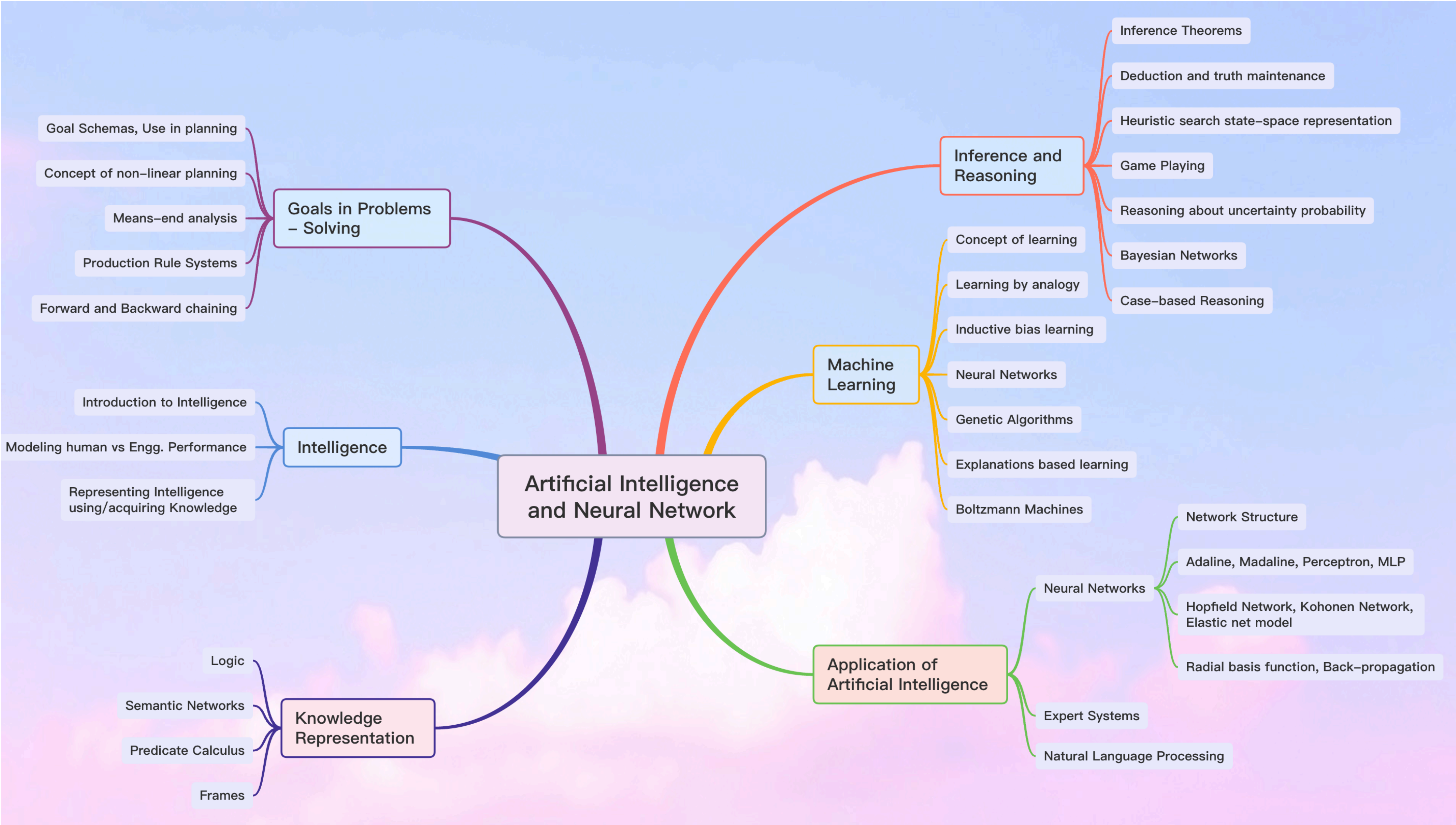
UNIT 3

Knowledge Representation



Artificial Intelligence and Neural Network
Goals in Problems – Solving
Goal Schemas, Use in planning
Concept of non–linear planning
Means–end analysis
Production Rule Systems
Forward and Backward chaining
Intelligence
Introduction to Intelligence
Modeling human vs Engg. Performance
Representing Intelligence using/acquiring Knowledge
Knowledge Representation
Logic
Semantic Networks
Predicate Calculus
Frames
Inference and Reasoning
Inference Theorems
Deduction and truth maintenance
Heuristic search state–space representation
Game Playing
Reasoning about uncertainty probability
Bayesian Networks
Case–based Reasoning
Machine Learning
Concept of learning
Learning by analogy
Inductive bias learning
Neural Networks
Genetic Algorithms
Explanations based learning
Boltzmann Machines
Application of Artificial Intelligence
Neural Networks
Network Structure
Adaline, Madaline, Perceptron, MLP
Hopfield Network, Kohonen Network, Elastic net model
Radial basis function, Back–propagation
Expert Systems
Natural Language Processing

Contents

Outline

- **Inference**
  - **Rules of Inference**
  - **Unification**
  - **Resolution Refutation System**
  - **Answer Extraction from RRS**
  - **Rule based Deduction System**
- Statistical Reasoning
  - Probability and Bayes Theorem
  - Causal Networks
  - Reasoning in Belief Network

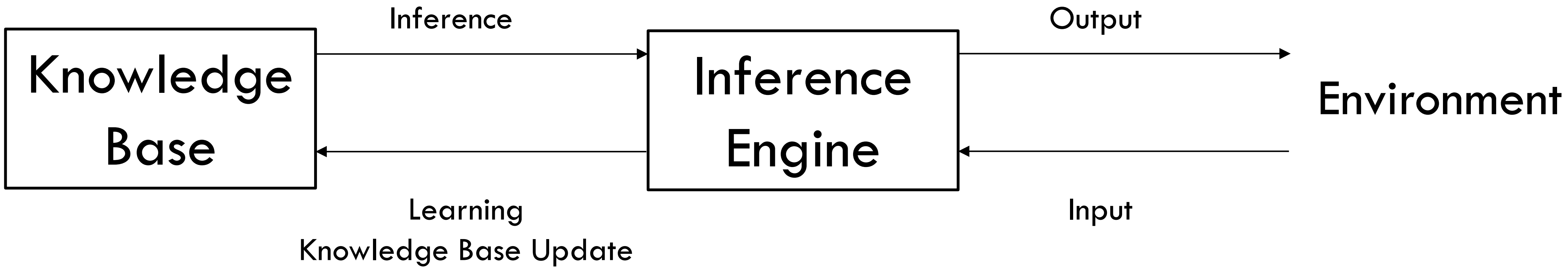
Knowledge Representation

- An area of AI whose fundamental goal is to represent knowledge in a manner that facilitates inferring or drawing conclusion from knowledge
- Analyses how to think formally, how to use symbol to represent a domain of discourse along with the function that allow inference about the objects

Knowledge Representation

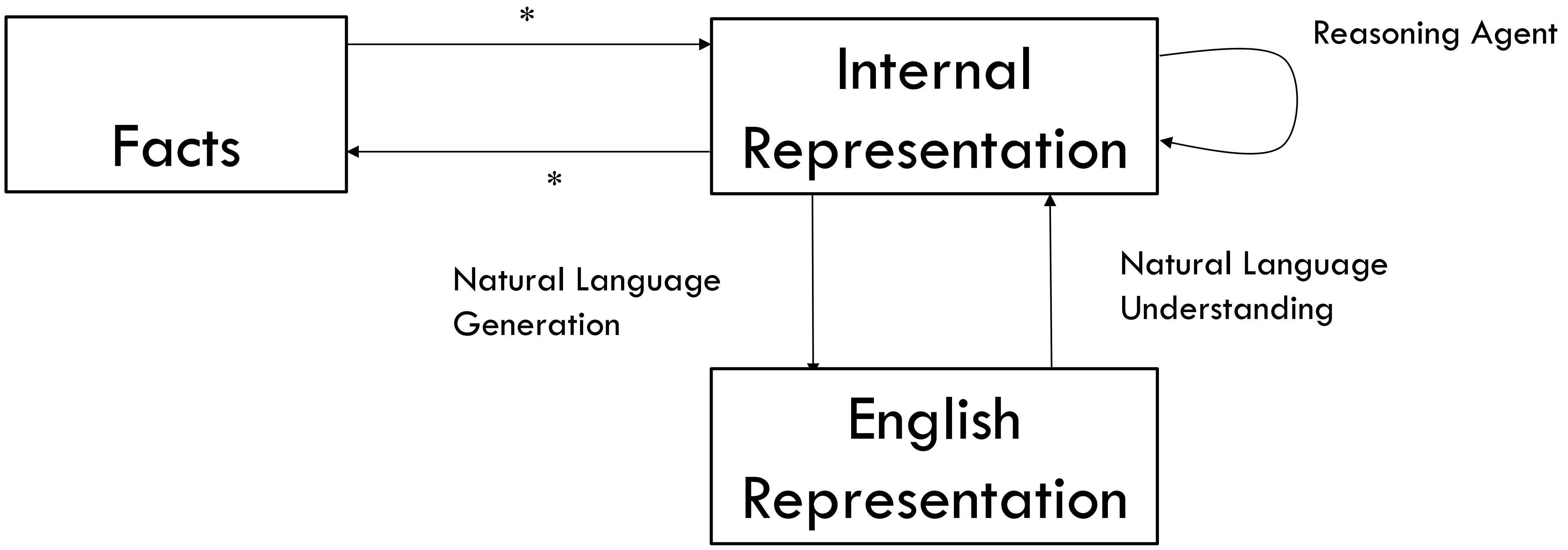
- Helps to address problems like:
  - How do we represent facts about the world?
  - How do we reason about them?
  - What representations are appropriate for dealing with the real world?
- Its objective is to express knowledge in a computer tractable form so that agent can perform well.

Knowledge Representation



Knowledge Representation

Figure: Mapping Facts and Representation



Knowledge Representation: Approaches

- A good system for knowledge representation should have
  - Representable Adequacy: Ability to represent all kind of knowledge that are needed in the domain
  - Inferential Adequacy: Ability to manipulate the representational structure in such a way as to derive new structures corresponding to new knowledge inferred from old
  - Inferential Efficiency: Ability to incorporate into the knowledge structure additional information that can be used to focus the attention of the inference mechanism in the most promising direction
  - Acquisitional Efficiency: Ability to acquire new information easily

Knowledge Representation: Types

- Simple Relational Knowledge
  - The simplest way to represent declarative facts is as a set of relations of the same sort used in database system
- Inheritable Knowledge
  - Structure must be designed to correspond to the inference mechanism that are desired
- Inferential Knowledge
  - Represents knowledge as formal logic
  - Based on reasoning from facts or from other inferential knowledge
  - Useless unless there is also an inference procedure that can exploit it
- Procedural (Imperative) Knowledge
  - Knowledge exercised in the performance of some task
  - Processed by an intelligent agent

Knowledge Representation: Issues

- Are any attributes of objects so basic that they have been occurred in almost every problem domain?
- Are there any important relationships that exist among attributes of objects
- At what level should knowledge be represented?
- How should sets of objects be represented?
- How can relevant parts be accessed when they are needed?

Knowledge Based Agent

- Knowledge Base: a set of sentences
- An agent having a knowledge base
- Each sentence in a knowledge base is expressed in a language called a knowledge representation language
- There must be a way to add new sentences to the knowledge base
- Logical Agents must infer from the knowledge base that has the information from the past or background knowledge

Knowledge Based Agent: Levels of Knowledge Base

- Knowledge Level
  - The most abstract level
  - Describes agent by saying what it knows
  - Example:
    - An intelligent taxi might know that the Bagmati Bridge connects Kathmandu with Lalitpur
- Logical Level
  - The level at which the knowledge is encoded into formal sentences
  - Example:
    - Joins(Bagmati bridge, Kathmandu, Lalitpur)
- Implementation Level
  - Physical representation of the sentences in the logical level
  - Example:
    - Objects, string, dams, etc.

Approaches of system building

- Declarative approach
  - Designing the representation language to make it easy to express the knowledge in the form of sentences
- Procedural approach
  - Encoded desired behaviour directly as program code

Logic

- Logic
- Syntax: Formal standard to express sentences so that the sentences are well formed
- Semantics: Has to do with the meaning of sentences
  - Defines the truth of the sentences with respect to respective possible world
- Connectives: Joins the different components of the sentence
- Model and Real World
- Entailment: the idea that a sentence follows logically from another sentence
  - Example: $\alpha \models \beta$,where α & β are sentences and β follows from α

Logic

- An inference algorithm that derives only entailed sentences is called sound or truth preserving
- Completeness is desirable
  - An inference algorithm is complete if it can derive any sentence that is entailed
- If knowledge base is true in the real world, then any sentence derived from the knowledge base by a sound inference procedure is also true in the real world

Logic

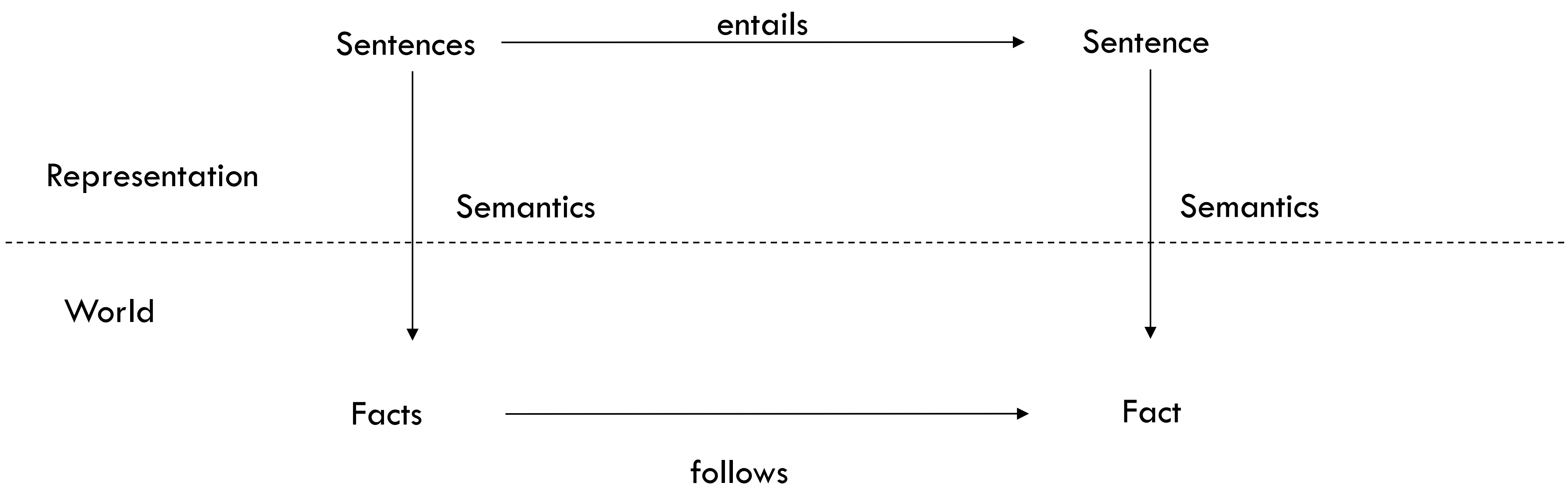


Figure: semantics map sentences in logic to fact in the world

Logic

- Example
- Knowledge Base
  - Socrates is a man
  - All men are Mortal
  - Äll men are kind
- Inference algorithm is applied to the above base
- Inferring “Socrates is Mortal”
- “Socrates is kind” follows the sentence “All men are Kind

Truth Table

P	Q	!P	PᵛQ	P^Q
False	False	True	False	False
False	True	True	True	False
True	False	False	True	False
True	True	False	True	True

Tautology and Validity

- A notation used in formal logic which is always true and valid.
- Example: A OR (NOT A)
  I am eating food OR I am not eating food
- If all the conditions for a statement is true its tautology
- Tautologies are also called valid sentences

Knowledge Models

- A model is a world in which a sentence is true under a particular interpretation
- There can be several models at once that have the same interpretations
- Types:
  - First order logic
  - Procedural Representation Model
  - Relational Representation Model
  - Hierarchical Representation Model
  - Semantic Nets

Knowledge Models: Types

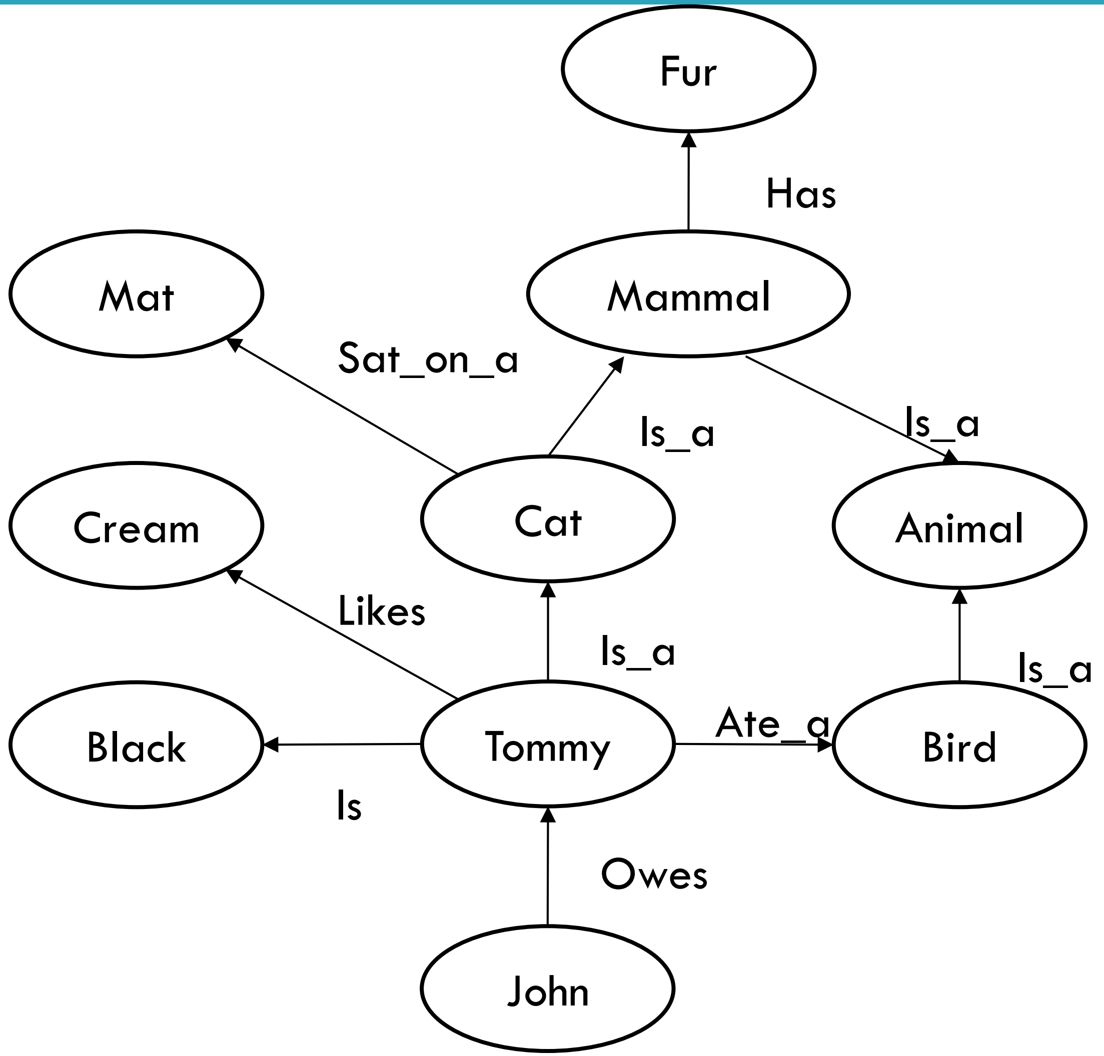
- First Order Logic
  - First Order Predicate Calculus
  - Consists of objects, predicates on objects, connectives and quantifiers
  - Predicates are the relations between objects or properties of the objects
  - Connectives and quantifiers allow for universal sentences
  - Relations between objects can be true or false
- Procedural Representation Model
  - This model of knowledge representation encodes facts along with the sequence of operations for manipulation and processing of the facts
  - Expert systems are based on this model
  - It works best when experts follow set of procedures for problem solving
  - Example: doctor making diagnosis

Knowledge Models: Types

- Relational Representation Model
  - Collection of knowledge are stored in tabular form
  - Mostly used in commercial databases, relational databases
  - The information is manipulated with relational calculus use a language like SQL, Oracle, etc.
  - Its flexible way of storing information by not good for storing complex relationships
  - Problem arises when more than one subject area is attempted
  - A new knowledge base from scratch has to be built for each area of expertise
- Hierarchical Representation Model
  - Based on inherited knowledge and the relationship and shared attributes between objects

Knowledge Models: Types

- Semantic Nets
  - Semantic networks are an alternative to predicate logic as a form of knowledge representation
  - The idea is that we can store our knowledge in the form of graph with nodes representing objects in the world and are representing relationships between those objects



Propositional Logic

- It is declarative sentences which can either be true or false but not both or neither
- A Very simple logic
- A Mathematical model that allows us to reason about the truth or falsehood of logical expressions
- There are sentences and connectives to describe an expression
- Its syntax defines allowable sentences
- Example:
  - Is it raining?
  - Is 2+2=5?
- Logical Connectives in Propositional Logic
  - ^ : Conjunction (and)
  - v : Disjunction (or)
  - ˥ : Negation (not)
  - => ➔ : Implication (if…then…)
  - <=> ⬄ : Logical Equivalence (If and only If)

Propositional Logic: Truth Tables

A	¬A
T	F
F	T

A	B	A^B
F	F	F
F	T	F
T	F	F
T	T	T

A	B	A˅B
F	F	F
F	T	T
T	F	T
T	T	T

A	B	A=>B
F	F	T
F	T	F
T	F	T
T	T	T

A	B	A<=>B
F	F	T
F	T	F
T	F	F
T	T	T

Propositional Logic

- Sentence Properties
  - T or F itself is a sentence
  - Individual Proposition symbols are sentences eg. P, Q, …
  - If s is a sentence, so is (s)
  - If S1 and S2 are sentences, so are: ¬S1, ¬S2, S1^S2, etc.
- Order of Precedence
  - ¬ : Negation (not)
  - ^ : Conjunction (and)
  - v : Disjunction (or)
  - => ➔ : Implication (if…then…)
  - <=> ⇔ : Logical Equivalence (If and only If)

Propositional Logic

- Atomic Sentences
  - Single sentence
  - T, F, P, Q,…
    where, each symbol stands for proposition that can be true or false.
  - Example: P=“Ram likes Rice”
    Q=“Sita is women”
- Complex Sentences
  - Sentences constructed from simple sentences using logical connectives
  - Example: P=“It is hot today”
    Q=“It is humid today”
    P^Q
    “It is hot and humid today”

Propositional Logic

- Unsatisfiable (Contradiction)
  - If all the sentences or statements are always false
  - Example: “There will be a clear sky during rainy day”
- Satisfiable
  - If at least one sentence in the knowledge base is true

Propositional Logic: Equivalence Laws

1. $P \Rightarrow Q \equiv \neg P \vee Q$
2. $P \Leftrightarrow Q \equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P)$
3. Distributive Laws
   $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
   $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
4. De-Morgan's Law
   $\neg(A \wedge B \wedge C) \equiv (\neg A) \vee (\neg B) \vee (\neg C)$
   $\neg(A \vee B \vee C) \equiv (\neg A) \wedge (\neg B) \wedge (\neg C)$

Propositional Logic: Inference Rules

1. Modus Ponen Rule
   Whenever any sentence of the form P =>Q and P are given, then the sentence Q can be inferred
   $$\frac{P \Rightarrow Q,\ P}{Q}$$
2. And Elimination
   $$\frac{A \wedge B}{A \mid B}$$
   sentence A or B can be inferred if A and B is given
3. And Introduction
   $$\frac{A,\ B,\ \ldots\ldots\ldots\ldots N}{A \wedge B \wedge \ldots\ldots \wedge N}$$
4. Or Introduction
   $$\frac{A,\ B,\ \ldots\ldots\ldots\ldots N}{A \vee B \vee \ldots\ldots \vee N}$$
5. Double Negation Elimination
   $$\frac{\neg\neg P}{P}$$

Propositional Logic: Inference Rules

6. Unit Resolution

$$\frac{A \vee B,\ \neg A}{B}$$

7. Modus Tollens

$$\frac{P \Rightarrow Q,\ \neg Q}{\neg P}$$

8. Resolution Chaining

$$\frac{P \Rightarrow Q,\ Q \Rightarrow R}{P \Rightarrow R}$$

$$\frac{\neg P \Rightarrow Q,\ Q \Rightarrow R}{\neg P \Rightarrow R}$$

Propositional Logic

- The semantics defines the rules for determining the truth of sentences with respect to a particular model, i.e. semantic must specify how to compute the truth value of any sentence in a given model.

Propositional Logic: BNF Grammar

- Backus Normal Form or Backus Naur Form
- It's a notation technique for context free grammars often used to describe the syntax of languages used in computing
- BNF can be used in two ways:
  - To generate strings belonging to the grammar
  - To recognize strings belonging to the grammar

Normal Forms of Propositional Logic Sentences

1. Conjunctive (disjunction of conjunction of literals) Normal Form
   - In which a sentence is written as the conjunction of literals

     $(A \vee B) \Rightarrow Q$

     $\equiv \quad \neg(A \vee B) \vee Q$

     $\equiv \quad (\neg A \wedge \neg B) \vee Q$

     $\equiv \quad (\neg A \vee Q) \wedge (\neg B \vee Q)$

2. Disjunctive (conjunction of disjunction of literals) Normal Form
   - In which a sentence is written as the disjunction of literals

     $(A \wedge Q) \vee (B \wedge Q)$

First Order Predicate Logic (FOPL)

- Propositional logic assumes that the world or system being modelled can be described in terms of fixed, known set of propositions
- This assumption can make it awkward or even impossible to specify many pieces of knowledge

- Example:
  - Consider a general sentence "if a person is rich then they have a nice car"
  - In propositional logic, we can generate rule for each person as
    - Bob_is_rich → Bob_has_a_nice_car
    - John_is_rich→John_has_a_nice_car
  - This seems to be an impractical way to represent knowledge, hence, generalization to represent this type of knowledge is a must

First Order Predicate Logic (FOPL)

- FOPL is a logic that gives us the ability to quantify over objects
- In FOPL, statements from a natural language like English are translated into symbolic structure composed of predicates, functions, variables, constants, quantifiers and logical connectives
- First Order Predicate Logic represents facts by separating classes and individuals and consider that world consists of different objects and relations between those objects

FOPL: Syntax

Sentence	→	AtomicSentence | (Sentence Connective Sentence) | Quantifier Variable,…Sentence | ⌉Sentence
AtomicSentence	→	Predicate(Term,….) | Term = Term
Term	→	Function (Term,…) | Constant | Variable
Connective	→	⌉ | ˅ | ˄ | => | <=>
Quantifier	→	∀ | ∃
Constant	→	A | X | John | …
Variable	→	a | x | s | …
Predicate	→	Before | HasColor | Raining | …
Function	→	Mother | Leftleg | …

FOPL: Syntax

- Constant Symbols are the strings that will be interpreted as representing objects
- Variable Symbols are used as place holders for quantifying over objects
- Predicate symbols are used to denote properties of objects and relationship among them
- Function Symbols map the specified number of input objects to objects
- Quantifiers are used to quantify objects
  - Universal Quantifier represents for all
  - Existential Quantifier represents the existence of an object

FOPL: Variable Scope

- The scope of the variable is in the sentence to which the quantifier syntactically applies
- In a block structured programming language, a variable in a logical expression refers to the closest quantifier within whose scope it appears
- In a well formed formula all the variables should be properly introduced

Relation Between Quantifiers

- $\forall x \neg P \equiv \neg \exists x P$
- $\neg \forall x P \equiv \exists x \neg P$
- $\forall x P \equiv \neg \exists x \neg P$
- $\exists x P \equiv \neg \forall x \neg P$
- $\forall x P(x) \cap Q(x) \equiv$
  $\forall x P(x) \cap \forall x Q(x)$
- $\exists x P(x) \cup Q(x) \equiv$
  $\exists x P(x) \cup \exists x Q(x)$

Examples

- All birds can't fly
  $\forall x\ Bird(x) \rightarrow \neg Fly(x)$
  OR
  $\neg(\exists x\ (Bird(x) \cap Fly(x))$
- Not all birds can fly
  $\neg(\forall x\ Bird(x)$
  OR
  Type equation here.
- If anyone can solve the problem then Raju can
  $\exists x Solves(x, problem) \rightarrow Solves(Raju, problem)$
- Try these
  - Nobody in electrical class is smarter than everyone in AI class
  - John hates all the people who don't hate themselves

Equality

- Can include equality as a primitive predicate in the logic or require it to be introduces and axiomitized as the identity relation
- Useful in representing certain types of knowledge
  - Example: Sita owns two cars $\exists x \exists y(Owns\ (Sita, x) \cap Owns\ (Sita, y) \cap Car(x) \cap Car(y) \cap \neg(x = y))$

- Try these:
  - There are exactly two purple flowers out of three
  - Everyone is married to exactly one person

Every gardener likes the sun.

$\forall x \text{ gardener}(x) \rightarrow \text{likes}(x,\text{Sun})$

You can fool some of the people all of the time.

$\exists x\ \forall t\ \ \text{person}(x) \wedge \text{time}(t) \rightarrow \text{can-fool}(x,t)$

You can fool all of the people some of the time.

$\forall x\ \exists t\ (\text{person}(x) \rightarrow \text{time}(t) \wedge \text{can-fool}(x,t))$

$\forall x\ (\text{person}(x) \rightarrow \exists t\ (\text{time}(t) \wedge \text{can-fool}(x,t)))$

All purple mushrooms are poisonous.

$\forall x\ (\text{mushroom}(x) \wedge \text{purple}(x)) \rightarrow \text{poisonous}(x)$

No purple mushroom is poisonous.

$\neg\exists x\ purple(x) \land mushroom(x) \land poisonous(x)$

$\forall x\ \ (mushroom(x) \land purple(x)) \rightarrow \neg poisonous(x)$

There are exactly two purple mushrooms.

$\exists x\ \exists y\ mushroom(x) \land purple(x) \land mushroom(y) \land purple(y)\ \text{^}\ \neg(x=y) \land \forall z\ (mushroom(z) \land purple(z)) \rightarrow ((x=z) \lor (y=z))$

Clinton is not tall.

$\neg tall(Clinton)$

X is above Y iff X is on directly on top of Y or there is a pile of one or more other objects directly on top of one another starting with X and ending with Y.

$\forall x\ \forall y\ above(x,y) \leftrightarrow (on(x,y) \lor\ \ \exists z\ (on(x,z) \land above(z,y)))$

Try few more

- Ram likes all kinds of food
- Anything anyone eats and is not killed by is food
- Rita eats samosa and is still alive
- Gita eats everything Rita eats
- Someone who hates something owned by another person will not love that person
- There is a barber in the town who shaves all men in the town who don't shaves themselves
- Everyone loves somebody
- No one likes everyone
- There is someone who is liked by everyone
- You can fool some of the people every time
- All employee earning Rs.200000|- or more per year pay taxes
- Some employee are sick today
- Nobody earns more than the chairman

Horn Clause

- Disjunction of literals of which at most one is positive is Horn Clause

$$P1 \cap P2 \cap \cdots \cap Pn \Rightarrow Q \equiv \neg P1 \cup \neg P2 \cup \cdots \cup \neg Pn \cup Q$$

- Clause with exactly one positive literals giving definite clause (fact)
- Horn clause with no positive literals can be written as an implication whose conclusion is the literal false

$$\neg x1 \cup \neg x2 \equiv x1 \cap x2 \Rightarrow False$$

Horn Clause

Reason for its importance

- Every horn clause can be written as an implication whose premises is a conjunction of positive literals and whose conclusion is a single positive literal Example: $\neg L1 \cup \neg L2 \cup B$ can be written as $L1 \cap L2 \Rightarrow B$
- Inference with horn clauses can be done with the forward chaining and backward chaining
- Deciding entailment with horn clauses can be done in time that is linear in the size of knowledge base

Well Formed Formula

- A sentence that has all its variables properly introduced using quantifiers is a well formed formula
- Example:
  $\forall x P(x, y)$ is not a well formed formula where x is bounded as universal quantifier and y is free
  $\forall x \exists y Q(x, y)$ is a well formed formula where both x and y are bounded

- Notes:
  - Predicate can't be quantifiers
  - Constant can't be negative
  - Letter cases must be well considered

Inference in FOL

- If x is a parent of y, then x is older than y
- If x is the mother of y then x is a parent of y
- Devaki is the mother of Krishna
- Conclusion:
  Devaki is older than Krishna

Mapping in FOL

- $\forall x \forall y \; parent(x, y) \Rightarrow older(x, y)$
- $\forall x \forall y \; mother(x, y) \Rightarrow parent(x, y)$
- $mother(Devaki, Krishna)$
- Conclusion:
  older(Devaki,Krishna)

Inference Rules in FOL

- Universal Instantiation
  - If a person is a student, studies in KEC and studies AI, then he/she is a third year student
  - $\forall x student(x) \cap studiesin(x, KEC) \cap studies(x, AI) \Rightarrow thirdyearstudent(x)$
- Existential Instantiation
  - There must be a topper in KEC
  - $\exists x student(x) \cap studiesin(x, KEC) \cap topper(x)$
- Propositionization
  - All people are kind
    $\forall x person(x) \Rightarrow kind(x)$
    It can be inferred as

$person(Ram) \Rightarrow kind(Ram)$

Inference Rules in FOL

- Generalized Modus Ponens
  - $\forall x student(x) \cap studieshard(x) \Rightarrow good\ student\ (x)$
  - $student(Arjun)$
  - $studieshard(Arjun)$
  - Conclusion: $goodstudent(Arjun)$
- Unification
  - $[knows(Sita, x),\ knows(Sita, Rita)]$ $x = Rita$
  - $[knows(Sita, x), knows(y, Rita)] \Rightarrow x = Rita, y = Sita$
  - $[knows(Sita, x), knows(y, ismother(y))] \Rightarrow y = Sita, x = mother(Sita)$
  - $[knows(Sita, x), knows(x, Rita)] \Rightarrow false$

Inference Rules in FOL

- Resolution
  - Produces proof by refutation (proof person or statement that is wrong)
  - Resolution can be applied to sentences in CNF (conjunctive normal form)
- Process of Resolution
  - Convert all sentences to CNF
  - Negate x
  - Add negate x to premises
  - Repeat until either a contradiction is detected or no progress is being made

CNF Conversion Process

1. Elimination of all implications with equivalence symbols
   - $P \rightarrow Q \equiv \neg P \cup Q$
   - $P <=> Q \equiv (\neg P \cup Q) \cap (\neg Q \cup P)$
2. Move $\neg$ inward (use De'Morgans law)
   - $\neg(P \cap Q) \equiv \neg P \cup \neg Q$
   - $\neg(P \cup Q) \equiv \neg P \cap \neg Q$
   - $\forall x\, \neg P \equiv \neg \exists x\, P$
   - $\neg \forall x\, P \equiv \exists x\, \neg P$
   - $\forall x P \equiv \neg \exists x \neg P$
   - $\exists x P \equiv \neg \forall x \neg P$
3. Standardize Variables
   - Rename variables if necessary so that all quantifiers have different variable assignments

CNF Conversion Process

4. Skolemization
   - The process of eliminating the existential quantifiers through a substitution process
   - The process requires that all such variables be replaced by short term functions, which can always assume a Skolen function, a correct value required for an existential quantifier variable

- If leftmost quantifier in an expression is existential quantifier (∃), replace all occurrence of the variables that quantifies with an arbitrary constant not appearing elsewhere in the expression and delete the quantifier
  - Example: $\exists x \exists y \forall z P(x, y, z) \cup Q(x, y) \equiv \forall z P(a, b, z) \cup Q(a, b)$

CNF Conversion Process

4. Skolemization
   - If existential quantifier (∃) is preceded by universal quantifier (∀), replace the existentially quantified variable by a function symbol whose arguments are variable appearing in those universal quantifiers

- Example:

$$\exists u \forall x \forall y \exists z\, P(f(u), x, y, z) \cup Q(x, y, z)$$
$$\equiv \forall x \forall y \exists z\, P(f(a), x, y, z) \cup Q(x, y, z)$$
$$\equiv \forall x \forall y\, P(f(a), x, y, f(x, y)) \cup Q(x, y, f(x, y))$$

5. Drop all universal quantifiers
6. Distribute ^over ˅

Example: Given Premises

1. If x is on top of y, y support x
2. If x is above y and they are touching each other, x is on top of y
3. Everything is on top of another thing
4. A cup is above a book
5. A cup is touching a book

- Answer:

Is the book supporting the cup?

Example: Solution

- $\forall x \forall y\ ontop(x,y) \Rightarrow supports\ (y,x)$
  $Implication\ Elimination$
  $\forall x \forall y \neg ontop(x,y)$
  $\quad \Rightarrow supports\ (y,x)$
  $Drop\ \ \forall x\ and\ \forall y$
  $\neg ontop(x,y)$
  $\quad \Rightarrow supports\ (y,x)$

- $\forall x,y\ above(x,y) \cap touch(x,y) \Rightarrow ontop(x,y)$
  $Implication\ Elimination$
  $\forall x,y\ \neg above(x,y)$
  $\quad \cup \neg touch(x,y)$
  $\quad \cup\ ontop(x,y)$
  $Drop\ \forall x,y$
  $\neg above(x,y) \cup \neg touch(x,y)$
  $\quad \cup\ ontop(x,y)$

Example: Solution

- $\forall x, y\ ontop(x, y)$

  $Drop\ \forall x, y$

  $ontop(x, y)$
- $above(cup, book)$
- $touch(cup, book)$

Solution

Conclusion

- $supports(book, cup)$
  Let
  $\neg supports(book, cup)$
  $using\ second\ and\ fifth\ conditions$
  $\neg above(x, y) \cup \neg touch(x, y)$
  $\cup\ ontop(x, y)$
  $touch(cup, book)$
  $\neg above(x, y) \cup ontop(x, y)$
  $using\ fourth\ condition$
  $above(cup, book)$

$ontop(x, y)$
$using\ first\ condition$
$\neg ontop\ (x, y) \cup supports(y, x)$
$supports(book, cup)$
$using\ assumed\ condition$
$\neg supports(book, cup)$
$Empty\ Clause$
$Hence, the\ book\ is\ supporting\ the\ cup$

Try these

- Every American who sells weapon to hostile nation is a criminal. The country Iraq is an enemy of America. All of the missiles in Iraq were sold by George. George is an American.
  Prove:
  George is a Criminal
- All Pompeians are Romans. All Romans were either loyal to Caesor or hated him.
  Everyone is loyal to someone.
  People only try to assassinate rulers they are not loyal to.
  Marcus tried to assassinate Caesor. Marcus was a Pompeian.
  Conclude:
  Did Marcus hate Caesor?

Forward Chaining

- One of the two main methods for reasoning using inference rules
- Can be described logically as repeated application of Modus Ponens
- It's a popular strategy of reasoning in expert system and production systems
- It starts with the available data and uses inference rules to extract more data until a goal is reached
- An inference engine using forward chaining searches the inference rules until it founds one where antecedent (If clause) is known to be true

Forward Chaining

- When it found if clause it can conclude or infer the consequent (then clause) to its data resulting in the addition of new information
- Example: (Animal Identification System)
  If X croaks and eats flies then it's a frog
  If X chirps and sings then it's a canary
- If X is a frog then X is green
  If X is a canary then X is yellow
  goal: colour of pet given that it croaks and eat flies

References

- Russell, S. and Norvig, P., 2011, Artificial Intelligence: A Modern Approach, Pearson, India.
- Rich, E. and Knight, K., 2004, Artificial Intelligence, Tata McGraw hill, India.

Thank You

Any Queries?

Now, Search for yourself.