Knowledge Representation and Reasoning: Logic

General features of a representation

Representational adequacy

The ability to represent all of the kinds of knowledge that are needed in a certain domain.

Inferential adequacy

The ability to represent all of the kinds of inferential procedures (procedures that manipulate the representational structures in such a way as to derive new structures corresponding to new knowledge inferred from old).

Inferential efficiency

The ability to represent efficient inference procedures (for instance, by incorporating into the knowledge structure additional information that can be used to focus the attention of the inference mechanisms in the most promising directions).

Acquisitional efficiency

The ability to acquire new information easily.

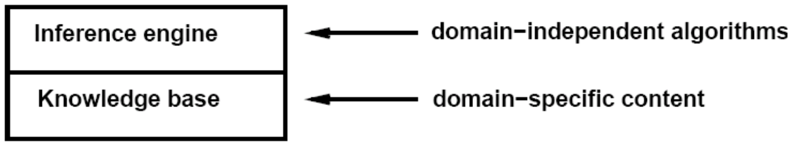
Logic

Logic (Knowledge-Based) agents combine general knowledge with current percepts to infer hidden aspects of current state prior to selecting actions

- Crucial in partially observable environments

Knowledge Base

Knowledge Base : set of sentences represented in a knowledge representation language and represents assertions about the world.



Then it can ask itself what to do-answers should follow from the KB
Agents can be viewed at the knowledge level
    i.e., what they know, regardless of how implemented
Or at the implementation level
    i.e., data structures in KB and algorithms that manipulate them

<u>Abilities KB agent</u>

Agent must be able to:

- Represent states and actions,
- Incorporate new percepts
- Update internal representation of the world
- Deduce hidden properties of the world
- Deduce appropriate actions

<u>Logic</u>

Logics are formal languages for representing information such that conclusions can be drawn: It has syntax and semantics:

- Syntax defines the sentences in the language
- Semantics define the meaning of sentences. that defines truth of a sentence in a world
-

E.g., the language of arithmetic
$x + 2 \geq y$ is a sentence; $x2 + y >$ is not a sentence
$x + 2 \geq y$ is true iff the number $x + 2$ is no less than the number y
$x + 2 \geq y$ is true in a world where $x=7$; $y=1$
$x + 2 \geq y$ is false in a world where $x=0$; $y=6$

Entailment

Entailment means that one thing follows from another:

$$KB \models \alpha$$

Knowledge base KB entails sentence α if and only if α is true in all worlds where KB is true

E.g., $x + y = 4$ entails $4 = x + y$

Entailment is a relationship between sentences (i.e., syntax) that is based on semantics

Models

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.

m is a model of a sentence α if α is true in m.
M(α) is the set of all models of α

Logical inference

If an inference algorithm i can derive α from KB, we write KB $|-_i$ α which is pronounced as " i derives α from KB"

If an algorithm only derives entailed sentences it is called *sound* or *truth preserving*.

- Otherwise it just makes things up.

i is sound if whenever KB |-i α it is also true that KB|= α

Completeness : the algorithm can derive any sentence that is entailed.

i is complete if whenever KB |= α it is also true that KB|-i α

Propositional Logic:Syntax

Propositional logic is the simplest logic. We use the symbols like P1, P2 to represent sentences. Propositional logic is defined as:

If S is a sentence, ¬S is a sentence (negation)
If S1 and S2 are sentences, S1 ^ S2 is a sentence (conjunction)
If S1 and S2 are sentences, S1 ∨ S2 is a sentence (disjunction)
If S1 and S2 are sentences, S1 ⇒ S2 is a sentence (implication)
If S1 and S2 are sentences, S1 ⇔ S2 is a sentence (biconditional)

Formal grammar for propositional logic can be given as below:

Sentennce	→ AutomicSentence | ComplexSedntence
AutomicSentence	→ True | False | Symbol
Symbol	→ P | Q | R …………
ComplexSentence	→ ¬Sentence
	| (Sentence ^ Sentence)
	| (Sentence ∨ Sentence)
	| (Sentence ⇒ Sentence)
	| (Sentence ⇔ Sentence)

Propositional Logic: Semantics

Each model specifes true/false for each proposition symbol

Rules for evaluating truth with respect to a model:
$\neg S$ is true if, S is false
S1 ^ S2 is true if, S1 is true and S2 is true
S1 $\vee$ S2 is true if, S1 is true or S2 is true
S1 $\Rightarrow$ S2 is true if, S1 is false or S2 is true
S1 $\Leftrightarrow$ S2 is true if, S1 $\Rightarrow$ S2 is true and S2 $\Rightarrow$ S1 is true

Truth Table showing the evaluation of semantics of complex sentences:

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

Logical equivalence
Two sentences α and ß are *logically equivalent* (α ≡ ß) iff true they are true inn same set of models
or Two sentences α and ß are *logically equivalent* (α ≡ ß) iff α \|= ß and ß \|= α.

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

Validity

A sentence is *valid* if it is true in all models,
e.g., *True*, A$\vee\neg$A, A $\Rightarrow$ A, (A $\wedge$ (A $\Rightarrow$ B)) $\Rightarrow$ B

Valid sentences are also known as tautologies
Every valid sentence is logically equivalent to True

Satisfiability

A sentence is *satisfiable* if it is true in *some* model

- e.g., $A \vee B$, C

A sentence is *unsatisfiable* if it is true in *no* models

- e.g., $A \neg \wedge A$

Validity and satisfiablity are related concepts

- α is valid iff $\neg\alpha$ is unsatisfiable
- α is satisfiable iff $\neg\alpha$ is not valid

Satisfiability is connected to inference via the following:

- $KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfiable

Inference rules in PL

Modus Ponens

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

And-elimination

$$\frac{\alpha \wedge \beta}{\alpha}$$

Monotonicity: the set of entailed sentences can only increase as information is added to the knowledge base.

For any sentence α and β if $KB \models \alpha$ then $KB \wedge \beta \models \alpha$.

Resolution

Unit resolution rule:

Unit resolution rule takes a clause – a disjunction of literals – and a literal and produces a new clause. Single literal is also called unit clause.

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k}$$

Where ℓ_i and m are complementary literals

Generalized resolution rule:

Generalized resolution rule takes two clauses of any length and produces a new clause as below.

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

For example

$$\frac{\ell_1 \vee \ell_2, \qquad \neg\ell_2 \vee \ell_3}{\ell_1 \vee \ell_3}$$

Resolution Uses CNF (Conjunctive normal form)

- Conjunction of disjunctions of literals (clauses)

The resolution rule is sound:

- Only entailed sentences are derived

Resolution is complete in the sense that it can always be used to either confirm or refute a sentence (it can not be used to enumerate true sentences.)

Conversion to CNF

A sentence that is expressed as a conjunction of disjunctions of literals is said to be in conjunctive normal form (CNF).
A sentence in CNF that contains only k literals per clause is said to be in k-CNF.

Let's illustrate the conversion to CNF by using an example.

$B \Leftrightarrow (A \vee C)$

- Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow ß$ with $(\alpha \Rightarrow ß) \wedge (ß \Rightarrow \alpha)$.
  - $(B \Rightarrow (A \vee C)) \wedge ((A \vee C) \Rightarrow B)$

- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow ß$ with $\neg\, \alpha \vee ß$.
  - $(\neg B \vee A \vee C) \wedge (\neg(A \vee C) \vee B)$

- Move $\neg$ inwards using de Morgan's rules and double-negation:
  - $(\neg B \vee A \vee C) \wedge ((\neg A \wedge \neg C) \vee B)$

- Apply distributivity law ($\wedge$ over $\vee$) and flatten:
  - $(\neg B \vee A \vee C) \wedge (\neg A \vee B) \wedge (\neg C \vee B)$

Resolution algorithm

Convert KB into CNF
Add negation of sentence to be entailed into KB i.e. $(KB \wedge \neg\alpha)$
Then apply resolution rule to resulting clauses.
The process continues until:

- There are no new clauses that can be added
    - Hence *KB* **does not** entail α
- Two clauses resolve to entail the empty clause.
    - Hence *KB* **does** entail α

Resolution example

Consider the knowledge base given as below:
$KB = (B \Leftrightarrow (A \vee C)) \wedge \neg B$

Prove that $\neg A$ can be inferred from above KB by using resolution.

Convert KB into CNF

$$B \Rightarrow (A \vee C)) \wedge ((A \vee C) \Rightarrow B) \wedge \neg B$$

$$(\neg B \vee A \vee C) \wedge (\neg(A \vee C) \vee B) \wedge \neg B$$

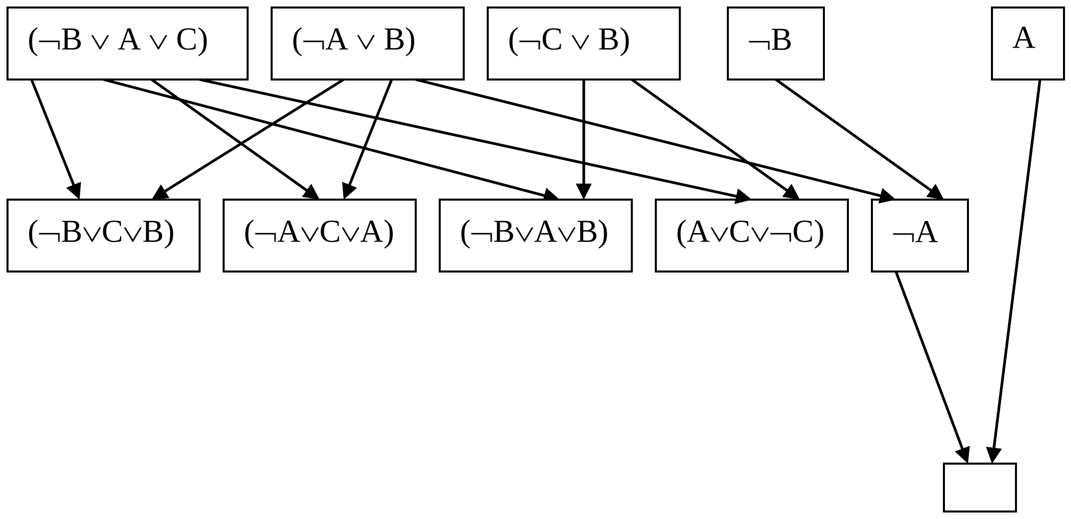
$$(\neg B \vee A \vee C) \wedge ((\neg A \wedge \neg C) \vee B) \wedge \neg B$$

$$(\neg B \vee A \vee C) \wedge (\neg A \vee B) \wedge (\neg C \vee B) \wedge \neg B$$

Add negation of sentence to be inferred from KB into KB

Now KB contains following sentences all in CNF
$(\neg B \vee A \vee C)$
$(\neg A \vee B)$
$(\neg C \vee B)$
$\neg B$
A

Now use Resolution algorithm



Forward and backward chaining

The completeness of resolution makes it a very important inference model. But in many practical situations full power of resolution is not needed. Real-world knowledge bases often contain only clauses of restricted kind called **Horn Clause.**
A Horn clauses is disjunction of literals with at most one positive literal
Three important properties of Horn clause are:

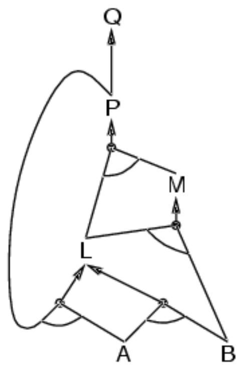
- ✓ Can be written as an implication
- ✓ Inference through forward chaining and backward chaining.
- ✓ Deciding entailment can be done in a time linear size of the knowledge base.

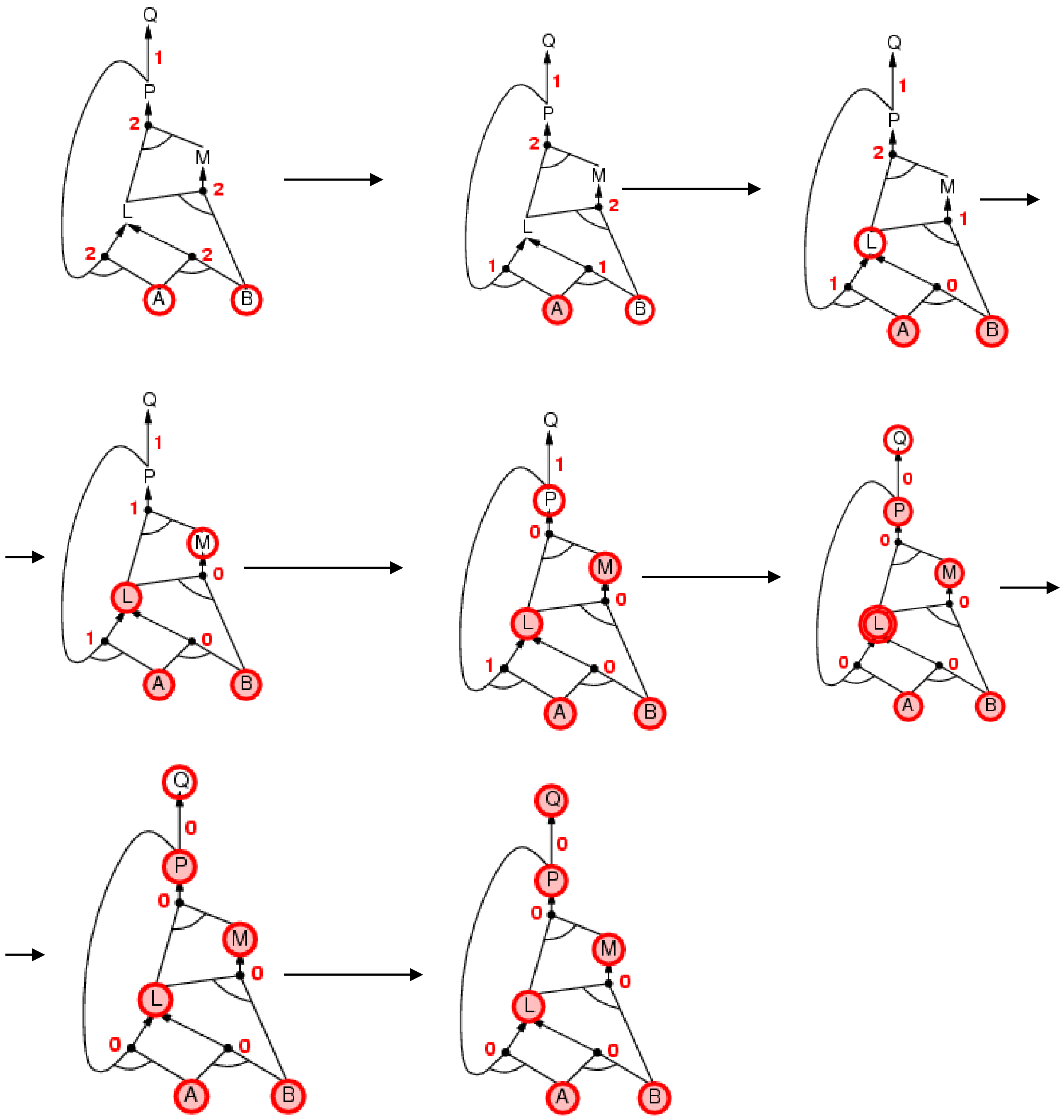
Forward chaining

Idea: fire any rule whose premises are satisfied in the *KB*,

- add its conclusion to the *KB*, until query is found

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B




Q
P
M
L
A
B

Backward chaining

Idea: work backwards from the query *q*: to prove *q* by BC,
Check if *q* is known already, or
Prove by BC all premises of some rule concluding *q*

Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal
1. has already been proved true, or
2. has already failed

Foe example for above KB

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Prove that Q can be inferred from above KB

We know $P \Rightarrow Q$, try to prove P
$L \wedge M \Rightarrow P$
Try to prove L and M
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
Try to prove B, L and A and P
A and B is already known, since $A \wedge B \Rightarrow L$, L is also known
Since, $B \wedge L \Rightarrow M$, M is also known
Since, $L \wedge M \Rightarrow P$, p is known, hence the proof

First-Order Logic

Pros and cons of propositional logic

Propositional logic is declarative
Propositional logic allows partial/disjunctive/negated information
- (unlike most data structures and databases)

Propositional logic is compositional:
  - meaning of $B \wedge P$ is derived from meaning of B and of P

Meaning in propositional logic is context-independent
- (unlike natural language, where meaning depends on context)

Propositional logic has very limited expressive power
- (unlike natural language)
- E.g., cannot say "pits cause breezes in adjacent squares"
    - except by writing one sentence for each square

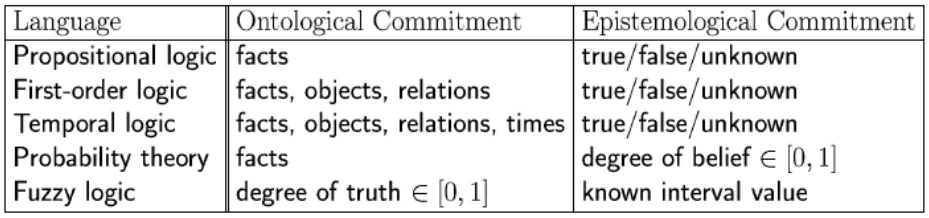
First-order logic

Whereas propositional logic assumes 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, …

Logics in General

The primary difference between PL and FOPL is their ontological commitment:
Ontological Commitment: What exists in the world — TRUTH
- PL: facts hold or do not hold.
- FL : objects with relations between them that hold or do not hold

Another difference is:
Epistemological Commitment: What an agent believes about facts — BELIEF



Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0,1]$
Fuzzy logic	degree of truth $\in [0,1]$	known interval value

Representing knowledge in first-order logic

The objects from the real world are represented by constant symbols (a,b,c,...). For instance, the symbol “Tom” may represent a certain individual called Tom.

Properties of objects may be represented by predicates applied to those objects (P(a), ...): e.g "male(Tom)" represents that Tom is a male.

Relationships between objects are represented by predicates with more arguments: "father(Tom, Bob)" represents the fact that Tom is the father of Bob.

The value of a predicate is one of the boolean constants T (i.e. true) or F (i.e. false)."father(Tom, Bob) = T" means that the sentence "Tom is the father of Bob" is true. "father(Tom, Bob) = F" means that the sentence "Tom is the father of Bob" is false.

Besides constants, the arguments of the predicates may be functions (f,g,...) or variables (x,y,...).

Function symbols denote mappings from elements of a domain (or tuples of elements of domains) to elements of a domain. For instance, weight is a function that maps objects to their weight: weight (Tom) = 150.Therefore the predicate greater-than (weight (Bob), 100) means that the weight of Bob is greater than 100. The arguments of a function may themselves be functions.

Variable symbols represent potentially any element of a domain and allow the formulation of general statements about the elements of the domain.

The quantifier’s ∀ and ∃ are used to build new formulas from old ones.
"∃x P(x)" expresses that there is at least one element of the domain that makes P(x) true.
"∃x mother(x, Bob)” means that there is x such that x is mother of Bob or, otherwise stated, Bob has a mother.
"∀x P(x)" expresses that for all elements of the domain P(x) is true.

Quantifiers
Allows us to express properties of collections of objects instead of enumerating objects by name. Two quantifiers are:
Universal: “for all” ∀
Existential: “there exists” ∃

Universal quantification:

∀<*Variables*> <*sentence*>
Everyone at VUB is smart:

$$\forall x\ At(x,VUB) \Rightarrow Smart(x)$$

∀x *P* is true in a model *m* iff *P* is true for all *x* in the model

Roughly speaking, equivalent to the conjunction of instantiations of *P*

$At(KingJohn,VUB) \Rightarrow Smart(KingJohn) \wedge \quad At(Richard,VUB) \Rightarrow Smart(Richard)$
$\quad \wedge At(VUB,VUB) \Rightarrow Smart(VUB) \wedge \ldots$

Typically, $\Rightarrow$ is the main connective with $\forall$

- A universally quantifier is also equivalent to a set of implications over all objects

Common mistake: using $\wedge$ as the main connective with $\forall$:
$\forall x\ At(x, VUB) \wedge Smart(x)$
means "Everyone is at VUB and everyone is smart"

Existential quantification

$\exists$<*variables*> <*sentence*>
Someone at VUB is smart:
$\quad \exists x\ At(x, VUB) \wedge Smart(x)$
$\exists x\ P$ is true in a model *m* iff *P* is true for at least one *x* in the model
Roughly speaking, equivalent to the disjunction of instantiations of *P*

$At(KingJohn,VUB) \wedge Smart(KingJohn) \vee At(Richard,VUB) \wedge Smart(Richard)$
$\vee At(VUB, VUB) \wedge Smart(VUB) \vee \ldots$

Typically, $\wedge$ is the main connective with $\exists$

Common mistake: using $\Rightarrow$ as the main connective with $\exists$:
$\exists x\ At(x, VUB) \Rightarrow Smart(x)$
is true even if there is anyone who is not at VUB!

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"

Quantifier duality: each can be expressed using the other
$\quad \forall x\ Likes(x,IceCream) \quad \equiv \quad \exists\neg x\ \neg Likes(x,IceCream)$
$\quad \exists x\ Likes(x,Broccoli) \quad \equiv \quad \neg\forall x\ \neg Likes(x,Broccoli)$

Equality

term1 = term2 is true under a given interpretation if and only if *term1* and *term2* refer to the same object
e.g.
Father (John) = Henary
Here Objects referred by father of john and Henary is same.

<u>Inference in first-order logic</u>

First Order Logic to Predicate Logic

First order inference can be done by converting the knowledge base to PL and using propositional inference.

- How to convert universal quantifiers?
  - Replace variable by ground term.
- How to convert existential quantifiers?
  - Skolemization.

Universal instantiation (UI)
Substitute ground term (term without variables) for the variables.
For example consider the following KB
$\forall$ x King (x) $\wedge$ Greedy (x) $\Rightarrow$ Evil(x)
King (John)
Greedy (John)
Brother (Richard, John)

It's UI is:

King (John) $\wedge$ Greedy (John) $\Rightarrow$ Evil(John)
King (Richard) $\wedge$ Greedy (Richard) $\Rightarrow$ Evil(Richard)
King (John)
Greedy (John)
Brother (Richard, John)

Note: Remove universally quantified sentences after universal instantiation.

Existential instantiation (EI)
For any sentence α and variable v in that, introduce a constant that is not in the KB (called skolem constant) and substitute that constant for v.
e.g.
Consider the sentence.
$\exists$ x crown(x) $\wedge$ OnHead(x, John)

After EI
$crown(C1) \wedge OnHead(C1, John)$ where C1 is Skolem Constant.

EI versus UI
UI can be applied several times to *add* new sentences; the new KB is logically equivalent to the old.
EI can be applied once to replace the existential sentence; the new KB is not equivalent to the old but is satisfiable if the old KB was satisfiable.

Note: After propositionalization, we can use any algorithm fool Pl to derive the conclusion.

<u>Unification and Lifting</u>

Instead of translating the knowledge base to PL, we can redefine the inference rules into FOL.

- Lifting: they only make those substitutions that are required to allow particular inferences to proceed.
- E.g. *generalized Modus Ponens*
    - To introduce substitutions different logical expressions have to be look identical
    - To identify the substitution we need unification algorithm.

Unification:
We can get the inference immediately if we can find a substitution α such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

α = {x/John,y/John} works

Unify(α ,β) = θ if $\alpha\theta = \theta\beta$

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x,OJ)	{fail}

Last unification is failed due to overlap of variables. x can not take the values of John and OJ at the same time.
We can avoid this problem by renaming to avoid the name clashes (standardizing apart)
e.g.

Unify{Knows(John,x) Knows(z,OJ) } = {x/OJ, z/John}

Another complication:
To unify *Knows(John,x)* and *Knows(y,z)*,
Unification of Knows*(John,x)* and *Knows(y,z)* gives α ={y/John, x/z } or α={y/John, x/John, z/John}

First unifier gives the result Knows(John,z) and second unifier gives the resultKnows(John, John). Second can be achieved from first by substituting john in place of z. The first unifier is more general than the second.

There is a single most general unifier (MGU) that is unique up to renaming of variables.
MGU = { y/John, x/z }

Generalized Modus Ponens (GMP)

$$\frac{p1', p2', \ldots , pn', (p1 \wedge p2 \wedge \ldots \wedge pn \Rightarrow q)}{q\theta} \qquad \text{where } pi'\theta = pi\theta \text{ for all } i$$

p1' is King(John)	p1 is King(x)
p2' is Greedy(y)	p2 is Greedy(x)
θ is {x/John,y/John}	q is Evil(x)
$q\theta$ is Evil(John)	

GMP used with KB of definite clauses (exactly one positive literal).

<u>Forward Chaining:</u>

We can use FC foe FOPL also:
Consider the following KB:
The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

Prove that Col. West is a criminal

Soln:
it is a crime for an American to sell weapons to hostile nations:
American(x) $\wedge$ Weapon(y) $\wedge$ Sells(x,y,z) $\wedge$ Hostile(z) $\Rightarrow$ Criminal(x)

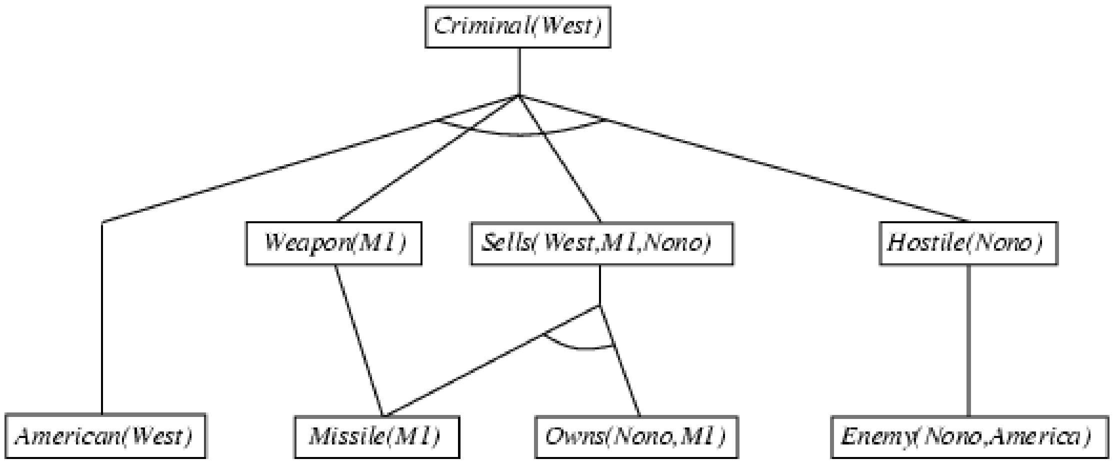
Nono … has some missiles, i.e.,
$\exists$x Owns(Nono,x) $\wedge$ Missile(x):
Use Existential Instantiation:
Owns(Nono,M1) and Missile(M1)

… all of its missiles were sold to it by Colonel West
Missile(x) $\wedge$ *Owns(Nono,x)* $\Rightarrow$ *Sells(West,x,Nono)*

Missiles are weapons:
Missile(x) $\Rightarrow$ *Weapon(x)*
An enemy of America counts as "hostile“:
Enemy(x,America) $\Rightarrow$ *Hostile(x)*

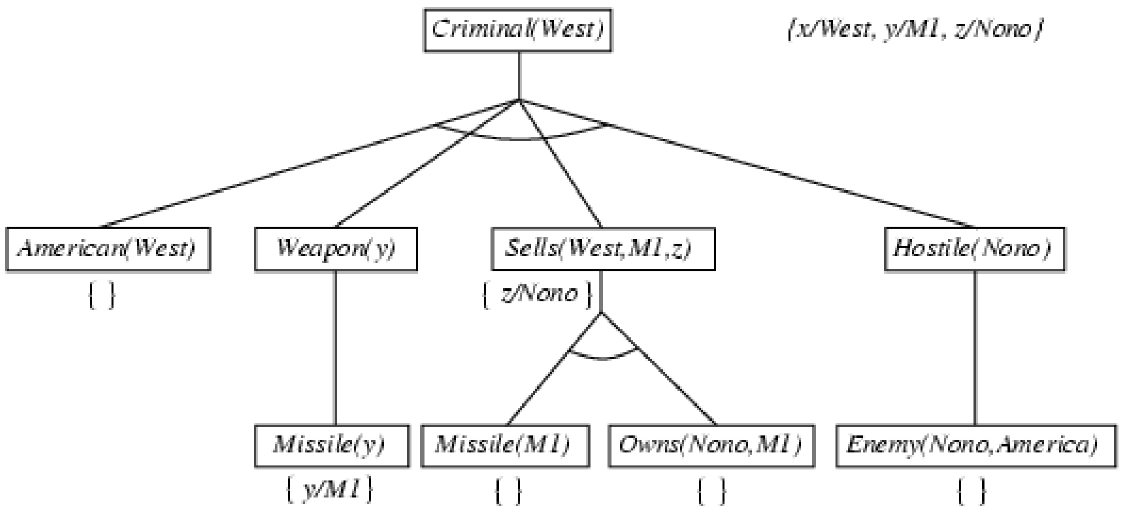
West, who is American …
American(West)

The country Nono, an enemy of America …
Enemy(Nono,America)



<u>Backward Chaining:</u>

We can also use BC algorithm for FOPL.
Eg. For above KB we can use BC al below:



Exercise:

- Robert is an American. All Americans who sells weapons are criminals. All missiles are weapons. Robert sells weapons. Prove that Robert is criminal.
- SRK is a superstar. All superstars are rich. Rich mans have fast cars. Fast cars consume a lot of petrol. Prove that SRK's car consumes a lot of petrol.
- All greedy leaders are autocrat. Autocrats are evils. Prashant is a greedy leader. Ram is a honest leader. Prove that Prashant is evil.
- All students of this class takes a course in BSC IT. BSC IT students are good in AI. Ritu is student of this class. Prove that Ritu is good in AI.