An Introduction to AI Planning

Ute Schmid
Applied CS/Cognitive Systems
Bamberg University

← Goals, Methods, Topics of AI
← Basic concepts of AI Planning
← Deductive Planning vs. State-based (Strips) Planning

Remark: Planning is typically introduced in the last third of an introductory AI lecture. Basic knowledge about problem solving (search algorithms) and First Order Logic is presumed.
## AI and Knowledge Based Systems

<table>
<thead>
<tr>
<th>Knowledge Base</th>
<th>Inference Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>→ Facts and Rules</td>
<td>→ Deduction (Theorem Proving),</td>
</tr>
<tr>
<td>→ Representation Formalisms (FOL-based, Production Rules)</td>
<td>→ Non-montonic/probabilistic reasoning</td>
</tr>
<tr>
<td>→ Different Kinds of Knowledge (Domain-Specific, Common-sense)</td>
<td>→ Abduction (Diagnosis)</td>
</tr>
<tr>
<td></td>
<td>→ Planning</td>
</tr>
<tr>
<td></td>
<td>→ Induction (Learning)</td>
</tr>
</tbody>
</table>
**AI and Knowledge Based Systems**

<table>
<thead>
<tr>
<th>Knowledge Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facts and Rules</td>
</tr>
<tr>
<td>Representation Formalisms (FOL-based, Production Rules)</td>
</tr>
<tr>
<td>Different Kinds of Knowledge (Domain-Specific, Common-sense)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inference Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deduction (Theorem Proving),</td>
</tr>
<tr>
<td>Non-montonic/probabilistic reasoning</td>
</tr>
<tr>
<td>Abduction (Diagnosis)</td>
</tr>
<tr>
<td>Planning</td>
</tr>
<tr>
<td>Induction (Learning)</td>
</tr>
</tbody>
</table>

**Broad View:**

AI Systems are Knowledge-Based Systems
Narrow View:

Knowledge-Based Systems is a sub-field of AI

Focus on: Knowledge Representation and Deductive Inference, Main Application: Expert-Systems
AI and Knowledge Based Systems

Knowledge Base
- Facts and Rules
- Representation Formalisms (FOL-based, Production Rules)
- Different Kinds of Knowledge (Domain-Specific, Common-sense)

Inference Algorithm
- Deduction (Theorem Proving),
- Non-montonic/probabilistic reasoning
- Abduction (Diagnosis)
- Planning
- Induction (Learning)

I take the broader view
## Goals of AI

<table>
<thead>
<tr>
<th>Engineering</th>
<th>Methods for solving problems on a computer algorithms/programs for inference, planning, learning, ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epistemology</td>
<td>Modeling of cognitive processes perception, reasoning, language understanding, ..</td>
</tr>
<tr>
<td>Formalisms</td>
<td>Development of formalisms for describing and evaluating problems and algorithms calculi of logic, graph theory, complexity theory, ...</td>
</tr>
</tbody>
</table>
## Goals of AI

| Engineering    | Methods for solving problems on a computer  
|               | algorithms/programs for inference, planning, learning, ... |
| Epistemology   | Modeling of cognitive processes  
|               | perception, reasoning, language understanding, .. |
| Formalisms     | Development of formalisms for describing and  
|               | evaluating problems and algorithms  
|               | calculi of logic, graph theory, complexity theory, ... |

→ **AI is inherently interdisciplinary!**
# Goals of AI

<table>
<thead>
<tr>
<th>Engineering</th>
<th>Methods for solving problems on a computer algorithms/programs for inference, planning, learning, ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epistemology</td>
<td>Modeling of cognitive processes perception, reasoning, language understanding, ..</td>
</tr>
<tr>
<td>Formalisms</td>
<td>Development of formalisms for describing and evaluating problems and algorithms calculi of logic, graph theory, complexity theory, ...</td>
</tr>
</tbody>
</table>

Mathematics, Computer Science, Neurosciences, Cognitive Psychology, Linguistics, Philosophy of Mind
Main Topics of AI

- Problem Solving/Planning
- Inference/Deduction
- Knowledge Representation
- Machine Learning
Main Topics of AI

- Problem Solving/Planning
- Inference/Deduction
- Knowledge Representation
- Machine Learning

**General methods, applied in:**
Game Playing, Expert Systems, Tutor Systems, Diagnosis, Automatic Theorem Proving, Program Synthesis, Cognitive Robotics, Multi-Agent-Systems, ...
Main Topics of AI

- Problem Solving/Planning: Seminar Topic
- Inference/Deduction
- Knowledge Representation
- Machine Learning: Lecture

General methods, applied in:
Game Playing, Expert Systems, Tutor Systems, Diagnosis, Automatic Theorem Proving, Program Synthesis, Cognitive Robotics, Multi-Agent-Systems, ...
A plan is a sequence of actions for transforming a given state into a state which fulfills a predefined set of goals.
A plan is a sequence of actions for transforming a given state into a state which fulfills a predefined set of goals.

AI planning ("action planning") deals with the development of representation languages for planning problems and with the development of algorithms for plan construction.
AI Planning

A **plan** is a sequence of actions for transforming a given state into a state which fulfills a predefined set of goals.

- Deductive Planning: Situation Calculus (Green, 1967)
- State-Based Planning: Strips (Fikes & Nilsson, 1971)
A plan is a sequence of actions for transforming a given state into a state which fulfills a predefined set of goals.

Alternative to Planning:
Reinforcement Learning
(Learn policies for selecting a suitable action to execute in a given state)
Applications

- System Control
  - autonomous systems
  - virtual agents
Applications

- Process Control:
  - Production of physical goods
  - Construction and configuration
  - Workflow management
  - Mission planning
  - Project planning
Blocks World

Running example:
Blocks World

Running example:

Objects:
A, B, C, D, E, F, Table
Running example:

Objects:
A, B, C, D, E, F, Table

Relations:
clear: \{C, E, F\}
ontable: \{A, E, D\}
on:\{(C, A), (B, D), (F, B)\}
A plan is a sequence of actions for transforming a given state into a state which fulfills a predefined set of goals.
A plan is a sequence of actions for transforming a given state into a state which fulfills a predefined set of goals.

(partial) description of a situation
C is on A and B is on D and F is on B and C is clear and E is clear and F is clear and A is on the table and E is on the table and D is on the table
States, Goals, Actions

A plan is a sequence of actions for transforming a given state into a state which fulfills a predefined set of goals.

Simple goal: D is clear
Conjunctive goal: D is clear and D is on A (disjunctive, quantified, ...)

An Introduction to AI Planning – p. 8
A plan is a sequence of actions for transforming a given state into a state which fulfills a predefined set of goals.

put block E on block F, put block C on the table
A plan is a sequence of actions for transforming a given state into a state which fulfills a predefined set of goals.

put block E on block F, put block C on the table

**Operator Scheme**: define actions over variables
put(?x, ?y): put a block on another block
puttable(?x): put a block on the table
States, Goals, Actions

A plan is a sequence of actions for transforming a given state into a state which fulfills a predefined set of goals.

put block E on block F, put block C on the table

Operator Scheme: define actions over variables
put(?x, ?y): put a block on another block
puttable(?x): put a block on the table

Operators have application conditions
e.g., to put a block somewhere, it must be clear
Planning Formalism

A planning formalism must provide

- a language to represent states, goals and actions.

- an algorithm for constructing a sequence of actions which transforms an initial state into a goal state.
A planning formalism must provide

- a **language** to represent states, goals and actions.
- Typically a subset of FOL for states and goals
- Actions are generated via instantiation of **operator schemes**
- an **algorithm** for constructing a sequence of actions which transforms an initial state into a goal state.
Planning Formalism

A planning formalism must provide

- a language to represent states, goals and actions.
  - Typically a subset of FOL for states and goals
  - Actions are generated via instantiation of operator schemes

- an algorithm for constructing a sequence of actions which transforms an initial state into a goal state.
  - Deductive Planning: Theorem Proving
  - State-Based Planning: Search in State-Space
Deductive Planning

Based on FOL, Situation Calculus (Green, 1967)
Deductive Planning

Based on FOL, Situation Calculus (Green, 1967)

- States are represented as conjunction of facts:
  
  Initial state \( s_1 \)
  
  clear(a, \( s_1 \)).
  clear(b, \( s_1 \)).
  on(b, c, \( s_1 \)).
  ontable(a, \( s_1 \)).
  ontable(c, \( s_1 \)).
Deductive Planning

Based on FOL, Situation Calculus (Green, 1967)

- States are represented as conjunction of facts:

  Initial state $s_1$
  clear(a, $s_1$).
  clear(b, $s_1$).
  on(b, c, $s_1$).
  ontable(a, $s_1$).
  ontable(c, $s_1$).

- Goals can be simple or conjunctions:
  on(a, b, S).

put A on B

An Introduction to AI Planning – p. 10
Deductive Planning

Based on FOL, Situation Calculus (Green, 1967)

- Operator Schemes are represented as **Effect Axioms**:
  \[
  \text{on}(X, Y, \text{put}(X, Y, S)) \leftarrow \text{clear}(X, S) \land \text{clear}(Y, S)
  \]
  “In a situation where blocks X and Y are clear, a new situation where \text{on}(X, Y)) holds can be reached by putting X on Y.”

Rewrite to clausal form:

\[
\neg \text{clear}(X, S) \lor \neg \text{clear}(Y, S) \lor \text{on}(X, Y, \text{put}(X, Y, S))
\]

Proof that the goal can be reached by resolution:

\( S = \text{put}(a, b, s_1) \)
Effect Axioms for Blocks World

on(X, Y, put(X, Y, S)) \iff clear(X, S) \land clear(Y, S)
clear(Z, put(X, Y, S)) \iff on(X, Z, S) \land clear(X, S) \land clear(Y, S)
clear(Y, puttable(X, S)) \iff on(X, Y, S) \land clear(X, S)
onetable(X, puttable(X, S)) \iff clear(X, S)

- Effect of put(X, Y, S): X is on Y,
  if Y was on Z, then Z becomes clear
- Effect of puttable(X, S): X is on the table,
  if X was on Y, then Y becomes clear
Planning in Situation Calculus

*Remember:* A planning formalism must provide

- a **language** to represent states, goals and actions.

- an **algorithm** for constructing a sequence of actions which transforms an initial state into a goal state.
Planning in Situation Calculus

Remember: A planning formalism must provide

- a **language** to represent states, goals and actions.
- **Situation calculus**: formula containing situation variables
- Actions are generated via instantiation of effect axioms
- an **algorithm** for constructing a sequence of actions which transforms an initial state into a goal state.
Planning in Situation Calculus

*Remember:* A planning formalism must provide

- a **language** to represent states, goals and actions.
- **Situation calculus:** formula containing situation variables
- Actions are generated via instantiation of **effect axioms**
- an **algorithm** for constructing a sequence of actions which transforms an initial state into a goal state.
- Resolution Proofs, action sequence returned via instantiation of situation variable
The Frame Problem

In general, additionally to the effect axioms, frame axioms are necessary.

Frame axioms state which relations do NOT change when an action is performed (common-sense knowledge)
The Frame Problem

- In general, additionally to the effect axioms, frame axioms are necessary.

- Frame axioms state which relations do NOT change when an action is performed (common-sense knowledge)

*If I put A on B, C is still on the table.*

![Diagram](image)
The Frame Problem

In general, additionally to the effect axioms, frame axioms are necessary.

Frame axioms state which relations do NOT change when an action is performed (common-sense knowledge)

For realistic problems, coming up with a suitable (complete and consistent) set of frame axioms is a very hard knowledge engineering problem.
Frame Axioms for Blocks World

clear(X, put(X, Y, S)) ← clear(X, S) ∨ clear(Y, S)
clear(Z, put(X, Y, S)) ← clear(X,S) ∨ clear(Y, S) ∨ clear(Z, S)
ontable(Y, put(X, Y, S)) ← clear(X, S) ∨ clear(Y, S) ∨ ontable(Y, S)
ontable(Z, put(X, Y, S)) ← clear(X, S) ∨ clear(Y, S) ∨ ontable(Z, S)
on(Y, Z, put(X, Y, S)) ← clear(X, S) ∨ clear(Y, S) ∨ on(Y, Z, S)
on(W, Z, put(X, Y, S)) ← clear(X, S) ∨ clear(Y, S) ∨ on(W, Z, S)
clear(Z, puttable(X, S)) ← clear(X, S) ∨ clear(Z, S)
ontable(Z, puttable(X, S)) ← clear(X, S) ∨ ontable(Z, S)
on(Y, Z, puttable(X, S)) ← clear(X, S) ∨ on(Y, Z, S)
clear(Z, puttable(X, S)) ← on(Y, X, S) ∨ clear(Y, S) ∨ clear(Z, S)
ontable(Z, puttable(X, S)) ← on(Y, X, S) ∨ clear(Y, S) ∨ ontable(Z, S)
on(W, Z, puttable(X, S)) ← on(Y, X, S) ∨ clear(Y, S) ∨ on(W, Z, S)
State-Based Planning

- Get rid of the frame problem:
  - Closed-world assumption
  - All relations not given explicitly in a state description are assumed to be false.

- Allows to calculate operator effects with simple set operations.

- The Strips Language (Fikes & Nilsson, 1971)
  (a very restricted subset of FOL)
Strips Language

State as set (conjunction) of positive literals. \{on(B, C),
clear(A), clear(B), ontable(A), ontable(C)\}
Strips Language

- State as set (conjunction) of positive literals. \{\text{on}(B, C), \text{clear}(A), \text{clear}(B), \text{ontable}(A), \text{ontable}(C)\}

- Goals as conjunctions of literals. \\
\{\text{on}(A, B), \text{on}(B, C)\}
Strips Language

- State as set (conjunction) of positive literals. \{on(B, C), clear(A), clear(B), ontable(A), ontable(C)\}

- Goals as conjunctions of literals. 
  \{on(A, B), on(B, C)\}

- Operator Schemes: Preconditions and ADD-DEL-Effects.

  **Operator:** put(?x, ?y)
  
  **PRE:** \{ontable(?x), clear(?x), clear(?y)\}
  
  **ADD:** \{on(?x, ?y)\}
  
  **DEL:** \{ontable(?x), clear(?y)\}
Operator Application

- Instantiate an operator wrt the current state $\leftrightarrow$ action $o$
- Calculate successor-state with set operations:

$$\text{Res}(o, s) = s \setminus \text{DEL}(o) \cup \text{ADD}(o) \text{ if } \text{PRE}(o) \subseteq s$$
Strips Example

Operator: put(?x, ?y)
PRE: \{ontable(?x), clear(?x), clear(?y)\}
ADD: \{on(?x, ?y)\}
DEL: \{ontable(?x), clear(?y)\}
Strips Example

Operator: \( \text{put}(?x, ?y) \)

PRE: \( \{\text{ontable}(?x), \text{clear}(?x), \text{clear}(?y)\} \)

ADD: \( \{\text{on}(?x, ?y)\} \)

DEL: \( \{\text{ontable}(?x), \text{clear}(?y)\} \)

\[ s_1 = \{\text{on}(B, C), \text{clear}(A), \text{clear}(B), \text{ontable}(A), \text{ontable}(C)\} \]
Strips Example

Operator: put(?x, ?y)
PRE: \{ontable(?x), clear(?x), clear(?y)\}
ADD: \{on(?x, ?y)\}
DEL: \{ontable(?x), clear(?y)\}

\[s_1 = \{\text{on}(B, C), \text{clear}(A), \text{clear}(B), \text{ontable}(A), \text{ontable}(C)\}\]

\[\text{PRE} \subset s_1 \text{ with } \{x \leftarrow A, y \leftarrow B\}\]
Strips Example

Operator: put(?x, ?y)
PRE: {ontable(?x), clear(?x), clear(?y)}
ADD: {on(?x, ?y)}
DEL: {ontable(?x), clear(?y)}

\[ s_1 = \{\text{on}(B, C), \text{clear}(A), \text{clear}(B), \text{ontable}(A), \text{ontable}(C)\} \]

Substract DEL
\[ \{\text{on}(B, C), \text{clear}(A), \text{ontable}(C)\} \]
Strips Example

Operator:  \( \text{put}(\?x, \?y) \)
PRE:      \( \{\text{ontable}(\?x), \text{clear}(\?x), \text{clear}(\?y)\} \)
ADD:      \( \{\text{on}(\?x, \?y)\} \)
DEL:      \( \{\text{ontable}(\?x), \text{clear}(\?y)\} \)

\[ s_1 = \{\text{on}(B, C), \text{clear}(A), \text{clear}(B), \text{ontable}(A), \text{ontable}(C)\} \]

Add ADD
\[ \{\text{on}(B, C), \text{clear}(A), \text{ontable}(C), \text{on}(A, B)\} = \text{Res}(o, s_1) \]
Operator: put(\( ?x, ?y \))
PRE: \{ ontable(?x), clear(?x), clear(?y) \}
ADD: \{ on(?x, ?y) \}
DEL: \{ ontable(?x), clear(?y) \}

\( s_1 = \{ \text{on}(B, C), \text{clear}(A), \text{clear}(B), \text{ontable}(A), \text{ontable}(C) \} \)

\( \{ \text{on}(B, C), \text{clear}(A), \text{ontable}(C), \text{on}(A, B) \} = \text{Res}(o, s_1) \)

Goal state because \( \{ \text{on}(A, B), \text{on}(B, C) \} \subseteq \text{Res}(o, s_1) \)
Blocksworld in Strips

put(?x, ?y)
PRE: {ontable(?x), clear(?x), clear(?y)}
ADD: {on(?x, ?y)}
DEL: {ontable(?x), clear(?y)}

put(?x, ?y)
PRE: {on(?x, ?z), clear(?x), clear(?y)}
ADD: {on(?x, ?y), clear(?z)}
DEL: {on(?x, ?z), clear(?y)}

puttable(?x)
PRE: {clear(?x), on(?x, ?y)}
ADD: {ontable(?x), clear(?y)}
DEL: {on(?x, ?y)}
Planning as State-Space Search

*Remember:* A planning formalism must provide

- a **language** to represent states, goals and actions.

- an **algorithm** for constructing a sequence of actions which transforms an initial state into a goal state.
Planning as State-Space Search

*Remember:* A planning formalism must provide

- a **language** to represent states, goals and actions.
  - Strips language
    - current standard: PDDL (Strips extension)
  - Planning problem $P = (I, G, O)$ with an initial state, a set of goals, and a set of operators.

- an **algorithm** for constructing a sequence of actions which transforms an initial state into a goal state.
Planning as State-Space Search

*Remember:* A planning formalism must provide

- a **language** to represent states, goals and actions.
  - **Strips language**
    - current standard: PDDL (Strips extension)
  - Planning problem $P = (I, G, O)$ with an initial state, a set of goals, and a set of operators.

- an **algorithm** for constructing a sequence of actions which transforms an initial state into a goal state.

$\rightarrow$ Operator application: a **single planning step**

$\rightarrow$ Planning: **search for a sequence of action applications** which transform an initial state into a goal state.
## State Space Complexity

<table>
<thead>
<tr>
<th># blocks</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td># states</td>
<td>1</td>
<td>3</td>
<td>13</td>
<td>73</td>
<td>501</td>
</tr>
<tr>
<td>approx.</td>
<td>$1.0 \times 10^0$</td>
<td>$3.0 \times 10^0$</td>
<td>$1.3 \times 10^1$</td>
<td>$7.3 \times 10^1$</td>
<td>$5.0 \times 10^2$</td>
</tr>
<tr>
<td># blocks</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td># states</td>
<td>4051</td>
<td>37633</td>
<td>394353</td>
<td>4596553</td>
<td>58941091</td>
</tr>
<tr>
<td>approx.</td>
<td>$4.1 \times 10^3$</td>
<td>$3.8 \times 10^4$</td>
<td>$3.9 \times 10^5$</td>
<td>$4.6 \times 10^6$</td>
<td>$5.9 \times 10^7$</td>
</tr>
<tr>
<td># blocks</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td># states</td>
<td>824073141</td>
<td>12470162233</td>
<td>202976401213</td>
<td>3535017524403</td>
<td>65573803186921</td>
</tr>
<tr>
<td>approx.</td>
<td>$8.2 \times 10^8$</td>
<td>$1.3 \times 10^{10}$</td>
<td>$2.0 \times 10^{11}$</td>
<td>$3.5 \times 10^{12}$</td>
<td>$6.6 \times 10^{13}$</td>
</tr>
</tbody>
</table>

Exponential growth of states

→ typically depth-first strategies are used
Progression vs. Regression Planning

Progression Planning: **Forward Search**

- **Start with** $s$ as **initial state**
- **Repeat**
  - Construct successor states by applying actions to the current state $s$
  - Eliminate cycles
  - If a deadend is reached: backtrack
- **Until** a goal state is reached or no new states can be generated (for finite domains)
Regression Planning: Backward Search

- Start with \( s \) as (partial) goal state
- Repeat
  - Construct successor states by backwards applying actions to the current state \( s \)
  - Eliminate cycles
  - If a deadend is reached: backtrack
- Until the initial state is reached or no new states can be generated (for finite domains)
Plan: Path from the initial state (leaf node) to the root (goal state)
Current Approaches

- State-based approaches dominate the field
  - Graphplan, Sat-Planning
- Special aspects:
  - Include (temporal/resource) constraints
  - Planning under uncertainty (conformant planning)
  - ...
Planning vs. Problem Solving

- **Problem solving**: domain specific representations
  - search algorithms (such as A*) must be adapted to problem
Planning vs. Problem Solving

- **Problem solving:**
  - domain specific representations
  - search algorithms (such as A*) must be adapted to problem

- **Planning:**
  - domain independent representation language
  - general-purpose algorithms for a given language
Planning vs. Human Problem Solving

Planning algorithms are (should be) **complete** (return a plan, if one exists) and **correct** (terminate with a legal action sequence or report failure).
Planning vs. Human Problem Solving

- Planning algorithms are (should be) **complete** (return a plan, if one exists) and **correct** (terminate with a legal action sequence or report failure)

- For complex problems, large parts of the search tree might be generated until a solution is found.
Planning vs. Human Problem Solving

Planning algorithms are (should be) **complete** (return a plan, if one exists) and **correct** (terminate with a legal action sequence or report failure)

For complex problems, large parts of the search tree might be generated until a solution is found.

Humans typically use **greedy search strategies involving heuristics**: try to reach a state which is more similar to the goal state in each step (experimental data, e.g. Hobbits and Orcs, Greeno, 1974) → **efficient but incomplete** (as the General Problem Solver, Newell & Simon, 1972)
Planning vs. Human Problem Solving

Hobbits and Orcs (Greeno, 1974)

Subjects have problems with the transformation from state (6) to (7). Here 2 and not only 1 passenger must be transported back to the left river bank. That is, there must be created a situation which is further removed from the goal state than the situation before.
Planning vs. Human Problem Solving

- Humans learn from problem solving experience (do not run into the same dead-ends again)

→ Research topic: strategy learning