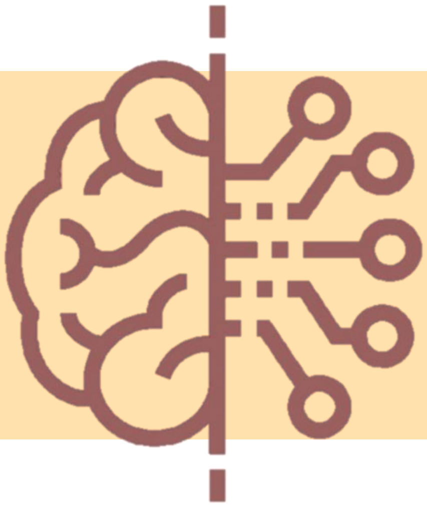
Knowledge Representation



Artificial Intelligence

School of Computing
Universiti Teknologi Malaysia

Outline

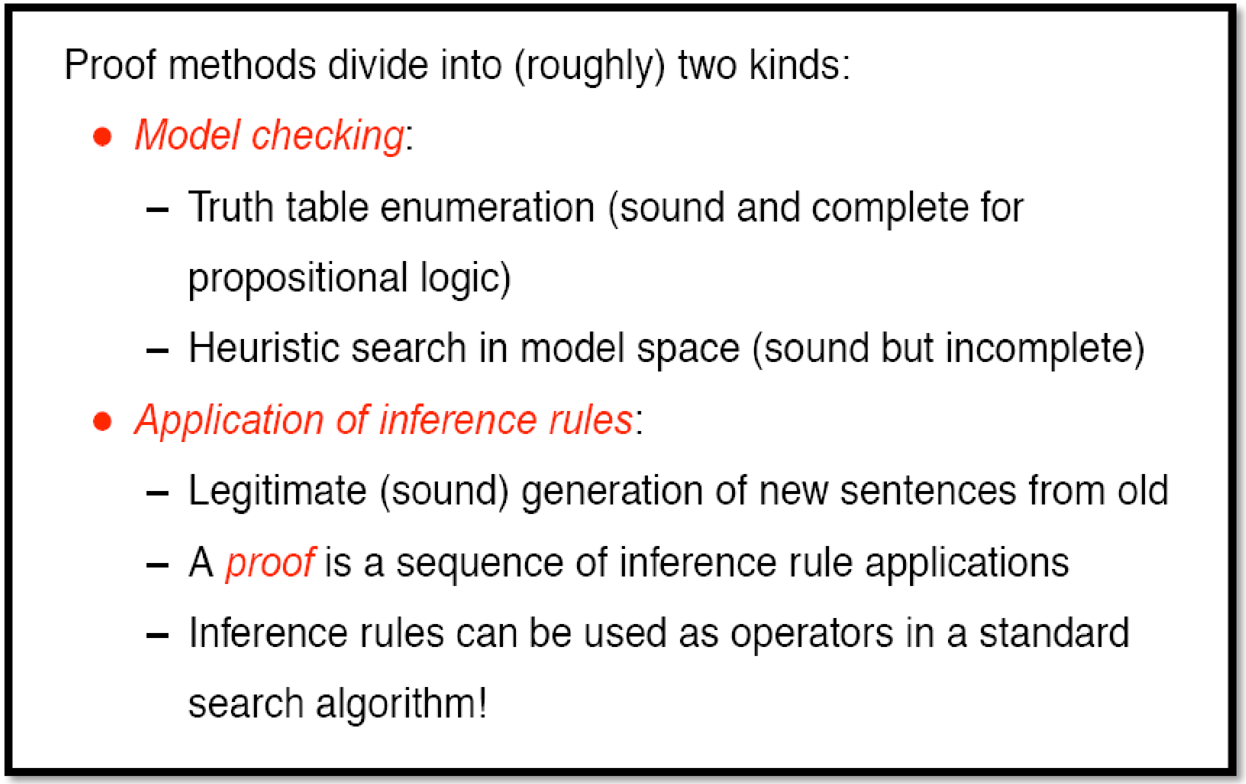
We will learn these topics:

1. Proof Method using Inference Rules
2. Proof Method using Resolution Refutation

Proof methods

Proof methods are techniques for constructing proofs.

- They are used to demonstrate that a conclusion follows from a set of premises.
- Model checking is a technique used to verify whether a system satisfies a given specification.
- There are several rules of inference for compound propositions and quantified statements, which can be combined to form formal proofs



Proof methods divide into (roughly) two kinds:

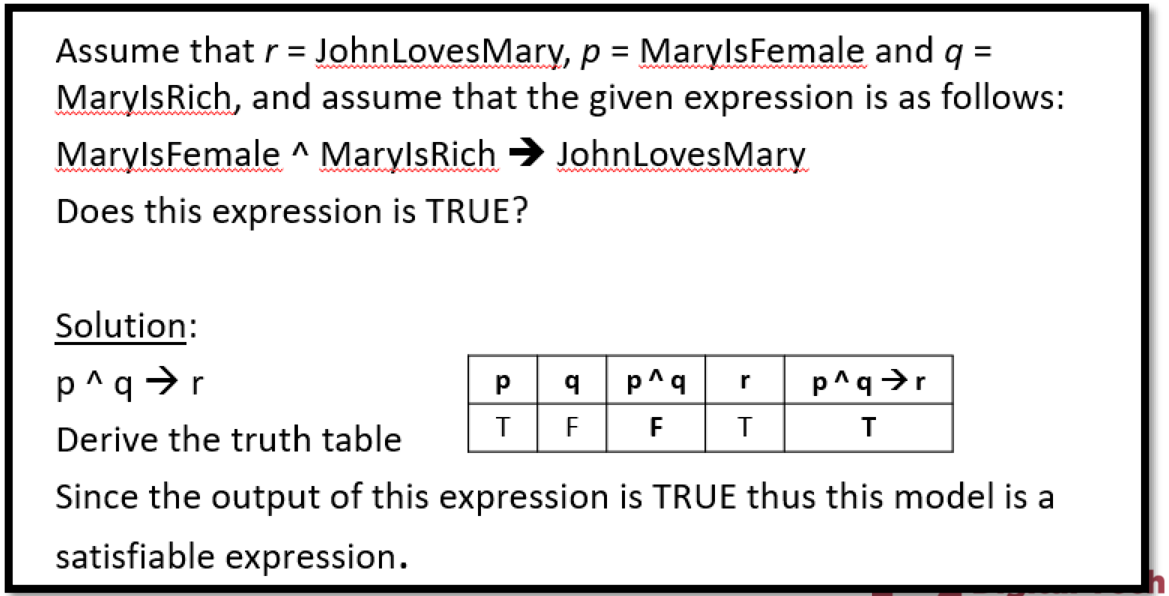
- *Model checking*:
  - Truth table enumeration (sound and complete for propositional logic)
  - Heuristic search in model space (sound but incomplete)
- *Application of inference rules*:
  - Legitimate (sound) generation of new sentences from old
  - A *proof* is a sequence of inference rule applications
  - Inference rules can be used as operators in a standard search algorithm!

Model Checking – Satisfiable Expression?

A satisfiable expression, in the context of propositional logic, refers to a logical statement or formula for which there exists a combination of truth values (assignments) to its variables that makes the expression true.

- In other words, a satisfiable expression can be made true under certain conditions by assigning appropriate truth values to its variables.

Use in Problem Solving: Satisfiability testing is a fundamental concept in various areas, including artificial intelligence, computer science, and formal verification. It is used to solve problems like Boolean satisfiability problems (SAT), where the goal is to find valid assignments for logical expressions.


Assume that *r* = JohnLovesMary, *p* = MaryIsFemale and *q* = MaryIsRich, and assume that the given expression is as follows:

MaryIsFemale ^ MaryIsRich → JohnLovesMary

Does this expression is TRUE?

Solution:

$p \wedge q \rightarrow r$

Derive the truth table

p	q	p ^ q	r	p ^ q → r
T	F	**F**	T	**T**

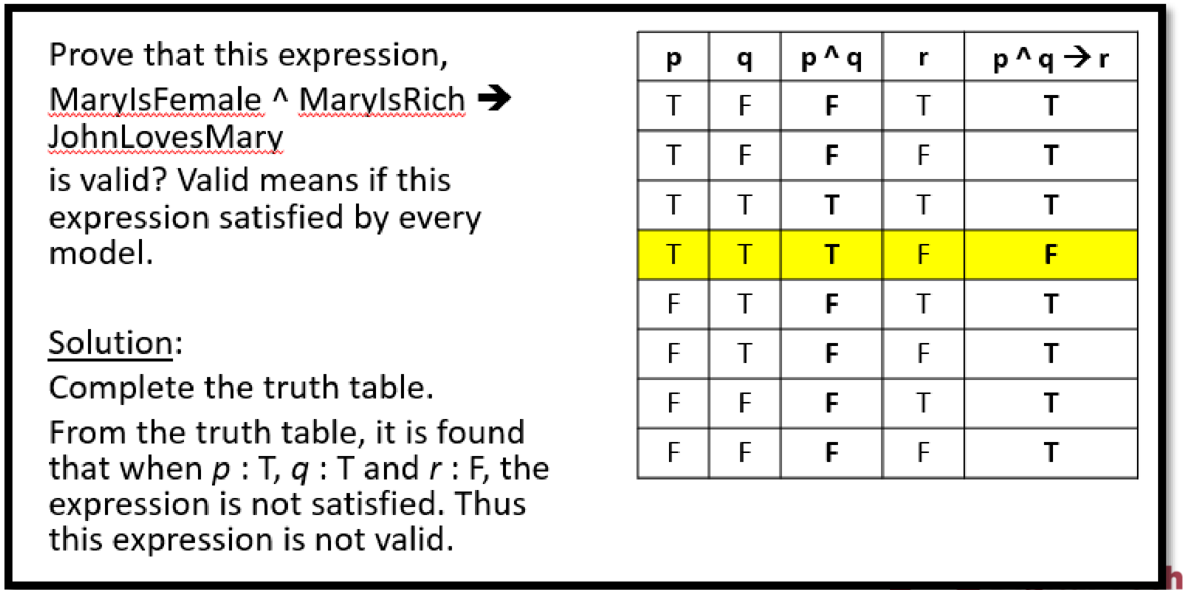
Since the output of this expression is TRUE thus this model is a satisfiable expression.

Model Checking – Valid Expression?

To be considered universally valid, an expression must hold true for every possible combination of truth values for its variables.

- If there is at least one combination that results in a false expression, then the expression is not universally valid**.**

Truth Preservation: Validity ensures that if the premises of an argument are true, the conclusion must also be true. In other words, valid arguments do not lead from true premises to a false conclusion. This property is essential for maintaining the integrity of logical reasoning.



Prove that this expression,
MaryIsFemale ^ MaryIsRich ➔ JohnLovesMary
is valid? Valid means if this expression satisfied by every model.

Solution:

Complete the truth table.

From the truth table, it is found that when *p* : T, *q* : T and *r* : F, the expression is not satisfied. Thus this expression is not valid.

p	q	p ^ q	r	p ^ q → r
T	F	**F**	T	**T**
T	F	**F**	F	**T**
T	T	**T**	T	**T**
T	T	**T**	F	**F**
F	T	**F**	T	**T**
F	T	**F**	F	**T**
F	F	**F**	T	**T**
F	F	**F**	F	**T**

Try this: Prove Validity and Satisfiability

1. Given *p* and *q* in the table below, how many models can be generated?
2. Complete the truth table below for $\neg((p \Rightarrow q) \vee (q \Rightarrow p))$
3. Let *p* is True, prove that this expression $p \vee \neg p$ is satisfiable or not?
4. Prove that this expression $\neg((p \Rightarrow q) \vee (q \Rightarrow p))$ is valid or not?

p	q	$(p \Rightarrow q) \vee (q \Rightarrow p)$	$\neg((p \Rightarrow q) \vee (q \Rightarrow p))$
T	T		
T	F		
F	T		
F	F		

Try this: Prove Validity and Satisfiability

2. Complete the truth table below for $\neg((p \Rightarrow q) \vee (q \Rightarrow p))$

p	q	$(p \Rightarrow q) \vee (q \Rightarrow p)$	$\neg((p \Rightarrow q) \vee (q \Rightarrow p))$
T	T		
T	F		
F	T		
F	F		

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

P→Q	Q→P	(P→Q) V (Q→P	~((P→Q) V (Q→P))

#1: Proof Method using Inference Rules

The proof process can be viewed as a *search* in which the operators are *inference rules*:

- *Modus Ponens (MP)*

$$\frac{\alpha, \quad \alpha \rightarrow \beta}{\beta} \qquad \frac{Takes(Joe,AI) \quad Takes(Joe,AI) \rightarrow Cool(Joe)}{Cool(Joe)}$$

- *And-Introduction (AI)*

$$\frac{\alpha \quad \beta}{\alpha \wedge \beta} \qquad \frac{Cool(Joe) \quad CSMajor(Joe)}{Cool(Joe) \wedge CSMajor(Joe)}$$

- *Universal Elimination (UE)*

$$\frac{\forall x \alpha}{\alpha\{x/\tau\}} \qquad \frac{\forall x\ Takes(x,AI) \rightarrow Cool(x)}{Takes(Pat,AI) \rightarrow Cool(Pat)}$$

τ must be a ground term: a term with no variables

Example of Proof Method using Inference Rules

Given a scenario as follows:

If it is hot and humid, then it is raining. If it is humid, then it is hot. It is humid in which,

H	It is hot.
D	It is humid.
R	It is raining.

Prove that is it raining by using inference rules.

MP: $\alpha, \alpha \rightarrow \beta \,/\, \beta$

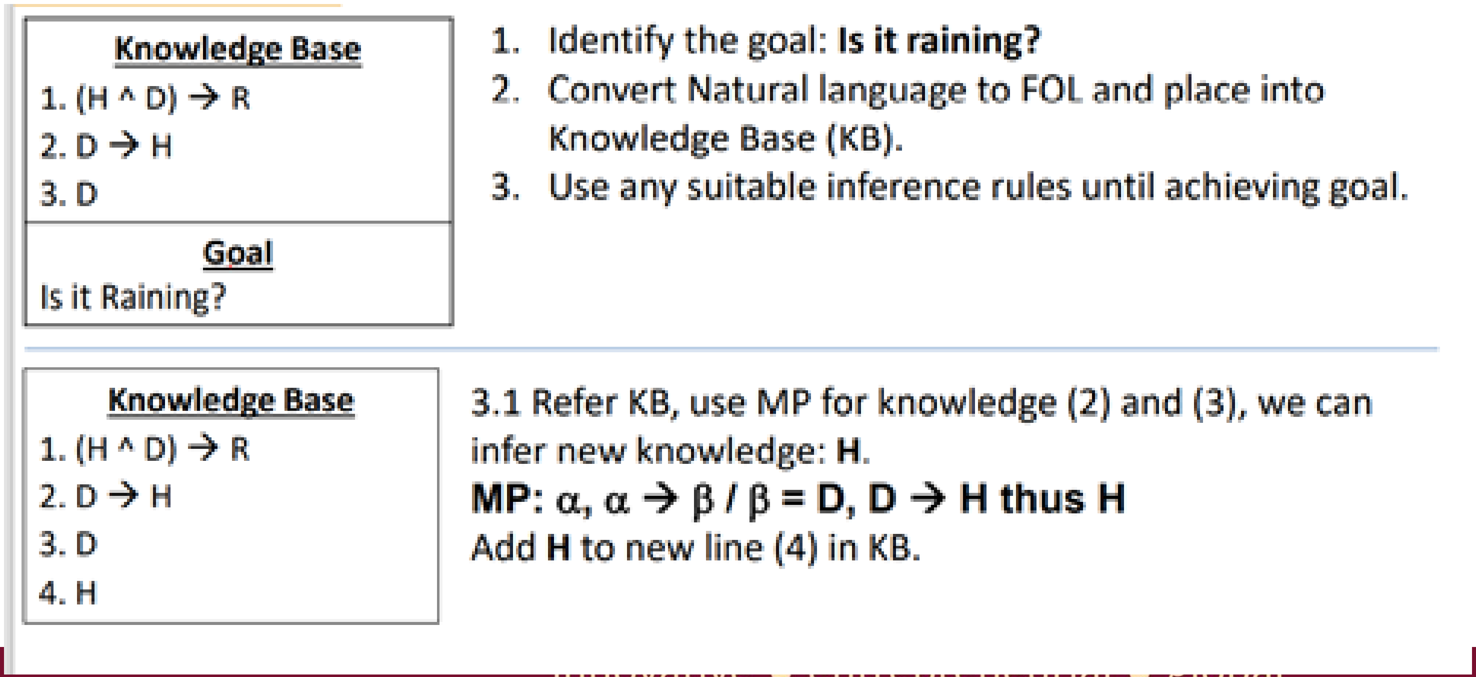
AI: $\alpha \;\; \beta \,/\, \alpha \wedge \beta$

UE: $\dfrac{\forall x \alpha}{\alpha\{x/\tau\}}$

Example of Proof Method using Inference Rules

If it is hot and humid, then it is raining. If it is humid, then it is hot. It is humid in which,

H	It is hot.
D	It is humid.
R	It is raining.



Knowledge Base
1. (H ^ D) → R
2. D → H
3. D

Goal
Is it Raining?

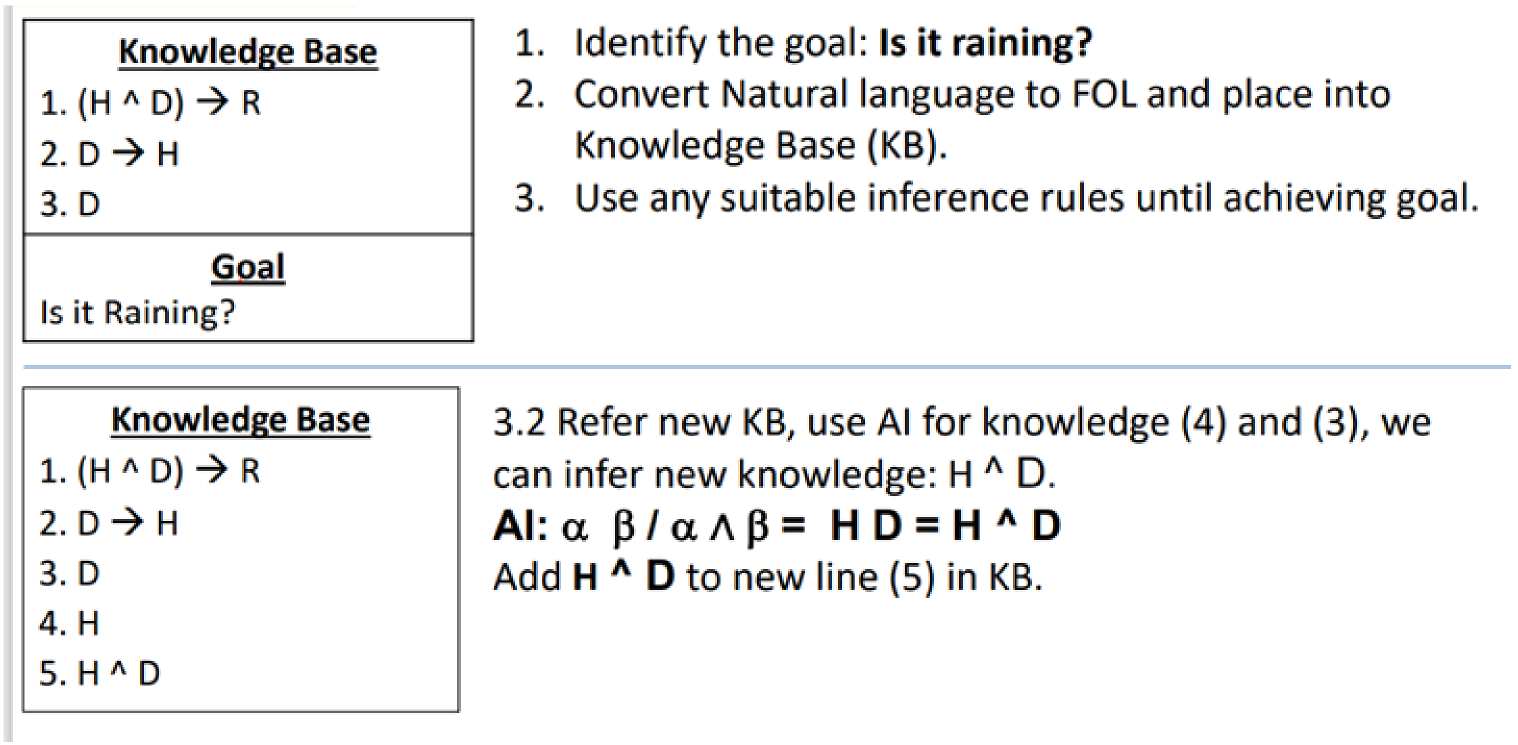
1. Identify the goal: **Is it raining?**
2. Convert Natural language to FOL and place into Knowledge Base (KB).
3. Use any suitable inference rules until achieving goal.

Knowledge Base
1. (H ^ D) → R
2. D → H
3. D
4. H

3.1 Refer KB, use MP for knowledge (2) and (3), we can infer new knowledge: **H**.
MP: α, $\alpha \rightarrow \beta$ / β = D, D → H thus H
Add **H** to new line (4) in KB.

Example of Proof Method using Inference Rules

Solution



Knowledge Base

1. (H ^ D) → R
2. D → H
3. D

Goal

Is it Raining?

1. Identify the goal: **Is it raining?**
2. Convert Natural language to FOL and place into Knowledge Base (KB).
3. Use any suitable inference rules until achieving goal.

Knowledge Base

1. (H ^ D) → R
2. D → H
3. D
4. H
5. H ^ D

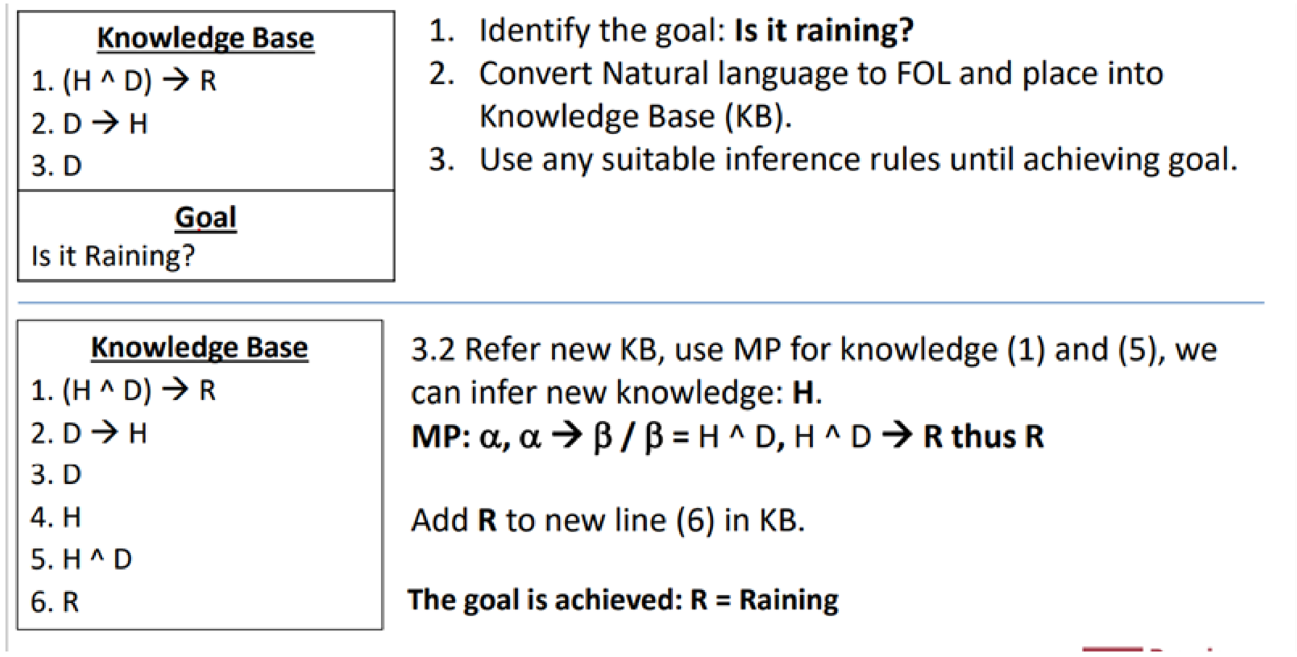
3.2 Refer new KB, use AI for knowledge (4) and (3), we can infer new knowledge: H ^ D.

AI: α β / $\alpha \wedge \beta$ = H D = H ^ D

Add **H ^ D** to new line (5) in KB.

Example of Proof Method using Inference Rules

Solution



Knowledge Base

1. $(H \wedge D) \rightarrow R$
2. $D \rightarrow H$
3. D

Goal

Is it Raining?

1. Identify the goal: **Is it raining?**
2. Convert Natural language to FOL and place into Knowledge Base (KB).
3. Use any suitable inference rules until achieving goal.

Knowledge Base

1. $(H \wedge D) \rightarrow R$
2. $D \rightarrow H$
3. D
4. H
5. $H \wedge D$
6. R

3.2 Refer new KB, use MP for knowledge (1) and (5), we can infer new knowledge: **H.**

MP: $\alpha, \alpha \rightarrow \beta \,/\, \beta$ = $H \wedge D$, $H \wedge D \rightarrow R$ **thus R**

Add **R** to new line (6) in KB.

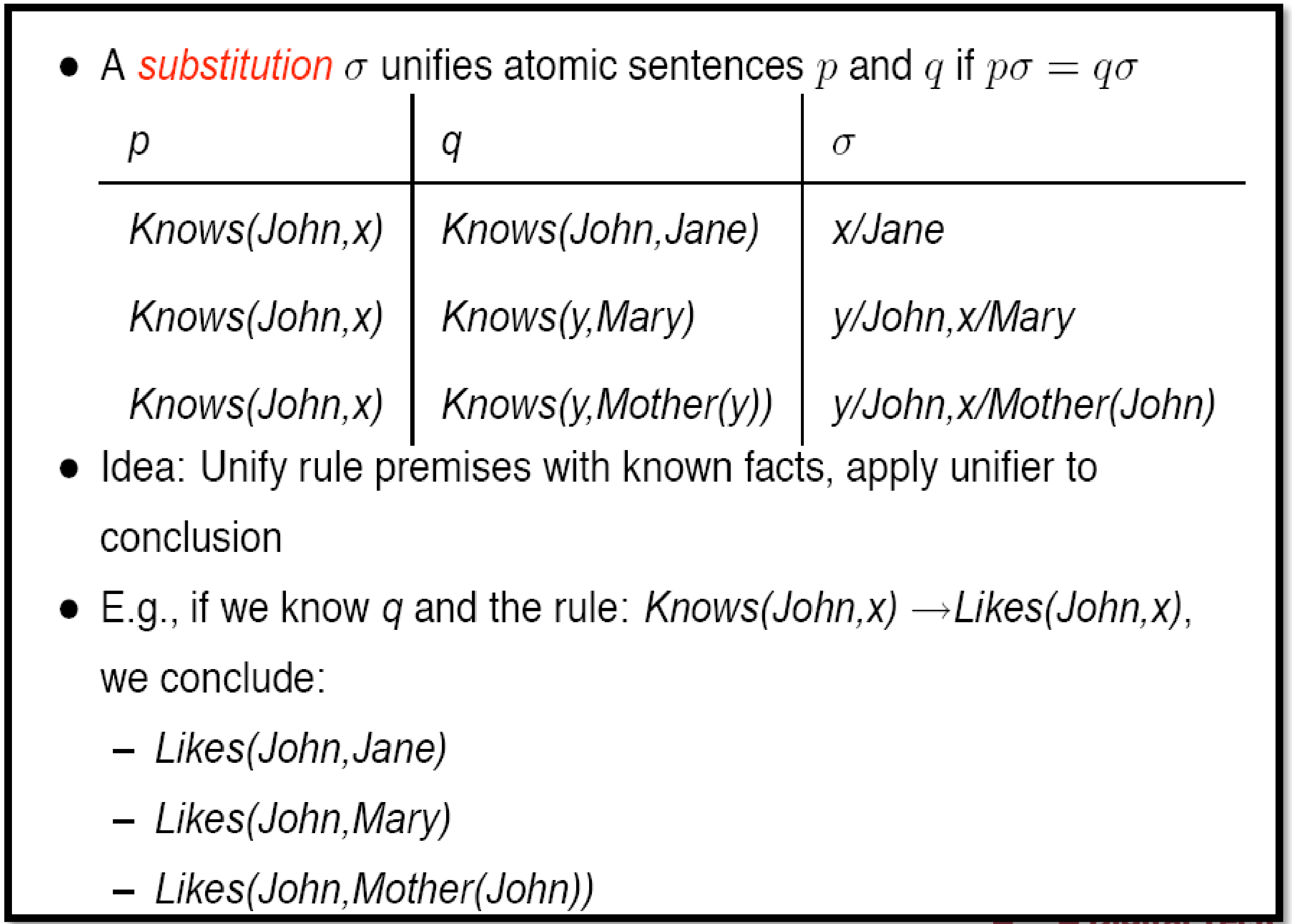
The goal is achieved: R = Raining

Premier Digital Tech University™

Unification

The purpose of unification in logic is to find a common ground or unify variables and terms in logical expressions.

Resolution in Theorem Proving: It's used to unify clauses (logical statements) in a way that eliminates variables, simplifies expressions, and helps derive conclusions or solve problems.

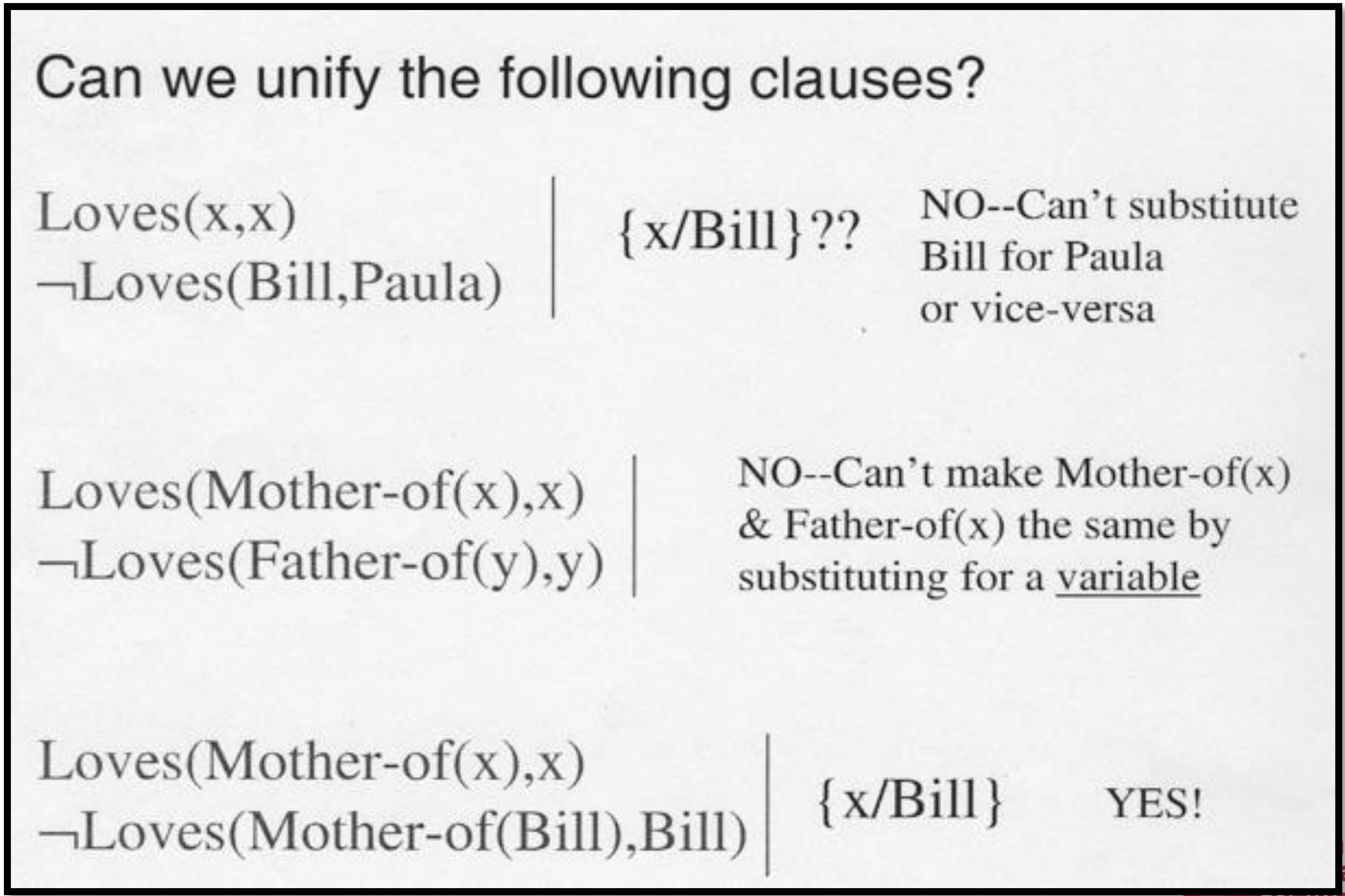


- A *substitution* σ unifies atomic sentences p and q if $p\sigma = q\sigma$

p	*q*	σ
Knows(John,x)	*Knows(John,Jane)*	*x/Jane*
Knows(John,x)	*Knows(y,Mary)*	*y/John,x/Mary*
Knows(John,x)	*Knows(y,Mother(y))*	*y/John,x/Mother(John)*

- Idea: Unify rule premises with known facts, apply unifier to conclusion
- E.g., if we know *q* and the rule: *Knows(John,x)* $\rightarrow$ *Likes(John,x)*, we conclude:
  - *Likes(John,Jane)*
  - *Likes(John,Mary)*
  - *Likes(John,Mother(John))*

Unification



Can we unify the following clauses?

Loves(x,x) ¬Loves(Bill,Paula)	{x/Bill}??	NO--Can't substitute Bill for Paula or vice-versa
Loves(Mother-of(x),x) ¬Loves(Father-of(y),y)		NO--Can't make Mother-of(x) & Father-of(x) the same by substituting for a variable
Loves(Mother-of(x),x) ¬Loves(Mother-of(Bill),Bill)	{x/Bill}	YES!

The unification algorithm

```
function UNIFY-VAR($var, x, \theta$) returns a substitution
    inputs: $var$, a variable
            $x$, any expression
            $\theta$, the substitution built up so far

    if $\{var/val\} \in \theta$ then return UNIFY($val, x, \theta$)
    else if $\{x/val\} \in \theta$ then return UNIFY($var, val, \theta$)
    else if OCCUR-CHECK?($var, x$) then return failure
    else return add $\{var/x\}$ to $\theta$
```

Forward and backward reasoning/chaining

- Situation: You have a collection of logical expressions (premises), and you are trying to prove some additional logical expression (the conclusion)
- You can:
  - Do forward reasoning: Start applying inference rules to the logical expressions you have, and stop if one of your results is the conclusion you want
  - Do backward reasoning: Start from the conclusion you want, and try to choose inference rules that will get you back to the logical expressions you have

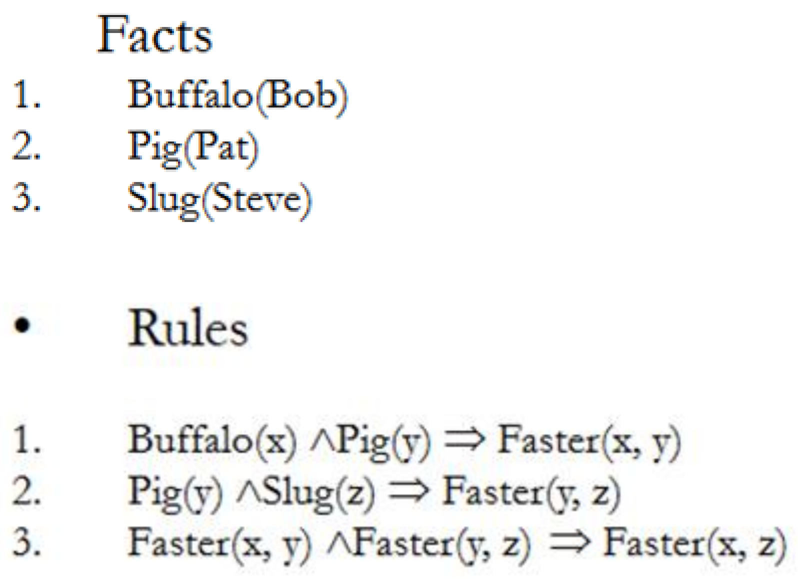
Try to prove this!

2 facts are given, prove this fact is true?

Bob is a buffalo	1. Buffalo(Bob)
Pat is a pig	2. Pig(Pat)
Buffaloes outrun pigs	3. $\forall x \forall y$ Buffalo(x) $\land$ Pig(y) $\rightarrow$ Faster(x,y)
AI 1 & 2	4. Buffalo(Bob) $\land$ Pig(Pat)
UE 3, x/Bob, y/Pat	5. Buffalo(Bob) $\land$ Pig(Pat) $\rightarrow$ Faster(Bob,Pat)
MP 4 & 5	6. Faster(Bob,Pat)

Prove using Forward chaining

Given the following facts and the rules prove that Bob is faster that Steve using forward chaining. Derive the new facts as well.



Facts

1. Buffalo(Bob)
2. Pig(Pat)
3. Slug(Steve)

- Rules

1. Buffalo(x) $\wedge$Pig(y) $\Rightarrow$ Faster(x, y)
2. Pig(y) $\wedge$Slug(z) $\Rightarrow$ Faster(y, z)
3. Faster(x, y) $\wedge$Faster(y, z) $\Rightarrow$ Faster(x, z)

Solution using Forward chaining

- Rules

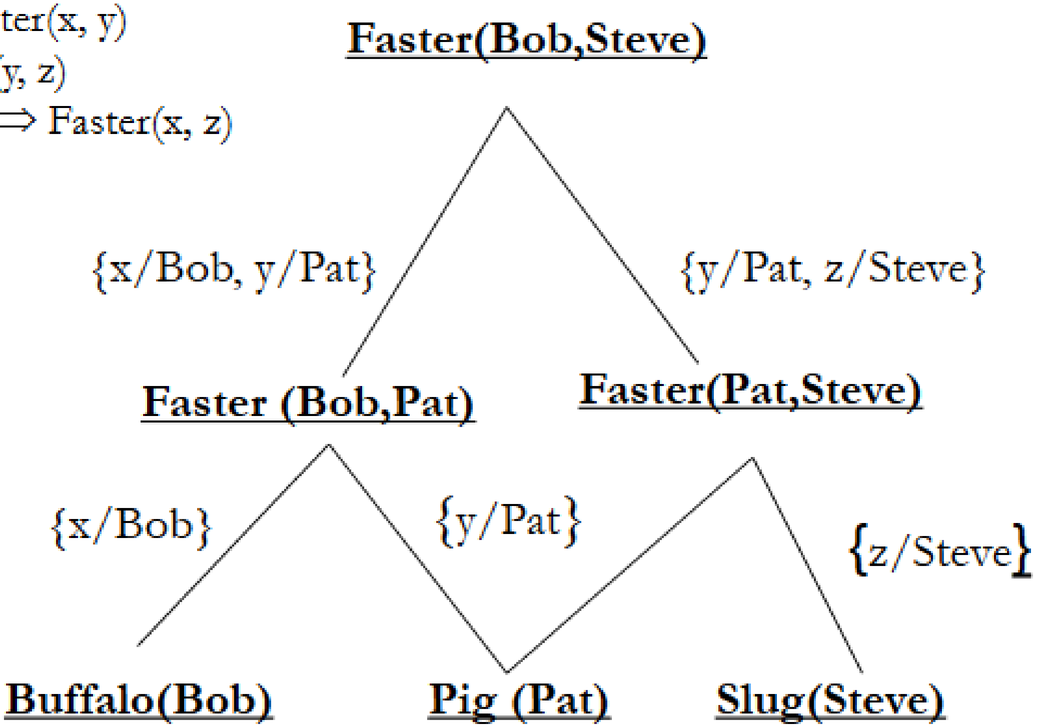
1. Buffalo(x) $\wedge$Pig(y) $\Rightarrow$ Faster(x, y)
2. Pig(y) $\wedge$Slug(z) $\Rightarrow$ Faster(y, z)
3. Faster(x, y) $\wedge$Faster(y, z) $\Rightarrow$ Faster(x, z)

Facts

1. Buffalo(Bob)
2. Pig(Pat)
3. Slug(Steve)

New facts

4. Faster(Bob,Pat)
5. Faster(Pat, Steve)
6. Faster (Bob, Steve)



We start from here

Prove using Backward chaining

Given the hypothesis namely **Pat is faster than Steve** and the following rules. Derive all possible facts in backward chaining tree.

$\mathrm{Pig}(y) \wedge \mathrm{Slug}(z) \Rightarrow \mathrm{Faster}(y, z)$

$\mathrm{Slimy}(a) \wedge \mathrm{Creeps}(a) \Rightarrow \mathrm{Slug}(a)$

$\mathrm{Pig}(\mathrm{Pat})$

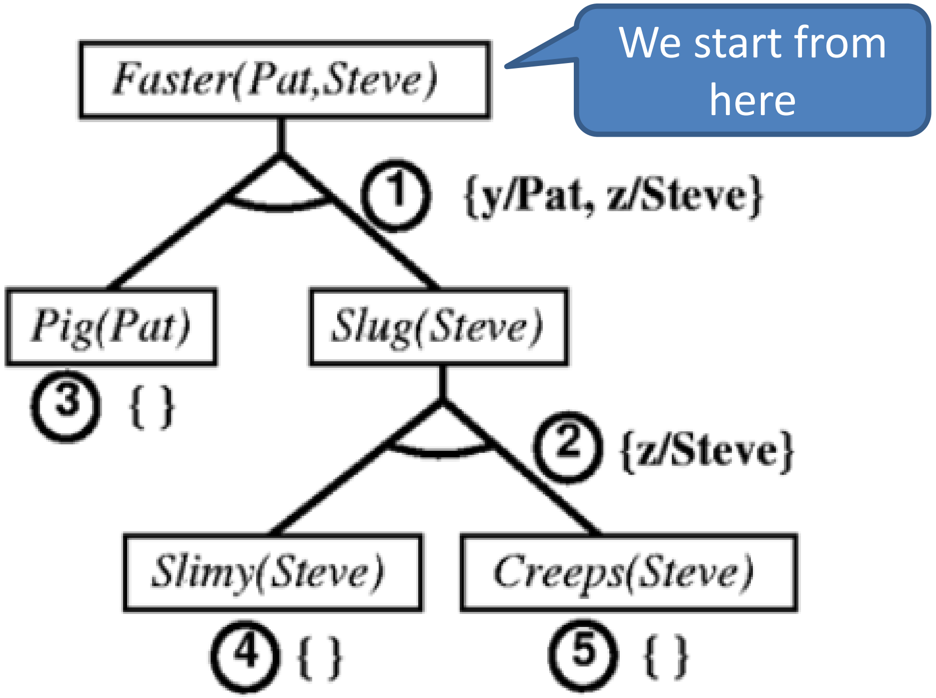
$\mathrm{Slimy}(\mathrm{Steve})$

$\mathrm{Creeps}(\mathrm{Steve})$

Solution using Backward chaining

Backward chaining starts with a hypothesis and work backwards , according to the rules in the knowledge base until reaching confirmed findings or facts.

$Pig(y) \land Slug(z) \Rightarrow Faster(y, z)$
$Slimy(a) \land Creeps(a) \Rightarrow Slug(a)$
$Pig(Pat)$
$Slimy(Steve)$
$Creeps(Steve)$



#2: Proof Method using Resolution Refutation

- The previous example was easy because it had very few clauses
- When we have a lot of clauses, we want to *focus* our search on the thing we would like to prove
- We can do this as follows:
  - Assume that our fact base is consistent (we can't derive NIL)
  - Add the *negation* of the thing we want to prove to the fact base
  - Show that the fact base is now inconsistent
  - Conclude the thing we want to prove

Proof Method using Resolution Refutation

- Resolution is a sound and complete inference method for first-order logic.
- Resolution is a *refutation* procedure: to prove that $KB \models \alpha$, we show that $KB \wedge \neg\alpha$ is unsatisfiable
- The knowledge base and $\neg\alpha$ are expressed in universally quantified, conjunctive normal form
- Like in propositional logic, the resolution inference rule combines two clauses to make a new one:

$$\begin{tikzcd} C_1 \arrow[dr] & & C_2 \arrow[dl] \\ & C & \end{tikzcd}$$

- Inference continues until an empty clause is derived (contradiction)

Conjunctive normal form (CNF) is an approach to Boolean logic that expresses formulas as conjunctions of clauses with an AND or OR. Each clause connected by a conjunction, or AND, must be either a literal or contain a disjunction, or OR operator. CNF is useful for automated theorem proving.

In conjunctive normal form, statements in Boolean logic are conjunctions of clauses with clauses of disjunctions. In other words, a statement is a series of ORs connected by ANDs.

For example:
(A OR B) AND (C OR D)
(A OR B) AND (NOT C OR B)
The clauses may also be literals:
A OR B
A AND B

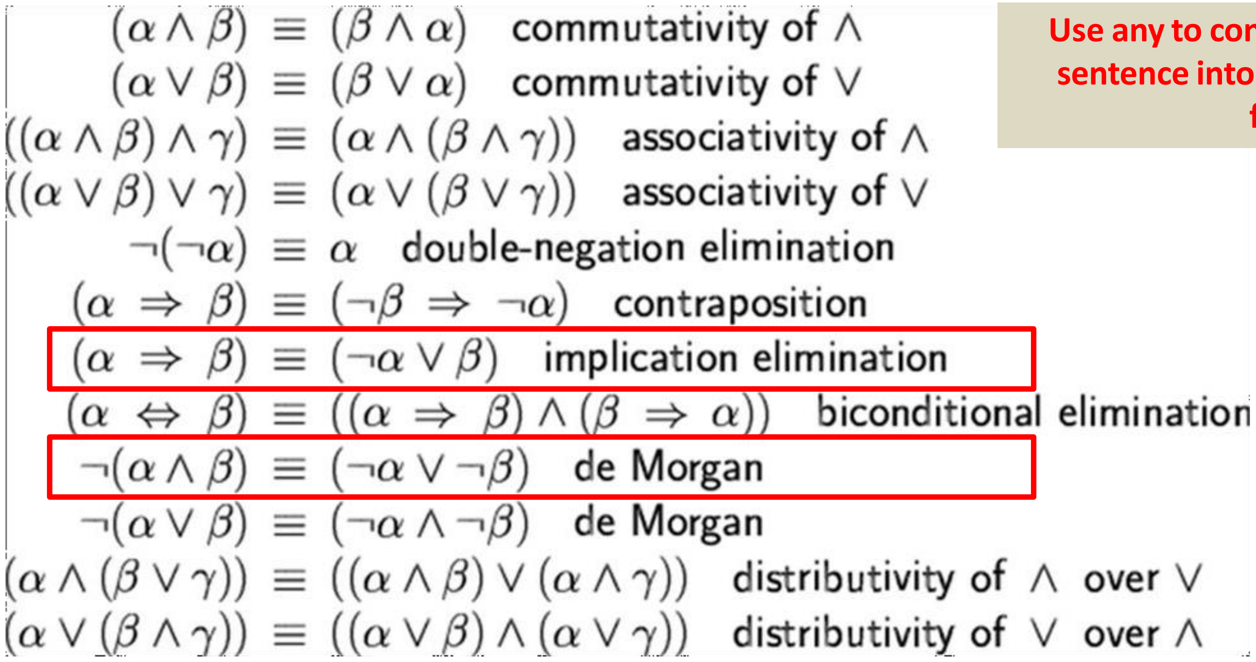
Literals are seen in CNF as conjunctions of literal clauses and conjunctions that happen to have a single clause. It is possible to convert statements into CNF that are written in another form, such as disjunctive normal form.

FOL Resolution

Resolution rule:

- Resolution refutation:
  - Convert all sentences to CNF
  - Negate the desired conclusion (converted to CNF)
  - Apply resolution rule until either
    - Derive false (a contradiction)
    - Can't apply any more
- Resolution refutation is sound and complete
  - If we derive a contradiction, then the conclusion follows from the axioms
  - If we can't apply any more, then the conclusion cannot be
    proved from the axioms.

Inference rules, equivalence



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

Use any to convert sentence into CNF form

Approach to convert FOL into CNF

1. Convert to negation normal form
   a. Eliminate implications & equivalence
   b. Move NOTs inwards using De Morgan
2. Standardize variables
3. Skolemization (to eliminate all existential quantifiers)
4. Drop all universal quantifiers
5. Distribute ORs inwards over ANDs

1. Convert to negation normal form

Eliminate implications & equivalence:

convert $p \rightarrow q$ to $\neg p \vee q$ and

convert $p \leftrightarrow q$ to $(p \vee \neg q) \wedge (\neg p \vee q)$

Approach #1.1: Implication elimination

Eg "Any mothers who love all babies, are also loved by some mothers"

$$\forall x\ \big(\forall y\ Babies(y) \rightarrow Loves(x,y)\big) \rightarrow (\exists y\ Loves(y,x))$$

$$\forall x\ \big(\forall y\ \neg Babies(y) \vee Loves(x,y)\big) \rightarrow (\exists y\ Loves(y,x))$$

$$\forall x\ \neg\big(\forall y\ \neg Babies(y) \vee Loves(x,y)\big) \vee (\exists y\ Loves(y,x))$$

1. Convert to negation normal form

De Morgan Laws:

convert $\neg(p \vee q)$ to $(\neg p) \wedge (\neg q)$;
convert $\neg(p \wedge q)$ to $(\neg p) \vee (\neg q)$;
convert $\neg\neg p$ to p;
convert $\neg(\forall x\, p(x))$ to $\exists x\, \neg p(x)$;
convert $\neg(\exists x\, p(x))$ to $\forall x\, \neg p(x)$

Approach #1.2: Der Morgan Laws

Eg "Any mothers who love all babies, are also loved by some mothers"

$$\forall x\, \neg\big(\forall y\, \neg Babies(y) \vee Loves(x,y)\big) \vee (\exists y\, Loves(y,x))$$
$$\forall x\, \big(\exists y\, \neg\, (\neg\, Babies(y) \vee Loves(x,y)\big)\ \vee (\exists y\, Loves(y,x))$$
$$\forall x\, \big(\exists y\, \neg\neg\, Babies(y) \wedge \neg Loves(x,y)\big) \vee \big(\exists y\, Loves(y,x)\big)$$
$$\forall x\, \big(\exists y\ \ Babies(y) \wedge \neg Loves(x,y)\big) \vee \big(\exists y\, Loves(y,x)\big)$$

UTM UNIVERSITI TEKNOLOGI MALAYSIA

www.utm.my

2. Standardize variables

For FOL that use the similar variable name twice, change the name of ONE of the variables to avoid confusion

Approach #2.1: Standardize Variables

$\forall x\, p(x) \vee \exists x\, q(x)$ change into $\forall x\, p(x) \vee \exists y\, q(y)$ and
$\forall x\, p(x) \vee \forall y\, q(y)$ change into $\forall x\, p(x) \vee \forall x\, q(x)$

Eg "Any mothers who love all babies, are also loved by some mothers"

$$\forall x \left(\exists y\ Babies(y) \wedge \neg Loves(x,y)\right) \vee \left(\exists y\ Loves(y,x)\right)$$
$$\forall x \left(\exists y\ Babies(y) \wedge \neg Loves(x,y)\right) \vee \left(\exists z\ Loves(z,x)\right)$$

Premier Digital Tech University™

3. Skolemization (eliminate all ∃)

- Move all quantifiers to the left, without changing their relative positions.

  Convert $p \wedge (\forall x\, q(x))$ to $\forall x (p \wedge q(x))$;
  convert $p \vee (\forall x\, q(x))$ to $\forall x (p \vee q(x))$;
  convert $p \wedge (\exists x\, q(x))$ to $\exists x (p \wedge q(x))$;
  convert $p \vee (\exists x\, q(x))$ to $\exists x (p \vee q(x))$

Approach #3.1: Skolemization

- Use Skolem function
  - If $\exists x\, p(x)$ then just pick one; call it x'
  - If the existential quantifier is under control of a universal quantifier, then the picked value has to be a function of the universally quantified variable: $\forall x, \exists x, p(x, y)$ then change to $\forall x, p(x, y(x))$

Eg "Any mothers who love all babies, are also loved by some mothers"

$$\forall x\ (\exists y\ Babies(y) \wedge \neg Loves(x,y)) \vee (\exists z\ Loves(z,x))$$
$$\forall x\ \exists z\ (\exists y\ Babies(y) \wedge \neg Loves(x,y)) \vee Loves(z,x)$$
$$\forall x\ \exists z\ \exists y\ (Babies(y) \wedge \neg Loves(x,y)) \vee Loves(z,x)$$
$$\forall x\ \exists y\ (Babies(y) \wedge \neg Loves(x,y)) \vee Loves(g(x),x)$$
$$\forall x\ (Babies(f(x)) \wedge \neg Loves(x,f(x))) \vee Loves(g(x),x)$$

4. Drop all universal quantifiers

By now, all the quantifiers have become universal ∀, therefore we can drop them because we can take for granted that all variables are universally quantified

Approach #4.1: Drop all universal quantifiers

Eg "Any mothers who love all babies, are also loved by some mothers"

$$\forall x \quad \big(Babies(f(x)) \land \neg Loves(x, f(x))\big) \lor Loves(g(x), x)$$

$$\big(Babies(f(x)) \land \neg Loves(x, f(x))\big) \lor Loves(g(x), x$$

5. Distribute ORs inwards over ANDs

Distribute Ors inwards over ANDs to create a conjunction of disjuncts.

– Replace $p \vee (q \wedge r)$ with $(p \vee q) \wedge (p \vee r)$

Eg "Any mothers who love all babies, are also loved by some mothers"

$$\big(Babies(f(x)) \wedge \neg Loves(x, f(x))\big) \vee Loves(g(x), x)$$

$$Babies(f(x)) \vee Loves(g(x), x)) \wedge (\neg Loves\big(x, f(x)\big) \vee Loves(g(x), x)$$

Example 1: Resolution refutation

Given the following sentences, proof that **is tweety is a bird?**

$\forall x : feathers(x) \rightarrow bird(x)$

$feathers(tweety)$

__**Solution:**__

1. Convert sentences into CNF and put into Knowledge Base (choose any approach #1-#5)
2. Negate goal and add this knowledge into Knowledge Base
3. Run proof tree until get NIL, otherwise the goal is false

Answer #1: Resolution refutation

1. Convert sentences into CNF,

- $\forall$x **feathers(x) → bird(x)**, since this FOL has implication, use approach IE and drop universal $\boxed{(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}}$

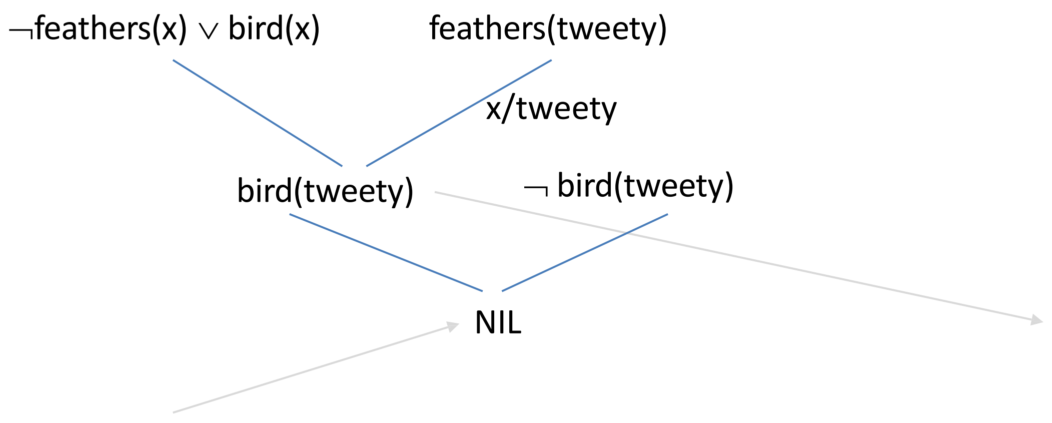
- **feathers (tweety)**, since this FOL atomic, just copy to KB
- Goal is "tweety is a bird', convert this to CNF and negate, ¬ **bird(tweety)**

2.Add to,

<u>Knowledge Base (CNF):</u>
1. ¬feathers(x) ∨ bird(x)
2. feathers(tweety)
3. ¬ bird(tweety)

Answer #1: Resolution refutation

3. Derive proof tree, try to get NIL. Refer knowledge in KB.



Knowledge Base (CNF):

1. ¬feathers(x) ∨ bird(x)
2. feathers(tweety)
3. ¬ bird(tweety)
4. bird(tweety)

4. Since NIL is achieved, thus it is proven that "tweety is a bird" is true.

Example 2: Resolution refutation

Given the following sentences, proof that **is Tom is a cat?**

$\forall x$ fur(x) → cat(x)

fur(Tom)

Solution:

1. Convert sentences into CNF and put into Knowledge Base (choose any approach #1-#5)
2. Negate goal and add this knowledge into Knowledge Base
3. Run proof tree until get NIL, otherwise the goal is false

Answer #2: Resolution refutation

1. Convert sentences into CNF,

   - $\forall x$ fur(x) → cat(x) since this FOL has implication, use approach IE and drop universal

     $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \lor \beta)$ implication elimination

   - **fur (Tom)**, since this FOL atomic, just copy to KB
   - Goal is "Tom is a cat', convert this to CNF and negate, ¬ **cat(Tom)**

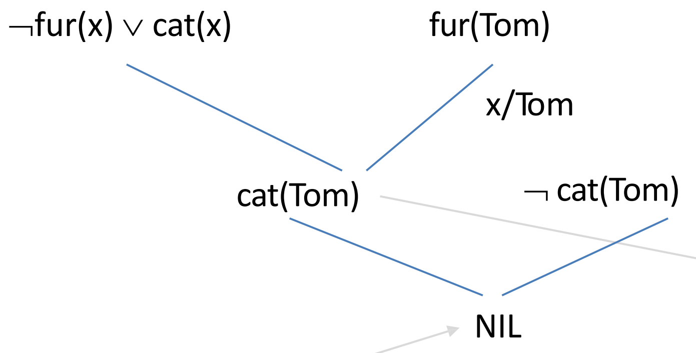
2. Add to,

- **<u>Knowledge Base (CNF):</u>**

1. ¬fur(x) ∨ cat(x)
2. fur(Tom)
3. ¬ cat(Tom)

Answer #2: Resolution refutation

3. Derive proof tree, try to get NIL. Refer knowledge in KB.



Knowledge Base (CNF):
1. ¬fur(x) ∨ cat(x)
2. fur(Tom)
3. ¬ cat(Tom)
4. cat(Tom)

4. Since NIL is achieved, thus it is proven that "Tom is a cat" is true.

Example 3: Resolution refutation

Given the following sentences, proof that, **Tom is cute?**

cat (y) → cute (y)
fur(y) Λ mammal (y) → cat(y)
fur(Tom)
mammal (Tom)

Solution:

1. Convert sentences into CNF and put into Knowledge Base (choose any approach #1-#5)
2. Negate goal and add this knowledge into Knowledge Base
3. Run proof tree until get NIL, otherwise the goal is false

Answer #3: Resolution refutation

1. Convert sentences into CNF,
   - **cat (y) → cute (y)**, since this FOL has implication, use approach IE

     $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination

   - **fur(y) Λ mammal (y) → cat(y)**, since this FOL has implication, use approach IE, now become ¬(fur(y) Λ mammal(y)) ∨ cat(y). Next, use De Morgan Laws to become, **¬fur(y) ∨ ¬ mammal(y) ∨ cat(y)**

     $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ de Morgan

   - **fur (Tom)**, since this FOL atomic, just copy to KB
   - **mammal (Tom)**, since this FOL atomic, just copy to KB
   - Goal is "Tom is cute', convert this to CNF and negate, ¬ **cute(Tom)**

1. Add to,

<u>**Knowledge Base (CNF):**</u>

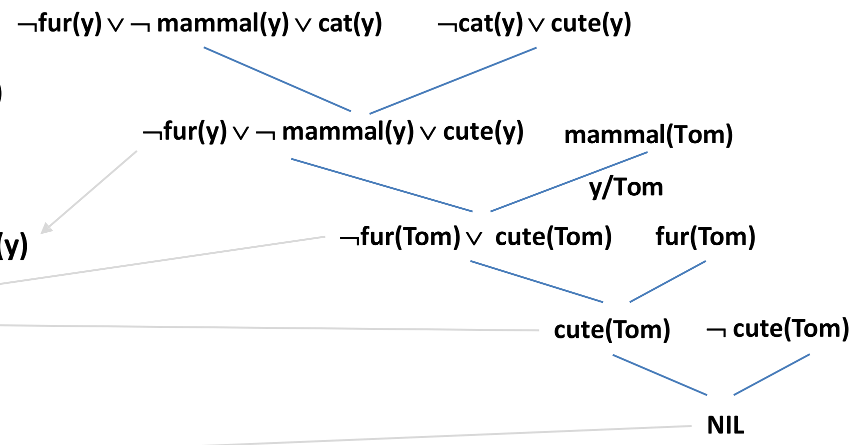
1. ¬cat(y) ∨ cute(y)
2. ¬fur(y) ∨ ¬ mammal(y) ∨ cat(y)
3. fur(Tom)
4. mammal(Tom)
5. ¬ cute(Tom)

Answer #3: Resolution refutation

3. Derive proof tree, try to get NIL. Refer knowledge in KB.

<u>**Knowledge Base (CNF):**</u>

1. ¬cat(y) ∨ cute(y)
2. ¬fur(y) ∨ ¬ mammal(y) ∨ cat(y)
3. fur(Tom)
4. mammal(Tom)
5. ¬ cute(Tom)
6. **¬fur(y) ∨ ¬ mammal(y) ∨ cute(y)**
7. **¬fur(Tom) ∨ cute(Tom)**
8. **cute(Tom)**



4. Since NIL is achieved, thus it is proven that “Tom is cute” is true.

What is the answer?

Based on the statement below, what will be the new knowledge can be derived.

$$\neg fur(y) \lor \neg\, mammal(y) \lor cat(y)$$

and

$$\neg cat(y) \lor cute(y)$$

A. **$\neg fur(y) \lor \neg\, mammal(y) \lor cute(y)$**
B. **$cute(y)$**
C. $\neg fur(y) \lor \neg\, mammal(y)$
D. $\neg fur(y) \lor \neg\, mammal(y) \lor \neg cat(y)$

Applications of First-Order Logic

- Prolog: a logic programming languages
- Production systems
- Semantic nets
- Automated theorem proving
- Planning

Summary

1. We learned three topics in this chapter: AI knowledge representation, propositional logic and predicate logic.
2. Knowledge representation is a part of Artificial intelligence which concerned with thinking that contributes to intelligent behavior.
3. Propositional logic is a branch of mathematical logic which studies the logical relationships between propositions (or statements, sentences, assertions) taken as a whole, and connected via logical connectives.
4. A predicate is an expression of one or more variables defined on some specific domain. A predicate with variables can be made a proposition by either assigning a value to the variable or by quantifying the variable