Logical reasoning systems

Knowledge-based system

Knowledge base
Inference engine

- **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 implicative 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.

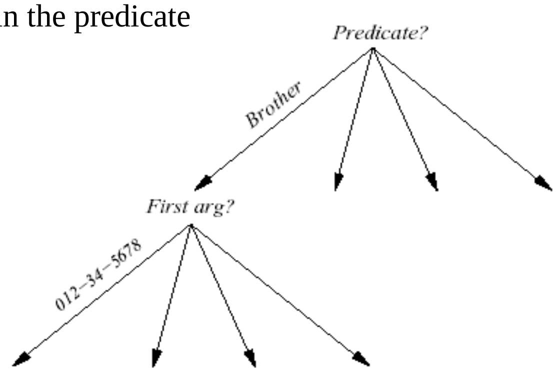
Key	Positive	Negative	Conclusion	Premise
Brother	*Brother(Richard,John)* *Brother(Ted,Jack)* *Brother(Jack,Bobbie)*	$\neg$*Brother(Ann,Sam)*	$Brother(x,y) \wedge Male(y) \Rightarrow Brother(y,x)$	$Brother(x,y) \wedge Male(y) \Rightarrow Brother(y,x)$ $Brother(x,y) \Rightarrow Male(x)$
Male	*Male(Jack)* *Male(Ted)* ...	$\neg$*Male(Ann)* ...	$Brother(x,y) \Rightarrow Male(x)$	$Brother(x,y) \wedge Male(y) \Rightarrow Brother(y,x)$

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)**

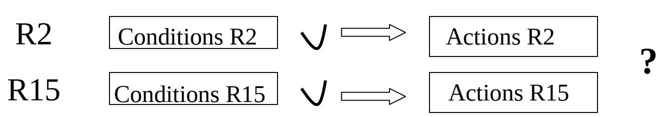
A special type of if – then rule

$$p_1 \land p_2 \land \dots \; p_n \Rightarrow a_1, a_2, \dots, a_k$$

- **Antecedent:** a conjuction 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 …

Production systems

- 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)

CIS587 - Artificial Intelligence

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 (N e t) 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:

$$A(x) \wedge B(x) \wedge C(y) \Rightarrow add \;\; D(x)$$

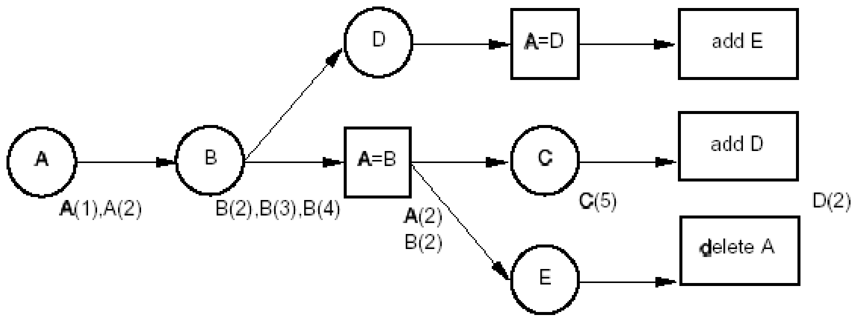
$$A(x) \wedge B(y) \wedge D(x) \Rightarrow add \;\; E(x)$$

$$A(x) \wedge B(x) \wedge E(z) \Rightarrow delete \;\; A(x)$$

- 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

CIS587 - Artificial Intelligence

Rete algorithm. Network.



Rules:

$$A(x) \wedge B(x) \wedge C(y) \Rightarrow add \;\; D(x)$$
$$A(x) \wedge B(y) \wedge D(x) \Rightarrow add \;\; E(x)$$
$$A(x) \wedge B(x) \wedge E(z) \Rightarrow delete \;\; A(x)$$

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.

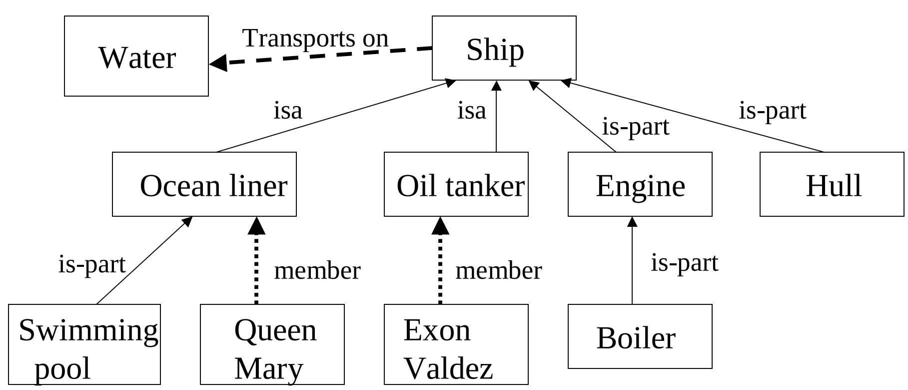
Semantic network systems

- 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) } Inheritance relation links
  - **Member links** (instance links) } Inheritance relation 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