- Handling Fluents
- Representing the Fluent axioms in the KB
- Hybrid Agent for Wumpus World
- Using Propositional Inferencing to make Plan

Logical State Estimation

- Earlier we used combination of KB inferencing and path-search algorithm to find a plan

- But we can do everything just using a SAT Solver.
    - Construct a big Propositional sentence that contains the following:
        - All initial axioms
        - All Fluent update rules for time step 1...t
        - $HaveGold^t \wedge ClimbedOut^t$

    - Give this full formula to a SAT Solver
        - If it gives a satisfiable assignment then extract a model out of it.
        - Try max t from 1,2,3... upto some threshold

- Might give spurious solutions if axiom set is not exhaustive
    - Good debugging tool

Logical State Estimation : Advantages

- No need to think about when to use A* when to use inference etc.
    - Generic solution for all situations
- Practical SAT solvers are powerful enough to handle most problems arising in the real world
- Does not work in Partially Observable settings
- Exercise : Find a solution to the Wolf-Cabbage-Sheep problem using SAT solvers

Knowledge Explosion

- As the number of steps increase, the knowledge base increases and hence time to make new inferences also increases.

- Can we ensure that inference takes time independent of step t?
  - One way is to save all previous inferences, so that we do not have to recompute them
  - Example : $WumpusAlive^1 \wedge L^1_{2,1} \wedge B_{2,1} \wedge (P_{3,1} \vee P_{2,3})$

- Keeping all previous inferences is costly
  - Typically some conservative under approximation is kept

Representation of the Wumpus World

- Propositional Logic is just ONE way to represent the wumpus world

- Any other natural representation?
  - 4X4 matrix
    - Not declarative, we need to say hardcode how new information is derived
    - Not clear how to say “[1,2] has a pit OR [2,1] has a pit”

Beyond Propositional Logic

- Advantages of Propositional Logic:
  - It is declarative : Inference is domain independent
  - Can handle partial information well : Disjunction, If else
  - It is compositional : Meaning of a formula can be derived from looking at the structure of the formula.

- Drawbacks of Propositional Logic:
  - $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
  - $(Forward^{t}) \Rightarrow (haveArrow^{t} \Leftrightarrow haveArrow^{t+1})$
    - We need to say state such properties for each i, j and for every time step t
  - We cannot say For every time step t $(Forward^{t}) \Rightarrow (haveArrow^{t} \Leftrightarrow haveArrow^{t+1})$
  - This would give a succinct representation of the Knowledge Base (This is more natural)

- First Order Logic extends Propositional Logic with this natural way of expression of properties
  - First Order Logic is more expressive than Propositional Logic

First Order Logic

- Derives its syntax from Natural Language
  - We have Nouns (Objects) : Wumpus, Pit, Square
  - We have Verbs, Adverbs, Adjectives (Relations) : is breezy, is adjacent to, is in

- Examples:
  - Objects : Person, University, Animal, Vertex, Numbers
  - Relations : Professor , Part of, brother of, is Prime ..
    - Functions (specialized relations) : Father, +, ...
    - Constants (specialized functions) : President of India, Director of IITM ...

- Functions and Constants have a special status compared to Relations

First Order Logic

- Pi is an irrational number
  - Objects : 1,2, 3.141, 4, 2.718 ...
  - Relations : Irrational - unary relation
  - Constant : Pi (refers to an object)

- India got Independence in 1947
  - Objects : India, 1947, Independence
  - Relations : gotIn - ternary relation, IsCountry

- One plus Two equals Three
  - Objects : One, Two, Three [One plus Two also refers to an object]
  - Relations : equals
  - Functions : Plus

Ontological and Epistemological Commitments

- Various logics typically differ in their Ontological and Epistemological commitments.
  - Ontological Commitment: What are the basic building blocks of the world?
  - Epistemological Commitment : What does the agent believe about the building blocks?

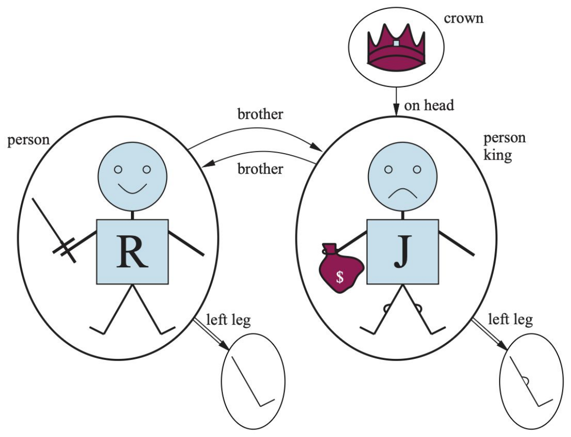
Logic	**Ontological Commitment**	**Epistemological Commitment**
Propositional Logic	Facts / Propositions	True / False / Unknown
First Order Logic	Facts about Objects / Relations	True / False / Unknown
Probabilistic Logic	Facts / Propositions	Degree of belief
Fuzzy Logic	Propositions with degrees of Truth	An interval of the belief

Models for First Order Logic

- Should tell us what are the underlying Objects, Relations, Functions and Constants

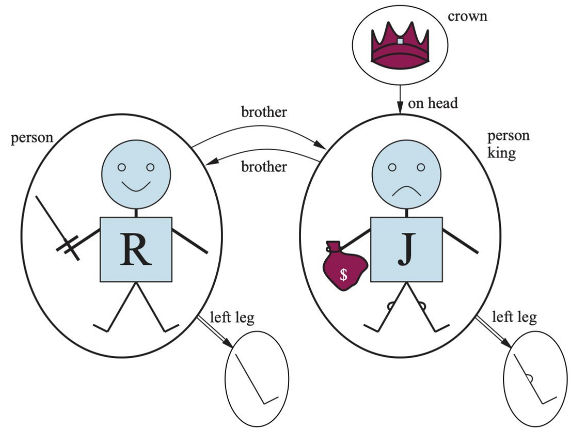
- Objects of a Model : Also called as Domain (is always non-empty)

- Example : Scenario with 5 objects:
  - P1 : Richard the Lionheart, King of England from 1189 to 1199
  - P2 : His younger brother, the evil King John, who ruled from 1199 to 1215
  - L1 : The left leg of Richard
  - L2 : The left leg of John
  - C: Crown.



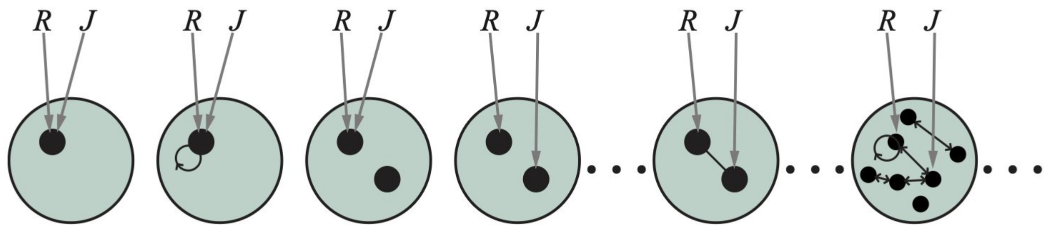
Models for First Order Logic

- Example : Scenario with 5 objects:
  - P1 : Richard the Lionheart, King of England from 1189 to 1199
  - P2 : His younger brother, the evil King John, who ruled from 1199 to 1215
  - L1 : The left leg of Richard
  - L2 : The left leg of John
  - C: Crown.



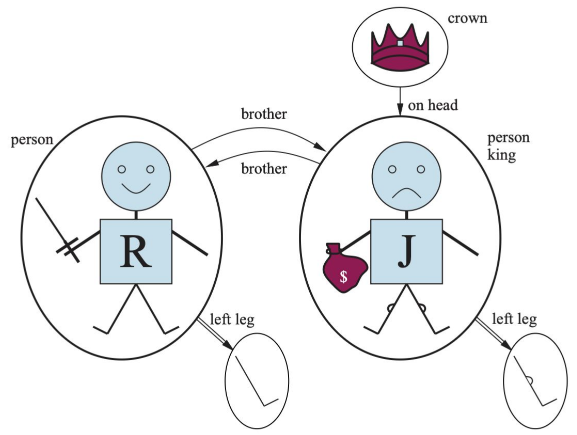
- Relations: Tuples of related objects
  - Brother : { (P1, P2), (P2, P1) }
  - OnHead : { (C, P2) }
  - Person : { (P1) , (P2) }
  - King : { (P2) }
  - Crown : { (C) }

- Constants
  - Richard : P1
  - John : P2



Models for First Order Logic

- Example : Scenario with 5 objects:
  - P1 : Richard the Lionheart, King of England from 1189 to 1199
  - P2 : His younger brother, the evil King John, who ruled from 1199 to 1215
  - L1 : The left leg of Richard
  - L2 : The left leg of John
  - C: Crown.



- Relations: Tuples of related objects
  - Brother : { (P1, P2), (P2, P1) }
  - OnHead : { (C, P2) }
  - Person : { (P1) , (P2) }
  - King : { (P2) }
  - Crown : { (C) }

- Constants
  - Richard : P1
  - John : P2

- Functions: LeftLeg
  - LeftLeg(Richard) = L1
  - LeftLeg(John) = L2

- Functions are total
  - Technically every object should have a LeftLeg
  - Solution : Map the remaining things to some "invisible" object
  - Safe as long as there are no assertions about such objects

Syntax of First Order Logic

- At the ground level we have Objects, Relations and Functions
- Correspondingly in the syntax we have Constant symbols, Predicate symbols and Function Symbols

- Every predicate and function symbol has a appropriate arity

- Model gives the interpretation for the Constants, Predicates and function symbols

Syntax of First Order Logic : Terms

- Terms refer to the domain elements
  - Constant Symbols are Terms : PresidentOfIndia, DirectorOfIITM
  - Can be more complex : Father(PresidentOfIndia), Secretary(DirectorOfIITM)
  - Terms can only point to a single object / Domain :
    - Brother(PresidentOfIndia) will not make sense if the President has more than 1 brothers
    - Father is a function, Brother is a binary relation
  - Terms can only use Functions and Constants (Terms cannot have Relations)
- Formally:
  - Every constant symbol c is a term
  - If $t_1\ t_2 \ldots t_n$ are terms and f is a function with arity n then $f(t_1\ t_2 \ldots t_n)$ is a term

Syntax of First Order Logic : Sentences

- Sentences state Facts
  - Brother(Richard, John)
  - Married(Father(Richard), Mother(John))
  - If R is a predicate of arity n and $t_1 t_2 \ldots t_n$ are terms then R ($t_1 t_2 \ldots t_n$) is an atomic sentence

- An atomic Sentence is True in the given model if the corresponding relation holds among the objects referred in the arguments

Syntax of First Order Logic : Sentences

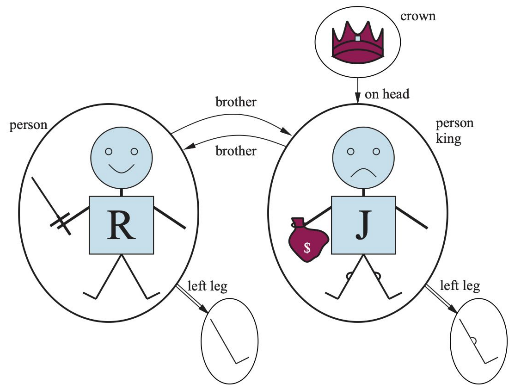
- Complex Sentences are built over the atomic sentences using connectives:

    - $\neg$ Brother(LeftLeg(Richard), John)
    - King(Richard) V King(John)
    - Brother(Richard, John) $\Leftrightarrow$ Brother(John, Richard)

- The connectives are the same that we had in Propositional Logic

Syntax of First Order Logic : Quantifiers

- Every King is a Person
  - $\forall x$ King(x) ⇒

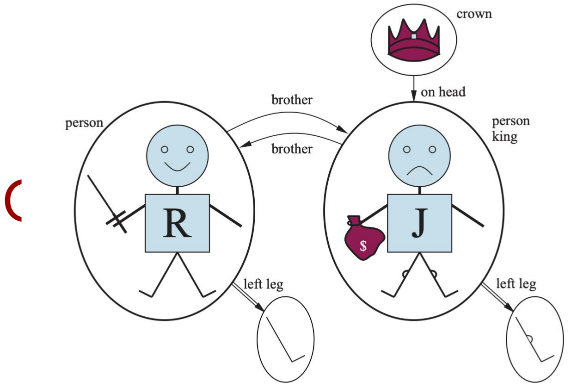


- Here x is a variable. What should x refer to?
  - A variable is also a Term
  - A term without variables is called a Ground Term

- Extended Interpretation : Interpretation of Ground terms + Interpretation of Variable(s)
  - $\forall x$ P is true in a model if P is true for all possible Extended Interpretations of x

Syntax of First Order Logic : Quantifiers

- There is a crown on John
  - ∃x Crown(x) □ ( )



- ∃x P is true in a model if P is true with at least one Extended Interpretation of x

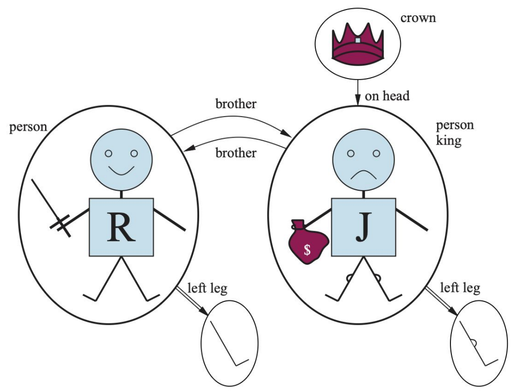
Syntax of First Order Logic : Nested Quantifiers

- All brothers are siblings
  - ∀x ∀y Brother(x,y) ⇒ Sibling(x,y)
- Every King has a crown on his head
  - ∀x King(x) ⇒ (∃y Crown(y) □ OnHead(y,x))

$$\begin{array}{lcl}
\neg\exists x\ P & \equiv & \forall x\ \neg P \\
\neg\forall x\ P & \equiv & \exists x\ \neg P \\
\forall x\ P & \equiv & \neg\exists x\ \neg P \\
\exists x\ P & \equiv & \neg\forall x\ \neg P
\end{array}$$

Syntax of First Order Logic : Equality

- Father of John ry
  Father(John) = ry



- Richard has at least two brothers
  ∃x ∃y Brother(x,Richard) □ Brother(y,Richard) □ ¬ (x = y)

- Equality gives more expressive power to the logic

Syntax of First Order Logic

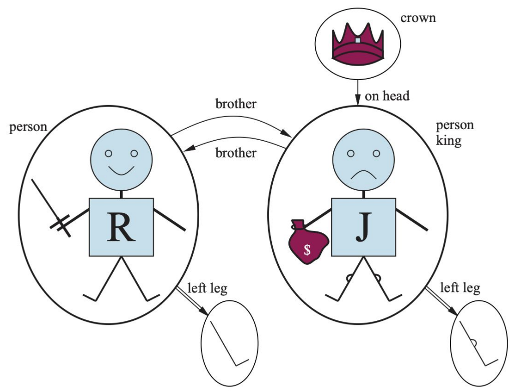
$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{Predicate} \mid \mathit{Predicate}(\mathit{Term}, \ldots) \mid \mathit{Term} = \mathit{Term} \\
\mathit{ComplexSentence} & \rightarrow & (\ \mathit{Sentence}\ ) \\
 & \mid & \neg\, \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \land \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \lor \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \ \Rightarrow\ \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \ \Leftrightarrow\ \mathit{Sentence} \\
 & \mid & \mathit{Quantifier}\ \mathit{Variable}, \ldots\ \mathit{Sentence} \\
 & & \\
\mathit{Term} & \rightarrow & \mathit{Function}(\mathit{Term}, \ldots) \\
 & \mid & \mathit{Constant} \\
 & \mid & \mathit{Variable} \\
 & & \\
\mathit{Quantifier} & \rightarrow & \forall \mid \exists \\
\mathit{Constant} & \rightarrow & A \mid X_1 \mid \mathit{John} \mid \cdots \\
\mathit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\mathit{Predicate} & \rightarrow & \mathit{True} \mid \mathit{False} \mid \mathit{After} \mid \mathit{Loves} \mid \mathit{Raining} \mid \cdots \\
\mathit{Function} & \rightarrow & \mathit{Mother} \mid \mathit{LeftLeg} \mid \cdots \\
\textsc{Operator Precedence} & : & \neg, =, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

Database semantics

- Richard has two brothers : John and Joffrey Brother(John, Richard) □ Brother(Geoffrey, Richard)
- John and Joffrey are different persons Brother(John, Richard) □ Brother(Geoffrey, Richard) □ ¬ (John = Geoffrey)
- There are no other brothers Brother(John, Richard) □ Brother(Joffrey, Richard) □ ¬ (John = Geoffrey) □ ∀x Brother(x, Richard) ⇒ (x = John) V (x = Geoffrey)
- Stating it in this detail every time is tedious. We might miss something.
  - Unique-names assumption : Every constant Refers to a distinct domain element
  - Closed world assumption : Every sentence not known to be true is false
  - Domain Closure: All domain elements are named by some constant
- With these assumptions, the first property already achieves what we intend to express

Using First Order Logic to populate KB

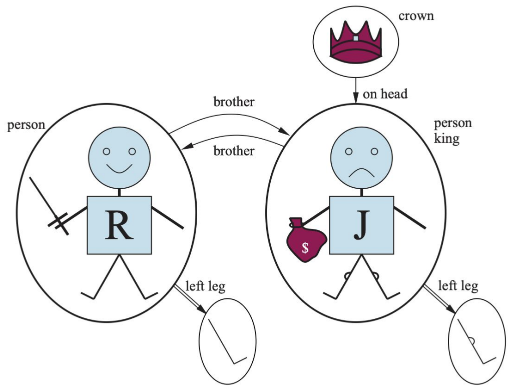
- TELL (KB, King(John))
- TELL (KB, Person(Richard))
- TELL(KB, ∀x King(x) ⇒ Person(x))
- TELL(KB, Crown(C))
- TELL(KB, OnHeac)



- ASK(KB, King(John))
- ASK(KB, King(Richard))
- ASK(KB, Person(John))
- ASK(KB, ∃x Person(x))

Ask Variables

- TELL (KB, King(John))
- TELL (KB, Person(Richard))
- TELL(KB, ∀x King(x) ⇒ Person(x))
- TELL(KB, Crown(C))
- TELL(KB, OnHead(C,John))



- ASKVARS (KB, Person (x))
  - Give all possible assignments to x, that makes it True : { x/John } { x/Richard }
- These are called substitutions
- More Examples:
  - ASKVARS (KB, OnHead(x,y))
    - Solution : {x/C, y/John}
  - ASKVARS (KB, ∃x Crown(C) Λ OnHead(x,y) Λ z = leftLeg(y))
    - Solution : {y/John, z/L2}

Using First Order Logic to populate KB : Family

- Suppose the model has:
  - Objects : People in a particular family
  - Relations : Male, Female, Parent, Sibling, Brother, Sister, Child, Daughter, Son, Spouse, Wife, Husband, GrandParent, GrandChild, Cousin, Aunt, Uncle
  - Functions : Father, Mother

- Axioms: Factual information from which useful conclusions can be derived
  - $\forall x\ \forall y\ Mother(x) = y \Leftrightarrow (\ Female(y) \land Parent(y,x)\ )$
  - $\forall x\ \forall y\ Sibling(x,y) \Leftrightarrow \exists p\ Parent(p,x) \land Parent(p, y) \land \neg\ (x = y)$
  - ………

Using First Order Logic to populate KB : Family

- Suppose the model has:
    - Objects : People in a particular family
    - Relations : Male, Female, Parent, Sibling, Brother, Sister, Child, Daughter, Son, Spouse, Wife, Husband, GrandParent, GrandChild, Cousin, Aunt, Uncle
    - Functions : Father, Mother

- We can also define new relation: (Special kind of Axioms)
    - $\forall x\ \forall y\ Nephew(y,x) \Leftrightarrow (\ Male(x) \land (\ Uncle(y,x) \lor Aunt(y,x)\ )$

Using First Order Logic to populate KB : Family

- Suppose the model has:
    - Objects : People in a particular family
    - Relations : Male, Female, Parent, Sibling, Brother, Sister, Child, Daughter, Son, Spouse, Wife, Husband, GrandParent, GrandChild, Cousin, Aunt, Uncle
    - Functions : Father, Mother

- Sentences entailed by Axioms are called Theorems
- Formally, KB only contains Axioms since Theorems do not add more information
- But practical implementations also add Theorems to KB to make the implementation more efficient.

Using First Order Logic to populate KB : Numbers

- Predicates: NatNum — Objects : 0, 1, 2, 3, ...
- Constants : 0 — 0 is interpreted as 0
- Functions : S — S(i) = i+1

- Peano Axioms :
  - NatNum(0)
  - $\forall x \ \text{NatNum}(x) \Rightarrow \text{NatNum}(S(x))$
  - $\forall x \ S(x) \neq 0$
  - $\forall x \ \forall y \ (x \neq y) \Rightarrow (S(x) \neq S(y))$
  - $\forall x \ \text{NatNum}(x) \Rightarrow +(x,0) = x$
  - $\forall x \ \forall y \ \text{NatNum}(x) \land \text{NatNum}(y) \Rightarrow +(S(x),y) = S(+(x,y))$

Using First Order Logic in Wumpus World

- Agent's Input:
  - Percept : Binary predicate [Stench, Breeze, Glitter, Bump, Scream], t

    Example : Percept([Stench, Breeze, None, None, None], 5)

- Perception Axioms: One axiom for every presence/absence of percept
  - Examples:
    - ∀t,b,g,w,c Percept ([Stench, b, g, w, c], t) ⇒ Stench(t)
    - ∀t,s,g,w,c Percept ([s, None, g, w, c], t) ⇒ ¬ Breeze(t)
    - ∀t,s,b,w,c Percept ([s, b, Glitter, w, c], t) ⇒ Glitter(t)
    - ....

Using First Order Logic in Wumpus World

- Agent’s Output:
    - TurnLeft, TurnRight, Forward, Grab, Climb, Shoot
    - Each of this is can be a Term

- ASKVARS(KB, BestAction(a,5)) what value of a satisfies
        - Reflex behavious can be directly expressed:
            - $\forall t \; Glitter(t) \Rightarrow BestAction(Grab, t)$

Using First Order Logic in Wumpus World

- Environment : Squares can be list terms [i,j]
  - ∀x,y,a,b Adjacent([x,y] , [a,b]) ⇔ (x = a ∧ (y = b+1 ∨ y = b - 1)) ∨ (y = b ∧ (x = a+1 ∨ x = a - 1))
- Pit can be a Unary predicate : Pit([x,y])
- Wumpus can be a constant : Referring to the Wumpus Object

Using First Order Logic in Wumpus World

- At(u, v, w) Object u is in square v at time w
  - Every object can be in at most one place in a given time
    - $\forall u \ \forall v \ \forall w \ \forall t \ (At(u, v, t) \land At(u, w, t)) \Rightarrow (v = w)$
  - Wumpus is at the same place all the time
    - $\exists x \ \exists y \ \forall t \ At(Wumpus, [x,y], t)$

Using First Order Logic in Wumpus World

- Neighbour of a breezy square contains a pit
  - $\forall v \quad Breezy(v) \Rightarrow \exists x \; (Adjacent(v,x) \land Pit(x))$
- HaveArrow updation:
  - $\forall t \; HaveArrow(t+1) \Leftrightarrow (HaveArrow(t) \land \neg Action(Shoot, t))$

Knowledge Engineering in First Order Logic

- Identify the Questions
  What questions will KB support? What facts will be available in KB?
- Assemble Relevant Knowledge
  Understand the scope of the KB
- Decide on the vocabulary
  Identify Objects / Relations / Functions / Constants
- Encode general knowledge about the domain and Problem Instance
- Test and Debug

Refer 8.4.2 that illustrates all these steps for a particular domain

Entailment, Validity and Satisfiability

- First Order Logic Formula α is VALID iff
  - for every model M and every extended interpretation σ for free variables of α over the domain of M we have $M, \sigma \models \alpha$

- First Order Logic Formula α is SATISFIABLE iff
  - there exists some model M and some extended interpretation σ for free variables of α over the domain of M such that $M, \sigma \models \alpha$

- All these notions are analogous to what we had in Propositional Logic

- So even in First Order Logic, we have :
  - $KB \models \alpha$ iff $(KB \land \neg\alpha)$ is not satisfiable

Universal Instantiation

- ∀x (King(x) □ Greedy(x)) ⇒ Evil(x)

- What all can we infer from this?
  - (King(Richard) □ Greedy(Richard)) ⇒ Evil(Richard)
  - (King(John) □ Greedy(John)) ⇒ Evil(John)
  - (King(Father(John)) □ Greedy(Father(John)) ⇒ Evil(Father(John))
  - .....

  - In general, for any ground term t, (King(t) □ Greedy(t)) ⇒ Evil(t)

Universal Instantiation

- ∀x (King(x) □ Greedy(x)) ⇒ Evil(x)

- If *θ* is some substitution then SUBST(*θ*, α) be the result of applying *θ* on α
  - Example: *θ* is { x / John }
    α is (King(x) □ Greedy(x)) ⇒ Evil(x))
    Then SUBST(*θ*, α) is (King(John) □ Greedy(John)) ⇒ Evil(John)

- Universal Instantiation Rule:
  ∀x α
  ———————————————————————
  SUBST({x/t }, α)   Where t is a ground term

Existential Instantiation

- Existential Instantiation Rule:
-

  $$\frac{\exists x \quad \alpha}{\mathrm{SUBST}(\{x/k\}, \alpha)}$$

  Where k is a new constant symbol that does not occur anywhere else in the knowledge base

- Here k is called a Skolem Constant.

- Similar to what we do in Proofs
  - Example : Suppose there exists a vertex in the graph / number such that the property does not hold. Let v be such a vertex/number.

First Order Inferencing

- Replace every formula of the form ∃x α by its existential instantiation
- Replace every formula of the form ∀x α by all possible universal instantiations.
- Now the KB contains only boolean combinations atomic sentences where the parameters are ground terms.
  - Replace each such atomic statement with a proposition
  - Example : King(Father(John)) is replaced with FatherofJohnIsKing
- This technique is called Propositionalization.
  - Original KB entails δ iff the Propositionalized KB entails Propositionalized δ
    - Needs proof
- Done? (End of discussion on First Order Inferencing?)
  - Propositionalization makes the KB infinite (in particular the Universal Instantiation step)

First Order Inferencing using Propositional Inferencing

- Herbrand's Theorem: A first order KB entails **δ** iff there exists a finite subset of the Propositionalized KB that entails Propositionalized **δ**

- Algorithm :
  Try out all possible subsets of the Propositionalized KB and check if it entails Propositionalized **δ**
  - First try all possible subsets where terms have depth 0 terms
  - Then try all possible subsets where terms have depth at most 1 terms
  - ....

- If the input is a YES instance, the algorithm always Returns YES.

- If the input is a NO instance then:
  - Algorithm gets stuck in an infinite loop

- Can we have an Algorithm that Returns NO for the negative instances?
  - No :(

Entailment problem for First Order Logic

- Entailment problem for First Order Logic is undecidable
  - The problem is Recursively enumerable but not Recursive
  - Satisfiability problem for First Order Logic is coRE but not Recursive

(If you do not know what Recursively enumerable / Recursive mean, you can safely ignore it)

- For Propositional Logic, we do not know of any fast algorithms for entailment, but heuristic based algorithms work well in practice

- For First Order Logic, we know that there cannot be an algorithm that terminates on all inputs and gives correct answer
  - But we will still try to build some heuristics based algorithms that (hopefully) work well in practice

Proof of Herbrand’s Theorem on Board