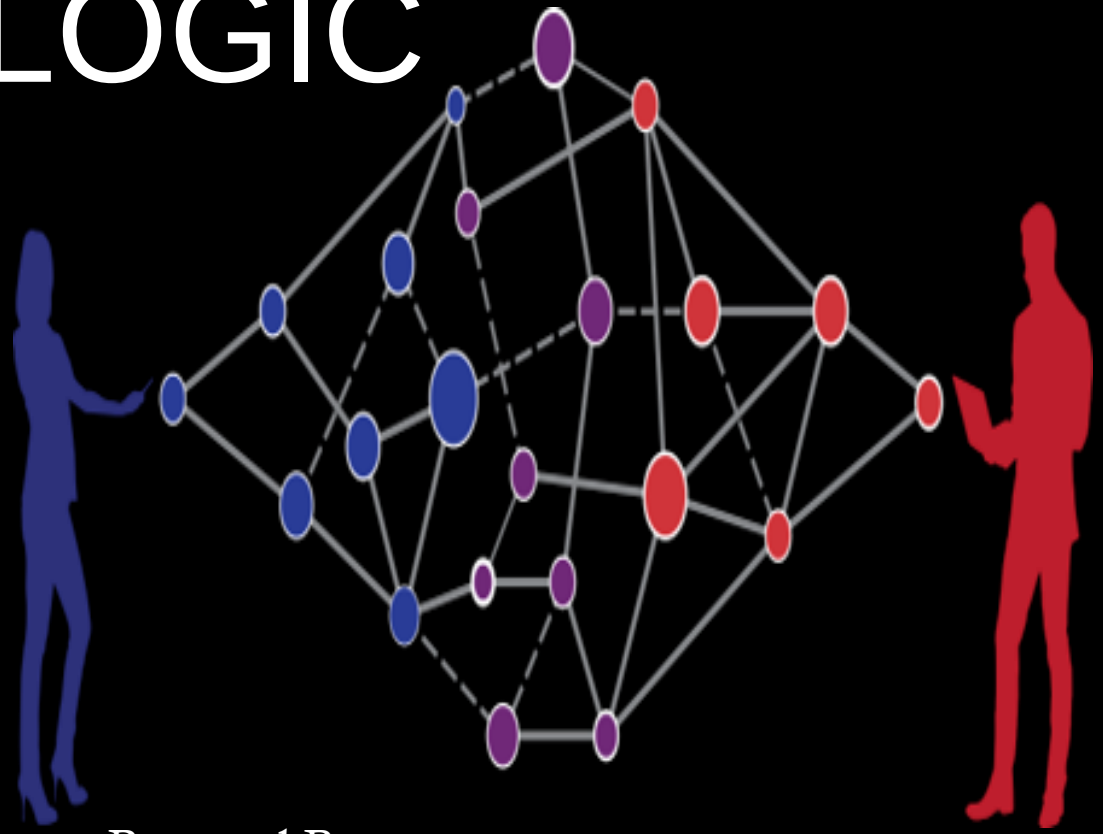
FIRST ORDER LOGIC



Prepared By

-Anooja Joy

FIRST-ORDER LOGIC

- **Propositional logic** Assumes that the world contains **facts** of from **yes or no.**
- There is a need for a language that is **declarative**(knowledge + inference based on truth relation between sentences), **compositional**(meaning of a sentence is meaning of its parts), **context-independent**, **more expressive**(to deal with partial information using disjunction and negation) and **unambiguous**.
- First-order logic is also called **Predicate logic** and **First-order predicate calculus (FOPL)**. It is a formal representation of logic in the form of **quantifier**s. In predicate logic, the **input** is taken as an **entity**, and the **output** it gives is either **true or false.**
- **First-order logic** Assumes that the world contains
  - **Objects** people, houses, numbers, theories, Donald Duck, colors, centuries, …
  - **Relations** red, round, prime, multistoried, … brother of, bigger than, part of, has color, occurred after, owns
  - **Functions** +, middle of, father of, one more than, beginning of, …

FORMAL LANGUAGES: ONTOLOGY AND EPISTEMOLOGY

- **Ontology** is the study of existence claim ie, what it assumes about **nature of reality**. Ie, an inventory of what exists. Ontological commitment means **"WHAT EXISTS IN THE WORLD"**.
- **Epistemology** is a major branch of philosophy that concerns the forms, nature, and preconditions of knowledge. Epistological commitment is a commitment that considers **possible states of knowledge** with respect to each fact. Eepistemological commitment means **"WHAT AN AGENT BELIEVES ABOUT FACTS".**

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

CONCEPTUALIZATION

- One plus two equals three
  - **Objects:** one, two, three, one plus two
  - **Function:** plus
  - **Relation:** equals
- Squares neighboring the wumpus are smelly
  - **Objects:** wumpus, square
  - **Property:** smelly
  - **Relations:** neighboring
- Evil King John ruled England in 1200
  - **Objects:** John, England, 1200
  - **Properties:** evil, king
  - **Relations:** ruled

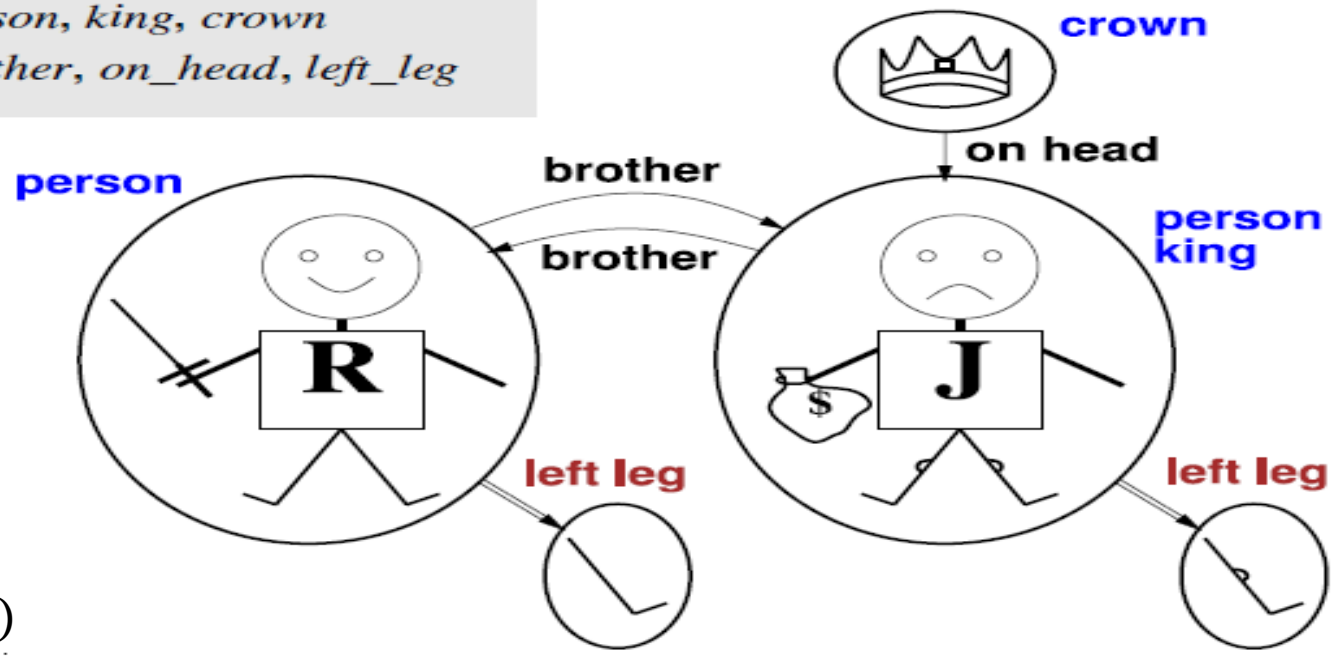
SYNTAX AND SEMANTICS OF FOL

MODELS FOR FOL

- **Model** contains **domain elements** and **relations** among them.
  - Models for **propositional logic** are just **sets of truth values** for the proposition symbols.
  - Models for **first-order logic** have **objects, relations** and **functions**.
- **Syntax:** It defines the way of representing the **given predicates**. As these predicates are represented via **quantifiers**, there are different types of quantifiers used.
- **Semantics:** It defines the sense of the given predicate. It allows to make more logical expression by devising its semantics. Semantics allow us to understand the sentence meaning.
- **Predicates:** express relationships between objects
- **Quantifiers:** Quantifiers are used to express properties of entire collections of objects, instead of enumerating the objects by name if a logic that allows object is found. It can restrict the scope of variables
- In FOL, **variables** refer to things in the world and can quantify over the-talk about all or some of them without naming them explicitly.

MODELS FOR FOL

Constants:	*KingJohn, Richard*
Predicates:	*person, king, crown*
Functions:	*brother, on_head, left_leg*



Objects (Ex:- Richard)
Relations (Ex:- King)
functions (Ex:- LeftLeg)

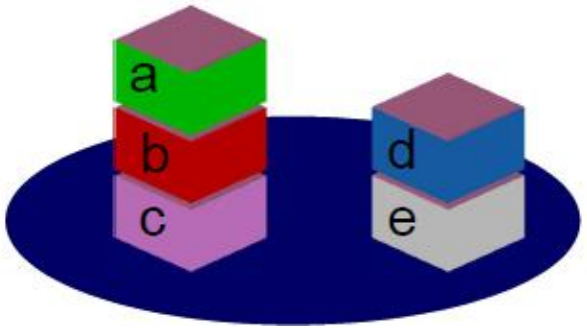
- The picture depicts Richard the Lionheart, King of England from 1189 to 1199; his younger brother, the evil King John, who ruled from 1199 to 1215; the left legs of Richard and John; and a crown.

MODELS FOR FOL: OBJECTS

- The model contain **5 objects**, 2 **binary relations**(brother, onhead), **3 unary relations**(person, person king, crown) and **1 unary function**(left leg)
- **Objects** refers to an entity that exists in the real world.
- The picture shows a model with five **objects:**
  - Richard the Lionheart
  - His younger brother
  - The evil King John
  - The left legs of Richard and John
  - A crown

MODELS FOR FOL: RELATIONS

- The objects in the model may be **related** in various ways, In the figure Richard and John are **brothers**. Formally speaking, a **relation** is just the **set of tuples of objects that are related.**
- A **tuple** is **a collection of Objects arranged in a fixed order and is written with angle brackets surrounding the objects.**
- **Eg:** The **brotherhood** relation in this model is the set {**<Richard the Lionheart, King John>**,**<King John, Richard the Lionheart>**} The crown is on King John's head, so the "**on head**" relation contains just one tuple, **<the crown, King John>**.
- The relation can be:
  - **binary relation** relating pairs of objects (**Eg:** "**Brother**")
  - **unary relation** representing a common object (**Eg:**"**Person**" representing both **Richard** and **John**)



Eg: Set of blocks {a, b, c, d, e}. The "On" relation includes: On = {<a,b>,<b,c> ,<d,e> }<a,b> represents the predicate On(A,B)

MODELS FOR FOL: FUNCTIONS

- Certain kinds of relationships are best considered as functions that relates an object to exactly one object.
- **Eg:**- each person has one left leg, so the model has a **unary** "**left leg**" **function** that includes the following mappings (Richard the Lionheart) ----> Richard's left leg

SYNTAX: ELEMENTS AND SYMBOLS OF FOL

- **The elements for which different symbols are defined are:**
  - **Objects(Constant Symbols):** It refers to an entity that exists in the real world. **For example,** Ram, John, etc. are referred to as Objects.
  - **Functions(Function Symbols):** Any function performed by the object/on the object. **For example,** LeftLeg, writes, eats, Sqrt, etc. are some of the functions.
  - **Relations(Predicate Symbols):** The relation of an object with the other object defines its relation**. For example,** brother, mother, king,>, =, … . etc. are some types of relations which exist in the real world.

1. **Constant Symbols:** These symbols are used to represent the objects. **Eg:** KingJohn, 2, Koblenz, C,…
2. **Function Symbols:** These symbols are used to represent the functions.**Eg:** Sqrt, LeftLegOf,…
3. **Predicate Symbols** are used to represent relations. **Eg:** brother, mother, king,>,
4. **Variable Symbols** x, y, a, b, …
5. **Connectives** ∧ ∨ ¬ ⇒ ⇔
6. **Equality:** ==
7. **Quantifiers** ∀ ∃

SYNTAX: ATOMIC AND COMPLEX SENTENCES

- **Atomic Sentences:** These sentences are formed via predicate symbols may or may not be followed by a list of terms.

Example:

Parents(Ram, Sita) where Ram and Sita are the parents.

Brother(KingJohn,RichardTheLionheart)

Length(LeftLegOf(KingJohn)))

- **Complex Sentences:** These sentences make use of logical connectives to construct more complex sentences from atomic sentences .

Example:

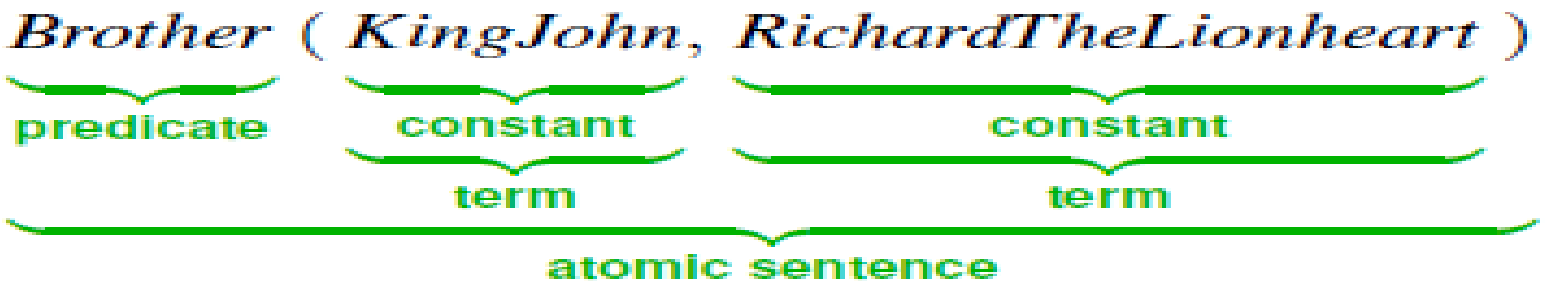
Sibling(KingJohn,Richard) ⇒ Sibling(Richard,KingJohn)

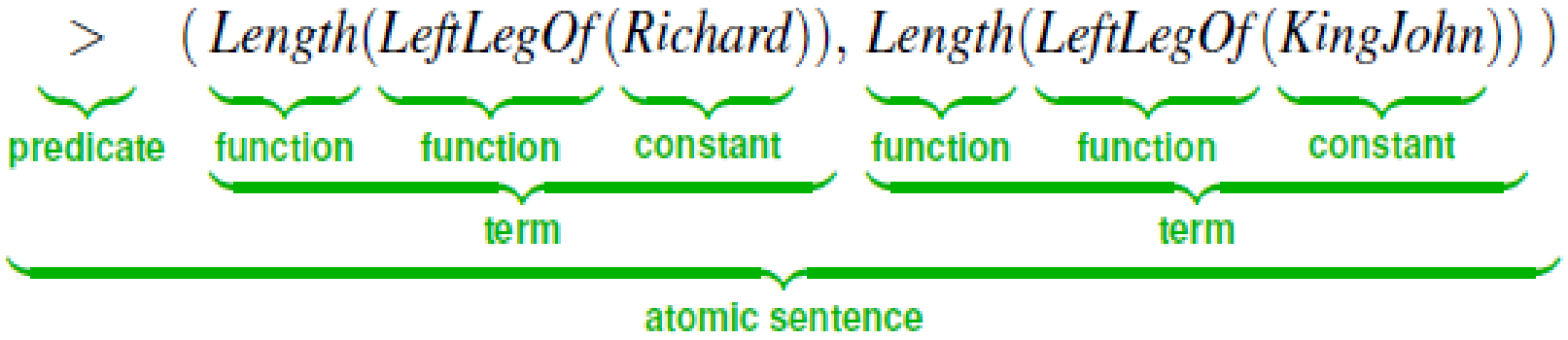
¬Brother (LeftLeg(Richard), John)

King (Richard) ∨ King (John)

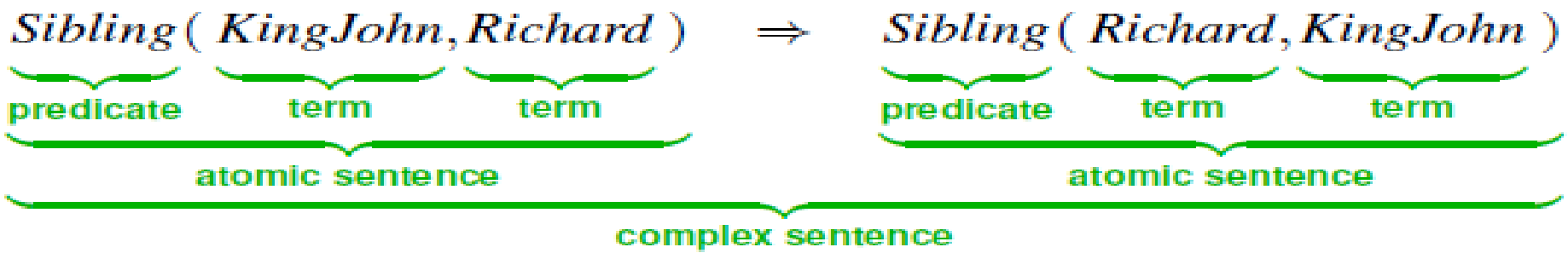
King (Richard)⇒ King(John)

EXAMPLES: ATOMIC SENTENCES





EXAMPLES: COMPLEX SENTENCES



SEMANTICS: INTENDED INTERPRETATION

- The semantics must **relate sentences** to models in order to **determine truth**. To do this, an **interpretation** is needed.
- **Interpretation specifying exactly which objects, relations and functions are referred to by the constant, predicate and function symbols.** Interpretation specifies referents for ie, it maps c**onstant symbols → objects, predicate symbols → relations, function symbols → functional relations**
- One possible interpretation called as the **intended interpretation**- is as follows;
  - **Richard** refers to Richard the Lionheart and **John** refers to the evil King John.
  - **Brother** refers to the brotherhood relation, that is the set of tuples of objects given in equation {(Richard the Lionheart, King John),(King John, Richard the Lionheart)}
  - **OnHead** refers to the **"on head" relation** that holds between the crown and King John; Person, King and Crown refer to the set of objects that are persons, kings and crowns.
  - **Leftleg** refers to the **"left leg" function**, that is, the mapping given in {(Richard the Lionheart, King John),(King John, Richard the Lionheart)}
- There are many other possible interpretations relating these symbols to this particular model.
- The **truth of any sentence** is determined by a **model** and an **interpretation** for the sentence symbols.

SEMANTICS: QUANTIFIERS

- Once we have a logic that allows objects, it is natural to want to express properties of entire collections of objects, instead of enumerating the objects by name. **Quantifiers** is a way for that.
- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- Quantifiers are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. These are thetypes of quantifier:

1. **Universal Quantifier, (for all, everyone, everything)**
2. **Existential quantifier, (for some, at least one)**
3. **Nested Quantifiers**

UNIVERSAL QUANTIFIER

- Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing. The Universal quantifier is represented by a symbol ∀, which resembles an inverted A.
- In general, **∀ x P is true** in a given model under a given interpretation **if P is true in all possible extended interpretations.**
- **Universal quantification (∀) Eg**
  - "**All kings are person**": ∀x King(x) ⇒ Person(x) which means For all x, if x is a king, then x is a person.Here x could be one of the following(5 extended interpretations): Richard, John, Richard's left leg, John's left leg, Crown

EXISTENTIAL QUANTIFIER

- Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something. It is denoted by the logical operator ∃, which resembles as inverted E. When it is used with a predicate variable then it is called as an existential quantifier.
- In general, **∃x P is true** in a given model under a given interpretation **if P is true in at least one extended interpretation that assigns x to a domain element.**
- Existential quantification (∃) Eg
  - **∃x Crown(x) ∧ OnHead(x, John)** which means there is an x such that x is a crown and x is on the John's head. Here , "Crown(crown) ∧ OnHead(crown, John)" is true

NESTED QUANTIFIERS

- It is the nesting of the same type of quantifier. One predicate is nested under the other predicate.
- Nested quantifiers Example
  - **$\forall x \, \forall y$ [Brother(x, y) $\Rightarrow$ Sibling(x, y)]**
  - **$\forall x \, \exists y$ Loves(x, y)**
  - **$\exists y \, \forall x$ Loves(x, y)**
- The **order of quantification is important**.
- **$\exists x \, \forall y$** is not the same as **$\forall y \, \exists x$.**

PROPERTIES OF QUANTIFIERS

Quantifier duality: each can be expressed using the other

- $\forall x \text{Likes}(x, \text{IceCream})$ is the same as $\neg \exists x \neg \text{Likes}(x, \text{IceCream})$
- $\exists x \text{Likes}(x, \text{Broccoli})$ is the same as $\neg \forall x \neg \text{Likes}(x, \text{Broccoli})$
- **De Morgan's rules for quantifiers**
  - $\forall x \neg P = \neg \exists x\ P$
  - $\neg\ \forall x\ P = \exists x \neg\ P$
  - $\forall x\ P = \neg \exists x\ \neg\ P$
  - $\neg\ \forall x \neg P = \exists x\ P$

PROPERTIES OF QUANTIFIERS

Quantifiers of same type commute

- ∀x ∀y is the same as ∀y ∀x
- ∃x ∃y is the same as ∃y ∃x

Quantifiers of different type do NOT commute

- ∃x ∀y is not the same as ∀y ∃x

Example

- ∃x ∀yLoves(x, y): "There is a person who loves everyone in the world"
- ∀y ∃xLoves(x, y): "Everyone in the world is loved by at least one person"

THINGS TO BE TAKEN CARE

- → is the main connective with ∀ where as ^ as the main connective with ∃
- It's a mistake to use ^ as the main connective with ∀ and Using → as the main connective with ∃.
- Example
- Correct:
  - ∀x (StudiesAt(x,Koblenz))→Smart(x)): "Everyone who studies at Koblenz is smart"
  - ∃x (StudiesAt(x,Landau) ^ Smart(x)): "There is someone who studies at Landau and is smart"
- Wrong:
  - ∀x (StudiesAt(x,Koblenz) ^ Smart(x)): "Everyone studies at Koblenz and is smart," i.e., "Everyone studies at Koblenz and everyone is smart"
  - ∃x (StudiesAt(x;Landau))→Smart(x)): "There is someone who, if he/she studies at Landau, is smart" This is true if there is anyone not studying at Landau

EQUALITY

- **term1 = term2** is true under a given interpretation if and only if **term1 and term2 refer to the same object .**
- Can be used to **state facts** about a given function

E.g., Father(John) = Henry

- Can be **used with negation to insist that two terms are not the same object**

USING FOL:

- **Example 1:** Lipton is a tea.
- **Solution:** Here, the object is Lipton. It will be represented as **Tea(Lipton). T**here is no requirement of quantifiers because the quantity is not specified in the given predicate
- **Example 2:** Every man is mortal.
- **Solution:** Here, the quantifier is the universal identifier, and the object is man. Let x be the man. Thus, it will be represented as **∀x: man(x) → mortal(x).**
- **Example 3:** All girls are beautiful.
- **Solution:** Here, we are talking about all girls. It means universal quantifier will be used. The object is girls. Let, y be the girls. Therefore, it will be represented as **∀y: girls(y) → beautiful(y).**
- **Example 4:** All that glitters is not gold.
- **Solution:** Here, we will represent gold as x. Therefore, it will be represented as **∀x: glitters(x) → ¬gold(x).**
- **Example 5:** Some boys are obedient.
- **Solution:** Here, boys are objects. The quantifier used will be existential quantifier. Let x be the boys. Thus, it will be represented as **∃x: boys(x) → obedient(x).**
- **Example 6:** Some cows are black and some cows are white.
- **Solution:** Let, x be the cows. Therefore, it will be represented as: **∃x: cows(x) → black(x) Λ white(x).**

USING FOL:

1. **All birds fly:** **$\forall x$ bird(x) $\rightarrow$fly(x).**
   In this question the predicate is "fly(bird)." And since there are all birds who fly so it will be represented as follows

2. **Every man respects his parent.** **$\forall x$ man(x) $\rightarrow$ respects (x, parent).**
   In this question, the predicate is "respect(x, y)," where x=man, and y= parent. Since there is every man so will use $\forall$, and it will be represented as follows

3. **Some boys play cricket.** **$\exists x$ boys(x) $\rightarrow$ play(x, cricket).**
   In this question, the predicate is "play(x, y)," where x= boys, and y= game. Since there are some boys so we will use **$\exists$, and it will be represented as**:

4. **Not all students like both Mathematics and Science** **$\neg\forall$ (x) [student(x) $\rightarrow$ like(x, Mathematics) $\wedge$ like(x, Science)].**
   In this question, the predicate is "like(x, y)," where x= student, and y= subject. Since there are not all students, so we will use **$\forall$ with negation, so** following representation for this:

5. **Only one student failed in Mathematics.** **$\exists$(x) [student(x) $\rightarrow$ failed (x, Mathematics) $\wedge\forall$ (y) [$\neg$(x==y) $\wedge$ student(y) $\rightarrow$ $\neg$failed (x, Mathematics)].**
   In this question, the predicate is "failed(x, y)," where x= student, and y= subject. Since there is only one student who failed in Mathematics, so we will use following representation for this.

USING FOL: KINSHIP DOMAIN

- One's mom is one's female parent: **∀x, y (Mother(x, y) ⇔ (Female(x)^Parent(x, y)))**
- One's husband is one's male spouse: **∀w,h Husband(h, w)⇔ Male(h) ΛSpouse(h, w)**
- "Brothers are siblings." **∀x, y (Brother(x, y))⇔ Sibling(x, y))**
- Parent and child are inverse relations: **∀ p, c Parent(p, c)⇔ Child(c, p)**
- A Grandparent is a Parent of one's Parent: **∀ g, c Grandparent(g,c) ⇔ ∃p Parent(g,p) ΛParent(p,c)**
- "A first cousin is a child of a parent's sibling." **∀x, y (FirstCousin(x, y)⇔ ∃p, ps (Parent(p, x)^Sibling(ps, p)^Parent(ps, y)))**

Practice:

- Male and female are disjoint categories
- A sibling is another child of one's parents

SOLUTION

- Male and female are disjoint categories:
- **$\forall x$ Male(x) $\Leftrightarrow \neg$Female(x)**
- A sibling is another child of one's parent:
- **$\forall x,y$ Sibling(x, y) $\Leftrightarrow$ $x \neq y$ ^ $\exists p$ Parent(p,x) ^ Parent(p, y)**
- Definition of (full) sibling in terms of Parent
- **$\forall x, y$ Sibling(x, y) $\Leftrightarrow$ ($\neg(x = y)$^ $\exists m, f$ ($\neg(m = f)$^ Parent(m, x)^Parent(f , x)^ Parent(m, y)^Parent(f , y)))**

FREE AND BOUND VARIABLES

- The quantifiers interact with variables which appear in a suitable way. There are two types of variables in First-order logic which are given below:
- **Free Variable:** A variable is said to be a free variable in a formula if it occurs outside the scope of the quantifier.

Example: ∀x ∃(y)[P (x, y, z)], where z is a free variable.

- **Bound Variable:** A variable is said to be a bound variable in a formula if it occurs within the scope of the quantifier.

Example: **∀x [A (x) B(y)], here x and y are the bound variables.**

INFERENCE IN FOL

INFERENCE IN FOL

- Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences.
- Two ideas:
  - convert the KB to propositional logic and use propositional inference
  - a shortcut that manipulates on first-order sentences directly

FOL INFERENCE RULES FOR QUANTIFIER:

- As propositional logic we also have inference rules in first-order logic, so following are some basic inference rules in FOL:

1. **Universal Generalization**
2. **Universal Instantiation**
3. **Existential Instantiation**
4. **Existential introduction**

1. UNIVERSAL GENERALIZATION:

- Universal generalization is a valid inference rule which states that if premise P(c) is true for any arbitrary element c in the universe of discourse, then we can have a conclusion as ∀ x P(x).
- It can be represented as:

$$\frac{P(c)}{\forall x\, P(x)}$$

- This rule can be used if we want to show that every element has a similar property.
- In this rule, x must not appear as a free variable.
- **Example:** Let's represent, P(c): "**A byte contains 8 bits**", so for **∀ x P(x)** "**All bytes contain 8 bits**.", it will also be true.

2. UNIVERSAL INSTANTIATION (UI):

- In this, we can infer any sentence by substituting a ground term (a term without variables) for the variables. In short, when we create the FOPL of the given statement, we can easily infer any other statement using that FOPL . It can be represented as:

∀ x P(x)

P(c)

- **Notation:** Let, **SUBST(θ , α)=** ∀v α /Subst({v/g}, α)
- means replace All occurrences of "v" with "g" in expression "α" be the result θ, where **v** is the variable and **g** is the ground term.
- **For example**: Every man is mortal. It is represented as **∀ x: man(x) → mortal(x).** In UI, we can infer different sentences as: **man(John) → mortal(John)** also **man(Aakash) → mortal(Aakash), etc.**
- **Universal instantiation** is also called as **universal elimination** or **UI** is a valid inference rule. It can be applied multiple times to add new sentences. The new KB is logically equivalent to the previous KB. As per UI, **we can infer any sentence obtained by substituting a ground term for the variable**. The UI rule state that we can infer any sentence P(c) by substituting a ground term c (a constant within domain x) from **∀ x P(x) for any object in the universe of discourse**.

EXAMPLES

- **Example:1.**
- IF "Every person like ice-cream"=> ∀x P(x) so we can infer that
  "John likes ice-cream" => P(c)
- **Example: 2.**
- Let's take a famous example,
- "All kings who are greedy are Evil." So let our knowledge base contains this detail as in the form of FOL:
- **∀x king(x) ∧ greedy (x) → Evil (x),**
- So from this information, we can infer any of the following statements using Universal Instantiation:
- **King(John) ∧ Greedy (John) → Evil (John),**
- **King(Richard) ∧ Greedy (Richard) → Evil (Richard),**
- **King(Father(John)) ∧ Greedy (Father(John)) → Evil (Father(John)),**

3. EXISTENTIAL INSTANTIATION(EI):

- In EI, the **variable is substituted** by a **single new constant symbol**:
  ∃ x P(x)

  P(c)
- This rule states that one can infer P(c) from the formula given in the form of ∃x P(x) for a new constant symbol c.
- **Notation:** Let, the variable be **v** which is replaced by a constant symbol **k** for any sentence α. ∃v α /Subst({v/k}, α)
- The value of **k** is unique as it does not appear for any other sentence in the knowledge base. Such type of constant symbols are known as **Skolem constant.** As a result, EI is a special case of **Skolemization process.**
- **Note: UI can be applied several times to produce many sentences**, whereas **EI can be applied once, and then the existentially quantified sentences can be discarded.**
  - **For example: ∃x:steal(x, Money).** We can infer from this: **steal(Thief, Money)**
- Existential instantiation is also called as **Existential Elimination**, which is a valid inference rule in first-order logic.

EXAMPLE

- From the given sentence: **∃x Crown(x) ∧ OnHead(x, John),**
- So we can infer: **Crown(K) ∧ OnHead(K, John),** as long as K does not appear in the knowledge base.
- The above used K is a constant symbol, which is called **Skolem constant**.
- The Existential instantiation is a special case of **Skolemization process**.

4. EXISTENTIAL INTRODUCTION

- An existential introduction is also known as an existential generalization, which is a valid inference rule in first-order logic. This rule states that if there is some element c in the universe of discourse which has a property P, then we can infer that there exists something in the universe which has the property P.
- It can be represented as:

P(c)

∃ x P(x)

- **Example: Let's say that,**
  "Priyanka got good marks in English."
  "Therefore, someone got good marks in English."

EXAMPLE

- For example, the following argument can be proven correct using the Universal Instantiation:**"No humans can fly. John Doe is human. Therefore John Doe can not fly."**
  First let us express this using the following notation:
  F(x) means **"x can fly."**
  H(x) means **"x is human."**
  d is a symbol representing **John Doe.**
  Then the argument is: $\forall x\,[H(x) \rightarrow \neg F(x)] \wedge H(d)\,] \rightarrow \neg F(d)$

1. $\forall x\,[H(x) \rightarrow \neg F(x)]$	**Hypothesis**
2. H(d)	**Hypothesis**
3. $H(d) \rightarrow \neg F(d)$	**Universal instantiation on 1.**
4. $\neg F(d)$	**Modus ponens on 2 and 3.**

REDUCTION TO PROPOSITIONAL INFERENCE

- Suppose the KB contains just the following:

  $\forall x\ \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$

  King(John)

  Greedy(John)

  Brother(Richard,John)

- Instantiating the universal sentence in all possible ways, we have:

  $\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$

  $\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$

  King(John)

  Greedy(John)

  Brother(Richard,John)

- **Discading quantifiers and applying inference rules to make KB propositional is known as propositionalization**.
- Proposition symbols are : King(John), Greedy(John), Evil(John), King(Richard), etc.
- What conclusion can you get?

PROBLEMS WITH PROPOSITIONALIZATION

- Propositionalization seems to generate lots of irrelevant sentences
- E.g., from:

  $\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)

  King(John)

  $\forall y$ Greedy(y)

  Brother(Richard,John)

- It seems obvious that Evil(John) is true, but propositionalization produces lots of facts such as Greedy(Richard) that are irrelevant
- When the KB includes a function symbol, the set of possible ground term substitutions is infinite like Father(Father(…(John)…))
- Theorem:
  - If a sentence is entailed by the original, first-order KB, then there is a proof involving just a finite subset of the propositionalized KB
  - First instantiation with constant symbols
  - Then terms with depth 1 (Father(John)) Then terms with depth 2 … Until the sentence is entailed

PROBLEMS WITH PROPOSITIONALIZATION

- Every FOL KB can be propositionalized so as to preserve entailment
- (A ground sentence is entailed by new KB iff entailed by original KB)
- Idea: propositionalize KB and query, apply resolution, return result
- Here the issue is **set of possible ground-term substitution is infinite.** This causes halting problem in Turing Machine
- ***Entailment of FOL is semidecidable-ie Algorithm says yes to every entailed sentences, but no algorithm exists that says no to every non-entailed sentence***
- **Hence propositionalization is inefficient.**

UNIFICATION AND LIFTING

FIRST ORDER INFERENCE RULE:

GENERALIZED MODUS PONEN

- For the inference process in FOL, **we have a single inference rule** which is called **Generalized Modus Ponens.** Generalized Modus Ponens can be summarized as, " P implies Q and P is asserted to be true, therefore Q must be True."
- According to Modus Ponens, for atomic sentences **pi, pi', q**. Where there is a substitution θ such that SUBST **(θ, pi',) = SUBST(θ, pi)**, it can be represented as:

$$\frac{p1', p2', \ldots, pn', (p1 \wedge p2 \wedge \ldots \wedge pn \Rightarrow q)}{SUBST(\theta, q)}$$

- Here the **conclusion** is the result of **applying substitution** to **consequent q.**
- GMP is **lifted version of Modus ponens-**it raises Modus ponens from **propositional logic to FOL.**

GENERALIZED MODUS PONEN EXAMPLE

- If there is some substitution θ that makes the premise of the implication identical to sentences already in the KB, then we assert the conclusion of the implication, after applying θ. Given query **Evil(x)**?

  ∀x King(x) ∧ Greedy(x) ⇒ Evil(x)

  King(John)

  Greedy(John) What's θ here? **θ=(x/John}**

- Instead of Greedy(John) ,we know every one is greedy

  ∀x King(x) ∧ Greedy(x) ⇒ Evil(x)

  King(John)

  ∀y Greedy(y) What's θ here? **θ=(x/John, y/john}**

- **Example: We will use this rule for Kings are evil, so we will find some x such that x is king, and x is greedy so we can infer that x is evil.**
- Here let say, 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)
- SUBST(θ,q) is **Evil(John)**.

UNIFICATION AND LIFTING

- Unification is a process of making **two different logical atomic expressions identical** by **finding a substitution**. Unification depends on the substitution process and is the **key component of First-order inference algorithms**.
- Unification is the process used by the lifted inference rules to find substituents that could give identical but different logical expressions. It means the meaning of the sentence should not be changed, but it should be expressed in multiple ways. The **UNIFY** algorithm in unification takes two sentences as input and then returns a unifier if one exists:

  UNIFY(p,q)= θ where SUBST(θ , p) = SUBST(θ, q)

- Unification is a lifted version of Modus Ponen as it uplifts the Modus Ponens from ground propositions to FOPL. It takes two literals as input and makes them identical using substitution.
- **Unification** is needed for **resolution** to work.

UNIFICATION

- **Unifier:** a substitution that makes two clauses resolvable
- **E.g.** Let's say there are two different expressions, **P(x, y), and P(a, f(z))**.
- In this example, we need t**o make both above statements identical to each other.** For this, we will **perform the substitution.**

  P(x, y)..... (i)
  P(a, f(z))..... (ii)

- Substitute **x** with **a**, and **y** with **f(z)** in the first expression, and it will be represented as **a/x** or **x/a** and **f(z)/y** or **y/f(z).** There are different interpretation for this notation. In **Russel an Norwig** the representation is **x/a, y/f(z)**
- With both the substitutions, the first expression will be identical to the second expression and the substitution set will be: **[x/a, y/f(z)]**

UNIFICATION EXAMPLE

- **Given KB:**
  - **Knows(John, Jane)**
  - **Knows(y, Bill)**
  - **Knows(y, Mother(y))**
  - **Knows(x, Elizabeth)**

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

- **a query Knows(John, x)** means **Every one knows John** The question is- **Whom does John knows? Ie like Does John know Jane?Does John know Bill?**
- **Answer to this query can be found by finding all sentences in KB that unifies Knows(John,x).** The UNIFY algorithm will search all the related sentences in the knowledge base, which could unify with **Knows(John,x)**.
- UNIFY (Knows(John, x), Knows(John, Jane))≡{x/Jane}
- UNIFY (Knows{John,x}, Knows{y, Bill})≡{x/Bill, y/John}
- UNIFY (Knows{John, x}, Knows{y, Mother(y)})≡{ y/John, x/Mother(John)}

UNIFICATION EXAMPLE

- **UNIFY (Knows{John,x}, Knows{x, Elizabeth})≡fails.** The last one failed because we have used the **same variable for two persons at the same time** ie **x cannot take both values John and Elizabeth.**
- It can be avoided by standardizing apart ie renaming
- UNIFY (Knows{John,x}, Knows{x, Elizabeth})≡ UNIFY (Knows{John,x}, Knows{x17, Elizabeth})≡ { x/Elizabeth, x17/ John }
- Consider the following two unifications
  - **UNIFY (Knows (John, x), Knows (y, z)) ={ y/John, x/ z }**
  - **UNIFY (Knows (John, x), Knows (y, z)) ={ y/John, x/John , z/John }**
- We say the first unifier is more general than the second . **It places fewer restrictions on the values of variables.** For every unifiable pairs of expressions, there is a single **most generalized unifier (MGU)** – E.g., the former unifier, { y/John, x/ z}, shown above

UNIFICATION RULES

- **Following are some basic conditions for unification:**
  1. **Predicate symbol must be same, atoms or expression with different predicate symbol can never be unified.**
  2. **Number of Arguments in both expressions must be identical.**
  3. **Unification will fail if there are two similar variables present in the same expression.**
- Rules for substitutions:
  1. **Can replace a variable by a constant.**
  2. **Can replace a variable by a variable.**
  3. **Can replace a variable by a function expression, as long as the function expression does not contain the variable.**

MOST GENERAL UNIFIER

- In cases **where there is more than one substitution choose the one that makes the least commitment (most general) about the bindings. The substitution variables are called Most General Unifier or MGU.**

= {y / John, x / z} not {y / John, x / John, z / John} not {y / John, x / z, z / Freda} …

- For each pair of atomic sentences, give the most general unifier if it exists.

1. Find the MGU of {p(f(a), g(Y))} and p(X, X)}

- Sol: S_0 =>{} Here, Ψ_1 = **p(f(a), g(Y))**, and Ψ_2 = **p(X, X)**
  SUBST θ= {X/ f(a) }
  S1 => Ψ1 = p(f(a), g(Y)), and **Ψ2 = p(f(a), f(a))**
  SUBST θ= {g(y)/f(a) }, **Unification failed.**

- Unification is not possible for these expressions.

MGU EXAMPLES

2. Find the MGU of {p(b, X, f(g(Z))) and p(Z, f(Y), f(Y))}

- Here, Ψ_1 = p(b, X, f(g(Z))) , and Ψ2 = p(Z, f(Y), f(Y))
  S0 => { p(b, X, f(g(Z))); p(Z, f(Y), f(Y))}
  SUBST θ={Z/b}
- S1 => { p(b, X, f(g(b))); p(b, f(Y), f(Y))}
  SUBST θ={X/f(Y)}
- S2 => { p(b, f(Y), f(g(b))); p(b, f(Y), f(Y))}
  SUBST θ= {Y/g(b)}
- S2 => { p(b, f(g(b)), f(g(b)); p(b, f(g(b)), f(g(b))} **Unified Successfully.**
  And Unifier = { Z/b, X/f(Y) , Y/g(b) }.

3. UNIFY(knows(Richard, x), knows(Richard, John))

- Here, Ψ1 = knows(Richard, x), and Ψ_2 = knows(Richard, John)
  S0 => { knows(Richard, x); knows(Richard, John)}
  SUBST θ= {x/John}
  S1 => { knows(Richard, John); knows(Richard, John)}, **Successfully Unified.**
  Unifier: {x/John}.

MGU EXAMPLES

4. Find the MGU of {p (X, X), and p (Z, f(Z))}

- Here, Ψ_1 = {p (X, X), and Ψ2 = p (Z, f(Z))
  S0 => {p (X, X), p (Z, f(Z))}
  SUBST θ= {X/Z}
  S1 => {p (Z, Z), p (Z, f(Z))}
  SUBST θ= {f(Z) /Z}, Unification Failed.
- **Hence, unification is not possible for these expressions.**

5. Find the MGU of UNIFY(prime (11), prime(y))

- Here, Ψ1 = {prime(11) , and Ψ2 = prime(y)}
  S0 => {prime(11) , prime(y)}
  SUBST θ= {y/11}
- S1 => {prime(11) , prime(11)} , **Successfully unified.**
  Unifier: {y/11}.

MGU EXAMPLES

6. Find the MGU of Q(a, g(x, a), f(y)), Q(a, g(f(b), a), x)}

- Here, Ψ_1 = Q(a, g(x, a), f(y)), and Ψ2 = Q(a, g(f(b), a), x)
  S0 => {Q(a, g(x, a), f(y)); Q(a, g(f(b), a), x)}
  SUBST θ= {x/f(b)}
  S1 => {Q(a, g(f(b), a), f(y)); Q(a, g(f(b), a), f(b))}
- SUBST θ= {y/b}
  S1 => {Q(a, g(f(b), a), f(b)); Q(a, g(f(b), a), f(b))}, Successfully Unified.
- **Unifier: [a/a, x/f(b), y/b].**
- **CHALLENGES**

1. P(A, B, B), P(x, y, z)
2. Q(y, G(A, B)), Q(G(x, x), y)
3. Older(Father(y), y), Older(Father(x), John)
4. Knows(Father(y), y), Knows(x, x)

RESOLUTION IN FOL

- Resolution means proving a conclusion S from given premises expressed in FOL.
- Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions. Unification is a key concept in proofs by resolutions.

RESOLUTION STEPS

1. Conversion of facts into first-order logic.
2. Convert all sentences to CNF(clausal form)
3. Negate conclusion S & convert result to CNF
4. Add negated conclusion S to the premise clauses
5. Draw resolution graph (unification), to better understand all the above steps, we will take an example in which we will apply resolution.Repeat until contradiction or no progress is made:
    a. Select 2 clauses (call them parent clauses)
    b. Resolve them together, performing all required unifications
    c. If resolvent is the empty clause, a contradiction has been found (i.e., S follows from the premises)
    d. If not, add resolvent to the premises

The Resolution Rule

Resolution relies on the following rule:

$$\frac{\neg\alpha \Rightarrow \beta,\ \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$ Resolution rule

equivalently,

$$\frac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma}$$ Resolution rule

Applying the resolution rule:

1. Find two sentences that contain the same literal, once in its positive form & once in its negative form:

CNF sentences → $\text{summer} \vee \text{winter},\ \neg\text{winter} \vee \text{cold}$

2. Use the resolution rule to eliminate the literal from both sentences

→ $\text{summer} \vee \text{cold}$

RESOLUTION INFERENCE RULE IN FOL

- The resolution inference rule:The resolution rule for first-order logic is simply a lifted version of the propositional rule. **Resolution can resolve two clauses if they contain complementary literals,** which are assumed to be standardized apart so that they share no variables.
- Where $\mathbf{l}_i$ and $\mathbf{m}_j$ are complementary literals.

$$\frac{l_1 \vee \dots \vee l_k, \quad m_1 \vee \dots \vee m_n}{\mathrm{SUBST}(\theta, l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n)}$$

- This rule is also called the **binary resolution rule** because it only resolves exactly two literals.

Resolution: brief summary

Full first-order version:

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

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$.

For example,

$$\frac{\neg Rich(x) \vee Unhappy(x) \quad Rich(Ken)}{Unhappy(Ken)}$$

with $\theta = \{x/Ken\}$

Apply resolution steps to $CNF(KB \wedge \neg\alpha)$; complete for FOL

EXAMPLE FOR RESOLUTION INFERENCE RULE

- Example:We can resolve two clauses which are given below:
- **[Animal (g(x) V Loves (f(x), x)] and [¬ Loves(a, b) V ¬Kills(a, b)]**
- Where two complimentary literals are: **Loves (f(x), x) and ¬ Loves (a, b)**
- These literals can be unified with unifier **θ= [a/f(x), and b/x]** , and it will generate a resolvent clause:
- **[Animal (g(x) V ¬ Kills(f(x), x)].**

EXAMPLE 1:

1. If something is intelligent, it has common sense
2. Deep Blue does not have common sense

- Prove that Deep Blue is not intelligent

1. $\forall x.I(x) \Rightarrow H(x)$
2. $\neg H(D)$

Conclusion: $\neg I(D)$

Denial: C3: $I(D)$

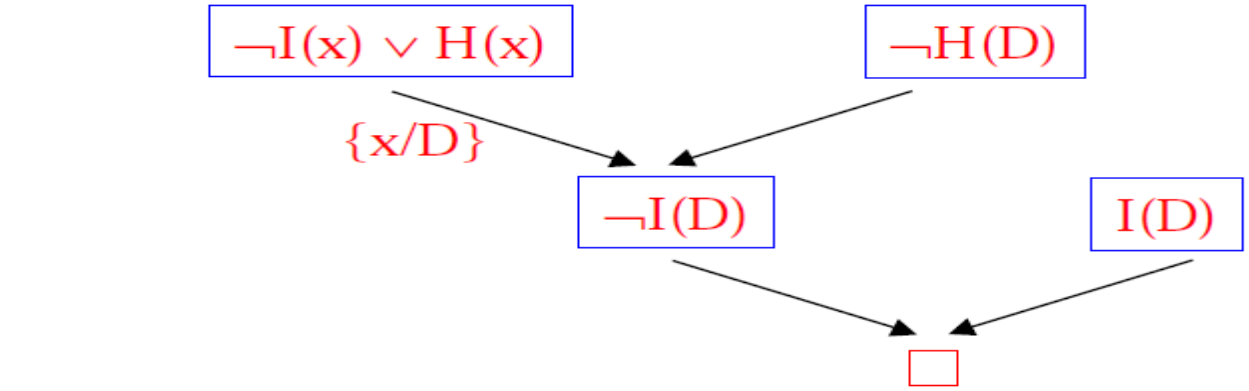
→

C1: $\neg I(x) \vee H(x)$

C2: $\neg H(D)$

CNF

A resolution proof of $\neg I(D)$:



Proof also written as:

C4: $\neg I(D)$ r[C1b, C2]

C5: □ r[C3, C4]

2nd literal, 1st clause