

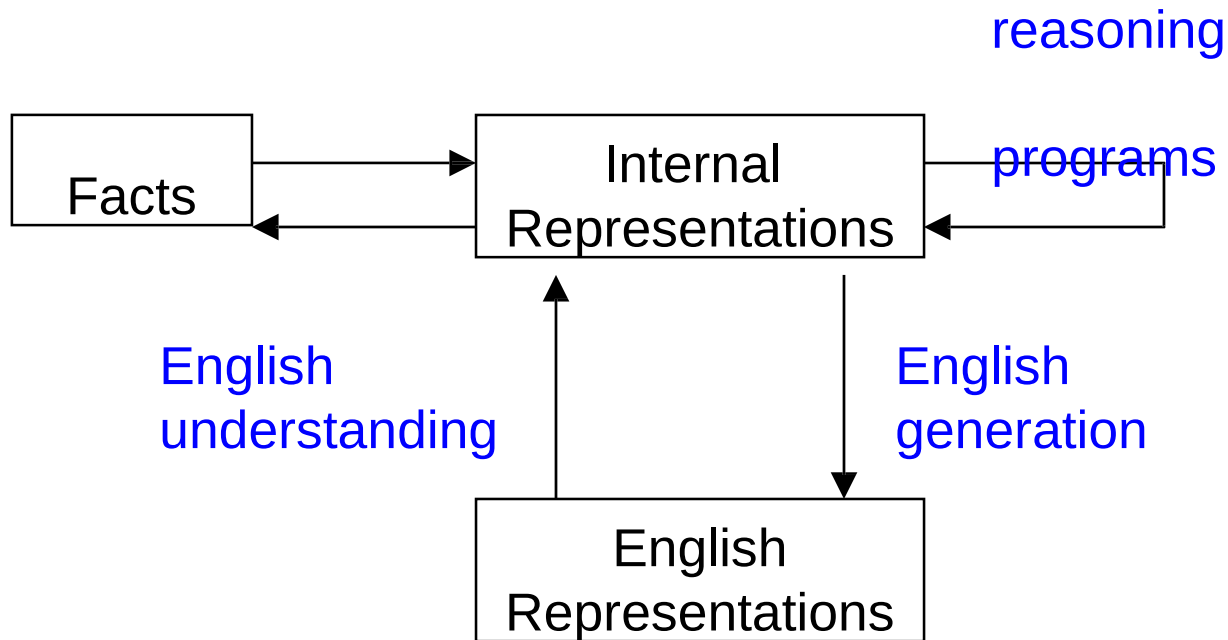
# **KNOWLEDGE REPRESENTATION & PREDICATE LOGIC**

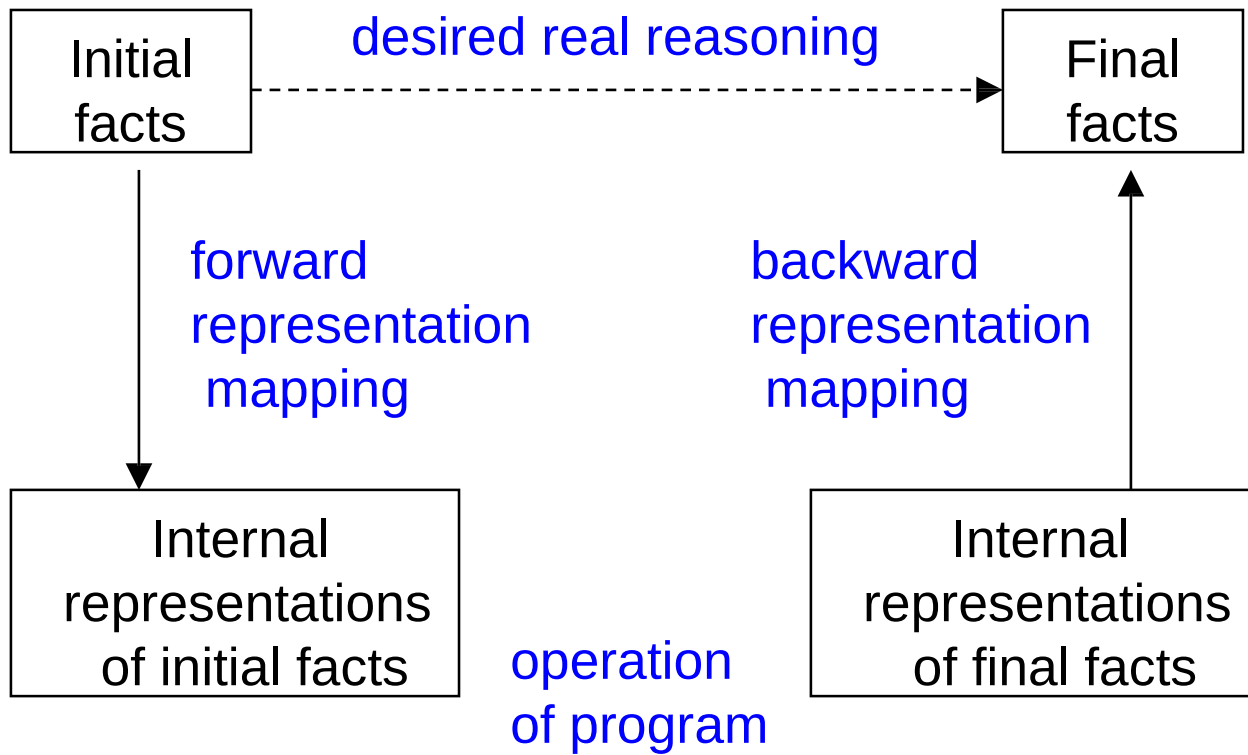
- To solve complex problems we need:
  1. Large amount of knowledge
  2. Mechanism for representation and manipulation of existing knowledge to create new solution.

## Knowledge Representation

- **Facts:** Things we want to represent. Truth in some relevant world.
- **Representation of facts.**

# Representation and Mapping





# Representation and Mapping

- Spot is a dog
- Every dog has a tail



Spot has a tail

# Representation and Mapping

- Spot is a dog

$\text{dog}(\text{Spot})$

- Every dog has a tail

$\forall x: \text{dog}(x) \rightarrow \text{hastail}(x)$



$\text{hastail}(\text{Spot})$

Spot has a tail

- Fact-representation mapping is **not one-to-one**.
- **Good representation** can make a **reasoning program trivial**.

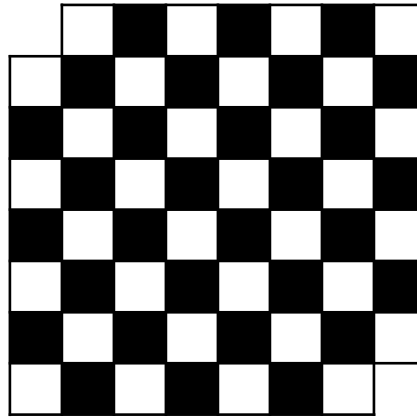
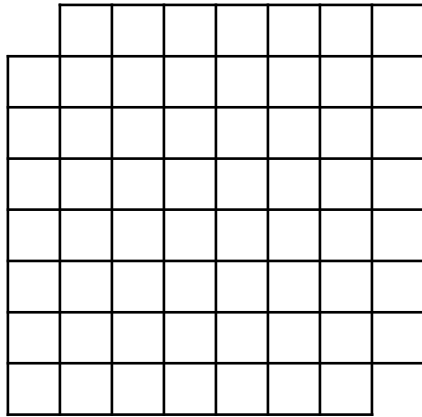
### The Multilated Checkerboard Problem

“Consider a normal checker board from which two squares, in opposite corners, have been removed.

The task is to cover all the remaining squares exactly with donimoes, each of which covers two squares. No overlapping, either of dominoes on top of each other or of dominoes over the boundary of the multilated board are allowed.

Can this task be done?”

# Representation and Mapping



No. black squares  
= 30

No. white square  
= 32



Good Knowledge representation should exhibit:

**1. Representational adequacy-**

Ability to **represent all kinds of knowledge** that are needed in the domain.

**2. Inferential adequacy-**

Ability to manipulate representational structures such that **new knowledge can be derived/inferred from the old.**

**3. Inferential efficiency-**

Ability to incorporate **additional information into an existing** knowledge base that can be used to focus the attention of **inference** mechanisms in the **most promising direction.**

**4. Acquisitional efficiency-**

**Ability to easily acquire new information.**

# Approaches to KR

## 1. Simple relational knowledge:

- Provides very weak inferential capabilities.
- May serve as the input to powerful inference engines.

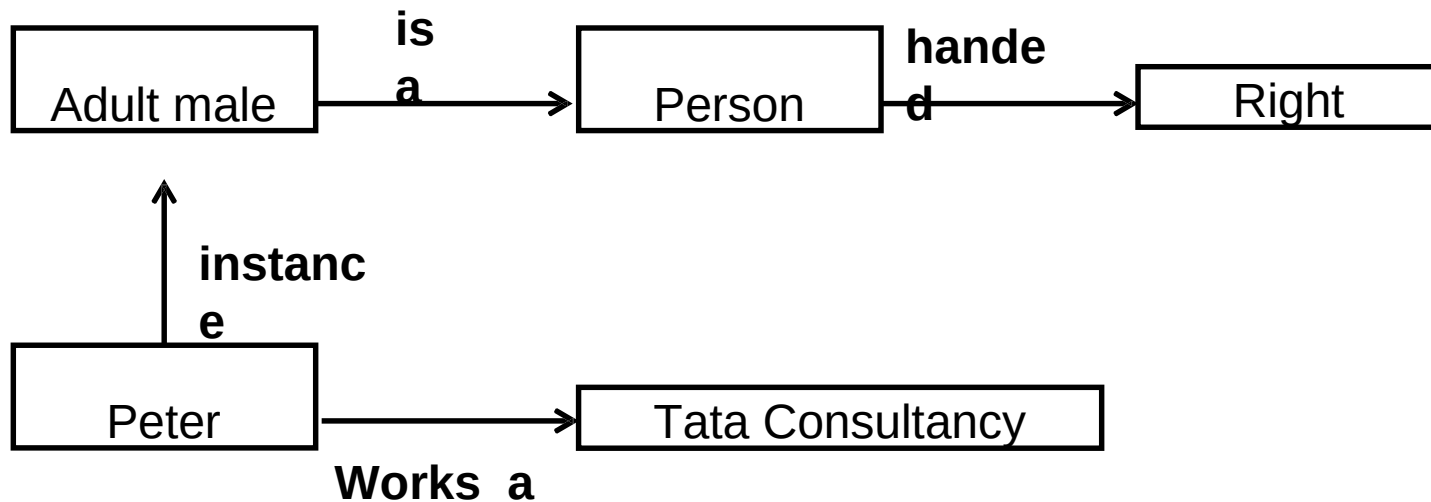
<b>Player</b>	<b>Height</b>	<b>Weight</b>	<b>handed</b>
Peter	6-0	180	right
Ajay	5-10	170	left
John	6-2	215	left
Vickey	6-3	205	right

Fails to infer “which right handed player can best face a particular bowler” .

# Approaches to KR

## Inheritable knowledge:

- Objects are organized into **classes** and classes are organized in a generalization **hierarchy**.
- **Inheritance** is a powerful form of inference, but **not adequate**.
- Ex. Property inheritance inference mechanism.



# Approaches to KR

## Inferential knowledge:

- Facts represented in a **logical form**, which facilitates **reasoning**.
- An **inference engine** is required.

ex. 1. “Marcus is a **man**”

2. “All **men** are **mortal**”

Implies:

3. “**Marcus** is **mortal**”

# Approaches to KR

## Procedural knowledge:

- Representation of “how to make it” rather than “what it is”.
- May have inferential efficiency, but no inferential adequacy and acquisitional efficiency.
- Ex. Writing LISP programs

# Issues in KR

1. Important Attributes: *Isa* and *instance* attributes.
2. Relationships among attributes: *inverses*, *existence* in a *Isa* hierarchy, *single-valued attributes*, techniques for reasoning about values.
3. Choosing the Granularity: High-level facts may not be adequate for inference. Low-level primitives may require a lot of storage.

- Ex: “john spotted sue”

[representation: *spotted(agent(john),object(sue))*]

Q1: “who spotted sue?” Q2: “Did john see sue?”

Ans1: “john”. Ans2: NO ANSWER!!!!

- Add detailed fact: *spotted(x,y)-->saw(x,y)* then Ans2: “Yes”.

- 4.Representing Set of Objects:
- 5. finding the right structure as needed.:

Ex: word “fly” can have multiple meanings:

1. “John flew to new york”
2. “John flew into a rage” [idiom]
3. “john flew a kite”

# Propositional logic

- Statements used in mathematics.
- **Proposition** :is a declarative sentence whose value is either true or false.

Examples:

- “The sky is blue.” **[Atomic Proposition]**
- “The sky is blue and the plants are green.”  
**[Molecular/Complex Proposition]**
- “Today is a rainy day” **[Atomic Proposition]**
- “Today is Sunday” **[Atomic Proposition]**
- “  $2*2=4$ ” **[Atomic Proposition]**



## Terminologies in propositional algebra:

Statement: sentence that can be true/false.

### Properties of statement:

✓ **Satisfiability**: a sentence is satisfiable if there is an interpretation for which it is true.

Eg. "we wear woollen cloths"

✓ **Contradiction**: no

✓ **Contradiction**: if there is **no** interpretation for which sentence is true.

Eg. "Japan is capital of India"

✓ **Validity**: a sentence is valid if it is true for every interpretation.

Eg. "Delhi is the capital of India"

# Inference rules:

Commutative	$p \wedge q \iff q \wedge p$	$p \vee q \iff q \vee p$
Associative	$(p \wedge q) \wedge r \iff p \wedge (q \wedge r)$	$(p \vee q) \vee r \iff p \vee (q \vee r)$
Distributive	$p \wedge (q \vee r) \iff (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \iff (p \vee q) \wedge (p \vee r)$
Identity	$p \wedge T \iff p$	$p \vee F \iff p$
Negation	$p \vee \sim p \iff T$	$p \wedge \sim p \iff F$
Double Negative	$\sim(\sim p) \iff p$	
Idempotent	$p \wedge p \iff p$	$p \vee p \iff p$
Universal Bound	$p \vee T \iff T$	$p \wedge F \iff F$
De Morgan's	$\sim(p \wedge q) \iff (\sim p) \vee (\sim q)$	$\sim(p \vee q) \iff (\sim p) \wedge (\sim q)$
Absorption	$p \vee (p \wedge q) \iff p$	$p \wedge (p \vee q) \iff p$
Conditional	$(p \implies q) \iff (\sim p \vee q)$	$\sim(p \implies q) \iff (p \wedge \sim q)$

Modus Ponens	$p \implies q$ $p$ $\therefore q$	Modus Tollens	$p \implies q$ $\sim q$ $\therefore \sim p$
Elimination	$p \vee q$ $\sim q$ $\therefore p$	Transitivity	$p \implies q$ $q \implies r$ $\therefore p \implies r$
Generalization	$p \implies p \vee q$ $q \implies p \vee q$	Specialization	$p \wedge q \implies p$ $p \wedge q \implies q$
Conjunction	$p$ $q$ $\therefore p \wedge q$	Contradiction Rule	$\sim p \implies F$ $\therefore p$

# INFERENCE RULES IN PROPOSITIONAL LOGIC

1. Idempotent rule:

$$P \wedge P \implies P \quad P \vee P \implies P$$

2. Commutative rule:

$$P \wedge Q \implies Q \wedge P \quad P \vee Q \implies Q \vee P$$

3. Associative rule:

$$P \wedge (Q \wedge R) \implies (P \wedge Q) \wedge R$$

$$P \vee (Q \vee R) \implies (P \vee Q) \vee R$$

#### 4. Distributive Rule:

$$P \vee (Q \wedge R) \implies (P \vee Q) \wedge (P \vee R)$$

$$P \wedge (Q \vee R) \implies (P \wedge Q) \vee (P \wedge R)$$

#### 5. De-Morgan's Rule:

$$\neg(P \vee Q) \implies \neg P \wedge \neg Q$$

$$\neg(P \wedge Q) \implies \neg P \vee \neg Q$$

#### 6. Implication elimination:

$$P \rightarrow Q \implies \neg P \vee Q$$

7. Bidirectional Implication elimination:

$$(P \leftrightarrow Q) \Rightarrow (P \rightarrow Q) \wedge (Q \rightarrow P)$$

8. Contrapositive rule:

$$P \rightarrow Q \Rightarrow \neg P \rightarrow \neg Q$$

9. Double Negation rule:

$$\neg(\neg P) \Rightarrow P$$

7. Absorption Rule:

$$P \vee (P \wedge Q) \Rightarrow P$$

$$P \wedge (P \vee Q) \Rightarrow P$$

## 11. Fundamental identities:

$$P \wedge \neg p \Rightarrow F \quad F \quad [\text{contradiction}]$$

$$P \vee \neg P \Rightarrow T$$

[Tautology]

$$P \vee T \Rightarrow P$$

$$P \vee F \Rightarrow P$$

$$P \vee \neg T \Rightarrow P$$

$$P \wedge F \Rightarrow F$$

$$P \wedge T \Rightarrow P$$

$$P \wedge T \Rightarrow P$$

## 12. Modus Ponens:

If **P** is true and **P→Q** then we can infer **Q** is also true.

$$\begin{array}{c} P \\ P \rightarrow Q \\ \hline \text{Hence, } Q \end{array}$$

## 13. Modus Tollens:

If **¬P** is true and **P→Q** then we can infer **¬Q**.

$$\begin{array}{c} \neg P \\ P \rightarrow Q \\ \hline \text{Hence, } \neg Q \end{array}$$



14. Chain rule:

If  $p \rightarrow q$  and  $q \rightarrow r$  then  $p \rightarrow r$

15. Disjunctive Syllogism:

if  $\neg p$  and  $p \vee q$  we can infer  $q$  is true.

16. AND elimination:

Given  $P$  and  $Q$  are true then we can deduce  $P$  and  $Q$  separately:

$$P \wedge Q \rightarrow P$$

$$P \wedge Q \rightarrow$$

$Q$

17. AND introduction:

Given **P** and **Q** are true then we deduce **P**  $\wedge$  **Q**

18. OR introduction:

Given P and Q are true then we can deduce P and Q separately:

$$P \rightarrow P \vee Q \quad Q \rightarrow P \vee Q$$

$\vee$

- Example:

“I will get wet if it rains and I go out of the house”

Let Propositions be:

**W** : “I will get wet “

**R** : “it rains “

**S** : “I go out of the house”

**$(S \wedge R) \rightarrow W$**

$\wedge$

# PREDICATE LOGIC

- Can represent **objects** and **quantification**
- Theorem proving is **semi-decidable**

## Representing simple facts

(Preposition) "SOCRATES IS A  
SOCRATESMAN -----1  
MAN"

"PLATO IS A  
MAN" -----2  
PLATOMAN

Fails to capture relationship between Socrates and man. We do not get any information about the objects involved Ex:

if asked a question : "who is a man?" we cannot get answer.

Using Predicate Logic however we can represent above facts as:      Man(Socretes)      and      Man(Plato)

# Using Predicate Logic

1. Marcus was a man.

`man(Marcus)`

# Using Predicate Logic

2. Marcus was a  
Pompeian.

Pompeian(Marcus)

- Quantifiers:
- 2 types:-
- Universal quantifier ( $\forall$ )
- $\forall x$ : means “for all” x
- It is used to represent phrase “ for all”.
- It says that something is true for all possible values of a variable.
- Ex. “ John loves everyone”
-



- Quantifiers:
- 2 types:-
- Universal quantifier ( $\forall$ )
- $\forall x$ : means “for all” x
- It is used to represent phrase “ for all”.
- It says that something is true for all possible values of a variable.
- Ex. “ John loves everyone”
- $\forall x$ : loves(John , x)

- Existential quantifier ( $\exists$ ):
- Used to represent the fact “there exists some”
- Ex:
- “some people like reading and hence they gain good knowledge”

$\exists x: \{ \text{[person}(x) \wedge \text{like}(x, \text{reading})]} \rightarrow \text{gain}(x, \text{knowledge}) \}$

- “lord Haggins has a crown on his head”
- “lord Haggins has a crown on his head”
- $\exists x: \text{crown}(x) \wedge \text{onhead}(x, \text{Haggins})$

# Nested Quantifiers

- We can use both  $\forall$  and  $\exists$  separately
- Ex: “everybody loves somebody”

$\forall x: \exists y: \text{loves} (x, y)$

$\forall x: \exists y: \text{loves} (x, y)$

- Connection between  $\forall$  and  $\exists$
- Connection between  $\forall$  and  $\exists$
- “everyone dislikes garlic”

➤ This can be also said as:  $\forall x: \neg \text{like} (x, \text{garlic})$

➤ This can be also said as:

“there does not exist someone who likes garlic”

$\neg \exists x: \text{like} (x, \text{garlic})$

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

$$\forall x: \text{Roman}(x) \rightarrow \text{loyalto}(x, \text{Caesar}) \vee \text{hate}(x, \text{Caesar})$$

4. Every one is loyal to someone.

$$\forall x: \exists y: \text{loyalto}(x, y) \quad \exists y: \forall x: \text{loyalto}(x, y)$$

• People **only** try to assassinate rulers they are not loyal to.

$$\forall x: \forall y: \text{person}(x) \wedge \text{ruler}(y) \wedge \text{tryassassinate}(x, y) \\ \rightarrow \neg \text{loyalto}(x, y)$$

6. “All Pompeians were Romans”

$\forall x: \text{Pompeian}(x) \rightarrow \text{Roman}(x)$

8.        Marcus tried to assassinate  
Caesar.

$\text{tryassassinate}(\text{Marcus}, \text{Caesar})$

# Some more examples

- “all indoor games are easy”

$\forall x: \text{indoor\_game}(x) \rightarrow \text{easy}(x)$

- “Rajiv likes only cricket”

$\text{Like}(\text{Rajiv}, \text{Cricket})$

- “Any person who is respected by every person is a king”

$\exists x: \forall y: \{ \text{person}(x) \wedge \text{person}(y) \wedge \text{respects}(y, x) \rightarrow \text{king}(x) \}$

God help those who help themselves

$\forall x: \text{helps}(\text{god}, \text{helps}(x, x))$

•Everyone who loves all animals is loved by someone”

$\forall x: [ \quad \quad \quad \forall y: \text{animal}(y) \rightarrow \text{loves}(x, y) ]$

everyone  
animals

who loves all

$\exists z: \text{loves}(z, x)$

there exist someone z and z loves x Thus

the predicate sentence is:

$\forall x: [ \forall y: \text{animal}(y) \rightarrow \text{loves}(x, y) ] \rightarrow [ \exists z: \text{loves}(z, x) ]$

# Computable functions and predicates

- “ Marcus was born in 40 A.D”

Born( Marcus, 40)

- “ All Pompeians died when volcano erupted in 79 A.D”

Erupted(volcano, 79)  $\wedge \forall x: [ \text{Pompeian} (x) \rightarrow \text{Died} (x, 79)]$

- “ no mortal lives longer than 150 years”

- How to solve ?

- let **t1 is time instance 1** and **t2 is time instance 2**

- We use computable function **gt( ... , ....)** which computes greater than.

$\forall x: \forall t1: \forall t2: \text{mortal} (x) \wedge \text{born} (x, t1) \wedge \text{gt}( t2 - t1, 150 ) \rightarrow$

dead ( x,



# Resolution algorithm in predicate logic

- Proof by refutation.
- INPUT: Predicate sentences in clausal form (CNF)
- (See conversion algo on next slide)
- Algorithm steps :-

Convert all the propositions of KB to clause form (S).

2. Negate  $\alpha$  and convert it to clause form. Add it to S.

3. Repeat until either a contradiction is found or no progress can be made.

2. Select two clauses  $(\alpha \vee \neg P)$  and  $(\gamma \vee P)$ .

a. Add the resolvent  $(\alpha \vee \gamma)$  to S.

# Conversion to Clause Form

1. Eliminate  $\rightarrow$ .

$$P \rightarrow Q \equiv \neg P \vee Q$$

1. Reduce the scope of each  $\neg$  to a single term.

$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

$$\neg \forall x: P \equiv \exists x: \neg P$$

$$\neg \exists x: P \equiv \forall x: \neg P$$

$$\neg \neg P \equiv P$$

- Standardize **variables** so that each quantifier binds a unique variable.

$$(\forall x: P(x)) \vee (\exists x: Q(x)) \equiv$$

$$(\forall x: P(x)) \vee (\exists y: Q(y))$$

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

$$\begin{aligned}
 & (\forall x: P(x)) \vee (\exists y: Q(y)) \\
 \equiv & \forall x: \exists y: (P(x) \vee Q(y))
 \end{aligned}$$

5. Eliminate  $\exists$  (Skolemization).

$$\exists x: P(x) \equiv P(c)$$

Skolem  
constant

$$\forall x: \exists y P(x, y) \equiv \forall x: P(x, f(x))$$

Skolem  
function

6. Drop  $\forall$ .

$$\forall x: P(x) \equiv P(x)$$

7. Convert the formula into a **conjunction of disjuncts**.

$$(P \wedge Q) \vee R \equiv (P \vee R) \wedge (Q \vee R)$$

- Create a **separate clause** corresponding to each conjunct.
- Standardize apart the **variables** in the set of obtained clauses.

- Example of conversion:

$$\forall x: \neg [ \text{Roman}(x) \rightarrow (\text{Pompeian}(x) \wedge \neg \text{hate}(x, \text{Caesar})) ]$$

After step 1: i.e. elimination of  $\rightarrow$  and  $\Leftrightarrow$  the above stmt becomes:

$$\forall x: \neg [ \neg \text{Roman}(x) \vee (\text{Pompeian}(x) \wedge \neg \text{hate}(x, \text{Caesar})) ]$$



After step 2: i.e. reducing scope of  $\neg$  the above stmt becomes:

$$\forall x: [ \text{Roman}(x) \wedge \neg (\text{Pompeian}(x) \wedge \neg \text{hate}(x, \text{Caesar})) ]$$

$$\forall x: [ \text{Roman}(x) \wedge (\neg \text{Pompeian}(x) \vee \text{hate}(x, \text{Caesar})) ]$$

- Example to demonstrate step 3:- i.e. standardization of variables.

$$\forall x: [ [ \forall y: \text{animal}(y) \rightarrow \text{loves}(x, y) ] \rightarrow [ \exists y: \text{loves}(y, x) ] ]$$

After step 3 above stmt becomes,

$$\forall x: [ [ \forall y: \text{animal}(y) \rightarrow \text{loves}(x, y) ] \rightarrow [ \exists z: \text{loves}(z, x) ] ]$$

- Example to demonstrate step 4: Move all quantifiers to the left without changing their relative order.

- 

- $\forall x: [ [ \forall y: \text{animal}(y) \wedge \exists z: \text{loves}(x, y) ] \vee \text{loves}(z, x) ]$

- After applying step 4 above stmt becomes:

- $\forall x: \forall y: \exists z: [ \text{animal}(y) \wedge \text{loves}(x, y) \vee \text{loves}(z, x) ]$

- After first 4 processing steps of conversion are carried out on original statement S, the statement is said to be in **PRENEX NORMAL FORM**

- Example to demonstrate step 5: skolemization ( i.e. elimination of  $\exists$  quantifier )

$\exists y$ : President (y) Can be transformed into  
President (**S1**)

where **S1** is a function that somehow produces a value that satisfies President (S1) – S1 called as **Skolem constant**

- Ex. 2:

$\exists y$ :  $\forall x$ : leads ( **y** , x )

Here value of y that satisfies 'leads' depends on particular value of x hence above stmt can be written as:

$\forall x$ : leads ( **f(x)** , x )

Where f(x) is **skolem function**.



- Example to demonstrate step 6: dropping prefix  $\forall$

$\forall x: \forall y: \forall z: [\neg \text{Roman}(x) \rightarrow \neg \text{know}(x, y) \rightarrow \text{hate}(y, z)]$

- After prefix dropped becomes,

$[\neg \text{Roman}(x) \rightarrow \neg \text{know}(x, y) \rightarrow \text{hate}(y, z)]$

- Example to demonstrate step 7: Convert the formula into a **conjunction of disjuncts**.(CNF)

- $\text{Roman}(x) \vee ( (\text{hate}(x, \text{caesar}) \wedge \neg \text{loyalto}(x, \text{caesar}) )$

- $\text{Roman}(x) \vee ( (\text{hate}(x, \text{caesar}) \wedge \neg \text{loyalto}(x, \text{caesar}) )$

- $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$

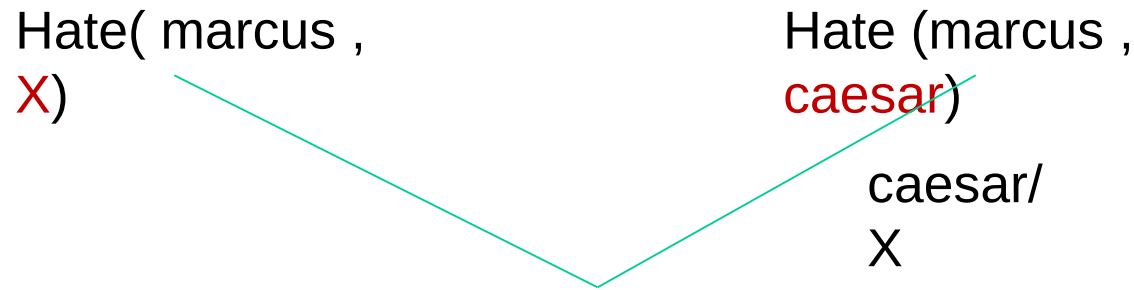
CLAUSE 1  $(P \vee Q) \wedge (P \vee R)$

**CLAUSE 1**  $(\text{Roman}(x) \vee (\text{hate}(x, \text{caesar}) \wedge \neg \text{loyalto}(x, \text{caesar}))$

**CLAUSE 2**  $(\text{Roman}(x) \vee \neg \text{loyalto}(x, \text{caesar}))$

# Unification

- It's a matching procedure that compares two literals and discovers whether there exists a set of substitutions that can make them identical.
- E.g.



e.g.  
2.

Hate(X,  
Y)

Hate( john,  
Z)

could be unified  
as:

John/  
X

and  $y/z$

## Unification:

$\text{UNIFY}(p, q) = \text{unifier } \theta \text{ where } \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$

$\forall x: \text{knows}(\text{John}, x) \rightarrow \text{hates}(\text{John}, x) \text{ knows}(\text{John}, \text{Jane})$

$\forall y: \text{knows}(y, \text{Leonid})$

$\forall y: \text{knows}(y, \text{mother}(y))$

$\forall x: \text{knows}(x, \text{Elizabeth})$

$\text{UNIFY}(\text{knows}(\text{John}, \underline{x}), \text{knows}(\text{John}, \underline{\text{Jane}})) = \{\text{Jane}/x\}$

$\text{UNIFY}(\text{knows}(\underline{\text{John}}, \underline{x}), \text{knows}(y, \underline{\text{Leonid}})) =$   
 $\{\text{Leonid}/x, \text{John}/y\}$

$\text{UNIFY}(\text{knows}(\text{John}, x), \text{knows}(y, \text{mother}(y))) =$   
 $\{\text{John}/y, \text{mother}(\text{John})/x\}$

$\text{UNIFY}(\text{knows}(\text{John}, \underline{x}), \text{knows}(\underline{x}, \text{Elizabeth})) = \text{FAIL}$

# Resolution algorithm

- It is used as inference mechanism.
- Pre-processing steps:
  1. Convert the given English sentence into predicate sentence.
  2. **Not all** of these sentences will be in clausal form (CNF).  
If any sentence is not in clausal form then convert it into clausal form.
  3. Give these sentences (clauses) as an **input** to resolution algorithm.

## Resolution algorithm steps:

A. Negate the proposition which is to be proved.

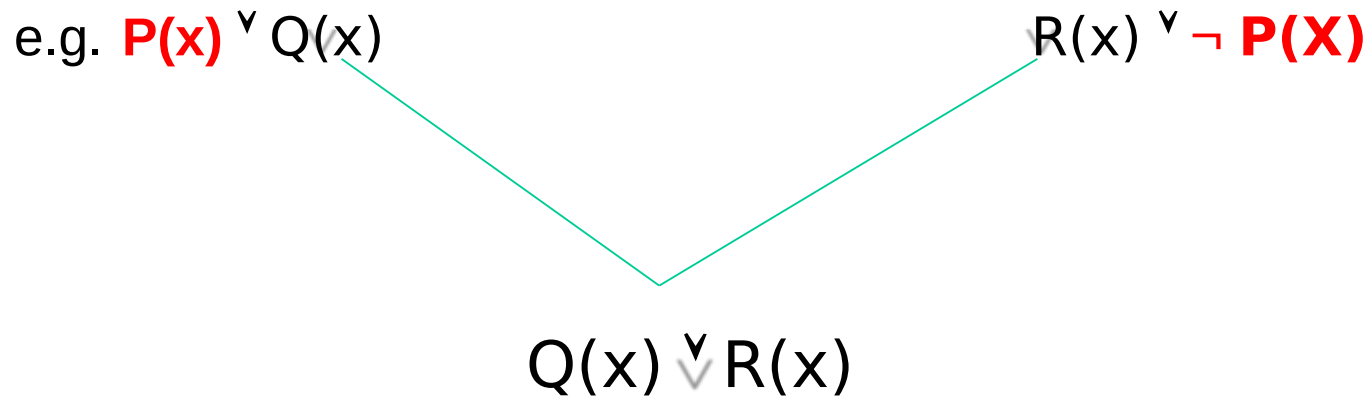
i.e.If we have to prove :-

like(tommy , cookies)            then assume  $\neg$  like(tommy,cookies) Add the resultant sentence to the set of sentences from step 3

B. Repeat until contradiction is found or no progress can be made:

- i. Select two clauses, call them parent clauses and resolve them together.

The resultant clause is called resolvent.



- ii. If resolvent contains empty clause then **contradiction** has been found.



- iii. If step ii. Results in empty clause , it means our assumption is wrong and the original clause (to be proved) has to be true.

# Example

1. Marcus was a man.
2. Marcus was a Pompeian.
3. All Pompeians were Romans.
4. Caesar was a ruler.
5. All Pompeians were either loyal to Caesar or hated him.
6. Every one is loyal to someone.
7. People only try to assassinate rulers they are not loyal to.
8. Marcus tried to assassinate Caesar.



1. “Marcus was a man”

**man(marcus)**

----- 1

2. “Marcus was a Pompeian”

**pompeian (marcus)**

----- 2

3. “All Pompeian's were Romans”

=>  $\forall x1: \text{pompeian}(x1) \rightarrow$   
 $\text{roman}(x1).$

=>  $\forall x1: \neg \text{pompeian}(x1) \vee \text{roman}(x1)$

**$\neg \text{pompeian}(x1) \vee \text{roman}(x1)$**

----- 3

4. “Caesar was a ruler”

**ruler (caesar)**

----- 4

5. “all romans were either loyalto caesar or hated him”

$\Rightarrow \forall x2: \text{roman}(x2) \rightarrow [ \text{loyalto}(x2, \text{caesar}) \vee \text{hate}(x2, \text{caesar}) ]$

$\Rightarrow \forall x2: \neg \text{roman}(x2) \vee \text{loyalto}(x2, \text{caesar}) \vee \text{hate}(x2, \text{caesar})$

$\Rightarrow \neg \text{roman}(x2) \vee \text{loyalto}(x2, \text{caesar}) \vee \text{hate}(x2, \text{caesar})$

**$\neg \text{roman} ( x2) \vee \text{loyalto} (x2, \text{caesar}) \vee \text{hate} (x2, \text{caesar})$**

----- 5

- “Every one is loyal to someone”

$\Rightarrow \forall x_3: \exists y_1: \text{loyalto}(x_3, y_1).$

Let  $f(x_3)$  be a skolem function then,

$\Rightarrow \forall x_3: \text{loyalto}(x_3, f(x_3)).$

$\Rightarrow \text{loyalto}(x_3, f(x_3))$

$\text{loyalto}(x_3, f(x_3))$
-------------------------------

 ----- 6

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

$$\Rightarrow \forall x_4: \forall y_2: [\text{man}(x_4) \wedge \text{ruler}(y_2) \wedge \text{tryassassinate}(x_4, y_2)]$$

$$\rightarrow \neg \text{loyalto}(x_4, y_2)$$

$$\Rightarrow \forall x_4: \forall y_2: \neg [\text{man}(x_4) \wedge \text{ruler}(y_2) \wedge \text{tryassassinate}(x_4, y_2)] \\ \vee \neg \text{loyalto}(x_4, y_2)$$

$$\Rightarrow \forall x_4: \forall y_2: \neg \text{man}(x_4) \vee \neg \text{ruler}(y_2) \vee \neg \text{tryassassinate}(x_4, y_2) \vee \\ \neg \text{loyalto}(x_4, y_2)$$

let  $f(x_4)$  be skolem function then,

$$\Rightarrow \Rightarrow \forall x_4: \neg \text{man}(x_4) \vee \neg \text{ruler}(f(x_4)) \vee$$

$$\neg \text{tryassassinate}(x_4, f(x_4)) \vee \neg \text{loyalto}(x_4, f(x_4))$$

$$\Rightarrow \neg \text{man}(x4) \vee \neg \text{ruler}(f(x4)) \vee \neg \text{tryassassinate}(x4, f(x4)) \vee \neg \text{loyalto}(x4, f(x4))$$

$$\neg \text{man}(x4) \vee \neg \text{ruler}(f(x4)) \vee \neg \text{tryassassinate}(x4, f(x4)) \vee \neg \text{loyalto}(x4, f(x4))$$

----- 7

8. "Marcus tried to assassinate Caesar"

tryassassinate(marcus , caesar)

tryassassinate( marcus ,  
**caesar** )

----- 8

To prove : marcus hate caesar      i.e.    hate(marcus, caesar)

- Assume  $\neg \text{hate}(\text{marcus}, \text{caesar})$

(5)

$\neg \text{roman}(x_2)$   
 $\vee \text{loyalto}(x_2, \text{caesar})$   
 $\vee \text{hate}(x_2, \text{caesar})$

$\text{hate}(x_2, \text{caesar})$

$x_2 / \text{marcus}$

$\neg \text{roman}(\text{marcus}) \vee \text{loyalto}(\text{marcus}, \text{caesar})$

$\neg \text{pompeian}(x_1) \vee \text{roman}(x_1)$

$\text{roman}(x_1)$  (3)

$x_1 / \text{marcus}$

(2)

$\text{pompeian}(\text{marcus})$

$\neg \text{pompeian}(\text{marcus}) \vee \text{loyalto}(\text{marcus}, \text{caesar})$

$\text{loyalto}(\text{marcus}, \text{caesar})$

(7)

$\neg \text{man}(x4) \vee \neg \text{ruler}(f(x4)) \vee \neg \text{tryassassinate}(x4, f(x4)) \vee$   
 $\neg \text{loyalto}(x4, f(x4))$

loyalto (marcus,  
caesar)

x4/ marcus  
f(x4)/  
caesar

$\neg \text{man}(\text{marcus}) \vee \neg \text{ruler}(\text{caesar}) \vee \neg \text{tryassassinate}(\text{marcus}, \text{caesar})$

(8)

tryassassinate( marcus , caesar )

$\neg \text{man}(\text{marcus}) \vee \neg \text{ruler}(\text{caesar})$

(1) man( marcus )

$\neg$   
ruler( caesar )

(4)

ruler( caesar )

$\neg$   
ruler( caesar )

E E

• Since we get an empty clause i.e. contradiction our assumption that  $\neg$  hate(marcus, caesar) is false

hence

hate(marcus, caesar) must be true.



- Consider the following paragraph:

“ anything anyone eats is called food. Milka likes all kind of food. Bread is a food. Mango is a food. Alka eats pizza. Alka eats everything milka eats.”

Translate the following sentences into (WFF) in predicate logic and then into set of clauses. Using resolution principle answer the following:

1. Does Milka like pizza?

2. what food Alka eats?                      [ Question answering]

- Solution:

1. “anything anyone eats is called food.”

$\forall x: \forall y: \text{eats}(x, y) \rightarrow \text{food}(y)$

$\Rightarrow \forall x: \forall y: \neg \text{eats}(x, y) \vee \text{food}(y)$

$\Rightarrow \neg \text{eats}(x, y) \vee \text{food}(y) \quad (1)$

2. “Milka likes all kind of food”

$\forall y1: \text{food}(y1) \rightarrow \text{like}(\text{milka}, y1)$

$\Rightarrow \forall y1: \neg \text{food}(y1) \vee \text{like}(\text{milka}, y1)$

$\Rightarrow \neg \text{food}(y1) \vee \text{like}(\text{milka}, y1) \quad (2)$

3. “Bread is a food”

$\text{food}(\text{bread}) \quad (3)$

4. “Mango is a food”

$\text{food}(\text{mango}) \quad (4)$

5. "Alka eats Pizza"

$\text{eats}(\text{alka}, \text{pizza})$

(5)

6. "Alka eats everything Milka eats"

$\forall x1: \text{eats}(\text{milka}, x1) \rightarrow \text{eats}(\text{alka}, x1)$

$\Rightarrow \quad \forall x1: \quad \neg \text{eats}(\text{milka}, x1) \vee \text{eats}(\text{alka}, x1)$

(6)

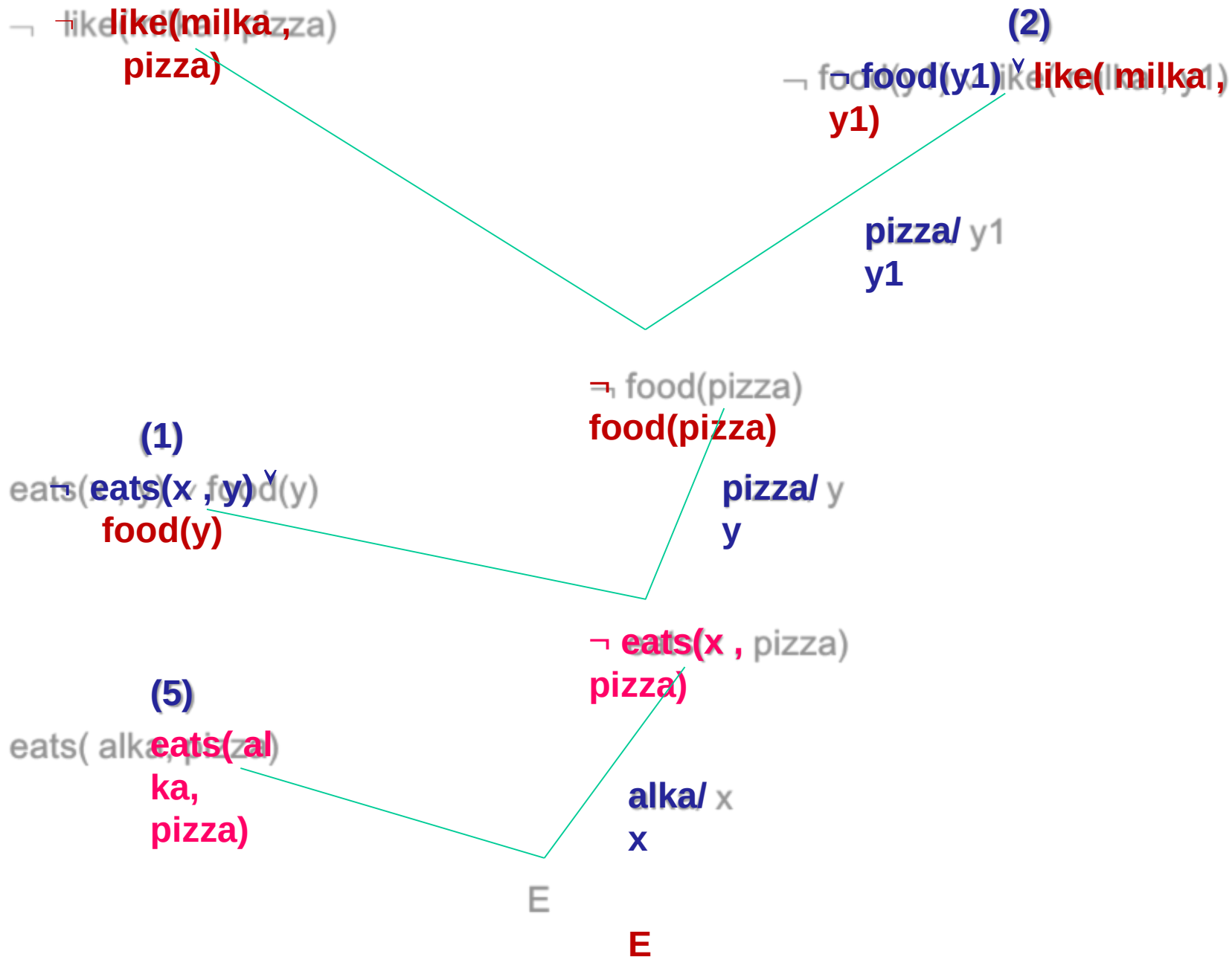
$\Rightarrow \quad \neg \text{eats}(\text{milka}, x1) \vee \text{eats}(\text{alka}, x1)$

Question to be answered : 1. "Does Milka likes Pizza ?"

assume : "Milka **does not** like  
Pizza"

$\neg \text{like}(\text{milka}, \text{pizza})$

(7)



Since  $\neg \text{like}(\text{milka}, \text{pizza})$  is contradiction  $\text{like}(\text{milka}, \text{pizza})$  is true

Question to be answered :

2. “ what

food Alka eats ?” **eats( alka, ??)**

**there exist something which Alka eats we have to find the value of x**

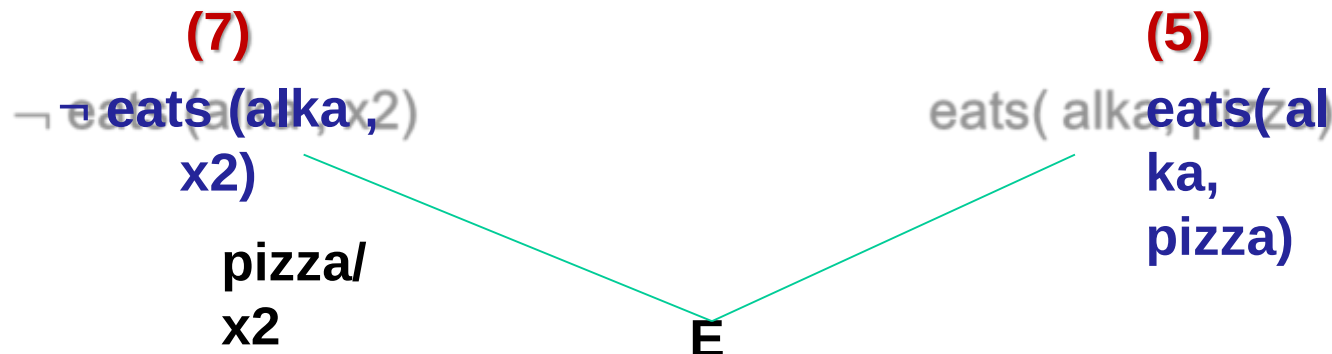
$\exists x: \text{eats} ( \text{alka}, x)$

Assume :      alka does not eat anything

$\neg [\exists x2: \text{eats} ( \text{alka}, x2)]$

**=>**       $\forall x2: \neg \text{eats} ( \text{alka}, x2)$

**=>**       $\neg \text{eats} ( \text{alka}, x2) \quad (7)$



- Therefore alka does not eat anything is false and
- Alka eats something is true.
- And x2 stores pizza

- Therefore we conclude :

eats ( alka, ??) answer is “pizza”

# Instance and Isa relationship

- “ Marcus is a man”

man(marcus)

OR

**instance**( marcus , man)

where marcus is an object/  
instance of class ‘man’

“ all pompeians were romans”

$\forall x: \text{pompeian}(x) \rightarrow \text{roman}(x).$

OR

$\forall x: \text{instance}(x, \text{pompeian}) \rightarrow \text{instance}(x, \text{roman}).$

- Isa Predicate :

“ all pompeians were romans”

$$\forall x: \text{pompeian}(x) \rightarrow \text{roman}(x).$$

OR

$$\forall x: \text{instance}(x, \text{pompeian}) \rightarrow \text{instance}(x, \text{roman}).\text{-----}(1)$$

- Now using isa predicate (1) becomes,

**Isa**( pompeian , roman)

which means pompeian is a subclass of roman class but it also requires extra axiom :

$$\forall x: \forall y: \forall z: \text{isa}(y, z) \wedge \text{instance}(x, y) \rightarrow \text{instance}(x, z)$$



# Using Predicate Logic

- Many English sentences are **ambiguous**.
- There is often a **choice** of how to represent knowledge.
- **Obvious information** may be necessary for reasoning
- We may not know in advance which **statements to deduce** ( $P$  or  $\neg P$ ).

THANK YOU

