Knowledge Representation and Reasoning

UNIT 3RD

PRASHANT TOMAR

Intelligent agents should have capacity for

- **Perceiving**, that is, acquiring information from environment,
- **Knowledge Representation**, that is, representing its understanding of the world,
- **Reasoning**, that is, inferring the implications of what it knows and of the choices it has, and
- **Acting**, that is, choosing what it want to do and carry it out.

Representation of knowledge and the reasoning process are central to the entire field of artificial intelligence. The primary component of a knowledge-based agent is its knowledge-base. A knowledge-base is a set of sentences. Each sentence is expressed in a language called the knowledge representation language. Sentences represent some assertions about the world. There must mechanisms to derive new sentences from old ones. This process is known as inferencing or reasoning.

Logic is the primary vehicle for representing and reasoning about knowledge. A logic consists of two parts-

- **A language (Formal Language)** and
- **A method of reasoning.**

The logical language has two aspects-

1. **Syntax and**
2. **Semantics**

Thus, to specify or define a particular logic, one needs to specify three things:

- **Syntax:** The atomic symbols of the logical language, and the rules for constructing well- formed, non-atomic expressions (symbol structures) of the logic. Syntax specifies the symbols in the language and how they can be combined to form sentences. **Hence facts about the world are represented as sentences in logic.**
- **Semantics:** The meanings of the atomic symbols of the logic, and the rules for determining the meanings of non-atomic expressions of the logic. It specifies what facts in the world a sentence refers to. Hence, also specifies how you assign a truth value to a sentence based on its meaning in the world. A fact is a claim about the world, and may be true or false.
- **Syntactic Inference Method:** The rules for determining a subset of logical expressions, called theorems of the logic. It refers to mechanical method for computing (deriving) new (true) sentences from existing sentences.

Facts are claims about the world that are True or False, whereas a representation is an expression (sentence) in some language that can be encoded in a computer program and stands for the objects and relations in the world.

There are a number of logical systems with different syntax and semantics.

- **Propositional logic**
- **First order predicate logic**
- **Temporal**
- **Modal**
- **Higher order logics**
- **Non-monotonic**

• **Propositional logic :** All objects described are fixed or unique

"John is a student" student(john)

Here John refers to one unique person.

• **First order predicate logic :** Objects described can be unique or variables to stand for a unique object

"All students are poor"

For All(S) [student(S) -> poor(S)]

Here **S** can be replaced by many different unique students.

• **Temporal** : Represents truth over time.

• **Modal :** Represents doubt

• **Higher order logics :** Allows variable to represent many relations between objects

• **Non-monotonic** :Represents defaults

- **Propositional logic:** Propositional logic consists of:

➢ The **logical values true** and **false** (T and F)

➢ **Propositions**: "Sentences," which are atomic (that is, they must be treated as indivisible units, with no internal structure), and Have a single logical value, either **true** or **false.**

➢ **Operators**, both unary and binary; when applied to logical values, yield logical values

- The usual operators are **and**, **or**, **not**, and **implies**
- In **propositional logic (PL)** an user defines a set of propositional symbols, like **P** and **Q**. User defines the semantics of each of these symbols. For example,
- **P** means "**It is hot**"
- **Q** means "**It is humid**"
- **R** means "**It is raining**"

A sentence (also called a formula or well-formed formula or wff) is defined as:

1. A symbol
2. 2. If S is a sentence, then ~S is a sentence, where "~" is the "not" logical operator
3. If S and T are sentences, then
   - (S v T),
   - (S ^ T),
   - (S => T), and
   - (S <=> T) are sentences, where the four logical connectives correspond to "**or**," "**and**," "**implies**," and "**if and only if**," respectively .

Examples of PL sentences:

- (P ^ Q) => R (here meaning "If it is hot and humid, then it is raining")
- Q => P (here meaning "If it is humid, then it is hot")
- Q (here meaning "It is humid.")

Implication (-->) means

P→Q(for sufficient condition)
if P is true then Q is true but if Q is true then P is not true
Ex. 1. If it is rain (T) then the road are wet(T).

What about of this proposition:

2. If the road are wet then it is rain (not implication)

Equivalence (<->) or Biconditional

P <->Q it means P→Q and Q→P

P is true then Q is true but if Q is true then P is true

Here are the binary operators that are traditionally used:

X	Y	AND $X \wedge Y$	OR $X \vee Y$	IMPLIES $X \Rightarrow Y$	BICONDITIONAL $X \Leftrightarrow Y$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

Logical expressions:

- All logical expressions can be computed with some combination of and ($\wedge$), or ($\vee$), and not ($\neg$) operators
- For example, logical implication can be computed this way:

X	Y	$\neg X$	$\neg X \vee Y$	$X \Rightarrow Y$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Notice that $\neg X \vee Y$ is equivalent to $X \Rightarrow Y$

Worlds:

A world is a collection of prepositions and logical expressions relating those prepositions

Example:

Propositions: JohnLikesMary, MaryIsFemale, MaryIsRich

Expressions:

$$\text{MaryIsFemale} \wedge \text{MaryIsRich} \Rightarrow \text{JohnLikesMary}$$

A proposition “says something” about the world, but since it is atomic, propositions tend to be very specialized and inflexible

Models:

- A model is an assignment of a truth value to each proposition, for example:
  - JohnLikesMary: T, MaryIsFemale: T, MaryIsRich: F
- An expression is satisfiable if there is a model for which the expression is true
  - For example, the above model satisfies the expression

    $MaryIsFemale \wedge MaryIsRich \Rightarrow JohnLikesMary$
- An expression is valid if it is satisfied by *every* model
  - This expression is *not* valid:

    $MaryIsFemale \wedge MaryIsRich \Rightarrow JohnLikesMary$

    because it is not satisfied by this model:

    JohnLikesMary: F, MaryIsFemale: T, MaryIsRich: T
  - But this expression *is* valid:

    $MaryIsFemale \wedge MaryIsRich \Rightarrow MaryIsFemale$

Inference rules in propositional logic

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$
$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$
$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$
$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$
$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$
$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$
$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan}$$
$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan}$$
$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$
$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

- **Ex. For propositional logic :**

1. Anil is intelligent (T/F)
2. Anil is hardworking.(T/F)

- Both are proposition

Intelligent(Anil) : where Anil is Object

Hardworking(Anil) : And hardworking or intelligent is the relation or function

- If Anil is intelligent and Anil is hardworking then Anil scores high mark.
- **It is called compound proposition**

- **First Order Predicate logic:** First order Predicate logic(FOPL) is one of the oldest and most important representation schemes for the knowledge. In FOPL, statements from a natural like English are translated into symbolic structure comprised of predicates, variable, contants, quantifiers and logical connectives.

- We can easily represent real –world facts as logical propositional written as well-formed formulas (w.ffs) in propositional logic. But some limitation in propositional logic. Suppose we want to represent the obvious fact stated by the classical sentence.

Socrates is a Man

we could write

SOCRATESMAN

Plato is Man

PLATOMAN

Both are separated assertion, we can represent these fact as-

MAN(SOCRATES)
MAN(PLATO)

Now we are in even more difficulties if we try to represent the equally classic sentence.
Ex. All men are mortal.

MOTALMAN

But that fails to capture the relationship b/t any individual being a Man & that individual being Mortal. So we need to write separate statement about the mortality of every know man ; to need **variable** and **quantifiers ,** so use the first order predicates logic. In predicate logic, we can represent real-world facts as statement written as wff's.

Syntax of First-Order Logic : The symbols and rules of combination permitted in FOPL are defined as follow……

1.Connectives: These are five basic connective symbol as follows:

$\wedge$ AND

$\vee$ OR

$\neg$ NOT

$\Rightarrow$IMPLICATION

$\Leftrightarrow$EQUIVALENCE (if and only if)

2. **Quantifiers:** Each quantifier defines a variable for the duration of the following expression, and indicates the truth of the expression…

The two quantifier symbol are-

- **Existential quantifier** "there exists" (∃)
  The expression is true for at least one value of the variable
- **Universal quantifier** "for all" ∀
  The expression is true for every possible value of the variable

3. **Constants**: Constants are fixed-value terms that belong to a given domain of discourse . Ex- Anil, 2, a,b,c…

4. Predicates: predicates symbols represent relation or properties of an object that is true and false. Capital letter and proper parenthesis such as P(x), BROTHER-OF(x,y), FATHER-OF (x,y) , >, …

5. **Function**: Function symbols represent relations defined on domain D. They Map n element (n>=0) to a single element of the domain. Ex. Sqrt, LeftArmOf, …

6. **Variables**: variables are term that are assume different values over a given domain. Ex.-x, y, z,a, b, …

Example:

1.Caesar was a ruler

Ruler(Caesar)

2. Every one is loyal to someone.

$\forall x: \exists x : loyal(x, y)$

3. All men are mortal

$\forall x: men(x) \rightarrow mortal(x)$

Everyone likes Ice-cream

$\forall x,$ likes(x, Ice-cream)

Someone likes Ice-cream

$\exists x,$ likes(x, Ice-cream)

All children like Ice-cream

$\forall x,$ child(x) $\Rightarrow$ likes(x, Ice-cream)

Everyone likes Ice-cream unless they are allergic to it

$\forall x,$ likes(x, Ice-cream) $\vee$ allergic(x, Ice-cream)

$\forall x, \neg$allergic (x, Ice-cream) $\Rightarrow$ likes(x, Ice-cream)

Properties of statement:

1. **Satisfiable:** A statement is satisfiable if there is some interpretation for which it is true.
2. **Valid or tautology:** A statement is valid if it true for every interpretation.
3. **Contradiction:** A statement is contradictory (unsatisfiable) if there is no interpretation for which it is true.
4. **Equivalence:** Two statement are equivalent if they have the same truth value under every interpretation .
5. **Logical consequences:** The sentence S is logical consequence of S1,S2….Sn if and only if (S1 $\wedge$**S2** $\wedge$**…….Sn)-> S is valid.**

6. **Inference Rules:** Inference means derived new fact from existing one.

Some Example of inference rule are follow…….

1. **MODUS PONENS:** From P and P→ Q, we can conclude Q . This is written as

P

P→Q

C:Q Where C for conclude

Ex.

All men are mortal : P

Socrates is a man: P→Q

Conclude: Q: Socrates is mortal

Notes: if premises are true, then the conclusion is necessarily true too.

2. Chain rule : From P→Q and Q→P we can conclude P→R

P→Q

Q→P

C: P→R

Properties of Quantifiers:

$\forall x\ \forall y$ is the same as $\forall y\ \forall x$

$\exists x\ \exists y$ is the same as $\exists y\ \exists x$

$\exists x\ \forall y$ is not the same as $\forall y\ \exists x$

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

"There is a person who loves everyone in the world"

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

"Everyone in the world is loved by at least one person"

Nesting Quantifiers:

Everyone likes some kind of food

$\forall y \exists x,\ food(x) \wedge likes(y, x)$

There is a kind of food that everyone likes

$\exists x \forall y,\ food(x) \wedge likes(y, x)$

Someone likes all kinds of food

$\exists y \forall x,\ food(x) \wedge likes(y, x)$

Every food has someone who likes it

$\forall x \exists y,\ food(x) \wedge likes(y, x)$

Example:

Quantifier Duality

Not everyone like McDonalds

$\neg(\forall x,\ \text{likes}(x, \text{McDonalds}))$

$\exists x, \neg\text{likes}(x, \text{McDonalds})$

No one likes McDonalds

$\neg(\exists x,\ \text{likes}(x, \text{McDonalds}))$

$\forall x, \neg\text{likes}(x, \text{McDonalds})$

1. Marcus was a man.
Man(marcus)

2. Marcus was a Pompeian
pompeian(marcus)

3. All pompeians were Romans
$\forall$x :pompeians(x)→romans(x)

4. Caesar was a ruler
caesar(ruler)

5. All Romans were either loyal to Caesar or hated him. -
$\forall$x :roman(x)→loyalto (x, caesar) v hate(x,caesar)

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 : $\exists$y person(x) $\wedge$ruler(y) $\wedge$tryassassinate(x,y)→ $\neg$loyalto(x,y)

8. Marcus tried to assassinate Caesar. ---- tryassassinate(marcus,caesar)

Conjunctive normal form(CNF): A sentence is written in conjunctive normal form look likes:…….
(AvB ¬C) ∧(BvD) ∧(¬ A) ∧(B **v**C)

(AvB ¬C) is a clause
Its outermost structure is conjunction. It's a conjunction of multiple units.these unit called "**clauses"**

A clause is the disjunction of many things. The unit that make up a clause are called literals. A literal is either a variable or negation of variable **.**
Ex. A,B , ¬C are literal
You can take any sentence in propositional logic write it in conjunctive normal form(CNF).

Convert to CNF:

1. **Eliminate arrows using definitions..**

A→B = ¬A v B

1. **Drive negation using de Morgan's Law**

¬(A v B)= ¬A ∧ ¬B

¬(A ∧ B)= ¬A v ¬ B

3. Distribute OR over AND

A v(B ∧C)=> (A v B) ∧(A v C)

Every sentence can be converted to CNF, but it may grow exponentially in size.

Ex.

(A v B) →(C →D)

¬(A v B) v (¬C v D)

(¬A ∧ ¬B) v (¬C v D)

(¬A v ¬C v D) ∧ (¬B v ¬ C v D) ……..CNF

Clausal form (which is also sometimes called "prenex normal form") is like CNF in its outer structure (a conjunction of disjunctions, or an "and" of "ors"). But it has no quantifiers.

FOPL Convert to clause form:

Rules:

1. Eliminate implication (arrow) using the fact..

   (X <=> Y) by expression ((X => Y) ^ (X => Y))

   that $x \Rightarrow y$ is equivalent to $\neg x \vee y$

2. Move the negation symbol to individual term using deMorgan's law

$\neg(\neg p) \equiv p$

$\neg(a \wedge b) \equiv (\neg a \vee \neg b)$

$\neg(a \vee b) \equiv (\neg a \wedge \neg b)$

$\neg\forall x, p(x) \equiv \exists x, \neg p(x)$

$\neg\exists x, p(x) \equiv \forall x, \neg p(x)$

3. Standardize variable : rename all variables so that each quantifier has its own unique variable name. since variables are just dummy name, this process can not effect the truth value of w.ff.

$\forall x: P(x) \ v \ \forall x: Q(x)$ would be written as
$\forall x: P(x) \ v \ \forall y: Q(y)$

4.Move all the quantifier to the left of the formula without changing their relative order. At this stage the formula is known as **prenex normal form.** E x.
$\forall x: P(x) \ v \ \forall y: Q(y)$

$\forall x: \forall y: P(x) \ v \ Q(y)$

5. Eliminate existential quantification by introducing Skolem functions(the function that is eliminate the existential quantifier).
i- If the leftmost quantifier in an expression is an existential quantifier the replace all occurrences of the variable with a arbitrary constant not appearing else where in expression and delete the quantifier.
$\exists x, \forall y \ (P(x ,y)$ is written as $\forall y \ (P(a ,y)$ where 'a' is constant not appearing in formula

ii- For each existential quantifier preceded by one or more universal quantifier , replace all occurrence of the existential quantifier variable by a function symbol not appearing else where in expression., And delete the existential quantifier.

$\forall x , \forall y, \exists z, (P(x , y) \text{ v } Q(z)$ is written as $\forall x , \forall y \ P(x , y) \text{ v } Q(f(x , y))$ ..

This function called **Skolem functions**, And argument of the function should math all variable in universal quantifier

6. Drop all universal quantifier put the remaining expression in CNF.

7. Distribute "and" over "or" to get a conjunction of disjunctions called conjunctive normal form. convert (P ^ Q) v R to (P v R) ^ (Q v R) convert (P v Q) v R to (P v Q v R)

8. Split each conjunct into a separate clause, which is just a disjunction ("or") of negated and non-negated predicates, called literals. This expression is said is to be **clausal form**.

9. Standardize variables in the set of clauses its means rename the variable so that no two clause make reference to the same variable

Example: All Romans who know Marcus either hate Caesar or think that anyone who hates anyone is crazy

$\forall x, [\ Roman(x) \wedge know(x, Marcus)\ ] \Rightarrow$
$[\ hate(x, Caesar) \vee (\forall y, \exists z, hate(y, z) \Rightarrow thinkCrazy(x, y))]$

1. $\forall x, \neg[\ Roman(x) \wedge know(x, Marcus)\ ] \vee$
   $[hate(x, Caesar) \vee$
   $(\forall y, \neg(\exists z, hate(y, z)) \vee thinkCrazy(x, y))]$

2. $\forall x, [\ \neg Roman(x) \vee \neg know(x, Marcus)\ ] \vee$
   $[hate(x, Caesar) \vee$
   $(\forall y, \forall z, \neg hate(y, z) \vee thinkCrazy(x, y))]$
3. Not necessary in our running example

4. $\forall x, \forall y, \forall z, [\neg Roman(x) \vee \neg know(x, Marcus)] \vee [hate(x, Caesar) \vee (\neg hate(y, z) \vee thinkCrazy(x, y))]$

5. Not necessary in our running example

6. $[\neg Roman(x) \vee \neg know(x, Marcus)] \vee [hate(x, Caesar) \vee (\neg hate(y, z) \vee thinkCrazy(x, y))]$

7. $\neg Roman(x) \vee \neg know(x, Marcus) \vee hate(x, Caesar) \vee \neg hate(y, z) \vee thinkCrazy(x, y)$

8. Not necessary in our running example

9. Not necessary in our running example

Final result:

$\neg Roman(x) \vee \neg know(x, Marcus) \vee hate(x, Caesar) \vee \neg hate(y, z) \vee thinkCrazy(x, y)$

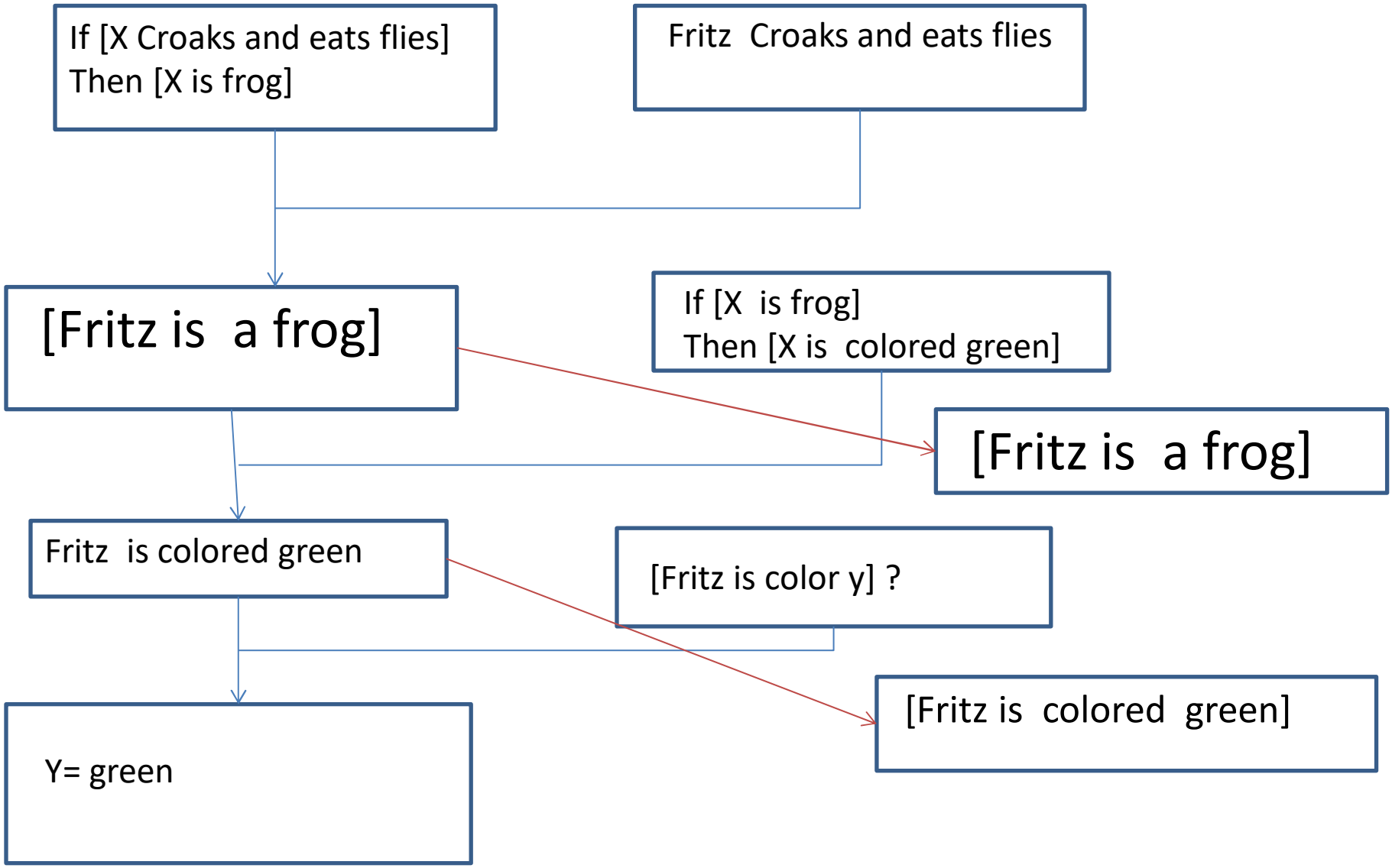
Forward chaining and Backward chaining:

You have a collection of logical expressions (premises), and you are trying to prove some additional logical expression (the conclusion). There is two way in which rules can be used in rule based system to draw a inference one is called **forward chaining** or **forward deduction** and other is called **backward chaining** or **backward deduction.**

1. **Forward chaining** : It is start with the available data and use inference rules to extract more data and stop if one of your results is the conclusion you want.
   So **forward chaining is a data driven method of deriving a particular goal from given a knowledge and set of inference rules.**
   A search control method is needed to select which element(s) of the knowledge base to apply the inference rule to at any point in deduction.

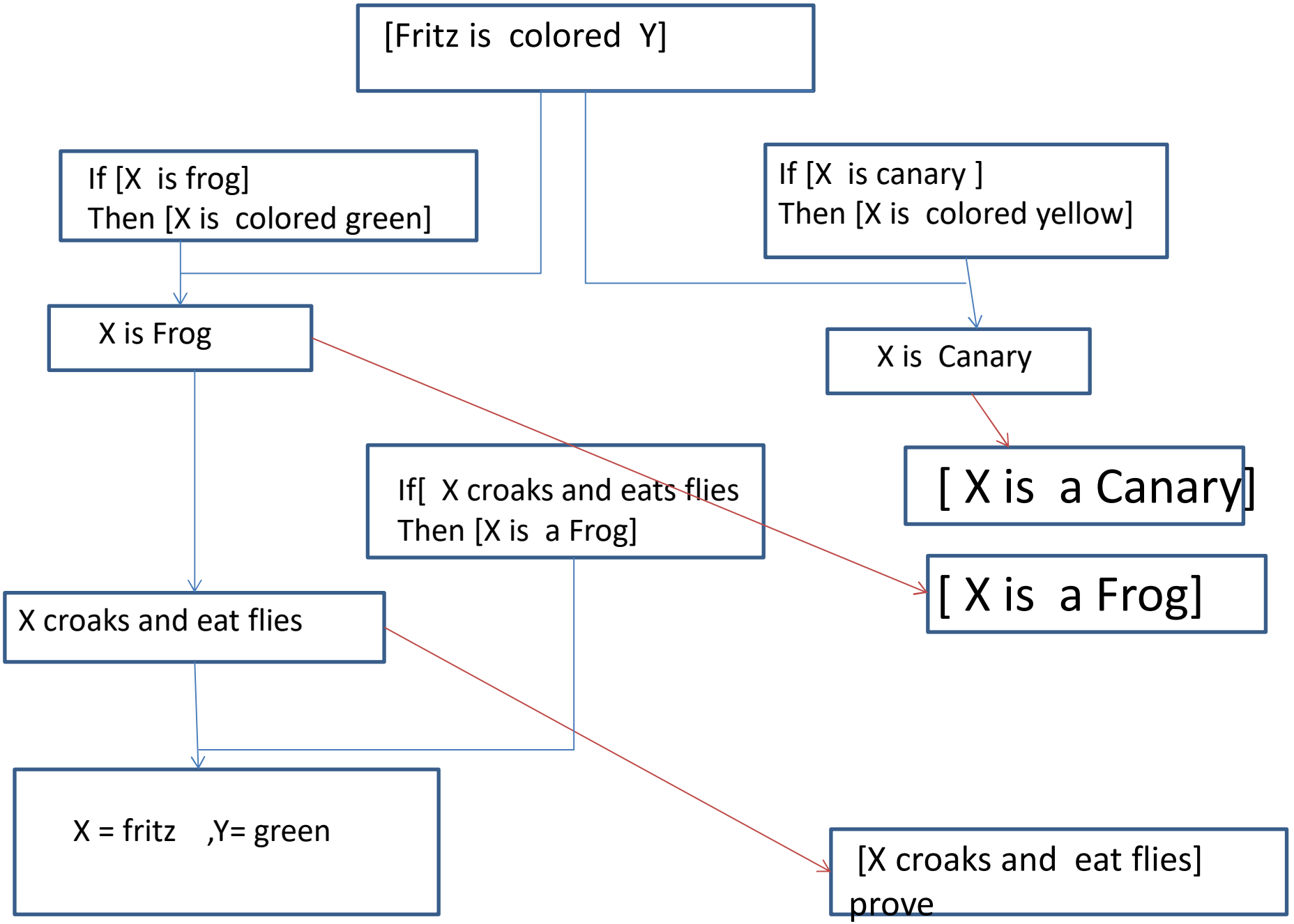
Example: suppose that the goal is to conclude the color of a pet named Fritz, given that he croaks and eats flies, and that the rule base contains the following four rules:

-if [X croaks and X eats flies] then [X is a frog]
-if [X chirps and sings] then [X is a canary]
-if [X is a frog] then [X is colored green]
-if [X is canary] then [X is color yellow]
-if [X croaks and eats flies] then [X is a frog]
-[Fritz croaks and eats flies]
Goal:
[Fritz is colored y] ?


If [X Croaks and eats flies]
Then [X is frog]
Fritz Croaks and eats flies
[Fritz is a frog]
If [X is frog]
Then [X is colored green]
[Fritz is a frog]
Fritz is colored green
[Fritz is color y] ?
[Fritz is colored green]
Y= green

2. Backward Chaining: In this, We Start from the conclusion (Goal), and try to choose inference rules that will get you back to the logical expressions you have. Goal is broken into many sub goal which can be solved easily.

Example.
KB:-
-if [X croaks and X eats flies] then [X is a frog]
-if [X chirps and sings] then [X is a canary]
-if [X is a frog] then [X is colored green]
-if [X is canary] then [X is color yellow]
-if [X croaks and eats flies] then [X is a frog]
-[Fritz croaks and eats flies]
Goal:
[Fritz is colored y] ?


[Fritz is colored Y]
If [X is frog]
Then [X is colored green]
If [X is canary]
Then [X is colored yellow]
X is Frog
X is Canary
[X is a Canary]
If[X croaks and eats flies
Then [X is a Frog]
[X is a Frog]
X croaks and eat flies
X = fritz ,Y= green
[X croaks and eat flies]
prove

Note : Forward chaining use Deductive Inference rule(Modus Ponen and backward chaining use Abductive Inference rule(Modus Ponen)

FC: Conclude "A" and "A" implies "B" to "B"
BC: Conclude from "B" and A->B to "A"

$$\frac{A \qquad A \rightarrow B}{B} \qquad\qquad \frac{B \qquad A \rightarrow B}{A}$$

Example: It is raining---A
If it is raining ,the road is wet ---A->B
The Road is wet.--B

Resolution

- Resolution technique uses proof by contradiction and is based on the fact that any sentence in propositional logic can be transformed into an equivalent sentence in conjunctive normal form .The resolution rule yields a sound and complete algorithm for deciding the *satisfiability* of a propositional formula
- The resolution procedure is a simple iterative procedure: at each step , two clauses, called the parent clauses, are compared(resolved), yielding a new clause that has been inferred from them The new clause represent that two parent clauses interact with each other.

Resolution: Propositional Form

Once sentences are in clausal form, we can apply the resolution inference process.

One rule is enough.

Given two clauses, this process is:

1.Find two complementary terms (e.g., A and ¬ A) in the two clauses.

2. cancel them

3. form a new clause containing all the remaining terms.

e.g. resolving

X v Y v Z

¬ X v A

Y v Z v A

which is a valid deduction.

Resolution in Predicate Logic Algorithm

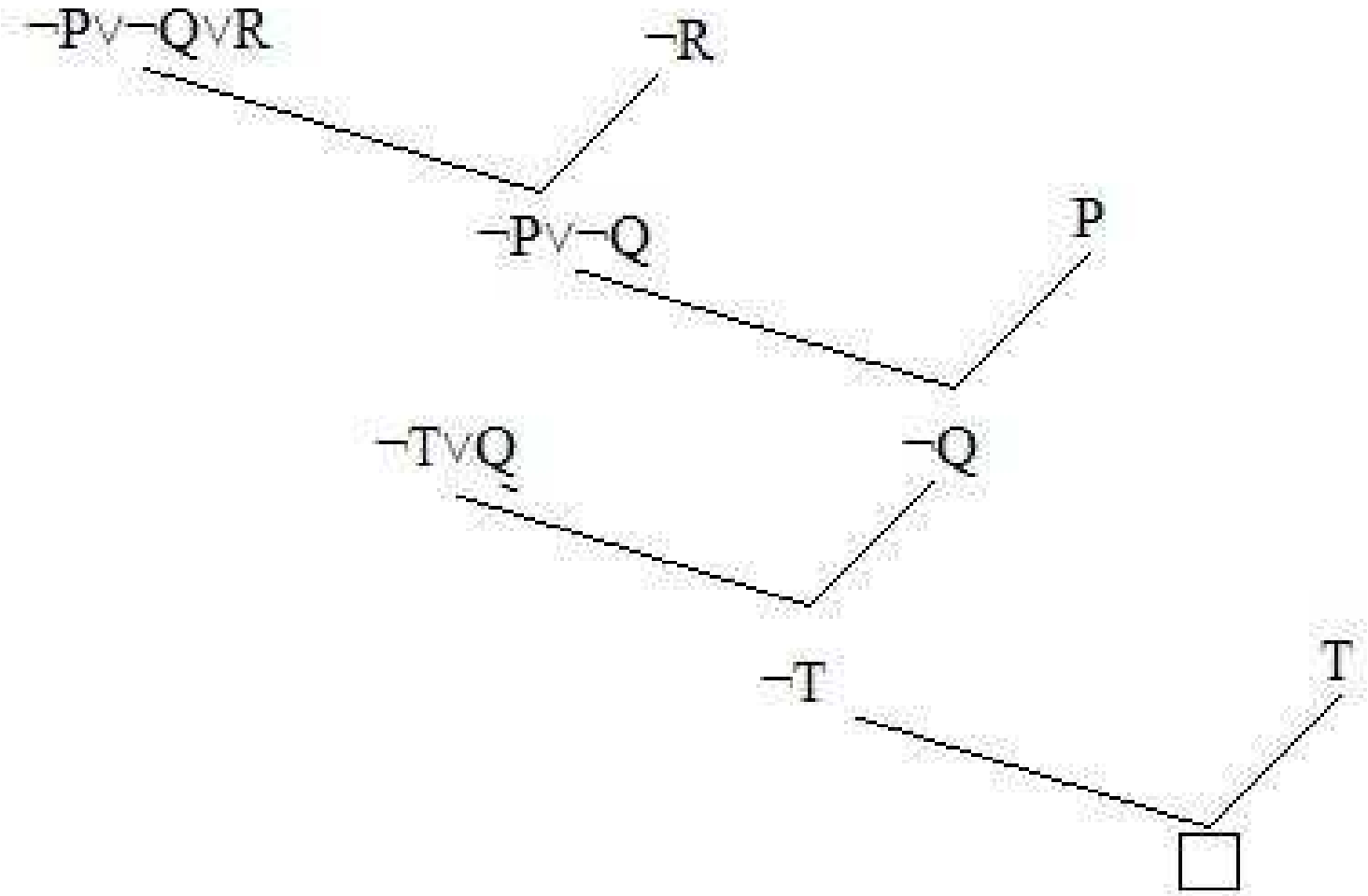
1. Convert all the statements of F to clause form
2. Negate P and convert the result to clause form. Add it to the set of clauses obtained in step 1.
3. Repeat until either a contradiction is found or no progress can be made or a predetermined amount of effort has been expended:

a) Select two clauses. Call these the parent clauses.

b) Resolve them together. The resulting clause, called the re-solvent, will be the disjunction of all of the literals of both of the parent clauses with the following exception: If there are any pairs of literals L and ¬L such that one of the parent clauses contains L and the other contains ¬L, then select one such pair and eliminate both L and ¬L from the re-solvent.

c) If the resolvent is the empty clause, then a contradiction has been found. If it is not then add it to the set of clauses available to the procedure.

Sl. No	Given Axioms	Converted to Clause Form
1	P	P
2	$(P \wedge Q) \rightarrow R$	$\neg P \vee Q \vee R$
3	$(S \vee T) \rightarrow Q$	$\neg S \vee Q$
4		$\neg T \vee Q$
5	T	T


¬P∨¬Q∨R
¬R
¬P∨¬Q
P
¬T∨Q
¬Q
¬T
T

Example……..
The Road is Wet
If It is raining , the road is wet
It is raining.

Resolution in Predicate Logic:

Let F be a set of given statements and S is a statement to be proved.

1. Covert all the statements of F to clause form.
2. Negate S and convert the result to clause form. Add it to the set of clauses obtained in 1.
3. Repeat until either a contradiction is found or no progress can be made.

a) Select two clauses. Call these parent clauses.

b) Resolve them together. The resolvent will be the disjunction of all of these literals of both clauses with appropriate substitutions performed and with the following exception: If there is a pair of literals T1 and ¬T2 such that one parent clause contains T1 and the other contains T2 and if T1 and T2 are unifiable, then neither T1 nor T2 should appear in the resolvent. Here T1 and T2 are called complimentary literals.

C) If the resolvent is the empty clause , then a contradiction has been found. If it is not, then add it to the set of clauses available to the procedure

In order to use the above resolution roofs, they must be converted to clause form as given below:

1. man (Marcus)
2. Pompeian(Marcus)
3. ~ Pompeian(x1) V roman (x1)
   (from Pompeian (x1) ---> Roman (x1))
4. ruler (caesar)
5. ~Roman (x2) V loyalto (x2, Caesar) V hate (x2, Caesar)
6. loyalto (x3, f1(x3))
7. ~man(x4) V ~ ruler(y1) V ~ try assassinate(x4,y1) ~ loyalto (x4, y1))
8. try assassinate (Marcus, Caesar)

Suppose our goal is to answer the question of the assertion S, ie., hate (Marcus, Caesar) . The resolution procedure has been illustrated below.

Example :

(a)

1. Marcus was a man
2. Marcus was a Pompeian.
3. Marcus was born in 40 AD.
4. All Pompeian's died when the Volcano erupted in 1979
5. No mortal lives longer than 150 years.
6. It is now 1991
7. Alive Man not dead

Goal**: Prove- "Marcus is dead" using resolution**

(b)

1. John likes all kinds of food.
2. Apples are food.
3. Chicken is food
4. Anything anyone eat and is not killed alive.
5. Sue eats everything bill eats.

(i)Translate these sentence into formulas in predicate logic.

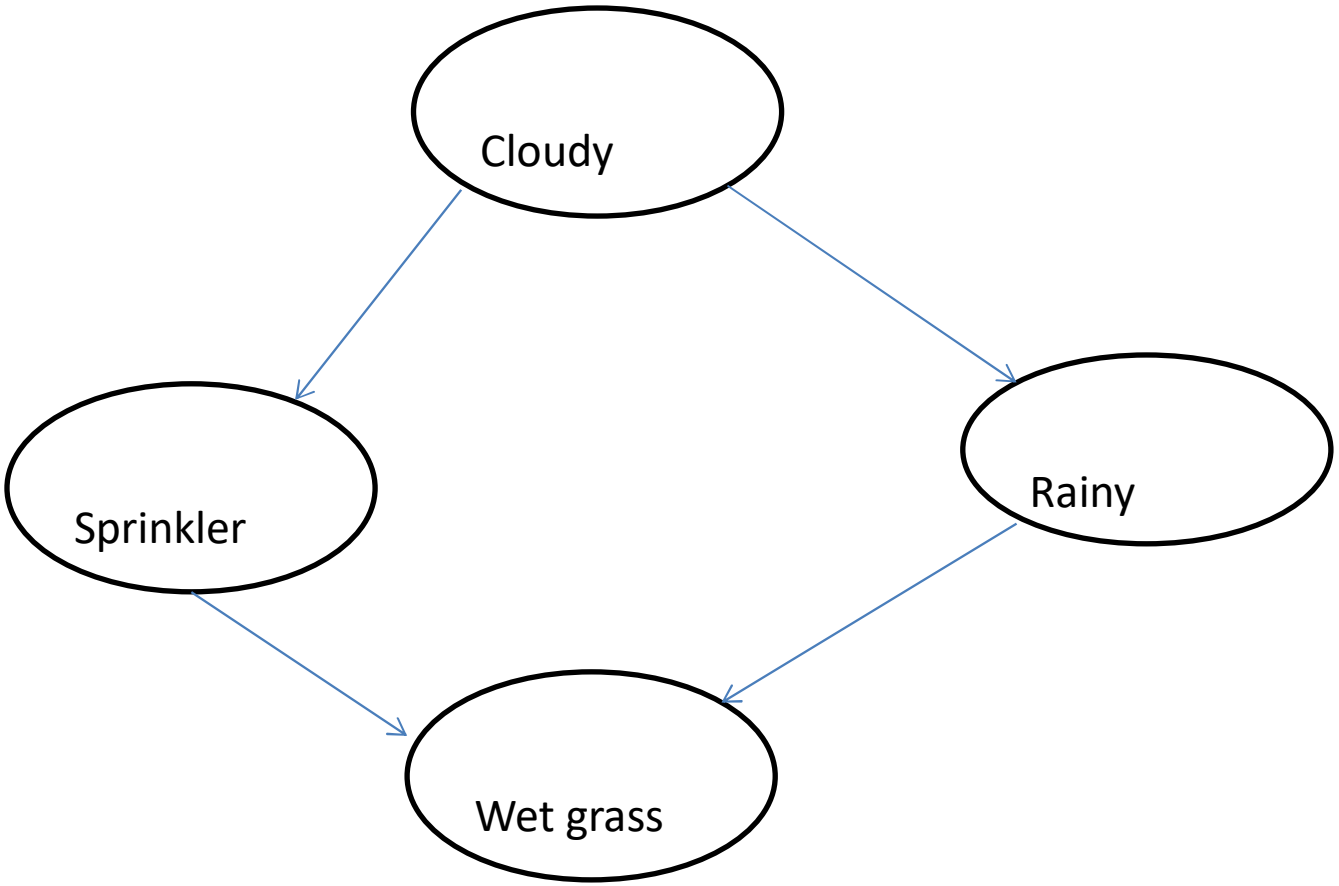
(ii) Prove that John likes peanut using backward chaining.

Bayesian Network: Representation and Syntax Bayes nets (BN) (also referred to as Probabilistic Graphical Models and Bayesian Belief Networks) are directed acyclic graphs (DAGs) where each node represents a random variable. The intuitive meaning of an arrow from a parent to a child is that the parent directly influences the child. These influences are quantified by conditional probabilities.

Syntax:

1. a set of nodes, one per variable
2. a directed, acyclic graph (link ≈ "directly influences")
3. a conditional distribution for each node given its parents:
   $P(X_i \mid \text{Parents}(X_i))$

- In the simplest case, conditional distribution represented as a conditional probability table (CPT) giving the distribution over X_i for each combination of parent values


Cloudy
Sprinkler
Rainy
Wet grass

P(C=F)	P(C=T)
0.5	0.5

C	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

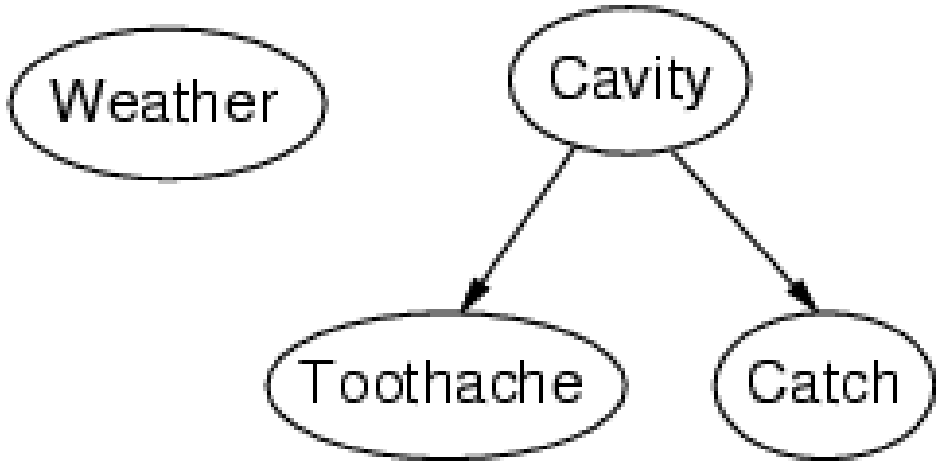
C	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8

S R	P(W=F)	P(W=T)
F F	1.0	0.0
T F	0.1	0.9
F T	0.1	0.9
T T	0.01	0.99

Consider another example, in which all nodes are binary, i.e., have two possible values, which we will denote by T (true) and F (false). We see that the event "grass is wet" (W=true) has two possible causes: either the water sprinker is on (S=true) or it is raining (R=true). The strength of this relationship is shown in the table. For example, we see that Pr(W=true | S=true, R=false) = 0.9 (second row), and hence, Pr(W=false | S=true, R=false) = 1 - 0.9 = 0.1, since each row must sum to one. Since the C node has no parents, its CPT specifies the prior probability that it is cloudy (in this case, 0.5). (Think of C as representing the season: if it is a cloudy season, it is less likely that the sprinkler is on and more likely that the rain is on.)

Exmple2:

- Topology of network encodes conditional independence assertions:



- *Weather* is independent of the other variables
- *Toothache* and *Catch* are conditionally independent given *Cavity*

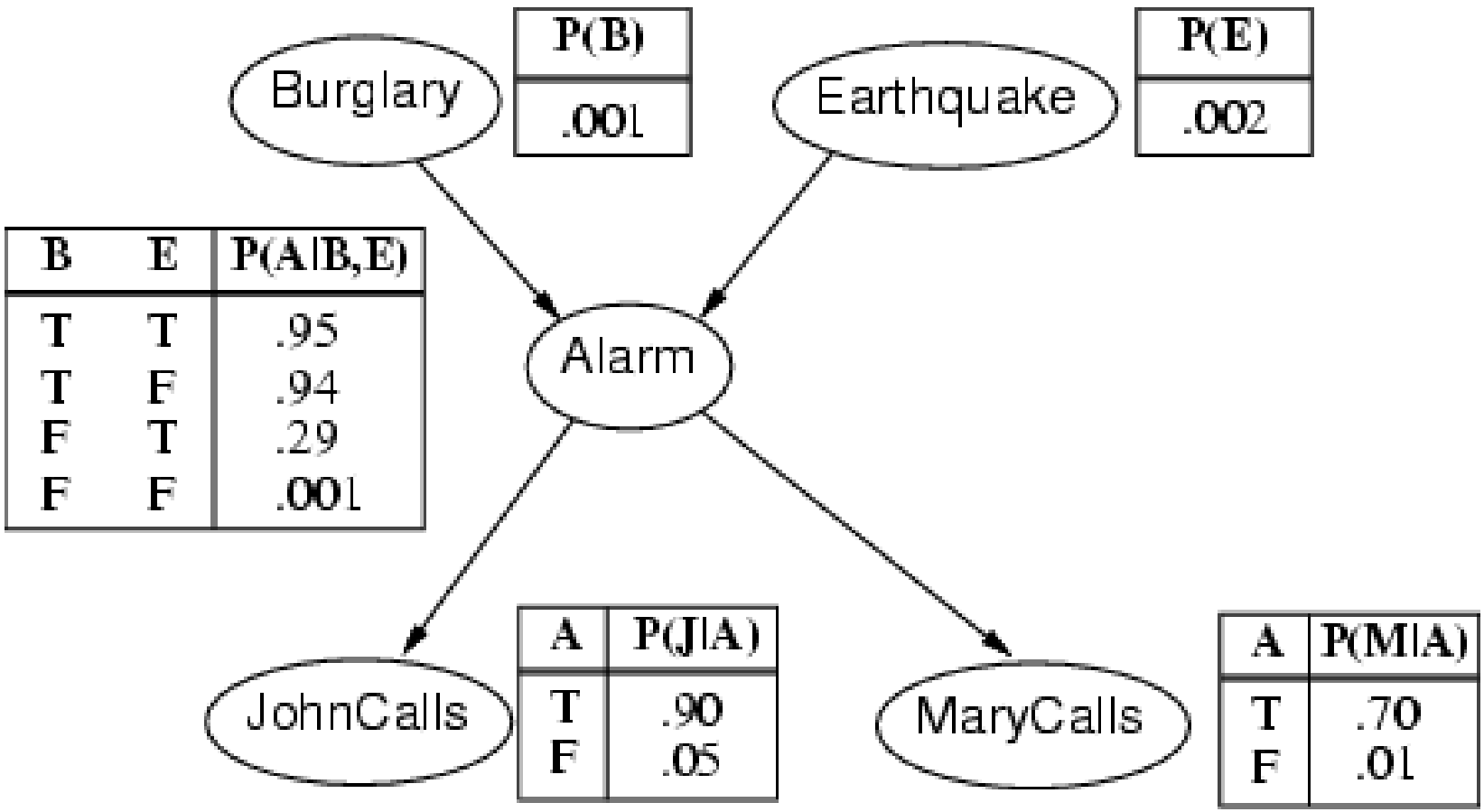
Example:-

I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

Variables: *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*

Network topology reflects "causal" knowledge:

- -A burglar can set the alarm off
- -An earthquake can set the alarm off
- -The alarm can cause Mary to call
- -The alarm can cause John to call


Burglary
P(B)
.001
Earthquake
P(E)
.002
B E P(A|B,E)
T T .95
T F .94
F T .29
F F .001
Alarm
JohnCalls
A P(J|A)
T .90
F .05
MaryCalls
A P(M|A)
T .70
F .01