Quick Recap

- Logical reasoning:
  - Syntax and semantics
    - Propositional logic
    - First-order logic
  - Proofs and resolution: Truth tables, inference rules, resolution
  - Applications

- Planning:
  - STRIPS notation for describing planning tasks.
  - State-space planning.
  - Plan-space planning.
Successes in Rule-Based Reasoning

- Expert systems
- Dendral (Buchanan et al., 1969)
- MYCIN (Feigenbaum, Buchanan, Shortliffe)
- PROSPECTOR (Duda et al., 1979)
- R1 (McDermott, 1982)

Material for next few slides on expert systems taken from:

DENRAL: Mass Spectrometry

- Infers molecular structure from the information provided by a mass spectrometer
- Generate-and-test method

\[
\text{if there are peaks at } x_1 \text{ and } x_2 \text{ s.t.}
\]
\[
x_1 + x_2 = M + 28
\]
\[
x_1 - 28 \text{ is a high peak}
\]
\[
x_2 - 28 \text{ is a high peak}
\]

At least one of \(x_1\) and \(x_2\) is high

then there is a ketone subgroup
MYCIN

- Diagnosis of blood infections
- 450 rules; performs as well as experts
- Incorporated certainty factors

If: (1) the strain of the organism is gram-positive, and
(2) the morphology of the organism is coccus, and
(3) the growth conformation of the organism is clumps,
then there is suggestive evidence (0.7) that the identity of the organism is staphylococcus.

More successes

• PROSPECTOR (Duda et al., 1979)
  - Correctly recommended exploratory drilling at geological site
  - Rule-based system founded on probability theory
• R1 (McDermott, 1982)
  - Designs configurations of computer components
  - About 10,000 rules
  - If: current context is ?x
    then: deactivate ?x context
    and activate ?y context
Cognitive modeling with rule-based systems

SOAR (Laird, Newell, Rosenbloom, 1987) is a general architecture for building intelligent systems.

- Long term memory consists of rules.
- Working memory = current state.
- All problem solving, incl. deciding what rule to execute, is state space search.
- Successful rule sequences are chunked into new rules.
- Control strategy embodied in terms of meta-rules.

Properties of knowledge-based systems

Advantages
1. Expressibility*: Human readable
2. Simplicity of inference procedures*: Rules/knowledge in same form
3. Modifiability*: Easy to change knowledge
4. Explainability: Answer “how” and “why” questions.
5. Machine readability
6. Parallelism*

Disadvantages
1. Difficulties in expressibility
2. Undesirable interactions among rules
3. Non-transparent behavior
4. Difficult debugging
5. Slow
6. Where does the knowledge base come from???
Other applications of FOL

- Prolog: a logic programming language
- Production systems
- Semantic nets
- Automated theory proving
- **Planning**

What is planning?

- A plan is a collection of actions for performing some task.
  
  *E.g. Assembling your new IKEA desk.*

- That are many programs that help humans formulate plans.

- But it is much more difficult to generate plans automatically.


- Autonav system used for autonomous navigation and finding imaging targets.

- Remote Agent system used to perform automatic fault detection and self-repair.

- First spacecraft to be controlled by AI system without human intervention.

Searching vs Planning

- Consider the task: *get milk, bananas, and a cordless drill*.
- Standard search algorithms seem to fail miserably:

  Why?

- Two main difficulties arise in complex search problems:
  - Branching factor is huge.
  - Difficult to find good heuristic functions.

Planning as search

- Planning problem is described like a search problem (states, operators, goals), but the problem representation is a lot more structured.

<table>
<thead>
<tr>
<th>Search</th>
<th>Planning</th>
</tr>
</thead>
<tbody>
<tr>
<td>States</td>
<td>Data structures</td>
</tr>
<tr>
<td>Actions</td>
<td>Code</td>
</tr>
<tr>
<td>Goal</td>
<td>Goal test</td>
</tr>
<tr>
<td>Plan</td>
<td>Sequence from $S_0$</td>
</tr>
<tr>
<td></td>
<td>Preconditions/outcomes</td>
</tr>
<tr>
<td></td>
<td>Logical sentence (conjunction)</td>
</tr>
<tr>
<td></td>
<td>Constraints on actions</td>
</tr>
</tbody>
</table>
Dimensions of Planning


Shakey: “The first electronic person”

- Designed in 1966, at the Stanford Research Institute.
- Able to plan and execute simple tasks related to navigation and object manipulation.
- Needed a language in which to express planning tasks.
STRIPS (Stanford Research Institute Planning System)

- **Domain**: set of typed objects represented as propositions.
- **States**: represented as first-order predicates over objects.

**Closed-world assumption:**
- everything not stated is false
- only objects in the world are the ones defined.

- **Operators**: defined in terms of:
  - **Preconditions**: when can the action be applied?
  - **Effects**: what happens after the action?
    (No explicit description of how the action should be executed.)

STRIPS representations

- **States** are represented as conjunctions of predicates:
  \[ \text{In(robot, room)} \land \text{Closed(door)} \land \ldots \]

- **Goals** are represented as conjunctions of predicates:
  \[ \text{In(robot, r)} \land \text{In(Charger, r)} \]

- **Operators**:
  - **Name**: \( \text{Go(x, y)} \)
  - **Preconditions** are represented as conjunctions: \( \text{At(robot, x)} \land \text{Path(x, y)} \)
  - **Postconditions** are represented as conjunctions: \( \text{At(robot, y)} \land \neg \text{At(robot, x)} \)

- Variables (e.g. \( x, y, r \)) can be instantiated only with objects of correct type.
STRIPS operator representation

- Operators: \{Name, Preconditions, Effects\}

- Preconditions are conjunctions of literals.

- Effects/Postconditions are represented in terms of:
  - Add-list: list of propositions that become true after the action.
  - Delete-list: list of propositions that become false after the action.

Semantics

- If the precondition is false in a world state:
  - the action does not change anything (since it cannot be applied).

- If the precondition is true:
  - Delete the items on the Delete-list.
  - Add the items on the Add-list.

  Order of operations is important here!

- This is a very restricted language, which means we can do efficient inference.
Example: Buying action

- **Action:**
  
  \( \text{Buy}(x) \) (where \( x \) is a good)

- **Precondition:**
  
  \( \text{At}(s), \text{Sells}(s, x, p), \text{HaveMoney}(p) \) (where \( s \) is a store, \( p \) is a price).

- **Effect:**
  
  - Delete-list: \( \text{HaveMoney}(p) \)
  - Add-list: \( \text{Have}(x) \)

Additional propositions can be added. E.g To show that now the store has the money, the stock has decreased, etc.

Example: Move action

- **Action:**
  
  - \( \text{Move}(\text{object}, \text{from}, \text{to}) \)

- **Preconditions:**
  
  - \( \text{At}(\text{object}, \text{from}), \text{Clear}(\text{to}), \text{Clear}(\text{object}) \)

- **Effects:**
  
  - Delete-list: \( \text{At}(\text{object}, \text{from}), \text{Clear}(\text{to}) \)
  - Add-list: \( \text{At}(\text{object}, \text{to}), \text{Clear}(\text{from}) \)
Welcome to the Blocks World!

Initial state = On(A,table) \land On(B,table) \land On(C,table) \land Clear(A) \land Clear(B) \land Clear(C)

Goal state = On(A,B) \land On(B,C)

Action = Move(b,x,y)
Precondition = On(b,x) \land Clear(b) \land Clear(y)
Effect = On(b,y) \land Clear(x) \land \neg On(b,x) \land \neg Clear(y)

Action = MoveToTable(b,x)
Preconditions = On(b,x) \land Clear(b)
Effect = On(b,Table) \land Clear(x) \land \neg On(b,x)

STRIPS state transitions
Pros and cons of STRIPS?

**Pros:**
- Since it is restricted, inference can be done efficiently.
- All operators can be viewed as simple deletions and additions of propositions to the knowledge base.

**Cons:**
- Assumes that a small number of propositions will change for each action (otherwise operators are hard to write down, and reasoning becomes expensive.)
- Limited language (preconditions and effects are expressed as conjunctions), so not applicable to all domains of interest.

Two basic approaches to planning

- **State-space planning** works at the level of states and operators.
  - Finding a plan is formulated as a search through state space for a path from the start state to the goal state(s).
  - Most similar to constructive search.

- **Plan-space planning** works at the level of plans.
Progression (forward) planning

1. Determine all operators that are applicable in the start state.
2. Ground the operators, by replacing any variables with constants.
3. Choose an operator to apply.
4. Determine the new content of the knowledge base, based on the operator description.

Repeat until goal state is reached.

Example: Supermarket domain

- In the start state we have $\text{At(Home)}$, which allows us to apply operators of the type $\text{Go(x,y)}$.
- The operator can be instantiated as $\text{Go(Home, HardwareStore)}$, $\text{Go(Home, GroceryStore)}$, $\text{Go(Home, School)}$, …
- If we choose to apply $\text{Go(Home, HardwareStore)}$, we will delete from the KB $\text{At(Home)}$ and add $\text{At(HardwareStore)}$.
- The new proposition enables new actions, e.g. $\text{Buy}$

- Note that now there are a lot of possible operators to consider!
Regression planning

- Pick actions that satisfy (some of) the goal propositions.

- Make a new goal, containing the preconditions of these actions, as well as any unsolved goal propositions.

- Repeat until the goal set is satisfied by the start state.

Example: Supermarket domain

- In the goal state we have \( \text{At(Home)} \land \text{Have(Milk)} \land \text{Have(Drill)} \).
- The action \( \text{Buy(Milk)} \) would allow us to achieve \( \text{Have(Milk)} \).
- To apply this action we need to have the precondition \( \text{At(GroceryStore)} \), so we add it to the set of propositions we want to achieve.
- Similarly, we want to achieve \( \text{At(HardwareStore)} \).

Note that in this case, the order in which we try to achieve these propositions matters!
State-space planning

- **Progressive planners** reason from the start space, trying to find the operators that can be applied. (-> match preconditions)

- **Regression planners** reason from the goal state, trying to find the actions that will lead to the goal. (-> match effects)

In both cases, the planners work with **sets of states**, instead of using individual states as in straight-forward search.

Analysis of STRIPS planning

- STRIPS planning is **SOUND**.
  - Only legal plans will be found.

- STRIPS planning is **NOT COMPLETE**.
  - Once a subgoal ordering is selected, no backtracking is allowed.

- STRIPS planning is **NOT OPTIMAL**.
  - No guarantee of finding shortest possible plan.

- STRIPS planning is **EXPENSIVE**. Typically NP-hard.
Another problem: Sussman anomaly

When using a stack of goals (i.e. linear planning), we clear block C, but then may repeatedly put A on top of B and remove it.

Variations of goal regression

- Using a **stack of goals** - this is called **linear planning**.
  - This is **sound but not complete**.
  - We may not find a plan even if one exists.

- Using a **set of goals** - this is called **non-linear planning**.
  - This is **sound and complete**!
  - May be more expensive (need to decide what to work on next.)
Total vs. Partial Order

- **Total order**: Plan is always a strict sequence of actions

  *E.g. To paint the ceiling*

  Plan 1:
  - Start ➔ Get Brush ➔ Get Ladder ➔ Paint ceiling ➔ Finish

  Plan 2:
  - Start ➔ Get Ladder ➔ Get Brush ➔ Paint ceiling ➔ Finish

- **Partial order**: Plan steps may be unordered

Two basic approaches to planning

- **State-space planning** works at the level of states and operators.

- **Plan-space planning** works at the level of plans.
  - Finding a plan is formulated as a search through space of plans.
  - Start with a partial (possibly incorrect) plan, then apply changes to make it a full (correct) plan.
  - Most similar to iterative improvement/repair.
Partial Order Planning (POP)

• Basic idea:
  – Search in plan space and use least commitment whenever possible.

  • In state space search:
    – Search space is a set of states of the world
    – Transitions between states are actions
    – Plan is path through space

  • In plan space search:
    – Search space is a set of partial plans
    – Transitions are plan operators

Important news

• Homework 2 posted. Due Feb. 19.

• Tutorial today, 4:30pm, ENGTR 3120

• Midterm is confirmed for Feb. 23 at the usual class time.
  • Last name A-O go to MCMED 504.
  • Last name P-Z go to MCMED 521.

Optional: Bring 1 page (single-side) of hand-written notes.