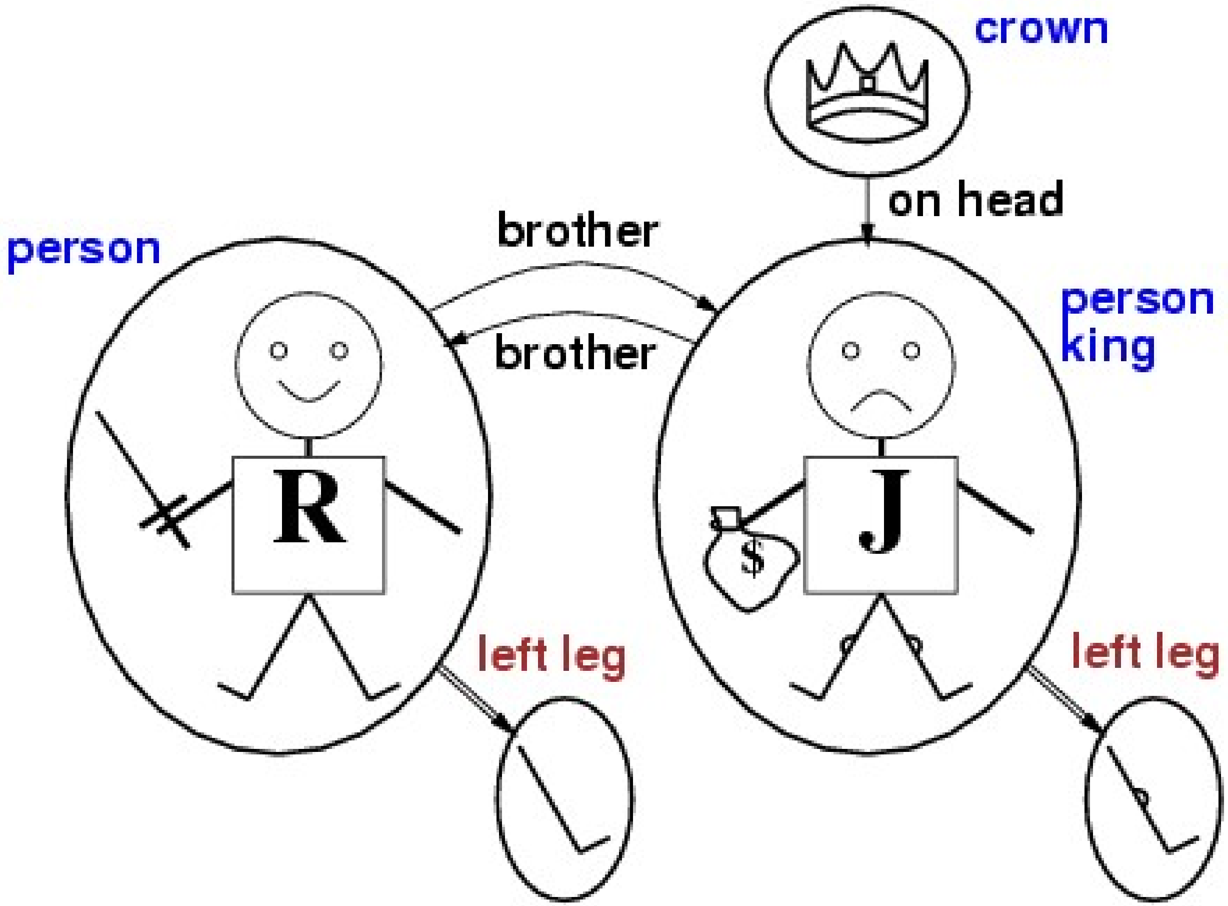
Predicate Logic

Limitations of propositional logic

- ☹ Propositional logic has limited expressive power
  - –unlike natural language
  - –All, some
  - –Logical relations or properties

The Real World



First-Order Logic

- Propositional logic assumes that the world contains facts.

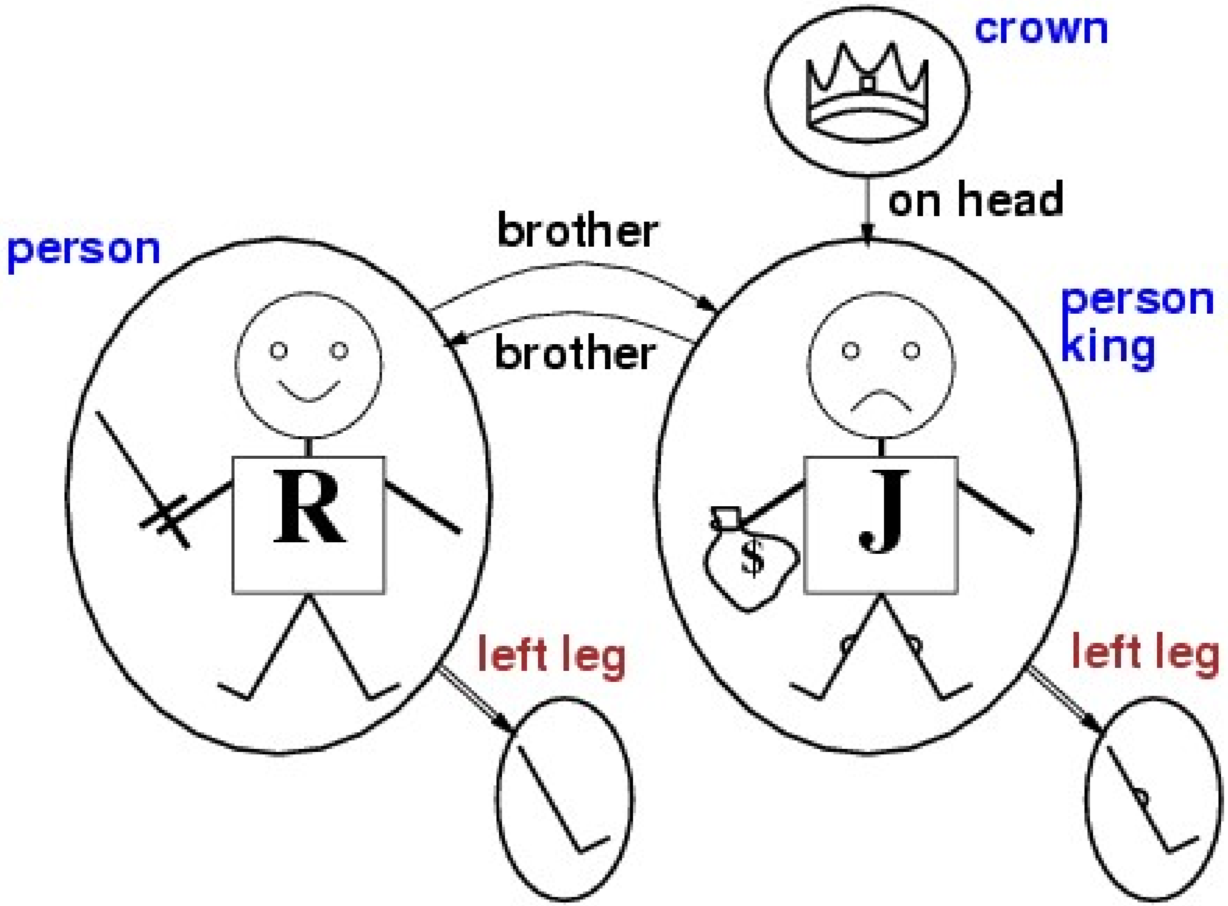
- First-order logic (like natural language) assumes the world contains

  – Objects: people, houses, numbers, colors, baseball games, wars, …

  – Relations: red, round, prime, brother of, bigger than, part of, comes between, …

  – Functions: father of, best friend, one more than, plus, …

Models for FOL: Graphical Example



First-order logic

- First-order logic (FOL) models the world in terms of
  - **Objects,** which are things with individual identities
  - **Properties** of objects that distinguish them from other objects
  - **Relations** that hold among sets of objects
  - **Functions,** which are a subset of relations where there is only one "value" for any given "input"
- Examples:
  - Objects: Students, lectures, companies, cars ...
  - Relations: Brother-of, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, ...
  - Properties: blue, oval, even, large, ...
  - Functions: father-of, best-friend, second-half, one-more-than ...

User provides

- **Constant symbols,** which represent individuals in the world
  - Mary
  - 3
  - Green
- **Function symbols,** map individuals to individuals
  - father-of(Mary) = John
  - color-of(Sky) = Blue
  - LeftLegOf(John)
  - Sqrt(3)
- **Predicate symbols,** which map individuals to truth values
  - greater(5,3)
  - green(Grass)
  - color(Grass, Green)

Relations

- Some relations are properties: they state some fact about a single object: Round(ball), Prime(7).

- n-ary relations state facts about two or more objects: Married(John,Mary), LargerThan(3,2).

- Some relations are functions: their value is another object: Plus(2,3), Father(Dan).

Tabular Representation

- A FOL model is basically equivalent to a relational database instance.
- Historically, the relational data model comes from FOL.

Student

s-id	Intelligence	Ranking
Jack	3	1
Kim	2	1
Paul	1	2

Course

c-id	Rating	Difficulty
101	3	1
102	2	2

Professor

p-id	Popularity	Teaching-a
Oliver	3	1
Jim	2	1

RA

s-id	p-id	Salary	Capability
Jack	Oliver	High	3
Kim	Oliver	Low	1
Paul	Jim	Med	2

Registration

s-id	c.id	Grade	Satisfaction
Jack	101	A	1
Jack	102	B	2
Kim	102	A	1
Paul	101	B	1

FOL Provides

- **Variable symbols**
  - E.g., x, y, foo
- **Connectives**
  - Same as in PL: not ($\neg$), and ($\wedge$), or ($\vee$), implies ($\rightarrow$), if and only if (biconditional $\leftrightarrow$)
- **Quantifiers**
  - Universal $\forall$**x** or **(Ax)**
  - Existential $\exists$**x** or **(Ex)**

Terms

- Term = logical expression that refers to an object.

- There are 2 kinds of terms:
  - constant symbols: Table, Computer
  - function symbols: LeftLeg(Pete), Sqrt(3), Plus(2,3) etc
- Functions can be nested:
  - Pat_Grandfather(x) = father(father(x))
- Terms can contain variables.
- No variables = **ground term**.

Atomic Sentences

- Atomic sentences state facts using terms and predicate symbols
  - P(x,y) interpreted as “x is P of y”

- Examples:

  LargerThan(2,3) is false.

  Brother_of(Mary,Pete) is false.

  Married(Father(Richard), Mother(John)) could be true or false

- Note: Functions do not state facts and form no sentence:
  - Brother(Pete) refers to John (his brother) and is neither true nor false.

- Brother_of(Pete,Brother(Pete)) is True.

Binary relation   Function

Complex Sentences

- A **complex sentence** is formed from atomic sentences connected by the logical connectives:
  - $\neg P$, $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$, $P \leftrightarrow Q$ where P and Q are sentences

More Examples

- Brother(Richard, John) $\wedge$ Brother(John, Richard)

- King(Richard) $\vee$ King(John)

- King(John) => $\neg$ King(Richard)

- LessThan(Plus(1,2) ,4) $\wedge$ GreaterThan(2,1)

More Examples

- Brother(Richard, John) $\wedge$ Brother(John, Richard)
- King(Richard) $\vee$ King(John)
- King(John) => $\neg$ King(Richard)
- LessThan(Plus(1,2) ,4) $\wedge$ GreaterThan(1,2)

(Semantics are the same as in propositional logic)

Universal Quantification ∀

- ∀ means "for all"

- Allows us to make statements about all objects that have certain properties

- Can now state general rules:

  $\forall$ x King(x) => Person(x)

  $\forall$ x Person(x) => HasHead(x)

  $(\forall x)$ dolphin(x) $\rightarrow$ mammal(x)

Existential Quantification $\exists$

- $\exists$ x means "there exists an x such that…." (at least one object x)
- Allows us to make statements about some object without naming it
- Examples:
  - $(\exists x)$ mammal(x) $\wedge$ lays-eggs(x)

Note that $\wedge$ is the natural connective to use with $\exists$

(And => is the natural connective to use with $\forall$)

More examples

For all real x, x>2 implies x>3.

$$\forall x\,[(x > 2) \Rightarrow (x > 3)] \quad x \in R \quad (false)$$

$$\exists x\,[(x^2 = -1)] \quad x \in R \quad (false)$$

There exists some real x whose square is minus 1.

Well formed formula

A **well-formed formula (wff)** is a sentence containing no "free" variables. That is, all variables are "bound" by universal or existential quantifiers.

$(\forall x)P(x,y)$?

has x bound as a universally quantified variable, but y is free.

Quantifiers

- Universal quantifiers are often used with "implies" to form "rules":

  $(\forall x)$ student(x) $\rightarrow$ smart(x) means "All students are smart"

- Universal quantification is *rarely* used to make blanket statements about every individual in the world:

  $(\forall x)$student(x)$\wedge$smart(x) ?

- Existential quantifiers are usually used with "and" to specify a list of properties about an individual:

  $(\exists x)$ student(x) $\wedge$ smart(x) means "There is a student who is smart"

Combining Quantifiers

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

- For everyone ("all x") there is someone ("y") that they love.

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

- there is someone ("y") who is loved by everyone

Clearer with parentheses: $\exists y \, (\forall x \;\; \text{Loves}(x,y) \,)$

Quantifier Scope

- Switching the order of universal quantifiers *does not* change the meaning:
  – $(\forall x)(\forall y)P(x,y) \leftrightarrow (\forall y)(\forall x)\ P(x,y)$
- Similarly, you can switch the order of existential quantifiers:
  – $(\exists x)(\exists y)P(x,y) \leftrightarrow (\exists y)(\exists x)\ P(x,y)$
- Switching the order of universals and existentials *does* change meaning:
  – Everyone likes someone: $(\forall x)(\exists y)$ likes(x,y)
  – Someone is liked by everyone: $(\exists y)(\forall x)$ likes(x,y)

Connections between All and Exists

We can relate sentences involving $\forall$ and $\exists$ using De Morgan's laws:

$$(\forall x)\ \neg P(x) \leftrightarrow \neg(\exists x)\ P(x)$$

$$\neg(\forall x)\ P \leftrightarrow (\exists x)\ \neg P(x)$$

$$(\forall x)\ P(x) \leftrightarrow \neg\ (\exists x)\ \neg P(x)$$

$$(\exists x)\ P(x) \leftrightarrow \neg(\forall x)\ \neg P(x)$$

De Morgan's Law for Quantifiers

De Morgan's Rule

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

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

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

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

Rule is simple: if you bring a negation inside a disjunction or a conjunction, always switch between them (or →and, and → or).

Using FOL

- We want to TELL things to the KB, e.g.
  TELL(KB, $\forall x, King(x) \Rightarrow Person(x)$)
  TELL(KB, *King(John)*)

  These sentences are assertions

- We also want to ASK things to the KB,
  ASK(KB, $\exists x, Person(x)$)

  these are queries or goals

  The KB should output x where Person(x) is true: {x/John,x/Richard,...}

Example: A simple genealogy KB by FOL

- **Build a small genealogy knowledge base using FOL that**
  - contains facts of immediate family relations (spouses, parents, etc.)
  - contains definitions of more complex relations (ancestors, relatives)
  - is able to answer queries about relationships between people
- **Predicates:**
  - parent(x, y), child(x, y), father(x, y), daughter(x, y), etc.
  - spouse(x, y), husband(x, y), wife(x,y)
  - ancestor(x, y), descendant(x, y)
  - male(x), female(y)
  - relative(x, y)
- **Facts:**
  - husband(Joe, Mary), son(Fred, Joe)
  - spouse(John, Nancy), male(John), son(Mark, Nancy)
  - father(Jack, Nancy), daughter(Linda, Jack)
  - daughter(Liz, Linda)
  - etc.

- **Rules for genealogical relations**
  - $(\forall x,y)$ parent(x, y) $\leftrightarrow$ child (y, x)
    $(\forall x,y)$ father(x, y) $\leftrightarrow$ parent(x, y) $\wedge$ male(x) (similarly for mother(x, y))
    $(\forall x,y)$ daughter(x, y) $\leftrightarrow$ child(x, y) $\wedge$ female(x) (similarly for son(x, y))
  - $(\forall x,y)$ husband(x, y) $\leftrightarrow$ spouse(x, y) $\wedge$ male(x) (similarly for wife(x, y))
    $(\forall x,y)$ spouse(x, y) $\leftrightarrow$ spouse(y, x) (**spouse relation is symmetric**)
  - $(\forall x,y)$ parent(x, y) $\rightarrow$ ancestor(x, y)
    $(\forall x,y)(\exists z)$ parent(x, z) $\wedge$ ancestor(z, y) $\rightarrow$ ancestor(x, y)
  - $(\forall x,y)$ descendant(x, y) $\leftrightarrow$ ancestor(y, x)
  - $(\forall x,y)(\exists z)$ ancestor(z, x) $\wedge$ ancestor(z, y) $\rightarrow$ relative(x, y)
    (related by common ancestry)
    $(\forall x,y)$ spouse(x, y) $\rightarrow$ relative(x, y) (related by marriage)
    $(\forall x,y)(\exists z)$ relative(z, x) $\wedge$ relative(z, y) $\rightarrow$ relative(x, y) (**transitive**)
    $(\forall x,y)$ relative(x, y) $\leftrightarrow$ relative(y, x) **(symmetric)**
- **Queries**
  - ancestor(Jack, Fred) /* the answer is yes */
  - relative(Liz, Joe) /* the answer is yes */
  - relative(Nancy, Matthew)
    /* no answer in general, no if under closed world assumption */
  - $(\exists z)$ ancestor(z, Fred) $\wedge$ ancestor(z, Liz)

Universal instantiation (a.k.a. universal elimination)

- If $(\forall x)\ P(x)$ is true, then $P(C)$ is true, where C is *any* constant in the domain of x
- Example:

  $(\forall x)$ eats(Ziggy, x) $\Rightarrow$ eats(Ziggy, IceCream)
- The variable symbol can be replaced by any ground term, i.e., any constant symbol or function symbol applied to ground terms only

Existential instantiation (a.k.a. existential elimination)

- From $(\exists x)\ P(x)$ infer $P(c)$
- Example:
  - $(\exists x)$ eats(Ziggy, x) $\rightarrow$ eats(Ziggy, Stuff)
- Note that the variable is replaced by a **brand-new constant** not occurring in this or any other sentence in the KB
- Also known as skolemization; constant is a **skolem constant**
- In other words, we don't want to accidentally draw other inferences about it by introducing the constant
- Convenient to use this to reason about the unknown object, rather than constantly manipulating the existential quantifier

Existential generalization (a.k.a. existential introduction)

- If P(c) is true, then $(\exists x)$ P(x) is inferred.
- Example

  eats(Ziggy, IceCream) $\Rightarrow$ $(\exists x)$ eats(Ziggy, x)
- All instances of the given constant symbol are replaced by the new variable symbol
- Note that the variable symbol cannot already exist anywhere in the expression

Semantics of FOL

- **Domain M:** the set of all objects in the world (of interest)
- **Interpretation I:** includes
  - Assign each constant to an object in M
  - Define each function of n arguments as a mapping M^n => M
  - Define each predicate of n arguments as a mapping M^n => {T, F}
  - Therefore, every ground predicate with any instantiation will have a truth value
  - In general there is an infinite number of interpretations because |M| is infinite
- **Define logical connectives:** **~, ^, ∨, =>, <=>** as in PL
- **Define semantics of (∀x) and (∃x)**
  - (∀x) P(x) is true iff P(x) is true under all interpretations
  - (∃x) P(x) is true iff P(x) is true under some interpretation

- **Model:** an interpretation of a set of sentences such that every sentence is *True*
- **A sentence is**
  - **satisfiable** if it is true under some interpretation
  - **valid** if it is true under all possible interpretations
  - **inconsistent** if there does not exist any interpretation under which the sentence is true
- **Logical consequence:** S |= X if all models of S are also models of X

Axioms, definitions and theorems

- **Axioms** are facts and rules that attempt to capture all of the (important) facts and concepts about a domain; axioms can be used to prove **theorems**
  - Mathematicians don't want any unnecessary (dependent) axioms –ones that can be derived from other axioms
  - Dependent axioms can make reasoning faster, however
  - Choosing a good set of axioms for a domain is a kind of design problem
- A **definition** of a predicate is of the form "p(X) $\leftrightarrow$ …" and can be decomposed into two parts
  - **Sufficient** description: "p(x) $\rightarrow$ …"
  - **Necessary** description "p(x) $\leftarrow$ …"
  - Some concepts don't have complete definitions (e.g., person(x))

More on definitions

- A **necessary** condition must be satisfied for a statement to be true.
- A **sufficient** condition, if satisfied, assures the statement's truth.
- Duality: "P is sufficient for Q" is the same as "Q is necessary for P."
- Examples: define father(x, y) by parent(x, y) and male(x)
  - parent(x, y) is a necessary (**but not sufficient**) description of father(x, y)
    - father(x, y) → parent(x, y)
  - parent(x, y) ^ male(x) ^ age(x, 35) is a **sufficient (but not necessary**) description of father(x, y):

    father(x, y) ← parent(x, y) ^ male(x) ^ age(x, 35)
  - parent(x, y) ^ male(x) is a **necessary and sufficient** description of father(x, y)

    parent(x, y) ^ male(x) ↔ father(x, y)

Resolution Refutation in Predicate Logic

- Resolution method is used to test unsatisfiability of a set S of clauses in Predicate Logic.
- It is an extension of resolution method for PL.
- The resolution principle basically checks whether empty clause is contained or derived from S.
- Resolution for the clauses containing no variables is very simple and is similar to PL.
- It becomes complicated when clauses contain variables.
- In such case, two complementary literals are resolved after proper substitutions so that both the literals have same arguments.

Example

- Consider two clauses C_1 and C_2 as follows:

$$C_1 = P(x) \ V \ Q(x)$$
$$C_2 = \sim P(f(x)) \ V \ R(x)$$

- Substitute 'f(a)' for 'x' in C_1 and 'a' for 'x' in C_2, where 'a' is a new constant from the domain, then

$$C_3 = P(f(a)) \ V \ Q(f(a))$$
$$C_4 = \sim P(f(a)) \ V \ R(a)$$

- Resolvent C of C_3 and C_4 is **[Q(f(a)) V R(a)]**
  - Here C_3 and C_4 do not have variables. They are called ground instances of C_1 and C_2 .
- In general, if we substitute 'f(x)' for 'x' in C_1, then

$$C'_1 = P(f(x)) \ V \ Q(f(x))$$

- Resolvent C' of C'_1 and C_2 is **[Q(f(x)) V R(x)]**
- We notice that C is an instance of C' .

Theorems

- Logical Consequence: L is a logical consequence of S iff $\{S \cup \sim L\} = \{ C_{1}, \ldots, C_n, \sim L\}$ is unsatisfiable.
  - A deduction of an empty clause from a set S of clauses is called a *resolution refutation* of S.
- Soundness and completeness of resolution: There is a *resolution refutation* of S if and only if S is *unsatisfiable* (inconsistent).
- L is a logical consequence of S if and only if there is a resolution refutation of $S \cup \{\sim L\}$.
- We can summarize that in order to show L to be a logical consequence the of set of formulae $\{\alpha_1, \ldots, \alpha_n\}$, use the procedure given on next slide.

Procedure

- Obtain a set S of all the clauses.
- Show that a set $S \cup \{ \sim L\}$ is unsatisfiable i.e.,
  - the set $S \cup \{ \sim L\}$ contains either empty clause or empty clause can be derived in finite steps using resolution method.
  - If so, then report *'Yes'* and conclude that L is a logical consequence of S and subsequently of formulae $\alpha_1, \ldots, \alpha_n$ otherwise report *'No'*.
- Resolution refutation algorithm finds a contradiction if one exists, if clauses to resolve at each step are chosen systematically.
- There exist various strategies for making the right choice that can speed up the process considerably.

Useful Tips

- Initially choose a clause from the negated goal clauses as one of the parents to be resolved.
  - This corresponds to intuition that the contradiction we are looking for must be because of the formula to be proved .
- Choose a resolvent and some existing clause if both contain complementary literals.
- If such clauses do not exists, then resolve any pairs of clauses that contain complementary literals.
- Whenever possible, resolve with the clauses with single literal.
  - Such resolution generate new clauses with fewer literals than the larger of their parent clauses and thus probably algorithm terminates faster.
- Eliminate tautologies and clauses that are subsumed by other clauses as soon as they are generated.

Logic Programming

- Logic programming is based on FOPL.
- Clause in logic programming is special form of FOPL formula.
- Program in logic programming is a collection of clauses.
- Queries are solved using resolution principle.
- A **clause** in logic programming is represented in a clausal notation as

  $A_1, \ldots, A_k \leftarrow B_1, \ldots, B_t$,

  where A_j are positive literals and B_k are negative literals.

Conversion of a Clause into Clausal Notation

- A clause in FOPL is a closed formula of the form,

  $$L_1 \vee \ldots \vee L_m$$

  where each L_k , $(1 \le k \le n)$ is a literal and all the variables occurring in $L_1, \ldots, L_m$ are universally quantified.

- Separate positive and negative literals in the clause as follows:

  $$(L_1 \vee \ldots \vee L_m)$$

  $$\cong \quad (A_1 \vee \ldots \vee A_k \vee \sim B_1 \vee \ldots \vee \sim B_t),$$

  where $m = k + t$, A_j, $(1 \le j \le k)$ are positive literals and B_j , $(1 \le j \le t)$ are negative literals

Cont…

$$(L_1 V \ldots V L_m) \cong (A_1 V \ldots VA_k \; V \sim B_1 V \ldots V\sim B_t),$$
$$\cong (A_1 V \ldots VA_k) V \sim (B_1 \Lambda \ldots \Lambda B_t)$$
$$\cong (B_1 \Lambda \ldots \Lambda B_t) \rightarrow (A_1 V \ldots VA_k)$$

{since $P \rightarrow Q \cong \sim P \; V \, Q$}

- Clausal notation is written in the form :

  $(A_1 V \ldots VA_k) \leftarrow (B_1 \Lambda \ldots \Lambda B_t)$ OR $A_1, \ldots, A_k \leftarrow B_1, \ldots, B_t$.

  Here A_j , $(1 \leq j \leq k)$ are positive literals and B_i , $(1 \leq i \leq t)$ are negative literals.
- Interpretations of $A \leftarrow B$ and $B \rightarrow A$ are same.
- In clausal notation, all variables are assumed to be universally quantified.
  - B_i , $(1 \leq i \leq t)$ (negative literals) are called *antecedents* and A_j , $(1 \leq j \leq k)$ (positive literals) are called *consequents*.
  - Commas in antecedent and consequent denote *conjunction* and *disjunction* respectively.

Cont…

- Applying the results of FOPL to the logic programs,
  - a goal G with respect to a program P (finite set of clauses) is solved by showing that the set of clauses $P \cup \{\sim G\}$ is unsatisfiable or there is a resolution refutation of $P \cup \{\sim G\}$.
- If so, then G is logical consequence of a program P.
- There are three basic statements. These are special forms of clauses.
  - facts,
  - rules and
  - queries.

Example

- Consider the following logic program.

  GRANDFATHER (x, y) ← FATHER (x, z) , PARENT (z, y)

  PARENT (x, y) ← FATHER (x, y)

  PARENT (x, y) ← MOTHER (x, y)

  FATHER (abraham, robert) ←

  FATHER (robert, mike) ←

- In FOL above program is represented as a set of clauses as:

S = { GRANDFATHER (x, y) V ~FATHER (x, z) V ~ PARENT (z, y),
PARENT(x, y) V ~ FATHER (x, y),
PARENT(x, y) V ~ MOTHER (x, y),
FATHER (abraham, robert),
FATHER (robert, mike)
}

Example

- Let us number the clauses of S as follows:
  i. GRANDFATHER(x, y) V ~FATHER(x, z) V ~PARENT(z, y)
  ii. PARENT(x, y) V ~ FATHER (x, y)
  iii. PARENT(x, y) V ~ MOTHER (x, y),
  iv. FATHER (abraham, robert)
  v. FATHER (robert, mike)
- *Simple queries :*
  - Ground Query
  Query:“Is abraham a grandfather of mike ?"

Example – Cont…

- Ground Query "Is abraham a grandfather of mike ?"

  ← GRANDFATHER (abraham, mike).

- In FOPL, ~GRANDFATHER(abraham, mike) is negation of goal { GRANDFATHER (abraham, mike).
- Include {~goal} in the set S and show using resolution refutation that S ∪ {~ goal} is unsatisfiable in order to conclude the goal.
- Let ~ goal is numbered as (vi) in continuation of first five clauses of S listed above.

  vi. ~ GRANDFATHER (abraham, mike)

- Resolution tree is given on next slide:

Resolution Tree

(i) (vi)

{x / abraham, y / mike}

(iv) ~ FATHER(abraham, z) V ~PARENT(z, mike)

{z / robert}

~ PARENT (robert, mike) (ii)

~ FATHER (robert, mike) (v)

Answer: Yes

□