# 1 Logic & AI: Propositional Logic

Search Algorithms only evaluate states, but do not have an "understanding" of the environment. This does mean, that a goal might not even be able to exist logically, but the search algorithms will still search for it.

**Propositional Logic** aims to improve on that aspect.

## 1.1 Logic

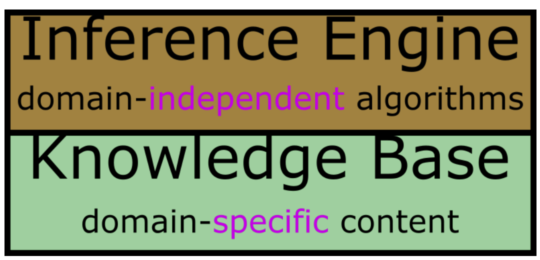
Logic is the key behind any formal knowledge. It allows to filter neccessary information from a set of information and to draw conclusions. In AI, any representation of knowledge is based on logic.

**Knowledge Base (KB)**

A knowledge base represents actual facts which exist in the real world. It is a central component of any knowledge-based agent. It is a collection of "sentences" in a formal language which describe the information related to the world.

**Inference Engine**

The inference engine is responsible for inferring new knowledge from the knowledge base. It is a central component of any knowledge-based agent.




**Knowledge-Based Agents**

A knowledge-based agent is a type of **intelligent agent** that uses a knowledge base and an inference engine to make decisions.

```
kb; // The knowledge base                t; // counter, indicating time
Function knowledge_based_agent(percept):
    tell(kb, make_percept_sentence(percept,t));
    action = ask(kb, make_action_query(t));
    tell(kb, make_percept_sentence(action,t)) t++;
    return action
```

- Represent states, actions...
- Incorporate new percepts and update knowledge base
- Deduce properties of the world and make decisions / actions

## 1.2 Syntax

A sentence in propositional logic follows the **Backus-Naur Form (BNF)**:

| | | |
|---|---|---|
| **Symbol:** | P, Q, R,... | Descriptor of a sentence |
| **Sentence:** | True \| False \| Symbol \|<br>$\neg$(Sentence) \|<br>(Sentence $\wedge$ Sentence) \|<br>(Sentence $\vee$ Sentence) \|<br>(Sentence $\Rightarrow$ Sentence) | Logical implication of a sentence |

## 1.3 Semantics

**Interpretation** specifies which symbols are true and which are false. Given a interpretation it should be possible to evaluate a sentence.

A truth table defines semantics of operators:

| $a$ | $b$ | $\neg a$ | $a \wedge b$ | $a \vee b$ | $a \Rightarrow b$ |
|---|---|---|---|---|---|
| false | false | true | false | false | true |
| false | true | true | false | true | true |
| true | false | false | false | true | false |
| true | true | false | true | true | true |

## 1.4 Tautology

A tautolgy is a sentence that is true for all possible interpretations.

| $P$ | $Q$ | $P \vee Q$ | $\neg P \wedge \neg Q$ | **(P $\vee$Q) $\vee$($\neg$P $\wedge\neg$Q)** |
|---|---|---|---|---|
| false | false | false | true | true |
| false | true | true | false | true |
| true | false | true | false | true |
| true | true | true | false | true |

## 1.5 Logical Equivalence

Two sentences are **logically equivalent** if they have the same truth value for every setting of their propositional variables.

| $P$ | $Q$ | **P$\vee$Q** | **$\neg$($\neg$P$\wedge\neg$Q)** |
|---|---|---|---|
| false | false | false | false |
| false | true | true | true |
| true | false | true | true |
| true | true | true | true |

| **Logical Law** | **Equivalence** |
|---|---|
| Commutativity | $(a \vee b) \equiv (b \vee a)$<br>$(a \wedge b) \equiv (b \wedge a)$ |
| Associativity | $((a \wedge b) \wedge c) \equiv (a \wedge (b \wedge c))$<br>$((a \vee b) \vee c) \equiv (a \vee (b \vee c))$ |
| Double Negation Elimination | $\neg(\neg a) \equiv a$ |
| Contraposition | $(a \Rightarrow b) \equiv (\neg b \Rightarrow \neg a)$ |
| Implication Elimination | $(a \Rightarrow b) \equiv (\neg a \vee b)$ |
| De Morgan's Laws | $\neg(a \wedge b) \equiv (\neg a \vee \neg b)$<br>$\neg(a \vee b) \equiv (\neg a \wedge \neg b)$ |
| Distributivity | $(a \wedge (b \vee c)) \equiv ((a \wedge b) \vee (a \wedge c))$<br>$(a \vee (b \wedge c)) \equiv ((a \vee b) \wedge (a \vee c))$ |

## 1.6 Inference / Entailment

A sentence is **entailed** by the knowledge base if, for every setting of the propositional variables, for which knowledge base is true, the sentence is also true.

Assume 2 sentences, $A$ and $A \Rightarrow B$:

| A | B | Knowledge base |
|---|---|---|
| false | false | false |
| false | true | false |
| true | false | false |
| true | true | true |

To find out whether a sentence $A$ is entailed by knowledge base as simple algorithm can be used:

**Basic Idea:**

1. Go through all possible setting of the propositional variables
2. If knowledge base is true and $A$ is false $\Rightarrow$ return false
3. Else $\Rightarrow$ return true

**Problem:** Not very efficient: The number of setting increases with $2^{\#\text{ propositional variables}}$

### 1.6.1 Principle of Non-Contradiction

"A cannot be ¬ A"

Two contradictionary statements cannot be true at the same time, as that would mean that anything could be true.

**Example:**

PetIsABird $\Rightarrow$ PetCanFly
PetIsAPenguin $\Rightarrow$ PetIsABird
PetIsAPenguin $\Rightarrow$ ¬(PetCanFly)
PetIsAPenguin

This would imply that a penguin can both fly and not fly. If you would work with this contradictionary predicate it could imply anything like:

PetCanFly ∨ MoonMadeOfCheese ≡ True

## 1.7 Conjunctive Normal Form (CNF)

The CNF is a way to write any knowledge base as a single formula:

**CNF Formula**

$$(\cdots \lor \cdots \lor \dots) \land (\cdots \lor \cdots \lor \dots) \land \dots$$

- Can be a symbol x or ¬(x) (**Literals**)
- Multiple fats in knowledge base are "AND"ed together

**Example:** RoommateWet $\Rightarrow$ (RoommateWetOfRain ∨ RoommateWetOfSprinklers)
becomes
(¬(RoommateWet) ∨ RoommateWetOfRain ∨ RoommateWetOfSprinklers)

## 1.8 Modus Ponens

Modus Ponens allows to form new sentences from existing ones:

Assume two sentences, $A$ and $A \Rightarrow B$: From this we can conclude the new sentence $B$.

### 1.8.1 Unit Resolution

Assume the sentences $l_1 \vee l_2 \vee \dots \vee l_k$ and $\neg(l_i)$.

From this we can conclude the new sentence: $l_1 \vee l_2 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k$

### 1.8.2 General Resolution

Assume two sentences $l_1 \vee l_2 \vee \dots \vee l_k$ and $m_1 \vee m_2 \vee \dots \vee m_n$ where for some $i, j$ $l_i = \neg(m_j)$.

From this we can conclude the new sentence: $l_1 \vee l_2 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee m_2 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n$

The same literal may appear multiple times; these need to be removed.

## 1.9 Resolution

**Satisfiable**

There exists a model that makes the modified knowledge base true, i.e., the modified knowledge base is consistent.

To see if a knowledge base is satisfiable, one can use a resolution algorithm.

### 1.9.1 Resolution Algorithm

**Basic Idea:** CNF formula for modified knowledge base is satisfiable if and only if sentence $A$ is **not entailed**. So to see if a sentence $A$ is entailed we can simply add ¬A to the knowledge base and see if it becomes inconsistent.

1. **Find** two clauses with complementary literals
2. **Apply** resolution
3. **Add** resulting clause (if not already there)
4. **Test**, if it results in the empty clause → formula is not satisfiable

#### Special Case: Horn Clauses

**Horn Clauses**

Horn clauses are implications with only positive (no negations) literals:

$$X_1 \wedge X_2 \wedge X_4 \Rightarrow X_3 \wedge X_6$$
$$\text{True} \Rightarrow X_1$$

To find out whether a literal $X_j$ is entailed:

1. Start from known imlications as far as possible
2. If $X_j$ is reachable it is entailed

To increase efficiency of this approach we can maintain a count of how many implications are already known to reduce the neccessary computations.

## 1.10 Limitations of Propositional Logic

- No notion of objects or relations:
  - Identifiers are merely suggestive, it does not neccessarily mean that the implied objects and relations actually exist or are real.
  - To this end, every identifier might as well be a single letter $A, B \dots$