St. Joseph's College of Engineering & Technology Palai

Department of Computer Science & Engineering

S8 CS

RT804 Artificial Intelligence

Module 4

Website: http://sites.google.com/site/sjcetcssz

Artificial Intelligence - Module 4

Syllabus

Knowledge and Reasoning –

Review of representation and reasoning with Logic – Inference in first order logic, Inference rules involving quantifiers, modus ponens,

Unification, forward and backward chaining – Resolution.

Contents

Part I. Propositional Calculus

Propositional calculus is a language. Using their words, phrases and senetences, we can represent and reason about properties and relationships in the world. Propositional logic is simple to deal with. Real world facts can be easily represented as logical propositions written as well formed formulas (wff's) in propositional logic.

1 Propositional Calculus Symbols

The symbols of propositional calculus are the

Propositional symbols:

P, Q, R, S. . . .

Truth symbols:

True, False

and connectives:

$\cap, \cup, \neg, \rightarrow, \equiv$

2 Propositional Calculus Sentences

The following are valid sentences;

$true$,

P,

Q,

R,

$\neg$,

$\neg false$,

$P \cup \neg P$,

$P \cap \neg P$,

$P \rightarrow Q$,

$P \cup Q \equiv R$

Conjunct

In the sentence

$$P \cap Q,$$

P and Q are called conjuncts.

Disjunct

In the sentence

$$P \cup Q,$$

P and Q are called disjuncts.

Equivalence

Two expressions in propositional calculus are equivalent if they have the same value under all truth value assignments.

$$\neg(\neg P) \equiv P$$

$$P \rightarrow Q \equiv \neg P \cup Q$$

$$\neg(P \cup Q) \equiv \neg P \cap \neg Q$$

$$\neg(P \cap Q) \equiv \neg P \cup \neg Q$$

$$P \cap Q \equiv Q \cap P$$

$$P \cup Q \equiv Q \cup P$$

$$(P \cap Q) \cap R \equiv P \cap (Q \cap R)$$

$$(P \cup Q) \cup R \equiv P \cup (Q \cup R)$$

$$(P \cup (Q \cap R) \equiv (P \cup Q) \cap (P \cup R)$$

$$(P \cap (Q \cup R) \equiv (P \cap Q) \cup (P \cap R)$$

These identities can be used to change propositional calculus expressions into a syntactically different but logically equivalent form.

3 Semantics of Propositional Calculus

A proposition symbol corresponds to a statement about the world.

For example, P may denote the statement “it is raining” or Q, the statement “I live in a wooden house”.

A proposition may be either true or false given some state of the world.

.

What is inference? Explain. (4marks)[MGU/May2008]

Discuss various inference rules in first order logic. (12marks)[MGU/June2007]

Explain inference process with an example in the first order logic. (12marks)[MGU/Jan2007]

Elaborate on two kinds of agents based on propositional logic. (4marks)[MGU/June2006]

How knowledge can be represented using first order logic? Give an example for making inferences in first order logic. (12marks)[MGU/June2006]

.

Explain the following. i)Knowledge and Reasoning. ii)Representation and Reasoning with logic. (12marks)[MGU/May2008]

Explain knowledge base. (4marks)[MGU/May2008]

Discuss about learning knowledge acquisition. (4marks)[MGU/Nov2010]

Discuss representation of knowledge and reasoning with logic. (12marks)[MGU/Nov2010]

Define Equivalence, Validity and Satisfiability. (4marks)[MGU/Jan2007]

Part II. Predicate Calculus

In propositional calculus, each atomic symbol P, Q etc.. denotes a proposition of some complexity. There is no way to access the components of an individual assertion. Predicate calculus provides this ability.

For eg. Instead of letting a single propositional symbol, P, denote the entire sentence “Marcus is a man”, we can create a predicate man that describes the relationship between man and Marcus. Man (Marcus)

Through inference rules we can manipulate predicate calculus expressions.

Predicate calculus allows expressions to contain variables.

4 The Syntax of Predicates

The symbols of predicate calculus consists of

1. The set of letters of the English alphabet. (both upper case and lower case)
2. The set of digits, 0, 1, 2. . . .9.
3. The underscore, _.

Symbols are used to denote objects, properties or relations in a world.

5 Predicate Calculus Terms

The following are some statements represented in predicate calculus.

f(X, Y)

father(david)

price(bananas)

likes(george, Susie)

friends(bill, george)

likes(X, george)

The predicate symbols in these expressions are f, father, price, likes, friends.

In the predicates above, david, bananas, george, Susie, bill are constant symbols.

The following represents some facts in the real world and corresponding predicate calculus expressions.

1. Marcus was a man.

   $man(Marcus)$

2. Marcus was a Pompeian.

   $pompeian(Marcus)$

3. All Pompeians were Romans.

   $\forall x\, pompeian(x) \rightarrow roman(x)$

4. Caesar was a ruler.

   $ruler(Caesar)$

5. All Romans were either loyal to Caesar or hated him.

   $\forall x\, roman(x) \rightarrow loyalto(x, Caeser) \ \cup \ hate(x, Caeser)$

6. Everyone is loyal to someone.

   $\forall x\, \exists y\, loyalto(x, y)$

7. People only try to assassinate rulers they are not loyal to.

   $\forall x\, \forall y\, person(x) \cap ruler(y) \cap tryassasinate(x, y) \ \rightarrow \ \neg loyalto(x, y)$

8. Marcus tried to assassinate Caesar.

   $tryassasinate(Marcus, Caesar)$

6 Quantifiers

Predicate calculus includes 2 symbols,

$\forall$ and $\exists$.

For example,

$\exists y\ friends(Y, peter)$

$\forall X\ likes(X, icecream)$

The universal quantifier, $\forall$, indicates that the sentence is true for all values of the variable.

The existential quantifier, $\exists$, indicates that the sentence is true for at least one value in the domain.

Some relationships between universal and existential quantifiers are given below.

$\vdash \exists X\ p(X) \equiv \forall X\ \vdash p(X)$

$\vdash \forall X\ p(X) \equiv \exists X\ \vdash p(X)$

$\exists X\ p(X) \equiv \exists Y\ p(Y)$

$\forall X\ q(X) \equiv \forall Y\ q(Y)$

$\forall X\ (p(X) \bigcap q(X)) \equiv \forall X\ p(X)\ \bigcap\ \forall Y\ q(Y)$

$\exists X\ (p(X) U q(X)) \equiv \exists X\ p(X)\ U\ \exists Y\ q(Y)$

7 Inference Rules

7.1 Modus Ponens

If the senetences $P$and $P \rightarrow Q$ are known to be true, then modus ponens lets us infer Q.

Example

Assume the following observations.

1. “if it is raining, then the ground will be wet”.
2. “it is raining”.

If P denotes “it is raining” and

Q denotes “the ground is wet”;

Then 1 becomes

$P \rightarrow Q$

2 becomes

P

With the application of modus ponens, we can infer Q.

That is

“ground is wet”.

Example 2

Consider the statements

“Socrates is a man”, and

“all men are mortal”.

These can be represented in predicate calculus.

“Socrates is a man” is represented as

$man(Socrates)$

“all men are mortal” is represented as

$\forall X\ (man(X) \rightarrow mortal(X)$

Since the variable X in the implication is universally quantified, we may substitute any value in the domain for x. By substituting “socrates” for X in the implication, we infer the expression,

$man(Socrates) \rightarrow mortal(Socrates)$

By considering the predicates

$man(Socrates)$ and

$man(Socrates) \rightarrow mortal(Socrates)$

Applying modus ponens, we get

$mortal(Socrates)$

That is

“Socrates is mortal”.

.

Describe modus ponens. (12marks)[MGU/June2007]

Part III. Unification

To apply inference rules such as modus ponens, an inference system must be able to determine when 2 expressions are the same or match. In predicate calculus, the process of matching 2 senetences is complicated by the existence of variables in the expressions.

Unification is an algorithm for determining the substitutions needed to make 2 predicate calculus expressions match. For example, in the earlier section, we substitute ‘socrates’ for X in the expression

$\forall X\ (man(X) \rightarrow mortal(X)$

This allowed the application of modus ponens to infer

$mortal(socrates)$

Example

$man(socrates)$

$\forall X\ (man(X) \rightarrow mortal(X))$

Here we substituted ’socrates’ for X.

This substitution is denoted by

$socrates/X.$

$socrates/X$is called a unifier.

Next let us see the unification algorithm. To simplify the manipulation of expressions, the algorithm assumes a slightly

modified syntax. By representing an expression as a list with the predicate or function names as the first element followed by its arguments, the manipulation of expressions is simplified.

PC syntax	LIST syntax
p (a,b)	(p a b)
friends (george, tom)	(friends george tom)
equal (eve, mother (cain))	(equal eve (mother cain))
man (socrates)	(man socrates)

8 Unification Algorithm

```
function unify (E1, E2)
{
    case
    {
        both E1 and E2 are constants or the empty list:
            if E1 == E2                then return { };
            else                       return FAIL;
        E1 is a variable:
            if E1 occurs in E2         then return FAIL;
            else                       return {E2 / E1};
        E2 is a variable:
            if E2 occurs in E1         then return FAIL;
            else                       return {E1 / E2};
        either E1 or E2 are empty:
                                       then return FAIL;
        otherwise:
        {
            HE1 = first element of E1;
            HE2 = first element of E2;
            SUBS1 = unify(HE1, HE2);
            If (SUBS1 == fail)         then return FAIL;
            TE1 = apply(SUBS1, rest of E!);
            TE2 = apply(SUBS1, rest of E2);
```

```
            SUBS2 = unify(TE1, TE2);
            if (SUBS2 == fail)          then return FAIL;
            else                        return composition (SUBS1, SUBS2);
        }
    }
}
```

Example:

Consider an example.

Given 2 predicates.

1. $parents\,(X,\ father(X),\ mother(bill))$
2. $parents\,(bill,\ father(bill),\ Y)$

These predicates can be represented in list syntax as

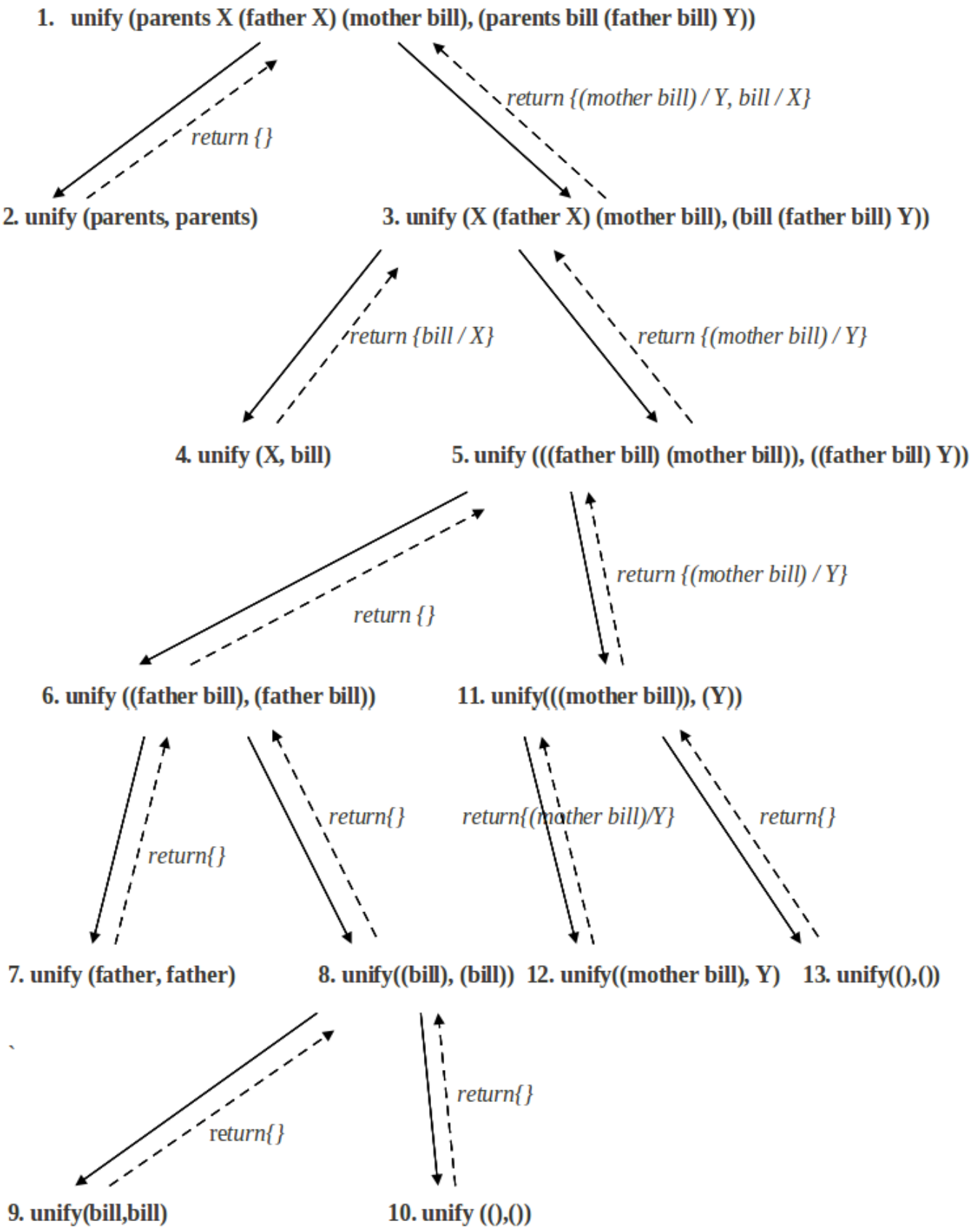
1 $\Longrightarrow (parents\ X\ (father\ X)\ (mother\ bill))$

2 $\Longrightarrow (parents\ bill\ (father\ bill)\ Y)$

To find out the unifier that match the above expressions call unify( );

$unify(\quad (parents\ X\ (father\ X)\ (mother\ bill)),\quad (parents\ bill\ (father\ bill)\ Y)\quad)$

A trace of the execution of the above call is shown below.



After the execution of the unification algorithm, we get the result as

{bill/X, (mother bill) / Y}.

.

What do you mean by unification? How is it useful in logic? (4marks)[MGU/June2007]

What is unification? (4marks)[MGU/Nov2010]

Part IV. Resolution

[Luger2002]

Resolution is a technique for proving theorems in the propositional or predicate calculus. Resolution proves a theorem by negating the statement to be proved and adding this negated goal to the set of axioms.

Resolution proofs involve the following steps.

1. Put the premises or axioms in to clause form.
2. Add the negation of what is to be proved, in clause form, to the set of axioms.
3. Resolve these clauses together, producing new clauses that logically follow from them.
4. Produce a contradiction by generating the empty clause.
5. The substitutions used to produce the empty clause are those under which the opposite of the negated goal is true.

Resolution requires that the axioms and the negation of the goal be placed in a normal form called clause form. Clause form represents the logical database as a set of disjunctions of literals.

9 Producing the Clause form

The resolution procedure requires all statements in the databse to be converted to a standard form called clause form. The form is referred to as conjunction of disjuncts.

The following is an example of a fact represented in clause form.

$$(\vdash dog(X)\, U\, animal(X)) \bigcap (\vdash animal(Y)\, U\, die(Y)) \bigcap (dog(fido))$$

The following is the algorithm for reducing any set of predicate calculus statements to clause form.

1. First we eliminate the $\rightarrow$ by using the equivalent form.

   For example

   $a \rightarrow b \equiv \vdash aUb$.

2. Next, reduce the scope of negation.

   $\vdash (\vdash a) \equiv a$

$\sim (\exists X)\, a(X) \;\equiv\; (\forall X)\; \sim a(X)$

$\sim (\forall X)\, b(X) \;\equiv\; (\exists X)\; \sim b(X)$

$\sim (a \bigcap b) \;\equiv \sim a\, U\; \sim b$

$\sim (aUb) \;\equiv \sim a \bigcap \sim b$

3. Standardize by renaming all variables so that variables bound by different quantifiers have unique names.

   If we have a statement,

   $((\forall X)\, a(X)\; U\; \forall X\; b(X)) \quad \equiv \quad (\forall X)\, a(X)\; U\; (\forall Y)\, b(Y)$

4. Move all quantifiers to the left without changing their order.

5. Eliminate all existential quantifiers by a process called skolemization.

   $(\forall X)(\exists Y)\; (mother(X, Y))$ is replaced with $(\forall X)\; mother(X, m(X))$

   $(\forall X)(\forall Y)(\exists Z)(\forall W)\; (foo(X, Y, Z, W))$ is replaced with $(\forall X)(\forall Y)(\forall W)\; (foo(X,\, Y,\, f(X, Y),\, W))$

6. Drop all universal quantifiers.

7. Convert the expression to the conjunct of disjuncts form using the following equivalences.

   $aU(bUc) \equiv (aUb)Uc$

   $a \bigcap (b \bigcap c) \equiv (a \bigcap b) \bigcap c$

   $a \bigcap (bUc)$is already in clause form.

   $aU(b \bigcap c) \equiv (aUb) \bigcap (aUc)$

8. Call each conjunct a separate clause.

   For eg.

   $(aUb) \bigcap (aUc)$

   Separate each conjunct as

   aUb and

   aUc

9. Standardize the variables apart again.

$(\forall X)\; (a(X) \bigcap b(X)) \quad \equiv \quad (\forall X)\, a(X) \bigcap (\forall Y)\, b(Y)$

After performing these nine steps, we will get the expression in clause form.

Example

Consider the following expression.

$(\forall X)\; \{\, [a(X) \bigcap b(X)] \;\rightarrow\; [c(X, I) \bigcap (\exists Y)\; ((\exists Z)\; [c(Y, Z) \;\rightarrow\; d(X, Y)\,])\,]\}\; U\; (\forall X)\; [e(X)]$

Convert this expression to clause form.

The above expression can be represented as,

$(\forall X)($

$$
\begin{array}{l}
\quad [a(X) \cap b(X)] \\
\quad \rightarrow \\
\quad [\\
\quad\quad c(X, I) \\
\quad\quad \cap \\
\quad\quad (\exists Y) \\
\quad\quad (\\
\quad\quad\quad (\exists z)[c(Y, Z)] \rightarrow d(X, Y) \\
\quad\quad) \\
\quad] \\
) \\
\cup \\
(\forall X)(e(X))
\end{array}
$$

Step 1. Eliminate the $\rightarrow$.

$$
\begin{array}{l}
(\forall X)(\\
\quad\quad [a(X) \cap b(X)] \\
\quad\quad \rightarrow \\
\quad\quad [\\
\quad\quad\quad c(X, I) \\
\quad\quad\quad \cap \\
\quad\quad\quad (\exists Y) \\
\quad\quad\quad (\\
\quad\quad\quad\quad (\exists z) \neg c(Y, Z) \cup d(X, Y) \\
\quad\quad\quad) \\
\quad\quad] \\
\quad) \\
\quad \cup \\
(\forall X)(e(X))
\end{array}
$$

Again,

$$
\begin{array}{l}
(\forall X)(\\
\quad\quad \neg [a(X) \cap b(X)] \\
\quad\quad \cup \\
\quad\quad [\\
\quad\quad\quad c(X, I)
\end{array}
$$

$$
\begin{array}{l}
\quad\quad\quad\quad \cap \\
\quad\quad\quad\quad (\exists Y) \\
\quad\quad\quad\quad (\\
\quad\quad\quad\quad\quad (\exists z) \neg c(Y,Z) \cup d(X,Y) \\
\quad\quad\quad\quad) \\
\quad\quad\quad] \\
\quad\quad) \\
\quad\quad \cup \\
(\forall X)(e(X))
\end{array}
$$

Thus after step 1, the expression becomes

$$(\forall X)\ (\ \neg[a(X) \bigcap b(X)]\ \cup\ [c(X,I) \bigcap (\exists Y)\ ((\exists Z)\ \neg c(Y,Z)\ \cup\ d(X,Y)\ )\ ])\ U\ (\forall X)\ (e(X))$$

Step 2: Reduce the scope of negation.

$$
\begin{array}{l}
(\forall X)(\\
\quad\quad [\neg a(X) \cup \neg b(X)] \\
\quad\quad \cup \\
\quad\quad [\\
\quad\quad\quad c(X,I) \\
\quad\quad\quad \cap \\
\quad\quad\quad (\exists Y) \\
\quad\quad\quad (\\
\quad\quad\quad\quad (\exists z) \neg c(Y,Z) \cup d(X,Y) \\
\quad\quad\quad) \\
\quad\quad] \\
\quad) \\
\cup \\
(\forall X)(e(X))
\end{array}
$$

After step 2, expression becomes,

$$(\forall X)\ (\ [\neg a(X)\ \cup\ \neg b(X)]\ \cup\ [c(X,I) \bigcap (\exists Y)\ ((\exists Z)\ \neg c(Y,Z)\ \cup\ d(X,Y)\ )\ ])\ U\ (\forall X)\ (e(X))$$

Step 3: Standardize by renaming the variables.

$$
\begin{array}{l}
(\forall X)(\\
\quad\quad [\neg a(X) \cup \neg b(X)] \\
\quad\quad \cup \\
\quad\quad [
\end{array}
$$

$$
\begin{array}{l}
\quad c(X,I) \\
\quad \cap \\
\quad (\exists Y) \\
\quad (\\
\quad\quad (\exists z) \neg c(Y,Z) \cup d(X,Y) \\
\quad) \\
\quad] \\
) \\
\cup \\
(\forall W)(e(W))
\end{array}
$$

That is,

$$(\forall X)\ (\ [\neg a(X)\ \cup\ \neg b(X)]\ \cup\ [c(X,I) \bigcap (\exists Y)\ ((\exists Z)\ \neg c(Y,Z)\ \cup\ d(X,Y)\ )\ ])\ U\ (\forall W)\ (e(W))$$

Step 4: Move all quantifiers to the left.

$$(\forall X)(\exists Y)(\exists Z)(\forall W)\ (\ [\neg a(X)\ \cup\ \neg b(X)]\ \cup\ [c(X,I) \bigcap (\ \neg c(Y,Z)\ \cup\ d(X,Y)\ )\ ])\ U\ e(W)$$

Step 5: Eliminate existential quantifiers.

$$(\forall X)(\forall W)\ (\ [\neg a(X)\ \cup\ \neg b(X)]\ \cup\ [c(X,I) \bigcap (\ \neg c(f(X),g(X))\ \cup\ d(X,f(X))\ )\ ])\ U\ e(W)$$

Step 6: Drop all universal quantifiers.

$$[\neg a(X)\ \cup\ \neg b(X)]\ \cup\ [c(X,I) \bigcap (\ \neg c(f(X),g(X))\ \cup\ d(X,f(X))\ )\ ]\ U\ e(W)$$

Step 7: Convert the expression to conjunct of disjuncts form.

$$
\begin{array}{l}
[\neg a(X)\ \cup\ \neg b(X)\ \cup\ c(X,I)\ U\ e(W)] \bigcap \\
\ [\neg a(X)\ \cup\ \neg b(X)\ \cup\ \neg c(f(X),g(X))\ \cup\ d(X,f(X))\ U\ e(W)]
\end{array}
$$

Step 8: Call each conjunct a separate clause.

1 $\Rightarrow \neg a(X)\ \cup\ \neg b(X)\ \cup\ c(X,I)\ U\ e(W)$

2 $\Rightarrow \neg a(X)\ \cup\ \neg b(X)\ \cup\ \neg c(f(X),g(X))\ \cup\ d(X,f(X))\ U\ e(W)$

Step 9: Standardize the variables apart again.

1 $\Rightarrow \neg a(X)\ \cup\ \neg b(X)\ \cup\ c(X,I)\ U\ e(W)$

2 $\Rightarrow \neg a(V)\ \cup\ \neg b(V)\ \cup\ \neg c(f(V),g(V))\ \cup\ d(V,f(V))\ U\ e(Z)$

This is the clause form generated.

10 The Resolution Proof Procedure

Let we are given the following axioms.

1. $bUc \rightarrow a$
2. b
3. $d \bigcap e \rightarrow c$
4. eUf
5. $d \bigcap \vdash f$

We need to prove ‘a’ from these axioms.

Step 1:

First convert the above predicates to clause form.

We get

1 $\Rightarrow$

$b \bigcap c \rightarrow a$

$\vdash (b \bigcap c) U a$

$\vdash bU \vdash cUa$

$aU \vdash bU \vdash c$

3 $\Rightarrow$

$d \bigcap e \rightarrow c$

$cU \vdash dU \vdash e$

We get the following clauses.

$aU \vdash bU \vdash c$

b

$cU \vdash dU \vdash e$

eUf

d

$\vdash f$

Step 2:

The goal to be proved, a, is negated and added to the clause set.

Now we have,

$a\,U\ \vdash b\,U\ \vdash c$

b

$c\ U\ \neg\ d\ U\ \neg\ e$

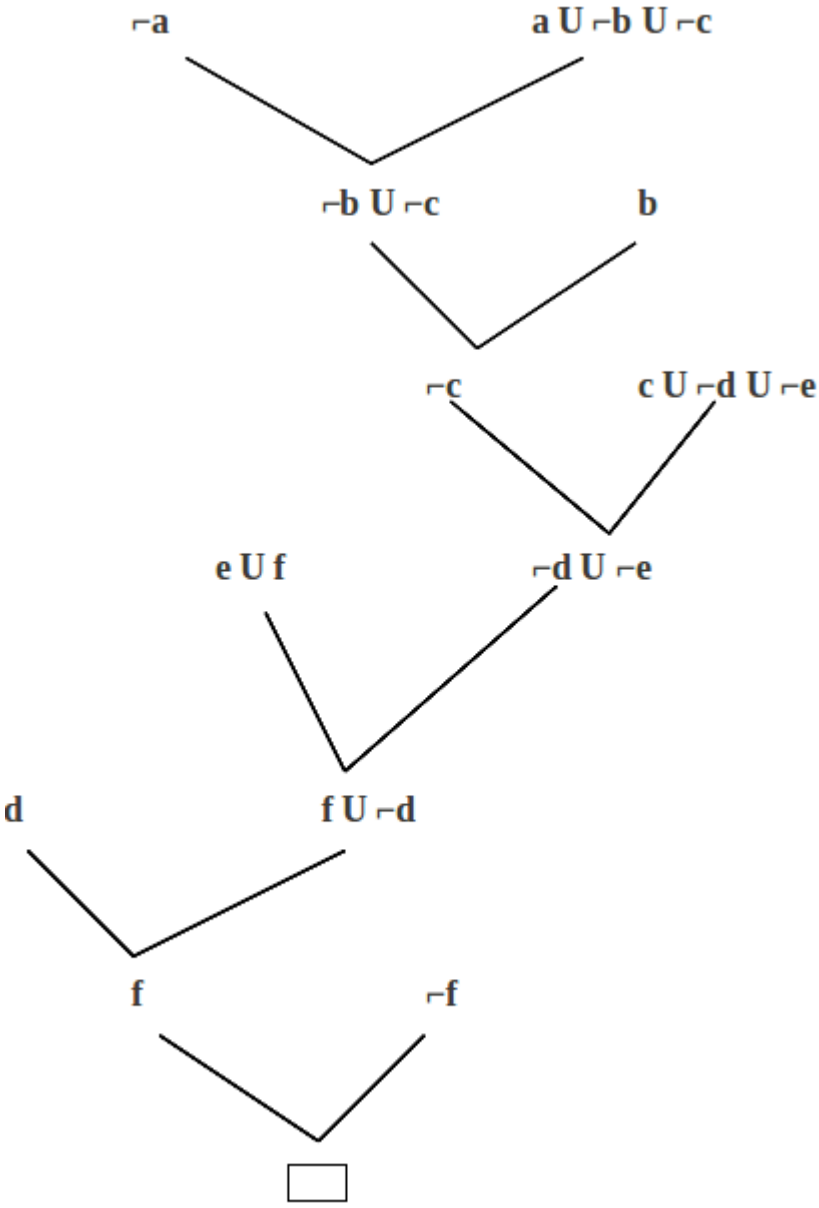
$e\ U\ f$

d

$\neg\ f$

$\neg\ a$

Step 3: Resolve the caluses together.



Step 4 and Step 5:

After resolving the clauses we get the empty clause. This means that a is true.

Example 2:

Consider the facts.

Anyone passing history exams and winning the lottery is happy. But anyone who studies or is lucky can pass all his exams. John did not study but he is lucky. Anyone who is lucky wins the lottery. Is john happy?

First the write the sentences in predicate form:

Anyone passing his history exams and winning the lottery is happy.

$\forall X\ (\ pass(X, history) \bigcap win(X, lottery)\ \rightarrow\ happy(X)\ )$

Anyone who studies or is lucky can pass all his exams.

$\forall X\ \forall Y\ (\ study(X)\ U\ lucky(X)\ \rightarrow\ pass(X, Y)\ )$

John did not study but he is lucky.

$\vdash study(john) \bigcap lucky(john)$

Anyone who is lucky wins the lottery.

$\forall X\ (\ lucky(X)\ \rightarrow\ win(X, lottery)\ )$

Step 1:

After changing these 4 predicate statements to clause form, we get

$\vdash pass(X, history)\ U\ \vdash win(X, lottery)\ U\ happy(X)$

$\vdash study(Y)\ U\ pass(Y, Z)$

$\vdash lucky(V)\ U\ pass(V, W)$

$\vdash study(john)$

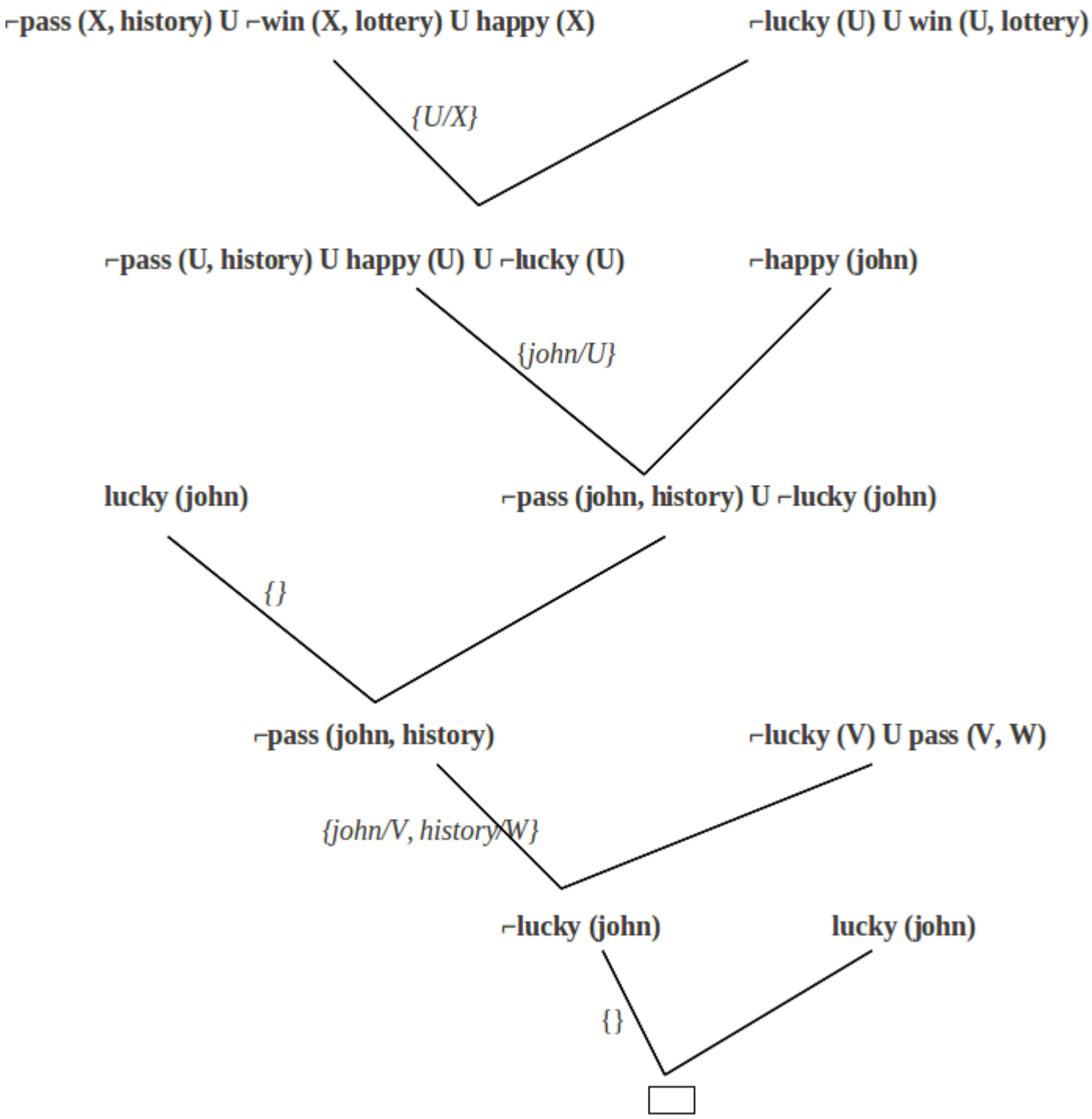
$lucky(john)$

$\vdash lucky(U)\ U\ win(U, lottery)$

Into these clauses is entered, in clause form, the negation of the conclusion.

$\vdash happy(john)$

The derivation of the contradiction is given below.



We get the empty clause. This proves that John is happy.

Example 3: Consider the following set of facts.

All people who are not poor and are smart are happy. Those people who read are not stupid. John can read and is wealthy. Happy people have exciting lives. Can anyone be found with an exciting life?

We assume

$$\forall X \ (\ smart(X) \ \equiv \neg \ stupid(X)\ )$$

and

$$\forall Y \ (\ wealthy(Y) \ \equiv \neg \ poor(Y)\ )$$

$$\forall X \ (\ \neg \ poor(X) \bigcap smart(X) \ \rightarrow \ happy(X)\ )$$

$$\forall Y \ (\ read(Y) \ \rightarrow \ smart(Y)\ )$$

$$read(john) \bigcap \neg \ poor(john)$$

$$\forall Z \ (\ happy(Z) \ \rightarrow \ exciting(Z)\ )$$

The negation of the conclusion is:

$$\neg \ \exists W \ (\ exciting(W)\ )$$

After transforming these in to clause form we get,

$$poor(X) \ U \ \neg \ smart(X) \ U \ happy(X)$$

$$\neg \ read(Y) \ U \ smart(Y)$$

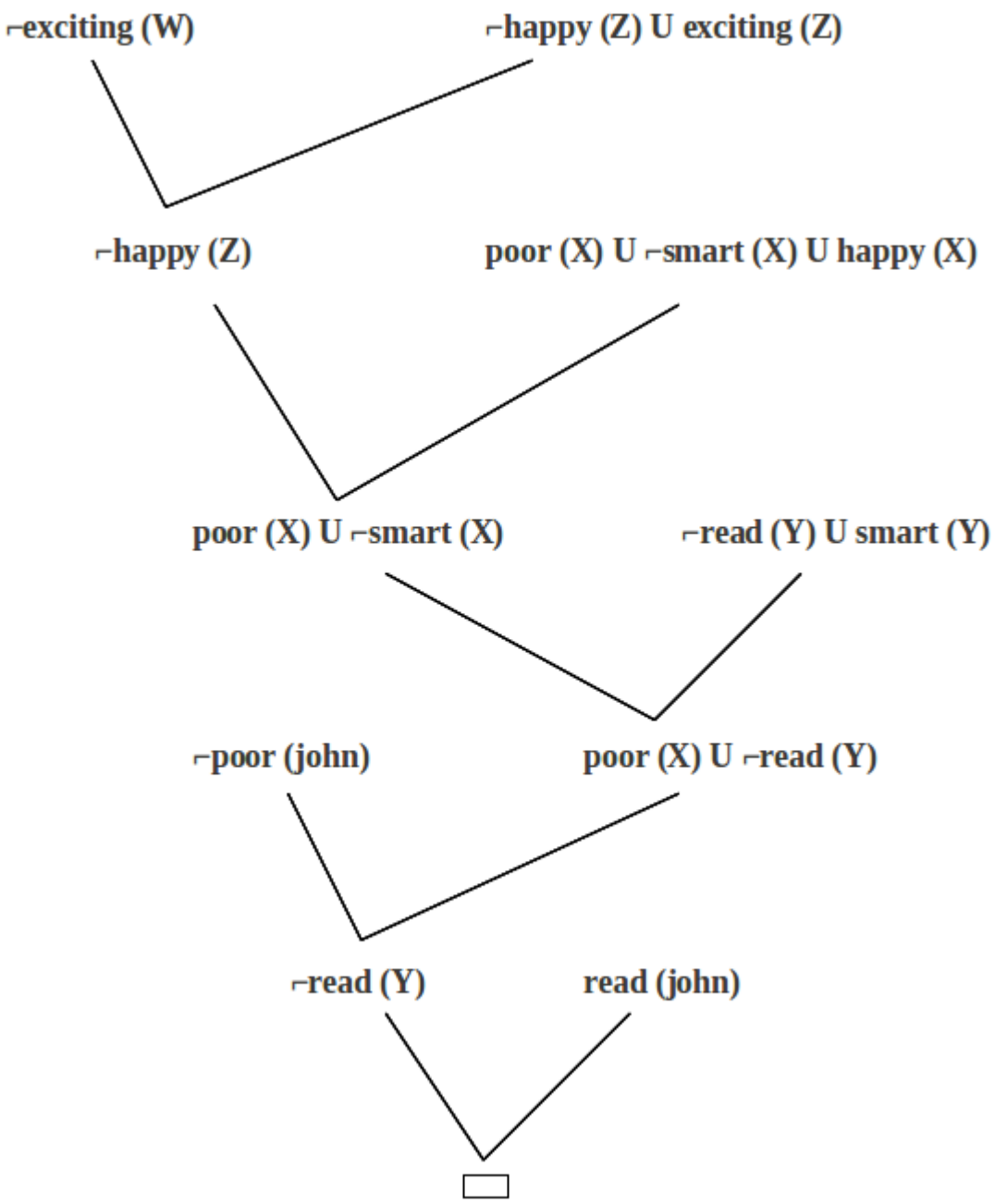
$$read(john)$$

$$\neg \ poor(john)$$

$$\neg \ happy(Z) \ U \ exciting(Z)$$

$$\neg \ exciting(W)$$

The resolution proof for this is given below.



Finally we get the empty clause.

This proves that some one can be found with an exciting life.

.

State and explain Resolution theorem. (4marks)[MGU/May2008]

Explain Resolution with a suitable example.

How can resolution be used to show that a sentence is valid/ Unsatisfiable? Explain. (12marks)[MGU/Jan2007]

What is a resolution and its closure? (4marks)[MGU/June2006]

Part V. Forward Chaining and Backward Chaining

[Russel2003]

The completeness of resolution makes it a very important inference method. In many practical situations, however, the full power of resolution is not needed. Real world knowledge bases often contain only clauses of a restricted kind called Horn clauses. A Horn clause is a disjunction of literals of which at most one is positive. Inference with Horn clauses can be done through the forward and backward chaining algorithms. In both these techniques, inference steps are obvious and easy to follow for humans.

11 Forward Chaining

The forward chaining method is very simple:

Start with the atomic sentences in the knowledge base and apply modus Ponens in the forward direction, adding new atomic sentences, until no further inferences can be made.

In many cases, the reasoning with forward chaining can be much more efficient than resolution theorem proving.

Consider the following problem:

The law says that it is a crime for an American to sell weapons to hostile nations. The country Iran, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West , who is American.

We need to prove that West is a criminal.

First we will represent these facts as first order definite clauses.

"...it is a crime for an American to sell weapons to hostile nations":

$American(x) \bigcap Weapon(y) \bigcap Sells(x, y, z) \bigcap Hostile(z) \rightarrow Criminal(x)$ $\Longrightarrow$rule 1

"Iran....has some missiles."

$\exists x \; Owns(Iran, x) \bigcap Missile(x)$

$\Longrightarrow$

$Owns(Iran, M1)$ $\Longrightarrow$rule 2

$Missile(M1)$ $\Longrightarrow$rule 3

"All of its missiles were sold to it by Colonel West"

$Missile(x) \bigcap Owns(Iran, x) \rightarrow Sells(West, x, Iran)$ $\Longrightarrow$ rule 4

Here we must provide the rule that

missiles are weapons.

$Missile(x) \rightarrow Weapon(x)$ $\Longrightarrow$rule 5

Here we must provide the fact that

enemy of America counts as hostile.

$Enemy(x, America) \rightarrow Hostile(x)$ $\Longrightarrow$rule 6

“West, who is American. . . ”

$American(West)$ $\Longrightarrow$ rule 7

“The country Iran, an enemy of America. . . ”

$Enemy(Iran, America)$ $\Longrightarrow$rule 8

Now we have the all the predicates.

Starting from the known facts, the forward chaining method triggers all the rules whose premises are satisfied, adding their conclusions to the known facts. This process repeats until the query is answered or no new facts are added.

In the first phase, rule 1 has unsatisfied premises.

Rule 2 $\Longrightarrow Owns(Iran, M1)$

Rule 3 $\Longrightarrow Missile(M1)$

Rule 4 $\Longrightarrow Missile(x) \bigcap Owns(Iran, x) \rightarrow Sells(West, x, Iran)$

Rule 4 is satisfied with {x/M1}, and

$Sells(West, M1, Iran)$ is added. $\Longrightarrow$Rule 9

Rule 3 $\Longrightarrow Missile(M1)$

Rule 5 $\Longrightarrow Missile(x) \rightarrow Weapon(x)$

Rule 5 is satisfied with {x/M1}, and

$Weapon(M1)$ is added. $\Longrightarrow$Rule 10

Rule 8 $\Longrightarrow Enemy(Iran, America)$

Rule 6 $\Longrightarrow Enemy(x, America) \rightarrow Hostile(x)$

Rule 6 is satisfied with {x / Iran},

$Hostile(Iran)$ is added. $\Longrightarrow$Rule 11

Rule 7 $\Longrightarrow American(West)$

Rule 9 $\Longrightarrow Sells(West, M1, Iran)$

Rule 10 $\Longrightarrow Weapon(M1)$

Rule 11 $\Longrightarrow Hostile(Iran)$

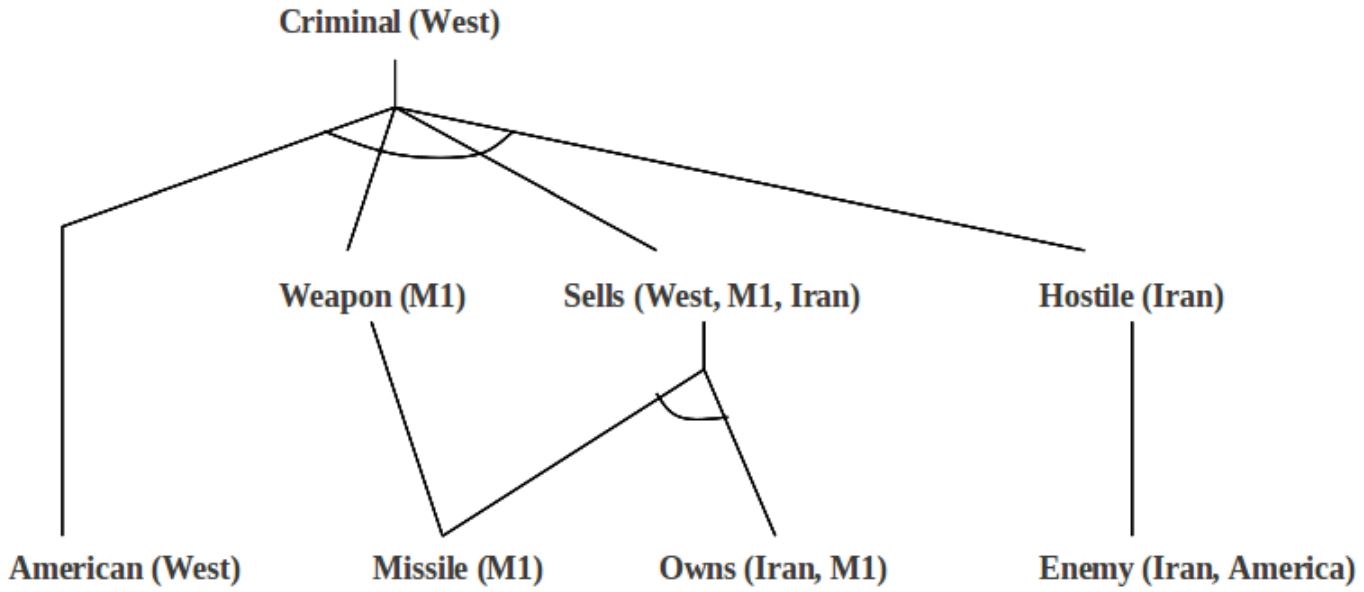
Rule 1 $\Longrightarrow American(x) \bigcap Weapon(y) \bigcap Sells(x, y, z) \bigcap Hostile(z) \rightarrow Criminal(x)$

In the second phase, rule 1 is satisfied with {x / West, y / M1, z / Iran),

$Criminal(West)$ is added.

The following figure shows the proof tree generated.

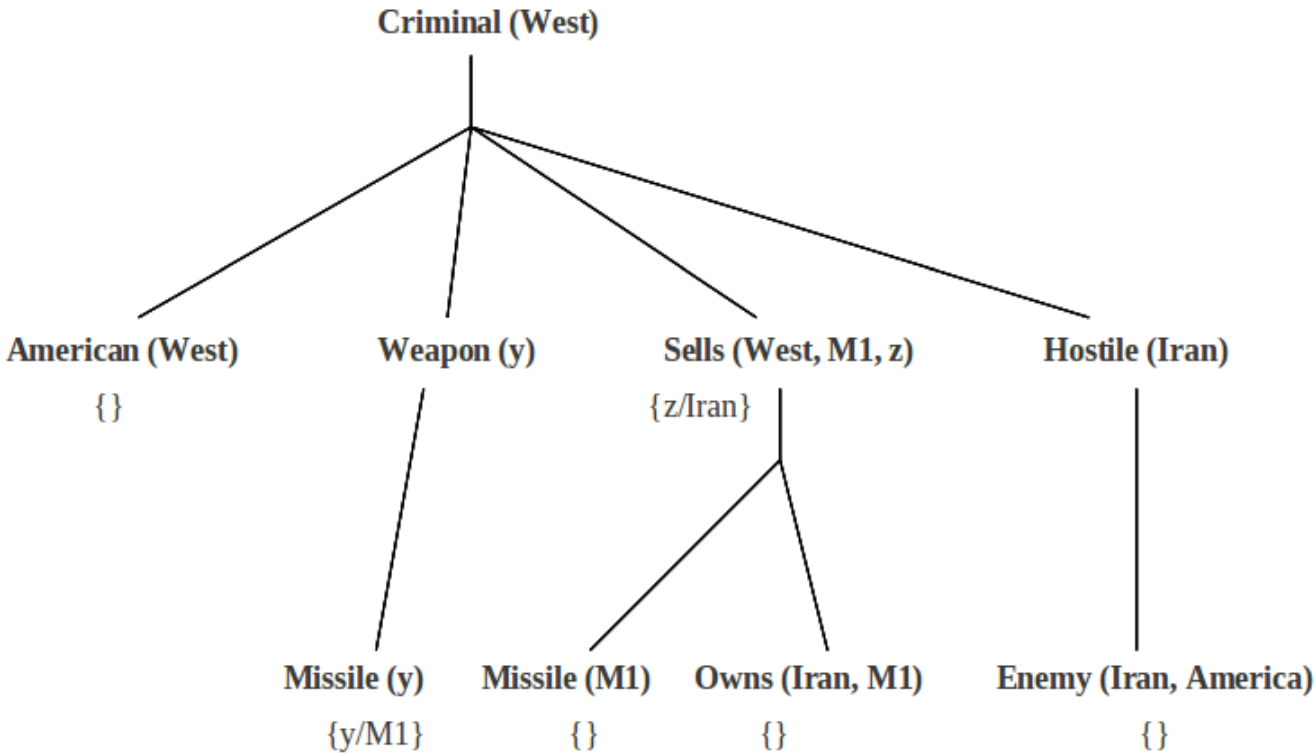
The initial facts appear at the bottom level, facts inferred on the first iteration in the middle level, and facts inferred on the second phase at the top level.



12 Backward Chaining

Many inference techniques use backward chaining approach. These algorithms work backward from the goal, chaining through rules to find known facts that support the proof. The list of goals can be thought of as a stack waiting to be worked on: if all of them can be satisfied, then the current branch of the proof succeeds. The backward chaining algorithm takes the first goal in the list and finds every clause in the knowledge base whose positive literal or head, unifies with the goal. Each such clause creates a new recursive call in which the premise, or body, of the clause is added to the goal stack.

The following figure shows the proof tree for deriving $Criminal(West)$ from the previous example.



.

Explain forward and backward chaining with suitable examples. (12marks)[MGU/May2008]

Explain, in detail, the forward chaining.

Compare forward and Backward chaining. (4marks)[MGU/June2007]

Differentiate between forward and backward chaining. (4marks)[MGU/Jan2007]

With an example, illustrate the the use of forward chaining algorithm. (12marks)[MGU/June2006]

Explain forward and backward chaining with suitable examples. (12marks)[MGU/Nov2010]

References

.

1. Rich, E; Knight, K. (1991). Artificial Intelligence. TMH.
2. Luger, G. (2005). Artificial Intelligence. Pearson Education.
3. Russel, S; Norvig, P. (2006). Artificial Intelligence. Pearson Education.