Logical inference problem:

- Given a knowledge base KB (a set of sentences) and a sentence $\alpha$, does the KB semantically entail $\alpha$?

$$KB \models \alpha$$

In other words: In all interpretations in which sentences in the KB are true, is also $\alpha$ true?

Logical inference problem in the first-order logic is **undecidable** !!! No procedure that can decide the entailment for all possible input sentences in finite number of steps.
Knowledge-based system

- **Knowledge base:**
  - A set of sentences that describe the world in some formal (representational) language (e.g. first-order logic)
  - Domain specific knowledge
- **Inference engine:**
  - A set of procedures that work upon the representational language and can infer new facts or answer KB queries (e.g. resolution algorithm, forward chaining)
  - Domain independent

Retrieval of KB information

- The reasoning algorithms operating upon the KB need to access and manipulate information stored there
  - Large KBs consist of thousands of sentences
- **Problem:** retrieval of sentences from the KB (e.g. for the purpose of unification)
  - Simple flat list of conjuncts can be very long and searching it exhaustively is inefficient
- **Solution:** indexing
  - Store and maintain the sentences in a table (hash table) according to predicate symbols they include
Table-based indexing of KBs

Assume the knowledge is expressed in the implicational form, with sentences corresponding to facts and rules

- For each predicate we store its:
  - positive literals
  - negative literals,
  - rules in which it occurs in the premise,
  - rules in which it occurs in the conclusion.

<table>
<thead>
<tr>
<th>Key</th>
<th>Positive</th>
<th>Negative</th>
<th>Conclusion</th>
<th>Premise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brother</td>
<td>Brother(Richard, John)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Brother(Ted, Jack)</td>
<td>¬Brother(Ann, Sam)</td>
<td>Brother(x, y) ∨ Male(y) ⇒ Brother(y, z)</td>
<td>Brother(x, z) ∨ Male(y) ⇒ Brother(y, z)</td>
</tr>
<tr>
<td>Male</td>
<td>Male(Ted)</td>
<td>¬Male(Ann)</td>
<td>Brother(x, y) ⇒ Male(y)</td>
<td>Brother(x, z) ∨ Male(y) ⇒ Brother(y, z)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Indexing and retrieval of KB information

**Problem:** the number of elements (clauses) with the same predicate can be very large!!

**Solution:** tree-based indexing (a form of combined indexing)

- structure the KB further, create tables for different symbols that occur in the predicate
Indexing of information in KBs

Problem: matching of sentences with variables
- Too many entries need to be searched and this even if the resulting set is small
Assume: Taxpayer(SSN, zipCode, net_income, dependents)
We want to match e.g.: Taxpayer(x, 15260, y, 5)
- Partial solution: cross-indexing
- Create more special tables combining predicates and arguments
e.g. have a table for: Taxpayer+zip_code+num_dependents
- Choose and search the most promising table for retrieval
- No universal solution for all possible matchings, since all the number of all tables would go up exponentially

Automated reasoning systems

Categories and main differences:
- **Theorem provers**
  – Prove sentences expressed in FOL
- **Deductive retrieval systems**
  – Systems based on rules (KBs in Horn form)
  – Prove theorems or infer new assertions (forward, backward chaining)
- **Production systems**
  – Systems based on rules with actions in antecedents
  – Forward chaining mode of operation
- **Semantic networks**
  – Graphical representation of the world, objects are represented as nodes in the graphs, relations as links
Production systems

Rule-based system, modus ponens is the main rule of inference
The knowledge base is divided into:
- rule memory (includes rules)
- working memory (includes facts)

A special type of if – then rule
\[ p_1 \land p_2 \land \ldots \land p_n \Rightarrow a_1, a_2, \ldots, a_k \]
- Antecedent: a conjunction of literal (facts, statements in predicate logic)
- Consequent: a conjunction of actions. An action can:
  - ADD the fact to the KB (working memory)
  - REMOVE the fact from the KB
  - QUERY the user, PRINTING a value, etc …

Use forward chaining to do reasoning:
- If the antecedent of the rule is satisfied (rule is said to be “active”) then its consequent can be executed (it is “fired”)

Problem: Two or more rules are active at the same time. Which one to execute next?

Strategy for selecting the rule to be fired from among possible candidates is called conflict resolution

Why do we care about the order?
- action of R2 can delete one of the preconditions of R15 and deactivate R15
- Note: this is not a problem in Horn KB (no deletions)
Production systems

- **Problems with production systems:**
  - Additions and Deletions can change a set of active rules;
  - If a rule contains variables testing all instances in which the rule is active may require a large number of unifications.
  - Conditions of many rules may overlap, thus requiring to repeat the same unifications multiple times.

- **Solution: Rete (Net) algorithm**
  - gives more efficient solution for managing a set of active rules and performing unifications
  - Implemented in the system **OPS-5** (used to implement XCON – an expert system for configuration of DEC computers)

Rete (Net) algorithm

- Assume a set of rules:
  \[
  A(x) \land B(x) \land C(y) \Rightarrow add \ D(x)
  \]
  \[
  A(x) \land B(y) \land D(x) \Rightarrow add \ E(x)
  \]
  \[
  A(x) \land B(x) \land E(z) \Rightarrow delete \ A(x)
  \]
- And facts: \(A(1), A(2), B(2), B(3), B(4), C(5)\)
- Rete:
  - Compiles the rules to a network that merges conditions of multiple rules together (avoids repeats)
  - Propagates valid unifications
  - Re-evaluates only changed conditions
**Rete algorithm. Network.**

```
A(1), A(2) ⊃ D
B(2), B(3), B(4) ⊃ A = B
A(2), B(2) ⊃ G
G(5) ⊃ add D
E ⊃ delete A
```

**Rules:**
- \( A(x) \land B(x) \land C(y) \Rightarrow \text{add } D(x) \)
- \( A(x) \land B(y) \land D(x) \Rightarrow \text{add } E(x) \)
- \( A(x) \land B(x) \land E(z) \Rightarrow \text{delete } A(x) \)

**Facts:**
- \( A(1), A(2), B(2), B(3), B(4), C(5) \)

---

**Conflict resolution strategies**

- **Problem:** Two or more rules are active at the same time. Which one to execute next?

- **Strategies:**
  - **No duplication.** Do not execute the same rule twice.
  - **Recency.** Rules referring to facts newly added to the working memory take precedence.
  - **Specificity.** Rules that are more specific are preferred.
  - **Priority levels.** Define priority of rules, actions based on expert opinion. Have multiple priority levels such that the higher priority rules fire first.
Semantic network systems

- Knowledge about the world described in terms of graphs. Nodes correspond to:
  - Concepts or objects in the domain.

  Links correspond to relations. Three kinds:
  - Subset links (isa, part-of links)
  - Member links (instance links)
  - Function links.

- Can be transformed to the first-order logic language
- Graphical representation is often easier to work with
  - better overall view on individual concepts and relations

Semantic network. Example.

Inferred properties:

Queen Mary is a ship
Queen Mary has a boiler