Chapter 5b

I. Problem-reduction representations and AND/OR graphs

We already know about the divide and conquer strategy, a solution to a problem can be obtained by decomposing it into smaller sub-problems. Each of this sub-problem can then be solved to get its sub solution. These sub solutions can then be recombined to get a solution as a whole. That is called is **Problem Reduction**. This method generates arc which is called as **AND** arcs. One AND arc may point to any number of successor nodes, all of which must be solved for an arc to point to a solution.

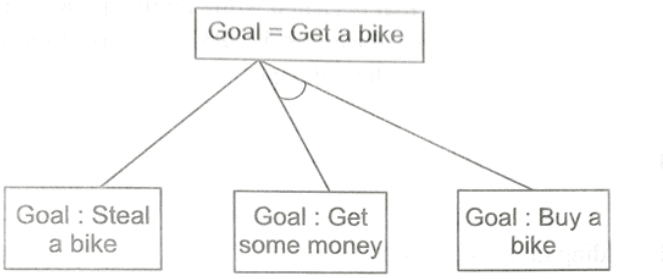

Fig: AND / OR Graph

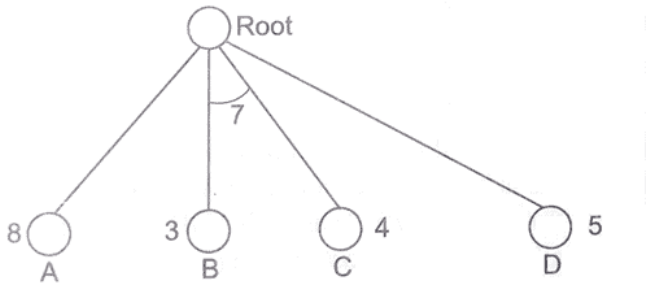

Fig: AND / OR Tree

II. Knowledge representation techniques

Knowledge Representation in AI describes the representation of knowledge. Basically, it is a study of how the **beliefs, intentions**, and **judgments** of an **intelligent agent** can be expressed suitably for automated reasoning. One of the primary purposes of Knowledge Representation includes modeling intelligent behavior for an agent.

Knowledge Representation and Reasoning (**KR, KRR**) represents information from the real world for a computer to understand and then utilize this knowledge to solve **complex real-life problems** like communicating with human beings in natural language. Knowledge representation in AI is not just about storing data in a database, it allows a machine to learn from that knowledge and behave intelligently like a human being.

The different kinds of knowledge that need to be represented in AI include:

- **Objects**
- **Events**
- **Performance**
- **Facts**
- **Meta-Knowledge**
- **Knowledge-base**

Now that you know about Knowledge representation in AI, let's move on and know about the different types of Knowledge.

More: https://www.javatpoint.com/knowledge-representation-in-ai

III. Uncertainty management (fuzzy logic)

https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_fuzzy_logic_systems.htm

IV. The resolution principle

Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions. It was invented by a Mathematician John Alan Robinson in the year 1965.

Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the **conjunctive normal form or clausal form**.

Clause: Disjunction of literals (an atomic sentence) is called a **clause**. It is also known as a unit clause.

Conjunctive Normal Form: A sentence represented as a conjunction of clauses is said to be **conjunctive normal form** or **CNF**

V. Logic Programming and SLD resolution

Logic programming is a programming paradigm which is largely based on formal logic. Any program written in a logic programming language is a set of sentences in logical form, expressing facts and rules about some problem domain. Major logic programming language families include Prolog, answer set programming (ASP) and Datalog. In all of these languages, rules are written in the form of *clauses*:

```
H :- B1, …, Bn.
```

and are read declaratively as logical implications:

```
H if B1 and … and Bn.
```

`H` is called the *head* of the rule and `B1`, ..., `Bn` is called the *body*. Facts are rules that have no body, and are written in the simplified form:

```
H.
```

In the simplest case in which `H`, `B1`, ..., `Bn` are all atomic formulae, these clauses are called definite clauses or Horn clauses. However, there are many extensions of this simple case, the most important one being the case in which conditions in the body of a clause can also be negations of atomic formulas. Logic programming languages that include this extension have the knowledge representation capabilities of a non-monotonic logic.

SLD Resolution

https://en.m.wikipedia.org/wiki/SLD_resolution

http://www.ale.cs.toronto.edu/docs/ref/ale_trale_ref/ale_trale_ref-node44.html

VI. Basic techniques in Logic Programming

https://en.wikipedia.org/wiki/Logic_programming#Prolog