = the partner of x works in y, values chipshop, notchipshop
- Children(x, y) = x has y children – values 15yes, 15no
- BuyHouse(x, y) = x wants to buy y – values yes, no
- We add one more value to attribute Income: huge
- Modified R3: $(HasChildren(x,no) \wedge BuyHouse(x,no)) \Rightarrow Savings(x,adequate)$
- Modified R4:
 $(HasPartner(x,yes) \wedge PartnerHasJob(x,yes) \wedge PartnerWorksIn(x,notchip\ shop)$
 $\wedge Children(x,15no)) \Rightarrow Income(x,adequate)$
- New R5: $Income(x,huge) \Rightarrow (Income(x,adequate) \wedge Savings(x,adequate))$
- C = coughing = person is coughing
- S = smoky = the environment is smoky
- W = wet = person is getting wet
- R = raining = it is raining
- B = burst_pipe = a pipe has burst
- A = alarm_rings = the alarm is ringing
- U = burglar = there is a burglary
- H = hot = the environment is hot
- F = fire = there is a fire
- R1: $C \Rightarrow S$
- R2: $(W \wedge \neg R) \Rightarrow B$
- R3: $(\neg C \wedge A) \Rightarrow U$
- R4: $(S \wedge H) \Rightarrow F$
- Coughing(x,y) = x describes whether y is coughing ; values of y are yes, no
- Smoky(x, y) = x describes whether y is smoky; values of y are yes, no
- Wet(x,y) = x describes whether y is getting wet; values of y are yes, no
- Weather(x, y) = y describes the weather outside of x; values of y are raining, not raining
- Alarm(x, y) = the alarm of x is y, values of y are ringing, not ringing
- Event(x, y) = there is a y occurring in x, values of y are fire, not fire
- Temperature(x, y) = the temperature in x is y
- R1: Coughing(x, yes) , then Smoky(x, yes)
- R2: $(Wet(x,yes) \wedge Weather(x,raining)) \Rightarrow Event(x,burst\ pipe)$
- R3: $(Wet(x,yes) \wedge Weather(x,raining)) \Rightarrow Event(x,burst\ pipe)$
- R4: $(Wet(x,yes) \wedge Weather(x,raining)) \Rightarrow Event(x,burst\ pipe)$

- Apply R1. The database is now $\{A, B, C, \neg D, \neg H, I\}$. The queue of rules (from front to end) is now $\{R2, R8\}$. Searching through the list of rules, we find that R3 matches and add R3 to our queue. The queue is now $\{R2, R8, R3\}$. We now apply R2.
- Apply R2. The database is now $\{A, B, C, \neg D, E, \neg H, I\}$. The queue of rules is now $\{R8, R3\}$. Searching through the list of rules, we find that R7 matches and add R7 to our queue. The queue is now $\{R8, R3, R7\}$. We now apply R8.
- Apply R8. The database is now $\{A, B, C, \neg D, E, \neg H, I, J\}$. The queue of rules is now $\{R3, R7\}$. Searching through the list of rules, we find that R9 matches and add R9 to our queue. The queue is now $\{R3, R7, R9\}$. We now apply R3.
- Apply R3. The database is now $\{A, B, C, \neg D, E, \neg H, I, J\}$. The queue of rules is $\{R7, R9\}$. Searching through the list of rules, we find no rules to add. We now apply R7.
- Apply R7. The database is now $\{A, B, C, \neg D, E, G, \neg H, I, J\}$. The queue of rules is $\{R9\}$. Searching through the list of rules, we find no rules to add. We now apply R9.
- Apply R9. The database is now $\{A, B, C, \neg D, E, G, \neg H, I, J, K\}$. The queue of rules is $\{\}$. Searching through the list of rules, we find no rules to add. No more rules match.
- Apply R5. The set of goals is now $\{E, F\}$. E is in the database. So, E is obtainable and can be removed from the set of goals. The set of goals is now $\{F\}$. F is not in the database. So, we must search for sub-goals of F . Searching through the list of rules, we see that only R4 matches. So, we apply R4.
- Apply R9. The set of goals is now $\{J, L\}$. J is not in the database. So, we must search for sub-goals of J . Searching through the list of rules, we find that only R8 matches. So, we apply R8.
- Apply R5. The set of goals is now $\{D\}$. D is not in the database. So, we must look for sub-goals of D . Searching through the list of rules, we see that only R1 matches. So, we apply R1.
- Apply R9. The set of goals is now $\{\neg E\}$. $\neg E$ is not in the database. So, we must look for sub-goals of $\neg E$. Searching through the list of rules, we see that only R3 matches. So, we apply R3.
- Apply R7. The set of goals is now $\{D, \neg E\}$. D is not in the database. So, we must look for sub-goals of D . Searching through the list of rules we see that only R1 matches. So, we apply R1.
- Apply R8. The set of goals is now $\{E, F\}$. E is not in the database. So, we must search for sub-goals of E . Searching through the list of rules, we see that only R4 matches. So, we apply R4.
- Apply R6. The set of goals is now $\{D, E\}$. D is not in the database. So, we must search for sub-goals of D . Searching through the list of rules, we see that only R1 matches. So, we apply R1.

SHORTHAND (Propositional) Solution:

Coughing \rightarrow Smoky
 Wet \wedge \neg Raining \rightarrow Burst
 \neg Coughing \wedge Ringing \rightarrow Burglar
 Smoky \wedge Hot \rightarrow Fire

PROPOSITIONAL LOGIC CONCEPTUALIZATION:

R1: $C \rightarrow S$	C – Coughing	A – Alarm	P – Burst Pipe
R2: $W \wedge \neg R \rightarrow P$	S – Smoky	B – Burglar	
R3: $\neg C \wedge A \rightarrow B$	W – Wet	H – Hot	
R4: $S \wedge H \rightarrow F$	R – Raining	F – Fire	

GOAL ORIENTED QUESTIONS

GOAL ONE -G1: F

F can only be reached by R4, so S and H must be in the fact's database. S can only be reached by R1, so C must also be in the facts database. No other rules can be applied, so the user must be asked about C and H.

Program: C – “Are you coughing?”

User: Yes C holds, continue questioning. Add C to the facts database.

or

User: No C fails, check next goal. Add \neg C to the facts database.

Program: H – “Are you hot?”

User: Yes H holds, follow R1 to deduce S, and R4 to deduce F.

Add H to the facts database. By conflict resolution, deduce

F.

Program: I deduce that there is a fire.

or

User: No H fails, check next goal. Add \neg H to the facts database.

GOAL TWO - G2: B

B can only be reached by R3, so \neg C and A must be in the facts database. No other rules can be applied so the user must be asked about both C and A.

Program: C – “Are you coughing?”

User: No \neg C holds, continue questioning. Add \neg C to the facts database.

or

User: Yes \neg C fails, check next goal. Add C to the facts database.

Program: A – “Is the alarm ringing?”

User: Yes A holds, use R3 to deduce B.

 Add A to the facts database. By conflict resolution, deduce B.

Program: I deduce that there is a burglar.

 or

User: No A fails, check next goal. Add $\neg A$ to the facts database.

GOAL THREE -G3: P

P can only be reached by R2, so W and $\neg R$ must be in the facts database.

No other rules can be applied, so the user must be asked about both W and R.

Program: W – “Are you wet?”

User: Yes W holds, continue questioning. Add W to the facts database.

 or

User: No W fails, check next goal. Add $\neg W$ to the facts database.

Program: R – “Is it raining?”

User: No $\neg R$ holds, use R2 to deduce P

 Add $\neg R$ to the facts database. By conflict resolution, deduce P.

Program: I deduce that a pipe has burst.

 or

User: Yes $\neg R$ fails, check next goal. Add $\neg R$ to the facts database.

Busse Notes, Problem 2

a) We start off with the database as $\{A, B, \neg D, \neg H, I\}$. Searching through the list of rules, we find that R1, R2, and R8 match. Following the conflict resolution rules, we apply R1 first.

The final content of the database after forward chaining is $\{A, B, C, \neg D, E, G, \neg H, I, J, K\}$.

b) We start off with the database as $\{A, B, D, E, I\}$. Searching through the list of rules, we find that R1 and R8 match. Following the conflict resolution rules, we apply R1 first.

- Apply R1. The database is now $\{A, B, C, D, E, I\}$. The queue of rules is now $\{R8\}$. Searching through the list of rules, we find that R4 matches and add R4 to our queue. The queue is now $\{R8, R4\}$. We now apply R8.

- Apply R8. The database is now $\{A, B, C, D, E, I, J\}$. The queue of rules is now $\{R4\}$. Searching through the list of rules, we find that R9 matches and add R9 to our queue. The queue is now $\{R4, R9\}$. We now apply R4.
- Apply R4. The database is now $\{A, B, C, D, E, F, I, J\}$. The queue of rules is now $\{R9\}$. Searching through the list of rules, we find that R5 matches and add R5 to our queue. The queue is now $\{R9, R5\}$. We now apply R9.
- Apply R9. The database is now $\{A, B, C, D, E, F, I, J, K\}$. The queue of rule sis now $\{\}$. Searching through the list of rules, we find no rules to add. No more rules match.

The final content of the database after forward chaining is $\{A, B, C, D, E, F, I, J, K\}$.

c) We start off with the set of goals (or hypotheses) as $\{L\}$. L is not in the database. So, we must look for sub-goals of L . Searching through the list of rules, we see that only the conclusion of R5 matches L . So, we apply R5.

- Apply R4. The set of goals is now $\{C, D\}$. C is not in the database. So, we must search for sub-goals of C . Searching through the list of rules, we see that only R1 matches. So, we apply R1.
 - Apply R1. The set of goals is now $\{A, B, D\}$. Both A and B are in the database. So, they are obtainable and can be removed from the set of goals. The set of goals is now $\{D\}$. Notice that $\neg D$ is in the database. So, D is unobtainable. The backwards search in this direction is rejected.

All searches for L failed. Thus, L is unobtainable.

No, the goal $\{L\}$ is not supported by the database $\{A, B, \neg D, E\}$.

d) We start off with the set of goals (or hypotheses) as $\{K, L\}$. K is not in the database. So, we must search for sub-goals of K . Searching through the list of rules, we find that only R9 matches. So, we apply R9.

- Apply R8. The set of goals is now $\{I, L\}$. I is in the database. So, I is obtainable and can be removed from the set of goals. The set of goals is now $\{L\}$. L is not in the database. So, we must search for sub-goals of L . Searching through the list of rules, we find that only R5 matches. So, we apply R5.
 - Apply R5. The set of goals is now $\{E, F\}$. E is not in the database. So, we must search for sub-goals of E . Searching through the list of rules, we find that both R2 and R3 match. So, we apply R2 first.
 - Apply R2. The set of goals is now $\{A, \neg D, F\}$. Both A and $\neg D$ are in the database. So, they are obtainable and can be removed from the set of goals. The set of goals is now $\{F\}$. F is not in the database. So, we must search for sub-goals of F . Searching through the list of rules, we find that only R4 matches. So, we apply R4.

- Apply R4. The set of goals is now $\{C, D\}$. C is not in the database. So, we must search for sub-goals of C . Searching through the list of rules, we find that only R1 matches.
 - Apply R1. The set of goals is now $\{A, B, D\}$. A is in the database. So, A is obtainable and can be removed from the set of goals. The set of goals is now $\{B, D\}$. B is not in the database. So, we must search for sub-goals of B . Searching through the list of rules, we find that no rule matches. So, B is unobtainable. The backwards search in this direction is rejected.

The set of goals is set back to $\{E, F\}$. We can now apply R3.

- Apply R3. The set of goals is now $\{C, \neg D, F\}$. C is not in the database. So, we must search for sub-goals of C . Searching through the list of rules, we find that only R1 matches.
 - Apply R1. The set of goals is now $\{A, B, \neg D, F\}$. A is in the database. So, A is obtainable and can be removed from the set of goals. The set of goals is now $\{B, \neg D, F\}$. B is not in the database. So, we must search for sub-goals of B . Searching through the list of rules, we find that no rule matches. So, B is unobtainable. The backwards search in this direction is rejected.

All searches for $\{K, L\}$ failed. So, $\{K, L\}$ is unobtainable.

No, the goal $\{K, L\}$ is not supported by $\{A, \neg D, \neg H, I\}$.

Busse Notes, Problem 4

a) We start off with the set of goals (or hypotheses) as $\{G\}$. G is not in the database. So, we must look for sub-goals of G . Searching through the list of rules, we see that the conclusions of R5 and R9 match with G . Following the conflict resolution rules, we apply R5 first.

- Apply R1. The set of goals is now $\{A, B\}$. Notice that $\neg A$ is in the database. So, A is unobtainable. The backwards search in this direction is rejected.

The set of goals is set back to $\{G\}$. We can now apply R9.

- Apply R3. The set of goals is now $\{\neg A, B\}$. $\neg A$ and B are both in the database. So, they are obtainable and can be removed from the set of goals. The set of goals is now $\{\}$. Hence, G is obtainable.

Yes, the goal $\{G\}$ is supported by the database $\{\neg A, B, C\}$.

b) We start off with the set of goals (or hypotheses) as $\{H\}$. H is not in the database. So, we must look for sub-goals of H . Searching through the list of rules, we see that the conclusions of R7 and R8 match H . Following the conflict resolution rules, we apply R7 first.

- Apply R1. The set of goals is now $\{A, B, \neg E\}$. Notice that $\neg A$ is in the database. So, A is unobtainable. The backwards search in this direction is rejected.

The set of goals is set back to $\{H\}$. We can now R8.

- Apply R4. The set of goals is now $\{B, C, F\}$. Both B and C are in the database. So, they are obtainable and can be removed from the set of goals. The set of goals is now $\{F\}$. F is not in the database. So, we must search for sub-goals of F . Searching through the list of rules, we see that only R2 matches. So, we apply R2.
 - Apply R2. The set of goals is now $\{\neg A\}$. $\neg A$ is in the database. So, $\neg A$ is obtainable and can be removed from the set of goals. The set of goals is now $\{\}$. Hence, H is obtainable.

Yes, the goal $\{H\}$ is supported by the database $\{\neg A, B, C\}$.

c) We start off with the set of goals (or hypotheses) as $\{I\}$. I is not in the database. So, we must look for sub-goals of I . Searching through the list of rules, we see that only R6 matches. So, we apply R6.

- Apply R1. The set of goals is now $\{A, B, E\}$. Notice that $\neg A$ is in the database. So, A is unobtainable. The backwards search in this direction is rejected.

All searches for I failed. Thus, I is unobtainable.

No, the goal $\{I\}$ is not supported by the database $\{\neg A, B, C\}$.

Below are other student solutions (written in a different form)

Busse Handout problem #2

a. DB = $\{A, B, \neg D, \neg H, I\}$

	Step 1	Step 2	Step 3	Step 4	Step 5
Applicable Rules	R1, R2, R8	R2, R3, R8	R7, R8	R7, R9	R9
Queue	R1 R2 R8	R2 R8 R3	R8 R7	R7 R9	R9
Rule fired, State of DB	R1 fired, DB = $\{A, B, C, \neg D, \neg H, I\}$	R2 fired, DB = $\{A, B, C, \neg D, E, \neg H, I\}$	R8 fired, DB = $\{A, B, C, \neg D, E, \neg H, I, J\}$	R7 fired, DB = $\{A, B, C, \neg D, E, G, \neg H, I, J\}$	R7 fired, final DB = $\{A, B, C, \neg D, E, G, \neg H, I, J, K\}$

b. DB = $\{A, B, D, E, I\}$

	Step 1	Step 2	Step 3	Step 4	Step 5
Applicable Rules	R1, R8	R4, R8	R4, R9	R5, R9	R5
Queue	R1 R8	R8 R4	R4 R9	R9 R5	R5
Rule fired, State of DB	R1 fired, DB = $\{A, B, C, D, E, I\}$	R8 fired, DB = $\{A, B, C, D, E, I, J\}$	R4 fired, DB = $\{A, B, C, D, E, F, I, J\}$	R9 fired, DB = $\{A, B, C, D, E, F, I, J, K\}$	R5 fired, final DB = $\{A, B, C, D, E, F, I, J, K, L\}$

c. DB = $\{A, B, \neg D, E\}$ Goal = $\{L\}$

Backward Chaining

Step 1	Step 2
The only rule that supports L is R5, $E \wedge F$. F is not in the DB so it becomes a sub goal	The only rule that supports F is R4, $C \wedge D$. However, since $\neg D$ is in the DB, F is not supported

d. DB = {A, $\neg D$, $\neg H$, I} Goal = {K, L}

Backward Chaining

Step 1	Step 2	Step 3	Step 4	Step 5
The only rule that supports L is R5, $E \wedge F$. F is not in the DB so it becomes a sub goal.	The only rule that supports F is R4, $C \wedge D$. However, since $\neg D$ is in the DB, F is not supported.	Move onto K	The only rule that supports K is R9, $J \rightarrow K$. J is not in the DB so it becomes a sub goal.	The only rule that supports J is R8, $I \rightarrow J$. Since I is in the DB, J is supported and thus K is supported

(2) Handout Problem #4

a. DB = { $\neg A$, B, C} Goal = {G}

Backward Chaining

Step 1	Step 2	Step 3
The first rule that supports G is R5, $D \rightarrow G$. D is not in the DB, so it is made a sub goal. However, the only rule supporting D is R1, $A \wedge B$. Therefore D is not supported	The next rule that supports G is R9, $\neg E \rightarrow G$. $\neg E$ is not in the DB, so it is made a sub goal.	R3 supports $\neg E$, given what we have in the DB. Therefore, G is supported

b. DB = { $\neg A$, B, C} Goal = {H}

Backward Chaining

Step 1	Step 2	Step 3	Step 4
The first rule that supports H is R7, $D \wedge \neg E$. From the previous example, we know $\neg E$ is supported, so D is made a sub goal. However, the only rule that supports D, R1, is not supported by the DB.	The next rule that supports H is R8, $E \wedge F$. Neither E or F are in the DB so we make them sub goals. F first.	The only rule that supports F is R2, which is a rule supported by the DB. So F is supported	R4 supports E, and the rule is supported by the DB. Therefore, E and F are supported which means the goal is supported

c. $DB = \{\neg A, B, C\}$ $Goal = \{I\}$

Backward Chaining

Step 1

The only rule that supports I is R6, $D \wedge E$. However, we know from the first part of this problem that R1 is the only rule that supports D. That rule, $A \wedge B$ is not supported at all, therefore the goal here is not supported.