CSE352 AI

HOMEWORK 1 SOLUTIONS

PART ONE

1. CONCEPTUALIZATION DEFINITION (NILSON)

- **Conceptualization** – step one of formalization of knowledge in declarative form.
- **C** = (**U**, **F**, **R**)
- **U** – Universe of discourse; it is a FINITE set of objects.

- **F** – Functional Basis Set; Set of functions (defined on **U**). Functions may be partial.
- **R** – Relational Basis Set; Set of relations defined on **U**.
- Remark: sets **R**, **F** are FINITE.

PROBLEM 1

Conceptualize the following situation using the above definition.

In a room there are 3 girls, 4 boys, and 2 cars – one red and one blue. **The following properties must be true.**

1. *Each girl likes exactly one boy.*

2. *Some boys like some girls.*

3*. Everybody (boys and girls) like some car.*

4*. Three boys like a red car.*

5. *Two girls like a blue car.*
Use the following notation

U – Universe of discourse is the set

U ={ o1, o2, o3, o4, o5, o6, o7, o8, o9 }

R – Relational Basis Set; Set of relations

R = { GIRL, BOY, CAR, REDCAR, BLUECAR, LIKE}

USE **INTENDED Interpretation, i.e.**

Relation **GIRL** is defined intuitively by a property *x is a girl*
Relation **BOY** is defined intuitively by a property *x is a boy*

Relation **CAR** is defined intuitively by a property *x is a car*

Relation **CAR** is defined intuitively by a property *x is a car*

Relation **REDCAR** is defined intuitively by a property *x is a red car*

Relation **BLUECAR** is defined intuitively by a property *x is a blue car*

Relation **LIKE** is defined intuitively by a property *x likes y*

Remark that the relations **GIRL, BOY, CAR, REDCAR, BLUECAR** are *one argument relations* and the relation

LIKE is a *two argument relation* and all of them are defined on the Universe **U**

SOLUTION:

We define, for example the relation **GIRL ⊆ U** (one argument relation) as

GIRL ={ o3, o6, o9 }

We define, for example the relation **BOY ⊆ U**
(one argument relation) as

BOY = { o2, o4, o5, o8}

Observe that the sets **GIRL** and **BOY** must be **disjoint-** as we use the **intended interpretation**

Observe that the sets **CAR, GIRL** and **BOY** must also be **disjoint-** as we use the **intended interpretation**

We must define now the relation **CAR ⊆ U**
(one argument relation) as

CAR ={ o1, o7}

We define, for example the relation **REDCAR ⊆ CAR ⊆ U**
(one argument relation) as

REDCAR ={ o7}

Observe that the sets **REDCAR** and **BLUECAR** must be **disjoint-** as we use the **intended interpretation**

We must hence define now the relation
BLUECAR⊆ CAR ⊆ U (one argument relation) as

BLUECAR ={ o1}

DEFINITION of the relation **LIKE**

Relation **LIKE** is defined intuitively by a property x likes y

LIKE 2 argument relation defined on **U**, i.e.

LIKE⊆ UxU

and must fulfill the following properties:

1. *Each girl likes exactly one boy.*
2. *Some boys like some girls.*
3*. Everybody (boys and girls) like some car*
4*. Three 3 boys like red car.*
5. *Two 2 girls like blue car*

Observe that the relation **LIKE** in order to fulfill the conditions **1.-5.** must be defined differently on different subsets of **U.**

We define first appropriate parts **LIKE1, LIKE2, …, LIKE5** that correspond to properties **1., 2., … 5.** and define **LIKE** as set union of all of them, i.e. we put

LIKE = LIKE1 ∨ LIKE2 ∨ …∨ LIKE5

PROPERTIES

1. *Each girl* ***likes*** *exactly one boy.*

We define **LIKE1** as follows

LIKE1⊆ GIRL x BOY ={ o3, o6, o9 }x { o2, o4, o5, o8} ⊆ UxU

LIKE1 ={(o3, o2), (o6, o8), (o6, o8)}

Observe that **LIKE1** is a FUNCTION on the set **GIRL**, and partial function on **U**

Observe that there are many ways of defining **LIKE1** – this is just my choice

2. *Some boys* ***like*** *some girls.*

We define **LIKE1** as follows

LIKE2 ⊆ BOYxGIRL ={ o2, o4, o5, o8}x { o3, o6, o9 } ⊆ UxU

LIKE2 ={(o4, o3), (o4, o3)}

Observe that there are many ways of defining **LIKE2** – this is just my choice

PART 2

PREDICATE LOGIC CONCEPTUALIZATION
Translations from Natural Language

Follow the steps:

1. **Identify** the domain: always a set $X \neq \varphi$
2. **Identify** predicates (simple: atomic)
3. **Identify** functions (if needed)
4. **Identify** the connectives
5. **Identify** the quantifiers $\forall x$, $\exists x$ or Restricted Quantifiers $\forall P(x)$, $\exists Q(x)$
6. **Write a formula using only symbols for 2 ,3, 4, 5**
   Use restricted domain quantifier translation rules, where needed
7. **Write LOGIC formula** – without Restricted quantifiers

PROBLEM 2

P1. Translate into plain English predicate logic examples from our **BOOK, page 23**

P2. Write a FULL solution following the Predicate Logic (1) Lecture Notes, with explanation and justification of correctness for the examples on the **book page 24**.

Solution

P1.

Translate the following statements in predicate logic to standard English.

$friends(Alison, Richard)$

Alison is friends with Richard.

$friends(father(fred), father(joe))$

Fred's father is friends with Joe's father.

Observe that the formula

$friends(father(fred), father(joe))$

is correct when we understand father(x) as fatherof(x)

$Likes(x, Richard)$

$x\ likes\ Richard$

$friends(alison, richard) \rightarrow likes(alison, richard)$

If Alison is friends with Richard then Alison likes Richard

$likes(alison, richard) \vee likes(alison, chocolate)$

Alison likes Richard or Alison likes chocolate.

$(likes(alison, richard) \vee likes(alison, chocolate)) \wedge \neg likes(alison, chocolate)$

Alison likes Richard or Alison likes chocolate and Alison doesn't like chocolate.

P2

Alison eats everything that she likes

Domain : $X \neq \varphi$

Predicates:

$likes(x, y)$ - x likes y

$eats(x, y)$ - x eats y

Functions: NONE, constant - Alison

Connectives: $\rightarrow$ - implies

Quantifiers: $\forall x$ - for all x

Restricted: $\forall likes(alison, x)$

Logic formula: $\forall x\ (likes(alison, x) \rightarrow eats(alison, x))$

There exists some bird that doesn't fly.

Domain : $X \neq \varphi$

Predicates:

$flies(x)$ - X can fly in the

$bird(x)$ - x is a bird

Functions: NONE

Connectives:

$\neg$ - not

$\wedge$ - and

Quantifiers: $\exists x$ - there exists an x

Restricted: $\exists bird(X)$

Restricted formula: $\exists \boldsymbol{bird(X)}\, \neg \boldsymbol{flies(X)}$

Logic formula: $\exists \boldsymbol{X}\, (\boldsymbol{bird(X)} \wedge \neg \boldsymbol{flies(X)})$

Every person has something that they love.

Domain : $X \neq \varphi$

Predicates:

$Person(x)$ - x is a person

$loves(x, y)$ - x loves y

Functions: none

Connectives: $\rightarrow$ - implies

Quantifiers: $\forall x$ - for all x, $\exists y$ - there exists y

Restricted: $\forall person(x)$

Restricted Formula $\forall person(x)\ \exists y\ \boldsymbol{loves(x, y)}$

Logic formula: $\forall \boldsymbol{x}\, (\boldsymbol{person(x)} \rightarrow \exists y\ \boldsymbol{loves(x, y)})$

P3. FOLLOW the ABOVE Steps to write detailed solution to **problems 2 , 3** from our BOOK, **page 39**

SOLUTION to problem 2, book page 39

Translate: "Every apple is either green or yellow"

1. Domain: $X \neq \varphi$
2. Predicates: A(x) – x is an Apple. G(x) – x is green. Y(x) – x is yellow.
3. Functions: (none)
4. Connectives: $\vee$ - " OR" – to represent "either green or yellow"
5. Quantifiers: $\forall_{A(x)}$ – "**All** apples" **(restricted)**

6. RESTRICTED FORMULA : $\forall_{A(x)} (G(x) \vee Y(x))$

7. LOGIC FORMULA: $\forall_x (A(x) \Rightarrow (G(x) \vee Y(x)))$

Translate: "No apple is blue"

1. Domain: $X \neq \varphi$
2. Predicates: A(x) – x is an Apple. B(x) – x is blue.
3. Functions: (none)
4. Connectives: $\neg$ - "**not**"
5. Quantifiers: $\exists_{A(x)}$ – "**some** apples" (restricted)
6. RESTRICTED FORMULA: $\neg \exists_{A(x)} B(x)$
7. LOGIC FORMULA: $\neg \exists x (A(x) \wedge B(x)$

BE CAREFUL! **YOU MUST ALWAYS DO DIRECT translation- NOT of some logically EQUIVALENT FORM LIKE (via de Morgan Laws)**

"All apples are not blue"

HERE IS TRANSLATION OF "**All apples are not blue"**

1. Domain: $X \neq \varphi$
2. Predicates: A(x) – x is an Apple. B(x) – x is blue.
3. Functions: (none)
4. Connectives: ¬ - "**not**"
5. Quantifiers: $\forall_{A(x)}$ – "**All** apples" (restricted)

6. RESTRICTED FORMULA : $\forall_{A(x)} \neg B(x)$

7. LOGIC FORMULA: $\forall_x (A(x) \Rightarrow \neg B(x))$

It is a different formula from

LOGIC FORMULA: $\neg \exists x (A(x) \wedge B(x)$

Translate: "If an apple is green then it is tasty"

1. Domain: $X \neq \varphi$
2. Predicates: A(x) – x is an Apple. G(x) – x is green. T(x) – x is tasty.
3. Functions: (none)
4. Connectives: $\wedge, \Rightarrow$
5. Quantifiers: no quantifiers
6. RESTRICTED FORMULA none
7. LOGIC FORMULA: $((A(x) \wedge G(x)) \Rightarrow T(x))$

This is called an OPEN Formula

Translate: "Every man likes all tasty apples"

1. Domain: $X \neq \varphi$

2. Predicates: A(x) – x is an Apple, M(x) – x is a man, T(x) – x is tasty. L(x,y) – x likes y for
3. Functions: (none)
4. Connectives: $\wedge$, $\Rightarrow$
5. Quantifiers: $\forall_{M(x)}$ – "**Every** man", $\forall(A(y) \wedge T(y))$ – " **All** tasty apples" **(restricted)**

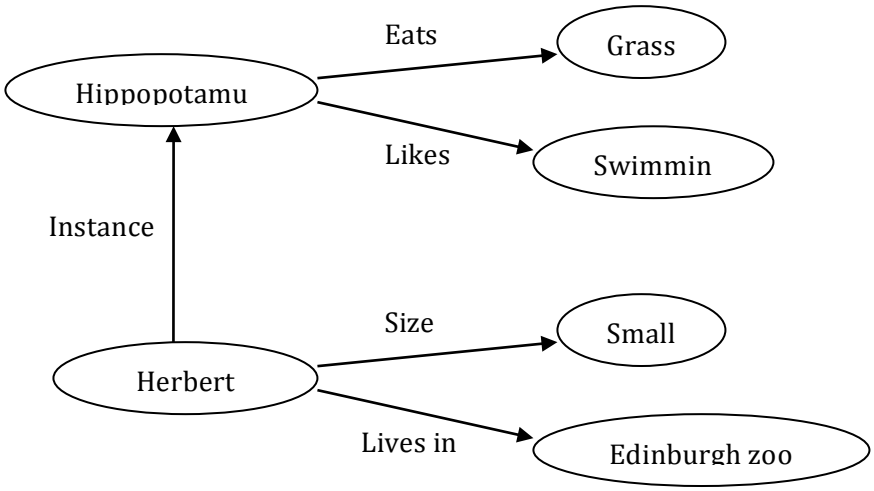
6. RESTRICTED FORMULA: $\forall_M(x)\ \forall(A(y) \wedge T(y))\ L(x,y)$

7. LOGIC FORMULA:

$$(\forall_x (M(x) \Rightarrow \forall_y ((A(y) \wedge T(y)) \Rightarrow L(x,y)))$$

SOLUTION to Problem 3, Book page 39

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 is INTENDED INTERPRETAION notation for constants, general notation for predicates.

- Hippopotamus(x) = H(x) = x is a hippopotamus
- Eats(x, y) = E(x, y) = x eats y

- Likes(x, y) = L(x, y) = x likes y
- Small(x) = S(x) = x is small
- LivesIn(x, y) = LI(x, y) = x lives in y

Here's the representation of the given data in predicate logic (under intended interpretation – we use names for CONSTANTS).

- $S(Herbert)$
- $LI(Herbert, Edinburgh\ zoo)$
- $\forall x(H(x) \Rightarrow E(x, Grass))$
- $\forall x(H(x) \Rightarrow L(x, Swimming))$

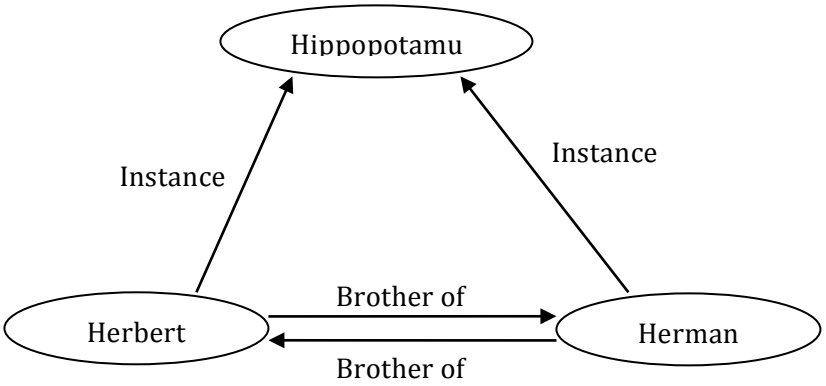
Notice that the last two representations, $\forall x(H(x) \Rightarrow E(x, Grass))$ and $\forall x(H(x) \Rightarrow L(x, Swimming))$, may be replaced by the equivalent expression-if we PROVE the equivalence!

- $\forall x(H(x) \Rightarrow (E(x, Grass) \bigcup L(x, Swimming)))$

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

- Herbert likes water.

- Herbert has a brother named Herman (who is also a hippopotamus).



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

- If Herbert is sad, then he eats ice cream.

- Define a new predicate: Feels(x, y) = F(x, y) = x feels y
- $F(Herbert, sad) \Rightarrow E(Herbert, ice\,cream)$

- Herbert likes sleeping or swimming.
  $L(Herbert, Sleeping) \cup L(Herbert, Swimming)$

2. Translations from Mathematics Language

PROBLEM 3

Here is a mathematical statement **S**:

For all natural numbers n the following implication holds:

IF n < 0, then there is a natural number m, such that

m+n <0

1. Re-write S as a "formula" **MF** that only uses mathematical and logical symbols
2. Translate your **MF** to a correct **logic formula LF**
3. Argue whether the statement **S** it true of false

Solution

1. Mathematical Formula

$$\forall n \in \mathbf{N}\ (n < 0 \Rightarrow \exists m \in \mathbf{N}(m+n< 0\ ))$$

2.Translation

1. **Domain:** $X \neq \varphi$
2. **Predicates:** $N(x)$ for $x \in \mathbf{N}$
   $G(x,y)$ for $x<y$

3.Functions:

f(x,y) for x+y

4.Constants: c for 0

5.Atomic formulas:

G(x, c) for x<0

G(f(y,x),c) for y+x<0

6. Connectives: $\Rightarrow$, $\wedge$

7. Quantifiers: $\forall_{N(x)}$, $\exists_{N(x)}$ restricted

8. **RESTRICTED FORMULA** :

$$\forall N(x)\ (G(x,c) \Rightarrow \exists N(y)\ G(f(y,x),c))$$

9. LOGIC FORMULA:

$$\forall x\,(N(x) \Rightarrow (G(x,c) \Rightarrow \exists y\,(N(y) \wedge G(f(y,x),c))))$$

3.The mathematical statement

$$\forall n \in N\ (n < 0 \Rightarrow \exists m \in N\,(m+n< 0))$$

is a **TRUE statement** because the **statement** $n > 0$ is **FALSE** for all natural numbers and by definition of implication we have that F$\Rightarrow$**Anything** is **TRUE.**

PART 3
RULE BASED SYSTEMS

PROBLEM 4

P1. CONCEPTUALIZE problem 5 from our BOOK, page 39 in **Propositional Logic** and solve it using Conflict Resolution.

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

- S = savings_adequate
- V = invest_savings
- I = income_adequate
- K = invest_stocks
- C = has_children
- P = has_partner
- J = partner_has_job

Here are the rules expressed in **propositional logic** conceptualization

- R1: $\neg$ S → V
- R2: S $\wedge$ I → K
- R3: $\neg$C → S
- R4: P $\wedge$ J → I
-

a) The initial 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 R2**. 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.
    - **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 $\{C, P, J\}$.

OBSERVE that the following CONCEPTUALIZATION is also correct

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

- S = savings_ NOTadequate
- A = savings_ adequate
- V = invest_savings
- I = income_adequate
- K = invest_stocks
- C = has_NOchildren
- P = has_partner
- J = partner_has_job
-

Here are the rules expressed in **propositional logic** conceptualization

- R1: S → V
- R2: A ∧ I → K

- R3: C → A
- R4: P ∧ J → I

P2. CONCEPTUALIZE problem 5 from our BOOK, page 39 in **Predicate Logic convention** using predicates
attribute(x, value of attribute),
attribute(object, value of attribute)

WRITE a format of a database TABLE needed for solution of P2

In order to express the rules in **PREDICATE FORM**, we must first define appropriate ATTRIBUTES and their values

We have the following ATTRIBUTES:

Savings
Values: **adequate, not adequate**

Income
Values: **adequate, not adequate**

InvestStocks
Values: **yes, no**

InvestSavings
Values: **yes, no**

Children
Values: **yes, no**

Partner
Values: **yes, no**

PartnerJob
Values: **yes, no**

DATA TABLE - example of a record

record	Savings	Income	Children	Partner	PartnerJob	InvestSavings	InvestStocks
o	not adequate	adequate	no	yes	no	yes	no

RULES:

R1: Savings(x, not adequate) → InvestSavings(x, yes)

R2: Savings(x, adequate) ∧ Income(x, adequate)→ InvestStocks(x, yes)

R3: Children(x, no) → Savings(x, adequate)

R4: Partner(x, yes) ∧ PartnerJob(x, yes) → Income(x, adequate)

Book page 47, Example

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

- 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

Here are the rules represented in propositional logic.

- R1: $C \Rightarrow S$
- R2: $(W \wedge \neg R) \Rightarrow B$
- R3: $(\neg C \wedge A) \Rightarrow U$
- R4: $(S \wedge H) \Rightarrow F$

SHORTHAND (Propositional) Solution:

Coughing → Smoky
Wet /\ ¬Raining → Burst
¬Coughing /\ Ringing → Burglar
Smoky /\ Hot → Fire

PROPOSITIONAL LOGIC CONCEPTUALIZATION:

R1: C → S

R2: W /\ ¬R → P
R3: ¬C /\ A → B
R4: S /\ H → F

C – Coughing
S – Smoky
W – Wet
R – Raining

A – Alarm P – Burst Pipe
B – Burglar
H – Hot
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 ¬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 ¬H to the facts database.

GOAL TWO - G2: B

B can only be reached by R3, so ¬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 ¬C holds, continue questioning. Add ¬C to the facts database.

or

User: Yes ¬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 ¬A to the facts database.

GOAL THREE -G3: P

P can only be reached by R2, so W and ¬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 ¬W to the facts database.

Program: R – "Is it raining?"
User: No ¬R holds, use R2 to deduce P
Add ¬R to the facts database. By conflict resolution, deduce P.
Program: I deduce that a pipe has burst.
or
User: Yes ¬R fails, check next goal. Add ¬R to the facts database.

CONFLICT RESOLUTION:

1. **Rules are numbered and ordered by number**

2. **Rules are scanned by RI in this order and inserted into a queue**
3. **RI fires a rule from the front of the queue and removes it**
4. **Principle: No rule is allowed to fire more then once on basis of the same contents of DBF – eliminates firing the same rule all the time**

Busse Handout problem #2

Part a. DB = {A, B, ¬D, ¬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, ¬D, ¬H, I}	R2 fired, DB = {A, B, C, ¬D, E, ¬H, I}	R8 fired, DB = {A, B, C, ¬D, E, ¬H, I, J}	R7 fired, DB = {A, B, C, ¬D, E, G, ¬H, I, J}	R7 fired, final DB = {A, B, C, ¬D, E, G, ¬H, I, J, K}

PART 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}

PART c. DB = {A, B, ¬D, E} Goal = {L}

- **Backward Chaining**

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

PART d. DB = {A, ¬D, ¬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 ∧ F. F is not in the DB so it becomes a sub goal.	The only rule that supports F is R4, C ∧ D. However, since ¬D is in the DB, F is not supported.	Move onto K	The only rule that supports K is R9, J -> K. J is not in the DB so it becomes a sub goal.	The only rule that supports J is R8, I -> J. Since I is in the DB, J is supported and thus K is supported

Busse Notes, Problem 2 – another Solution

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.

- 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.

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 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 rues, we see 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 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 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 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 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 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 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 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 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 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 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 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 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.
    - 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 $\{\neg A, B, C\}$.

(2) Handout Problem #4 - another Solution

a. DB = {¬A, B, C} Goal = {G}
Backward Chaining

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

b. DB = {¬A, B, C} Goal = {H}
Backward Chaining

Step 1	Step 2	Step 3	Step 4
The first rule that supports H is R7, D Λ ¬E. From the previous example, we know ¬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 Λ F. Neither E or F are in the DB so we make them sub	The only rule that supports F is R2, which is a rule supported by the DB.	R4 supports E, and the rule is supported by the DB. Therefore, E and F are supported which means the goal is

	goals. F first.	So F is supported	supported

c. DB = {¬A, B, C} Goal = {I}

Backward Chaining

Step 1
The only rule that supports I is R6, D Λ E. However, we know from the first part of this problem that R1 is the only rule that supports D. That rule, A Λ B is not supported at all, therefore the goal here is not supported.