WEEK 5 ARTIFICIAL INTELLIGENCE

LECTURE 21

This file, "Lecture 21: Expert Systems," defines and describes expert systems (ES), which are computer programs designed to mimic human expertise in specific domains by applying inference methods to a body of knowledge.

Key aspects of expert systems covered in the lecture include:

- **Definition:** Various definitions highlight that ES aim to solve complex problems typically requiring human expertise, make reasoned judgments, or provide assistance in areas where human skills might be fallible or scarce.
- **Applications:** Examples of ES applications include medical diagnosis (e.g., arrhythmia recognition, early melanoma diagnosis, breast cancer diagnosis), computer configuration, and machine fault diagnosis.
- **Characteristics:** ES can explain their reasoning, display intelligent behavior, draw conclusions from complex relationships, and provide portable knowledge.
- **When to Use ES:** They are beneficial for tasks with high potential payoff, capturing irreplaceable human expertise, providing expertise in multiple locations or hostile environments, and sharing knowledge for training and development.
- **Components:** The core components of an expert system are:
  - **Knowledge base:** Stores relevant information, data, rules, cases, and relationships (often as facts and rules).
  - **Inference engine:** Seeks information and relationships from the knowledge base to provide answers, predictions, and suggestions.
  - **User interface:** Facilitates interaction with end-users and helps in knowledge base development.
  - **Explanation facility:** Explains the system's reasoning.
  - **Working memory:** A database of facts used by the rules.
  - **Agenda:** A prioritized list of rules created by the inference engine.
  - **Knowledge acquisition facility:** An automatic way for users to enter knowledge.
- **Types of Reasoning:** The file briefly mentions fuzzy logic, backward chaining (working backward from conclusions), and forward chaining (working forward from facts).
- **Knowledge Engineering:** This is the process of acquiring domain knowledge and building it into the knowledge base. It involves knowledge acquisition (transferring knowledge from humans or the environment to computers), knowledge representation, and inference. The **Knowledge Engineer** mediates between the human expert and the knowledge base, eliciting and representing knowledge.
- **Components of Knowledge:** Knowledge in ES includes facts, procedural rules (well-defined sequences of events), and heuristic rules (rules of thumb gained by experience).
- **Early Expert Systems:** The lecture lists several historical expert systems like DENDRAL (chemical constituents), MYCIN (infectious diseases), PROSPECTOR (mineral diagnosis), and XCON/R1 (computer configuration).

LECTURE 22

This file, "lecture22.pdf," discusses Expert Applications Systems, with a particular focus on MYCIN.

Here's a summary of the key points:

- **MYCIN (1972-1980):** An early expert system for blood disease diagnosis, using 50-500 rules acquired from experts. It used "certainty factors" (CF) to handle uncertainty, defined by a measure of belief (MB) and a measure of disbelief (MD).
- **Rule Format:** Rules in MYCIN were typically "if-then" statements, with examples provided in both human-readable and MYCIN formats.
- **Advantages of Expert Systems:**
  - **Economical:** Lower cost per user.
  - **Availability:** Accessible anytime, anywhere.
  - **Response Time:** Often faster than human experts.
  - **Reliability:** Logical deduction from existing facts with no distraction, fatigue, or emotional involvement.
  - **Explanation:** Can show reasoning steps.
  - **Intellectual Property & Permanence:** Knowledge is retained even if human experts leave.
  - **Multiple Experts:** Can draw from more knowledge and prevent skewed decision-making.
  - **Other benefits:** No emotion, high efficiency, expertise in a domain, no memory limitation, regular updates, high security, considers all facts.
- **Limitations/Disadvantages of Expert Systems:**
  - Not widely used or tested, limited to narrow problems.
  - Cannot deal with "mixed" knowledge or refine their own knowledge base.
  - Possibility of error, difficult to maintain, high development costs.
  - Raise legal and ethical concerns.
  - **Limited Knowledge:** "shallow" knowledge, no "deep" or "common-sense" understanding, "closed world" (only knows what it's told).
  - **Mechanical Reasoning:** May not select the most appropriate method, some "easy" problems are computationally expensive.
  - **Lack of Trust:** Users may be hesitant to rely on machines for critical decisions.
- **Areas of Application:** Financial services, mechanical engineering, telecommunications, healthcare (medical diagnoses), agriculture (crop damage), customer service, transportation, and law. Specific applications include credit granting, information management, embedded AI, plant layout, hospitals, help desks, employee performance evaluation, loan analysis, virus detection, repair, shipping, marketing, and warehouse optimization.
- **Summary of Components:** Expert systems or knowledge-based systems use if-then rules and primarily consist of a knowledge base and an inference engine. They can be more reliable than humans but are often difficult and expensive to develop.

LECTURE 23
This document, "lecture23.pdf," introduces planning in artificial intelligence, focusing on how a sequence of actions can be found to accomplish a specific task.

Key concepts covered include:

- **PLANNER:** The process of searching through a space of actions to find a sequence to achieve a task, highlighting the complexity of world states and the "frame problem" (characterizing what changes and what doesn't with an action).

- **Planning Definition:** Planning is defined as a series of actions to achieve a goal, often using the Planning Domain Definition Language (PDDL). States are represented by conjunctions of predicates, and action schemas include an action, preconditions, and effects.
- **Defining a Planning Problem:** A planning problem is defined by a set of action schemas (the planning domain), an initial state, and a goal state, both represented by conjunctions of predicates.
- **Examples:** The document provides detailed examples of planning problems, including:
    - **Air Cargo Transportation:** Loading, unloading, and flying cargo between airports.
    - **The Spare Tire Problem:** Changing a flat tire.
    - **The Blocks World:** Building a three-block tower, illustrating predicates like `on(X,Y)`, `clear(X)`, `gripping(X)`, `gripping()`, and `ontable(X)`.
- **Actions and Rules in Blocks World:** Specific rules and actions (pickup, putdown, stack, unstack) are defined for the blocks world problem.
- **STRIPS:** This system describes operators as triples: preconditions (P), an add list (A) of new state descriptions, and a delete list (D) of state descriptions to remove.
- **Planning in Situation Calculus:** A method using First-Order Logic (FOL) to describe change as a sequence of situations, with predicates having an extra situation argument.
- **Forward and Backward Search:**
    - **Forward state-space search (Progression search):** Starts from the initial state and searches forward to the goal. It can be prone to exploring irrelevant actions due to large state spaces.
    - **Backward relevant space search (Regression search):** Starts from the goal and applies actions backward, only considering actions relevant to the goal.

The document uses figures and examples from "Artificial Intelligence: A Modern Approach" by Russel and Norvig.V
LECTURE 24
This document, "Lecture 24: Planning graph," by V. Susheela Devi, covers the concept of planning graphs (PGs) in artificial intelligence.

Here's a summary of the key points:

- **Partial Order Planner (POP):** The lecture briefly introduces POPs as a regression planner that starts with goals and works backward, extending a minimal plan with start and finish states by achieving preconditions.
- **Planning Graph (PG) Structure:** A PG is a directed graph organized into alternating levels of states (S_i) and actions (A_i). S_0 is the initial state. S_i contains all literals that could hold at time 'i', and A_i contains all actions whose preconditions are satisfied at time 'i'. PGs are designed for propositional planning problems (no variables).
- **Observations in PGs:**
    - Actions connect preconditions in S_i to effects in S_{i+1}.
    - Literals appear due to actions or persistence actions (represented by small squares), which indicate a literal persists if not negated.
- **Mutex Links:** PGs include mutual exclusion (mutex) links, shown as gray lines, which indicate actions or literals that cannot occur together.
    - **Action Mutex Conditions:** Inconsistent effects (one action negates another's

    effect), interference (one action's effect negates another's precondition), and competing needs (preconditions of two actions are mutually exclusive).
  - **Literal Mutex Conditions:** One literal is the negation of another, or each possible pair of actions achieving the two literals is mutually exclusive (inconsistent support).
- **Estimation from PGs:** PGs provide information for solving planning problems.
  - If a goal literal never appears, the problem is unsolvable.
  - **Level cost:** The level at which a goal literal first appears.
  - **Heuristics for conjunctions of literals:** Max-level, level-sum, and set-level heuristics (finds level where all goal literals appear without being mutex).
- **GRAPHPLAN Algorithm:** This algorithm extracts a plan from the PG by iteratively expanding the graph until a solution is found or no solution is possible.
- **Disadvantages of PGs:**
  - Limited to propositional planning.
  - Approximation of the search space, potentially leading to suboptimal plans.
  - Heavily reliant on heuristics for effectiveness.
  - Can be complex and computationally intensive to build and maintain.
  - Less suitable for dynamic environments.
  - Do not guarantee optimal solutions.
- **Advantages of PGs:**
  - Provide a structured representation that is easier to understand and manipulate.
  - Improved performance through the use of derived heuristics.
  - Applications in robotics, autonomous vehicles, logistics, resource management, and game playing.
- **Planning in the Real World:** The lecture briefly touches on additional concepts required for real-world planning, including hierarchical plans, complex conditions (e.g., universal quantification), time (deadlines, durations, time windows), and resources (budget, workers, resource consumption/generation).

LECTURE 25
This document, "lecture25.pdf," focuses on planning in real-world scenarios, particularly in the context of Artificial Intelligence.

Here's a summary of the key topics:

- **Planning and Scheduling:** Discusses the two phases: planning (choosing actions with order constraints) and scheduling (adding temporal and resource information). It introduces job shop scheduling problems with examples like assembling cars, considering duration and resource constraints.
- **Solving Scheduling Problems:** Explains the Critical Path Method (CPM) to determine earliest and latest start times (ES and LS) for actions, defining slack and critical path.
- **Hierarchical Planning (HTN Planning):** Introduces the concept of hierarchical decomposition to manage complexity. It distinguishes between primitive tasks and High-Level Actions (HLAs), which can be refined into a series of other actions. The document explains how HTN planning starts with a top-level action and repeatedly refines HLAs until a primitive plan is achieved, emphasizing its efficiency and human readability due to its hierarchical nature. It also mentions the "Downward Refinement Property."

- **Contingent Planning:** Deals with plans that include conditional branching based on observations, suitable for partially observable and non-deterministic environments. An example involving painting a chair and table is provided.
- **Conditional Planning:** Focuses on handling incomplete information by constructing plans that account for all possible situations, using sensing actions to check conditions. It also contrasts this with execution monitoring, which defers dealing with conditions until they arise.
- **Replanning:** Explains the need for new plans when initial plans fail or the world model is incorrect. It outlines three modes of replanning: action monitoring, plan monitoring, and goal monitoring, and discusses limitations such as dead-ends or unknown preconditions.
- **Issues Addressed by Planners in Real Domains:** Summarizes how planners use explicit domain descriptions, problem decomposition, handle conditional effects, universally quantified objects, errors in domain descriptions, sensing for more information, and adapt to dynamically changing environments.
- **Multiagent Planning:** Explores scenarios with multiple agents having their own goals, which can lead to cooperation or hindrance. It defines multieffector and multibody planning, and uses a doubles tennis problem as an example to illustrate coordination challenges.