REPRESENTATION OF KNOWLEDGE

Game playing - Knowledge representation, Knowledge representation using Predicate logic, Introduction to predicate calculus, Resolution, Use of predicate calculus, Knowledge representation using other logic-Structured representation of knowledge.

Knowledge Representation (KR)

Given the world

- Express the general facts or beliefs using a language
- Determine what else we should (not) believe

Logic consists of

- A language which tells us how to build up sentences in the language (i.e., syntax), and and what the sentences mean (i.e., semantics)
- An inference procedure which tells us which sentences are valid inferences from other sentences

![Diagram](chart.png)

Add more facts
Delete existing facts
Pose queries

Contains:
- Facts (quantified or not)
  + Function implementations

Result
Domain independent algorithms

ASK

Inference engine

TELL

Knowledge Base

Domain specific content

function KB-AGENT(percept) returns an action
static: KB, a knowledge base
        t, a counter, initially 0, indicating time
TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
action ← ASK(KB, MAKE-ACTION-QUERY(t))
TELL(KB, MAKE-ACTION-SENTENCE(action, t))
t ← t + 1
return action

Wumpus World Problem:
**Percepts** Breeze, Glitter, Smell

**Actions** Left turn, Right turn, Forward, Grab, Release, Shoot

**Goals** Get gold back to start without entering pit or wumpus square

**Environment**
- Squares adjacent to wumpus are smelly
- Squares adjacent to pit are breezy
- Glitter if and only if gold is in the same square
- Shooting kills the wumpus if you are facing it
- Shooting uses up the only arrow
- Grabbing picks up the gold if in the same square
- Releasing drops the gold in the same square

**Characteristics of Wumpus World:**

- **Deterministic?** Yes – outcome exactly specified.
- **Accessible?** No – only local perception.
- **Static?** Yes – Wumpus and pits do not move.
- **Discrete?** Yes
- **Episodic?** (Yes) – because static.
Logics are formal languages for representing information such that conclusions can be drawn.

Syntax defines the sentences in the language.

Semantics define the “meaning” of sentences; i.e., define 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 $x = 0, y = 6$. 
Propositional logic

The symbols of propositional calculus are

The propositional symbols:
P, Q, R, S, ...

The truth symbols:
true, false

and connectives:
∧, ∨, ¬, →, ≡

Every propositional symbol and truth symbol is a sentence.
Examples: true, P, Q, R.

The negation of a sentence is a sentence.
Examples: ¬P, ¬ false.

The conjunction, or and, of two sentences is a sentence.
Example: P ∧ ¬P

The disjunction, or or, of two sentences is a sentence.
Example: P ∨ ¬P

The implication of one sentence from another is a sentence.
Example: P → Q

The equivalence of two sentences is a sentence.
Example: P ∨ Q ⇔ R

Legal sentences are also called well-formed formulas or WFFs.

An interpretation of a set of propositions is the assignment of a truth value, either T or F to each propositional symbol.

The symbol true is always assigned T, and the symbol false is assigned F.
The truth assignment of *negation*, \( \neg P \), where \( P \) is any propositional symbol, is \( F \) if the assignment to \( P \) is \( T \), and is \( T \) if the assignment to \( P \) is \( F \).

The truth assignment of *conjunction*, \( \wedge \), is \( T \) only when both conjuncts have truth value \( T \); otherwise it is \( F \).

Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms:

**Conjunctive Normal Form (CNF—universal)**

conjunction of \( \neg \) disjunctions of literals

E.g., \( (A \lor \neg B) \land (B \lor \neg C \lor \neg D) \)

**Disjunctive Normal Form (DNF—universal)**

disjunction of \( \neg \) conjunctions of literals

E.g., \( (A \land B) \lor (A \land \neg C) \lor (A \land \neg D) \lor (\neg B \land \neg C) \lor (\neg B \land \neg D) \)

**Horn Form (restricted)**

conjunction of Horn clauses (clauses with \( \leq 1 \) positive literal)

E.g., \( (A \lor \neg B) \land (B \lor \neg C \lor \neg D) \)

Often written as set of implications:
\[ B \Rightarrow A \text{ and } (C \land D) \Rightarrow B \]
**Entailment:**

\[ KB \models Sen1 \]

The knowledge Base “KB” entails a Sentence “Sen1” if and only if Sen1 is true in the domain where the KB is true.

**Representation:** Sentences \( \models \) Sentence

\[ \text{Refers to} \quad \text{(Semantics)} \]

World \( \models \) Facts \( \Rightarrow \) Fact

**CNF:**

Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms

- **Conjunctive Normal Form (CNF—universal)**
  
  \[ \text{conjunction of disjunctions of literals} \]

  E.g., \( (A \lor \neg B) \land (B \lor \neg C \lor \neg D) \)

- **Disjunctive Normal Form (DNF—universal)**
  
  \[ \text{disjunction of conjunctions of literals} \]

  E.g., \( (A \land B) \lor (A \land \neg C) \lor (A \land \neg D) \lor (\neg B \land \neg C) \lor (\neg B \land \neg D) \)

- **Horn Form (restricted)**
  
  \[ \text{conjunction of Horn clauses (clauses with } \leq 1 \text{ positive literal)} \]

  E.g., \( (A \lor \neg B) \land (B \lor \neg C \lor \neg D) \)

  Often written as set of implications:

  \( B \Rightarrow A \) and \( (C \land D) \Rightarrow B \)
◊ Modus Ponens or Implication-Elimination: (From an implication and the premise of the implication, you can infer the conclusion.)

$$\alpha \Rightarrow \beta, \alpha \quad \therefore \quad \beta$$

◊ And-Elimination: (From a conjunction, you can infer any of the conjuncts.)

$$\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n \quad \therefore \quad \alpha_i$$

◊ And-Introduction: (From a list of sentences, you can infer their conjunction.)

$$\alpha_1, \alpha_2, \ldots, \alpha_n \quad \therefore \quad \alpha_1 \land \alpha_2 \land \ldots \land \alpha_n$$

◊ Or-Introduction: (From a sentence, you can infer its disjunction with anything else at all.)

$$\alpha_i \quad \therefore \quad \alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n$$

◊ Double-Negation Elimination: (From a doubly negated sentence, you can infer a positive sentence.)

$$\neg \neg \alpha \quad \therefore \quad \alpha$$

◊ Unit Resolution: (From a disjunction, if one of the disjuncts is false, then you can infer the other one is true.)

$$\alpha \lor \beta, \neg \beta \quad \therefore \quad \alpha$$

◊ Resolution: (This is the most difficult. Because $\beta$ cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)

$$\alpha \lor \beta, \neg \beta \lor \gamma \quad \text{or equivalently} \quad \neg \alpha \Rightarrow \beta, \beta \Rightarrow \gamma \quad \therefore \quad \neg \alpha \Rightarrow \gamma$$
Resolution

- Resolution is a sound and complete inference procedure.

- Reminder: Resolution rule for propositional logic:
  - \( P_1 \lor P_2 \lor \ldots \lor P_n \)
  - \( \neg P_1 \lor Q_2 \lor \ldots \lor Q_m \)
  - \text{Resolvent: } P_2 \lor \ldots \lor P_n \lor Q_2 \lor \ldots \lor Q_m

- Examples:
  - \( P \land \neg P \lor Q \) : derive \( Q \) (Modus Ponens)
  - \( (\neg P \lor Q) \land (\neg Q \lor R) \) : derive \( \neg P \lor R \)
  - \( P \land \neg P \) : derive False [contradiction!]
  - \( (P \lor Q) \land (\neg P \lor \neg Q) \) : derive True

```
function PL-RESOLUTION(KB, \alpha) returns true or false

clauses ← the set of clauses in the CNF representation of \( KB \land \neg \alpha \)
new ← \{\}

loop do
    for each \( C_i, C_j \) in clauses do
        resolvents ← PL-RESOLVE(C_i, C_j)
        if resolvents contains the empty clause then return true
        new ← new ∪ resolvents
    if new ⊆ clauses then return false
    clauses ← clauses ∪ new
```

FOL:

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, war

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

Syntax of FOL: Basic elements

- Constants: John, 2, DIT,
- Predicates: Brother, >,
- Functions: Sqrt, LeftLegOf,
- Variables: x, y, a, b,
- Connectives: ¬, ⇒, ∧, ∨, ⇔
- Equality: =
- Quantifiers: ∀, ∃

Atomic sentences:

Atomic sentence = predicate(term₁,...,termₙ) or term₁ = term₂

Term = function(term₁,...,termₙ) or constant or variable

Example:

\[ \text{Brother(John,RichardTheLionheart) > (Length(LeftLegOf(Richard)), Length(LeftLegOf(John)))} \]

Complex sentences:

Complex sentences are made from atomic sentences using connectives

¬S₁, S₁ ∧ S₂, S₁ ∨ S₂, S₁ ⇒ S₂, S₁ ⇔ S₂,

Example:

\[ \text{Sibling(John,Richard) ⇒ Sibling(Richard,John)} \]

\[ >(1,2) ∨ ≤ (1,2) \]

\[ >(1,2) ∧ ¬ >(1,2) \]
Universal quantification:

$\forall <variables> <sentence>$

Everyone at SVCET is smart:

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

$\forall x \, P$ is true in a model $m$ iff $P$ is true with $x$ being each possible object in the model

Existential quantification:

$\exists <variables> <sentence>$

Someone at SVCET is smart:

$\exists x \, At(x, SVCET) \land Smart(x)$

$\exists x \, P$ is true in a model $m$ iff $P$ is true with $x$ being some possible object in the model

Quantifier duality:

each can be expressed using the other

$\forall x \, Likes(x, IceCream)$ is same as $\neg \exists x \, \neg Likes(x, IceCream)$

$\exists x \, Likes(x, Coffee)$ is same as $\neg \forall x \, \neg Likes(x, Coffee)$
Equality:

- \( \text{term}_1 = \text{term}_2 \) is true under a given interpretation if and only if \( \text{term}_1 \) and \( \text{term}_2 \) refer to the same object.

Example definition of Sibling in terms of Parent:

\[ \forall x, y \ Sibling(x, y) \equiv [\neg(x = y) \land \exists m, f - (m = f) \land Parent(m, x) \land Parent(f, x) \land Parent(m, y) \land Parent(f, y)] \]

The kinship domain Using FOL:

- Brothers are siblings
  \[ \forall x, y \ Brother(x, y) \equiv Sibling(x, y) \]

- One's mother is one's female parent
  \[ \forall m, c \ Mother(c) = m \equiv (Female(m) \land Parent(m, c)) \]

- "Sibling" is symmetric
  \[ \forall x, y \ Sibling(x, y) \equiv Sibling(y, x) \]

Knowledge engineering in FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

HORN CLAUSE:

- A Horn clause is a sentence of the form:
  \[ (\forall x) \ P_1(x) \land P_2(x) \land ... \land P_n(x) \rightarrow Q(x) \]

John likes to eat everything.

\[ \forall X \ \text{food}(X) \rightarrow \text{likes}(\text{john}, X) \]

John likes at least one dish Jane likes.

\[ \exists F \ \text{food}(F) \land \text{likes}(\text{Jane}, F) \land \text{likes}(\text{john}, F) \]

Everybody likes some food.

\[ \forall X \ \exists F \ \text{food}(F) \land \text{likes}(X, F) \]
There is a food that everyone likes.
\[ \exists F \forall X \text{food}(F) \land \text{likes}(X,F) \]

Whenever someone eats a spicy dish, they're happy.
\[ \forall X \exists F \text{food}(F) \land \text{spicy}(F) \land \text{eats}(X,F) \rightarrow \text{happy}(X) \]

John's meals are spicy:
\[ \forall X \text{meal-off}(\text{John},X) \rightarrow \text{spicy}(X) \]

Every city has a dogcatcher who has been bitten by every dog in town.
\[ \forall T \exists C \forall D \text{city}(C) \rightarrow (\text{dogcatcher}(C,T) \land (\text{dog}(D) \land \text{lives-in}(D,T) \rightarrow \text{bit}(D,C)) \]

For every set \( x \), there is a set \( y \), such that the cardinality of \( y \) is greater than the cardinality of \( x \).
\[ \forall X \exists Y \forall U \exists V \text{set}(X) \rightarrow (\text{set}(Y) \land \text{cardinality}(X,U) \land \text{cardinality}(Y,V) \land \text{greater-than}(V,U)) \]

**Generalized Modus Ponens:**

\[ \forall x \text{King}(x) \land \text{Greedy}(x) \rightarrow \text{Evil}(x) \]

\[ p_1', p_2', \ldots, p_n', (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \]

\[ \text{Subst}(\theta, q) \]

Example:
\[ p_1' \text{ is } \text{King}(\text{John}) \quad p_1 \text{ is } \text{King}(x) \]
\[ p_2' \text{ is } \text{Greedy}(y) \quad p_2 \text{ is } \text{Greedy}(x) \]
\[ \theta \text{ is } \{ x/\text{John}, y/\text{John} \} \quad q \text{ is } \text{Evil}(x) \]
\[ \text{Subst}(\theta, q) \text{ is } \text{Evil}(\text{John}) \]

- Implicit assumption that all variables universally quantified

- GMP is sound
  
  Only derives sentences that are logically entailed

- GMP is complete for a KB consisting of definite clauses
  
  Complete: derives all sentences that entailed

- Definite clause: disjunction of literals of which exactly 1 is positive,
  
  e.g., \( \text{King}(x) \land \text{Greedy}(x) \rightarrow \text{Evil}(x) \)
  
  NOT(\text{King}(x)) \lor \text{NOT}(\text{Greedy}(x)) \lor \text{Evil}(x)
• **Forward-chaining**
  - Uses GMP to add new atomic sentences
  - Useful for systems that make inferences as information streams in
  - Requires KB to be in form of first-order definite clauses

• **Backward-chaining**
  - Works backwards from a query to try to construct a proof
  - Can suffer from repeated states and incompleteness
  - Useful for query-driven inference

• **Resolution-based inference (FOL)**
  - Refutation-complete for general KB
    - Can be used to confirm or refute a sentence \( p \) (but not to generate all entailed sentences)
  - Requires FOL KB to be reduced to CNF
  - Uses generalized version of propositional inference rule

---

**Unification:**

- Unification is a **"pattern-matching"** procedure
  - Takes two atomic sentences, called literals, as input
  - Returns “Failure” if they do not match and a substitution list, \( \theta \), if they do
  - That is, \( \text{unify}(p, q) = \theta \) means \( \text{subst}(\theta, p) = \text{subst}(\theta, q) \) for two atomic sentences, \( p \) and \( q \)
  - \( \theta \) is called the most general unifier (mgu)
  - All variables in the given two literals are implicitly universally quantified
  - To make literals match, replace (universally quantified) variables by terms

**Unification Algorithm:**

```plaintext
procedure unify(p, q, \theta)
    Scan p and q left-to-right and find the first corresponding terms where p and q “disagree” (i.e., p and q not equal)
    If there is no disagreement, return \( \theta \) (success!)
    Let \( r \) and \( s \) be the terms in \( p \) and \( q \), respectively, where disagreement first occurs
    If variable(\( r \)) then {
        Let \( \theta = \text{union}(\theta, \{ r/s \}) \)
        Return unify(subst(\theta, p), subst(\theta, q), \theta)
    }
    else if variable(\( s \)) then {
        Let \( \theta = \text{union}(\theta, \{ s/r \}) \)
        Return unify(subst(\theta, p), subst(\theta, q), \theta)
    }
    else return “Failure”
end
```
Forward Chaining Algorithm

```plaintext
function FOL-FC-Ask(KB, α) returns a substitution or false

repeat until new is empty
    new ← \{
    for each sentence \( r \) in KB do
        (\( p_1 \land \ldots \land p_n \Rightarrow q \)) ← STANDARDIZE-APART(r)
        for each \( \theta \) such that (\( p_1 \land \ldots \land p_n \theta \) = (\( p'_1 \land \ldots \land p'_n \theta \))
            for some \( p'_1, \ldots, p'_n \) in KB
                \( q' \leftarrow \text{SUBST}(\theta, q) \)
                if \( q' \) is not a renaming of a sentence already in \( KB \) or \( \text{new} \) then do
                    add \( q' \) to \( \text{new} \)
                    \( \phi \leftarrow \text{UNIFY}(q', \alpha) \)
                    if \( \phi \) is not fail then return \( \phi \)
            add \( \text{new} \) to \( KB \)

return false
```

EXAMPLE:

- 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.

  it is a crime for an American to sell weapons to hostile nations:
  \( \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \)

  Nono ... has some missiles,
  i.e., \( \exists x \text{ Owns}(\text{Nono},x) \land \text{Missile}(x) \):
  \( \text{Owns}(\text{Nono},M_1) \) and \( \text{Missile}(M_1) \)

  All of its missiles were sold to it by Colonel West
  \( \text{Missile}(x) \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono}) \)

  Missiles are weapons:
  \( \text{Missile}(x) \Rightarrow \text{Weapon}(x) \)

  An enemy of America counts as "hostile":
  \( \text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \)

  West, who is American ...
  \( \text{American}(\text{West}) \)

  The country Nono, an enemy of America ...
  \( \text{Enemy}(\text{Nono}, \text{America}) \)
Backward chaining:

- Backward-chaining deduction using GMP is also complete for KBs containing only Horn clauses
- Proofs start with the goal query, find rules with that conclusion, and then prove each of the antecedents in the implication
- Keep going until you reach premises
- Avoid loops: check if new subgoal is already on the goal stack
- Avoid repeated work: check if new subgoal
  - Has already been proved true
  - Has already failed

It is a crime for an American to sell weapons to hostile nations:

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

None has some missiles,

\[ \exists x \, \text{Owns}(\text{None},x) \land \text{Missile}(x) \]

\[ \text{Owns(} \text{None}, \text{M}_1 \text{) and Missile(} \text{M}_1 \text{)} \]

All of its missiles were sold to it by Colonel West

\[ \text{Missile}(x) \land \text{Owns(} \text{None},x) \Rightarrow \text{Sells(} \text{West},x,\text{None}) \]

Missiles are weapons:

\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]

An enemy of America counts as "hostile":

\[ \text{Enemy}(x,\text{America}) \Rightarrow \text{Hostile}(x) \]
West, who is American

\[ \text{American}(\text{West}) \]

The country Nono, an enemy of America ...

\[ \text{Enemy}(\text{Nono}, \text{America}) \]

Can be converted to CNF

Query: Criminal(West)?

First Order Logic: (FOL)

```
function FOL-BC-ASK(KB, goals, \theta) returns a set of substitutions
  inputs: KB, a knowledge base
           goals, a list of conjuncts forming a query
           \theta, the current substitution, initially the empty substitution \{ \}
  local variables: ans, a set of substitutions, initially empty
  if goals is empty then return \{ \theta \}
  \text{else}
  \q' \leftarrow \text{SUBST}(\theta, \text{FIRST}(goals))
  \text{for each } r \text{ in } KB \text{ where } \text{STANDARDIZE-APART}(r) = (p_1 \land \ldots \land p_n \Rightarrow q)
  \text{and } \theta' \leftarrow \text{UNIFY}(q, \q') \text{ succeeds}
  \text{ans} \leftarrow \text{FOL-BC-ASK}(KB, [p_1, \ldots, p_n|\text{REST(goals)}], \text{COMPOSE}(\theta, \theta')) \cup \text{ans}
  \text{return ans}
```
Resolution in first-order logic

- Given sentences
  \[ P_1 \lor \ldots \lor P_n \]
  \[ Q_1 \lor \ldots \lor Q_m \]
- in conjunctive normal form:
  - each \( P_i \) and \( Q_k \) is a literal, i.e., a positive or negated predicate symbol with its terms,
  - if \( P_i \) and \( \neg Q_k \) unify with substitution list \( \theta \), then derive the resolvent sentence
  - \( \text{subst}(\theta, P_1 \lor \ldots \lor P_{j-1} \lor P_j \lor P_{j+1} \ldots P_n \lor Q_1 \lor \ldots \lor Q_{k-1} \lor Q_{k+1} \lor \ldots \lor Q_m) \)

Example
- from clause \( P(x, f(x)) \lor P(x, f(y)) \lor Q(y) \)
- and clause \( \neg P(z, f(a)) \lor \neg Q(z) \)
- derive resolvent \( P(z, f(y)) \lor Q(y) \lor \neg Q(z) \)
- using \( \theta = \{x/z\} \)

Converting sentences to CNF

1. Eliminate all \( \leftrightarrow \) connectives
   \( (P \leftrightarrow Q) \Rightarrow ((P \rightarrow Q) \land (Q \rightarrow P)) \)

2. Eliminate all \( \rightarrow \) connectives
   \( (P \rightarrow Q) \Rightarrow (\neg P \lor Q) \)

3. Reduce the scope of each negation symbol to a single predicate
   \( \neg \neg P \Rightarrow P \)
   \( \neg (P \lor Q) \Rightarrow \neg P \land \neg Q \)
   \( \neg (P \land Q) \Rightarrow \neg P \lor \neg Q \)
   \( \neg (\forall x)P \Rightarrow (\exists x)\neg P \)
   \( \neg (\exists x)P \Rightarrow (\forall x)\neg P \)

4. Standardize variables: rename all variables so that each quantifier has its own unique variable name

Converting sentences to clausal form Skolem constants and functions

5. Eliminate existential quantification by introducing Skolem constants/functions
   \( (\exists x)P(x) \Rightarrow P(c) \)
   - \( c \) is a Skolem constant (a brand-new constant symbol that is not used in any other sentence)
   \( (\forall x)(\exists y)P(x, y) \Rightarrow (\forall x)P(x, f(x)) \)
   - since \( \exists y \) is within the scope of a universally quantified variable, use a Skolem function \( f \) to construct a new value that depends on the universally quantified variable
f must be a brand-new function name not occurring in any other sentence in the KB.
E.g., (\forall x) (\exists y) loves(x, y) \Rightarrow (\forall x) loves(x, f(x))

In this case, f(x) specifies the person that x loves

6. Remove universal quantifiers by (1) moving them all to the left end; (2) making the scope of each the entire sentence; and (3) dropping the “prefix” part
Ex: (\forall x) P(x) \Rightarrow P(x)

7. Put into conjunctive normal form (conjunction of disjunctions) using distributive and associative laws
(P \land Q) \lor R \Rightarrow (P \lor R) \land (Q \lor R)
(P \lor Q) \lor R \Rightarrow (P \lor Q \lor R)

8. Split conjuncts into separate clauses

9. Standardize variables so each clause contains only variable names that do not occur in any other clause

Resolution Proof

Resolution by refutation:

- Given a consistent set of axioms KB and goal sentence Q, show that KB |= Q
- Proof by contradiction: Add \neg Q to KB and try to prove false.
  i.e., (KB |= Q) \leftrightarrow (KB \lor \neg Q |= \text{False})
- Resolution is refutation complete: it can establish that a given sentence Q is entailed by KB, but can’t (in general) be used to generate all logical consequences of a set of sentences
- Also, it cannot be used to prove that Q is not entailed by KB.
- Resolution won’t always give an answer since entailment is only semidecidable
  And you can’t just run two proofs in parallel, one trying to prove Q and the other trying to prove \neg Q, since KB might not entail either one
Refutation resolution proof tree

\[-\text{allergies}(w) \lor \text{sneeze}(w) \quad \neg \text{cat}(y) \lor \neg \text{allergic-to-cats}(z) \lor \text{allergies}(z)\]

\[w/z\]

\[-\text{cat}(y) \lor \text{sneeze}(z) \lor \neg \text{allergic-to-cats}(z) \quad \text{cat}(\text{Felix})\]

\[y/\text{Felix}\]

\[\text{sneeze}(z) \lor \neg \text{allergic-to-cats}(z) \quad \text{allergic-to-cats}(\text{Lise})\]

\[z/\text{Lise}\]

\[\text{sneeze}(\text{Lise}) \quad \neg \text{sneeze}(\text{Lise})\]

\[\text{false}\]

\[\text{negated query}\]

- How to convert FOL sentences to conjunctive normal form (a.k.a. CNF, clause form): normalization and skolemization
- How to unify two argument lists, i.e., how to find their most general unifier (mgu) q: unification
- How to determine which two clauses in KB should be resolved next (among all resolvable pairs of clauses) : resolution (search) strategy

It is a crime for an American to sell weapons to hostile nations:

\[\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)\]

Nono ... has some missiles,

\[\exists x \ \text{Owns}(\text{Nono},x) \land \text{Missile}(x)\]

\[\text{Owns}(\text{Nono},M_1) \text{ and Missile}(M_1)\]

all of its missiles were sold to it by Colonel West

\[\text{Missile}(x) \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono})\]

Missiles are weapons:

\[\text{Missile}(x) \Rightarrow \text{Weapon}(x)\]
An enemy of America counts as "hostile":

\[ \text{Enemy}(x, \text{America}) \implies \text{Hostile}(x) \]

West, who is American

\[ \text{American}(\text{West}) \]

The country Nono, an enemy of America ...

\[ \text{Enemy}(\text{Nono}, \text{America}) \]

Can be converted to CNF

Query: Criminal(\text{West})?

\[ \neg \text{American}(x) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(x, y, z) \lor \neg \text{Hostile}(z) \lor \text{Criminal}(x) \rightarrow \neg \text{Criminal}(\text{West}) \]

\[ \neg \text{American}(\text{West}) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West}, y, z) \lor \neg \text{Hostile}(z) \]

\[ \neg \text{Missile}(x) \lor \neg \text{Weapon}(x) \]

\[ \neg \text{Missile}(y) \lor \neg \text{Sells}(\text{West}, y, z) \lor \neg \text{Hostile}(z) \]

\[ \neg \text{Missile}(x) \lor \neg \text{Owens}(\text{Nono}, y) \lor \text{Sells}(\text{West}, x, \text{Nono}) \]

\[ \neg \text{Sells}(\text{West}, x, \text{Nono}) \lor \neg \text{Hostile}(z) \]

\[ \neg \text{Missile}(y) \lor \neg \text{Owens}(\text{Nono}, y) \lor \neg \text{Hostile}(z) \]

\[ \neg \text{Missile}(y) \lor \neg \text{Owens}(\text{Nono}, y) \lor \neg \text{Hostile}(z) \]

\[ \neg \text{Owens}(\text{Nono}, x) \lor \neg \text{Hostile}(\text{Nono}) \]

\[ \neg \text{Missile}(y) \lor \neg \text{Owens}(\text{Nono}, y) \lor \neg \text{Hostile}(\text{Nono}) \]

\[ \neg \text{Hostile}(\text{Nono}) \]

\[ \neg \text{Hostile}(\text{Nono}) \]

\[ \neg \text{Hostile}(\text{Nono}) \]

\[ \neg \text{Hostile}(\text{Nono}) \]

\[ \neg \text{Hostile}(\text{Nono}) \]
STRUCTURED REPRESENTATION OF KNOWLEDGE

Representing knowledge using logical formalism, like predicate logic, has several advantages. They can be combined with powerful inference mechanisms like resolution, which makes reasoning with facts easy. But using logical formalism complex structures of the world, objects and their relationships, events, sequences of events etc. can not be described easily.

A good system for the representation of structured knowledge in a particular domain should posses the following four properties:

(i) Representational Adequacy:- The ability to represent all kinds of knowledge that are needed in that domain.

(ii) Inferential Adequacy :- The ability to manipulate the represented structure and infer new structures.

(iii) Inferential Efficiency:- The ability to incorporate additional information into the knowledge structure that will aid the inference mechanisms.

(iv) Acquisitional Efficiency :- The ability to acquire new information easily, either by direct insertion or by program control.

The techniques that have been developed in AI systems to accomplish these objectives fall under two categories:

1. Declarative Methods:- In these knowledge is represented as static collection of facts which are manipulated by general procedures. Here the facts need to be stored only one and they can be used in any number of ways. Facts can be easily added to declarative systems without changing the general procedures.

2. Procedural Method:- In these knowledge is represented as procedures. Default reasoning and probabilistic reasoning are examples of procedural methods. In these, heuristic knowledge of “How to do things efficiently “can be easily represented.

In practice most of the knowledge representation employ a combination of both. Most of the knowledge representation structures have been developed to handle programs that handle natural
language input. One of the reasons that knowledge structures are so important is that they provide a way to represent information about commonly occurring patterns of things. Such descriptions are sometimes called schema. One definition of schema is

“Schema refers to an active organization of the past reactions, or of past experience, which must always be supposed to be operating in any well adapted organic response”.

By using schemas, people as well as programs can exploit the fact that the real world is not random. There are several types of schemas that have proved useful in AI programs. They include

(i) **Frames:** Used to describe a collection of attributes that a given object possesses (e.g. description of a chair).

(ii) **Scripts:** Used to describe common sequence of events (e.g. a restaurant scene).

(iii) **Stereotypes:** Used to describe characteristics of people.

(iv) **Rule models:** Used to describe common features shared among a set of rules in a production system.

Frames and scripts are used very extensively in a variety of AI programs. Before selecting any specific knowledge representation structure, the following issues have to be considered.

(i) The basis properties of objects, if any, which are common to every problem domain must be identified and handled appropriately.

(ii) The entire knowledge should be represented as a good set of primitives.

(iii) Mechanisms must be devised to access relevant parts in a large knowledge base.