Principles of AI Planning
2. Transition systems and planning tasks

Malte Helmert and Bernhard Nebel

Albert-Ludwigs-Universität Freiburg

April 27th, 2010
Transition systems
Definition (transition system)

A transition system is a 5-tuple \( \mathcal{T} = \langle S, L, T, s_0, S_\star \rangle \) where

- \( S \) is a finite set of states,
- \( L \) is a finite set of (transition) labels,
- \( T \subseteq S \times L \times S \) is the transition relation,
- \( s_0 \in S \) is the initial state, and
- \( S_\star \subseteq S \) is the set of goal states.

We say that \( \mathcal{T} \) has the transition \( \langle s, \ell, s' \rangle \) if \( \langle s, \ell, s' \rangle \in T \).

We also write this \( s \xrightarrow{\ell} s' \), or \( s \rightarrow s' \) when not interested in \( \ell \).

Note: Transition systems are also called state spaces.
Transition systems are often depicted as **directed arc-labeled graphs** with marks to indicate the initial state and goal states.
Transition system terminology

We use common graph theory terms for transition systems:

- \( s' \) successor of \( s \) if \( s \rightarrow s' \)
- \( s \) predecessor of \( s' \) if \( s \rightarrow s' \)
- \( s' \) reachable from \( s \) if there exists a sequence of transitions
  \( s^{(0)} \xrightarrow{\ell_1} s^{(1)}, \ldots, s^{(n-1)} \xrightarrow{\ell_n} s^{(n)} \) s.t. \( s^{(0)} = s \) and \( s^{(n)} = s' \)
  - Note: \( n = 0 \) possible; then \( s = s' \)
  - \( \ell_1, \ldots, \ell_n \) is called path from \( s \) to \( s' \)
  - \( s^{(0)}, \ldots, s^{(n)} \) is also called path from \( s \) to \( s' \)
  - length of that path is \( n \)
- additional terms: strongly connected, weakly connected, strong/weak connected components, \ldots
Transition system terminology (ctd.)

Some additional terminology:

- $s'$ reachable (without reference state) means reachable from initial state $s_0$
- solution or goal path from $s$: path from $s$ to some $s' \in S^*$
  - if $s$ is omitted, $s = s_0$ is implied
- transition system solvable if a goal path from $s_0$ exists
Definition (deterministic transition system)

A transition system with transitions \( T \) is called deterministic if for all states \( s \) and labels \( \ell \), there is at most one state \( s' \) with \( s \xrightarrow{\ell} s' \).

Example: previously shown transition system
Running example: blocks world

Throughout the course, we will often use the blocks world domain as an example.

In the blocks world, a number of differently coloured blocks are arranged on our table.

Our job is to rearrange them according to a given goal.
Blocks world rules

Location on the table does not matter.

\[\begin{array}{ccc}
\text{Location on the table does not matter.} & = & \\
\includegraphics[width=1in]{block1.png} & = & \includegraphics[width=1in]{block2.png}
\end{array}\]

Location on a block does not matter.

\[\begin{array}{ccc}
\text{Location on a block does not matter.} & = & \\
\includegraphics[width=1in]{block3.png} & = & \includegraphics[width=1in]{block4.png}
\end{array}\]
Blocks world rules (ctd.)

At most one block may be below a block.

At most one block may be on top of a block.
Blocks world transition system for three blocks

(Transition labels omitted for clarity.)
## Blocks world computational properties

<table>
<thead>
<tr>
<th>blocks</th>
<th>states</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>73</td>
</tr>
<tr>
<td>5</td>
<td>501</td>
</tr>
<tr>
<td>6</td>
<td>4051</td>
</tr>
<tr>
<td>7</td>
<td>37633</td>
</tr>
<tr>
<td>8</td>
<td>394353</td>
</tr>
<tr>
<td>9</td>
<td>4596553</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>blocks</th>
<th>states</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>58941091</td>
</tr>
<tr>
<td>11</td>
<td>824073141</td>
</tr>
<tr>
<td>12</td>
<td>12470162233</td>
</tr>
<tr>
<td>13</td>
<td>202976401213</td>
</tr>
<tr>
<td>14</td>
<td>3535017524403</td>
</tr>
<tr>
<td>15</td>
<td>65573803186921</td>
</tr>
<tr>
<td>16</td>
<td>1290434218669921</td>
</tr>
<tr>
<td>17</td>
<td>26846616451246353</td>
</tr>
<tr>
<td>18</td>
<td>588633468315403843</td>
</tr>
</tbody>
</table>

- **Finding a solution** is polynomial time in the number of blocks (move everything onto the table and then construct the goal configuration).
- **Finding a shortest solution** is NP-complete (for a compact description of the problem).
Planning tasks
Compact representations

- Classical (i.e., deterministic) planning is in essence the problem of finding solutions in huge transition systems.
- The transition systems we are usually interested in are too large to explicitly enumerate all states or transitions.
- Hence, the input to a planning algorithm must be given in a more concise form.
- In the rest of chapter, we discuss how to represent planning tasks in a suitable way.
State variables

How to represent huge state sets without enumerating them?

- represent different aspects of the world in terms of different state variables

→ a state is a valuation of state variables

- $n$ state variables with $m$ possible values each induce $m^n$ different states

→ exponentially more compact than “flat” representations

- Example: $n$ variables suffice for blocks world with $n$ blocks
Blocks world with finite-domain state variables

Describe blocks world state with three state variables:

- *location-of-A*: \{B, C, table\}
- *location-of-B*: \{A, C, table\}
- *location-of-C*: \{A, B, table\}

**Example**

\[
\begin{align*}
s(\text{location-of-A}) &= \text{table} \\
s(\text{location-of-B}) &= A \\
s(\text{location-of-C}) &= \text{table}
\end{align*}
\]

Not all valuations correspond to intended blocks world states. **Example**: \(s\) with \(s(\text{location-of-A}) = B, s(\text{location-of-B}) = A\).
Boolean state variables

Problem:

- How to succinctly represent transitions and goal states?

Idea: Use propositional logic

- state variables: propositional variables (0 or 1)
- goal states: defined by a propositional formula
- transitions: defined by actions given by
  - precondition: when is the action applicable?
  - effect: how does it change the valuation?

Note: general finite-domain state variables can be compactly encoded as Boolean variables
Blocks world with Boolean state variables

Example

\[
\begin{align*}
    s(A\text{-}on\text{-}B) &= 0 \\
    s(A\text{-}on\text{-}C) &= 0 \\
    s(A\text{-}on\text{-}table) &= 1 \\
    s(B\text{-}on\text{-}A) &= 1 \\
    s(B\text{-}on\text{-}C) &= 0 \\
    s(B\text{-}on\text{-}table) &= 0 \\
    s(C\text{-}on\text{-}A) &= 0 \\
    s(C\text{-}on\text{-}B) &= 0 \\
    s(C\text{-}on\text{-}table) &= 1
\end{align*}
\]
Definition (propositional formula)

Let $A$ be a set of atomic propositions (here: state variables). The propositional formulae over $A$ are constructed by finite application of the following rules:

- $\top$ and $\bot$ are propositional formulae (truth and falsity).
- For all $a \in A$, $a$ is a propositional formula (atom).
- If $\varphi$ is a propositional formula, then so is $\neg \varphi$ (negation).
- If $\varphi$ and $\psi$ are propositional formula, then so are $(\varphi \lor \psi)$ (disjunction) and $(\varphi \land \psi)$ (conjunction).

Note: We often omit the word “propositional”.
Propositional logic conventions

Abbreviations:

- $(\varphi \rightarrow \psi)$ is short for $(\neg \varphi \lor \psi)$ (implication)
- $(\varphi \leftrightarrow \psi)$ is short for $((\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi))$ (equivalence)
- Parentheses omitted when not necessary
- $(\neg)$ binds more tightly than binary connectives
- $(\land)$ binds more tightly than $(\lor)$ than $(\rightarrow)$ than $(\leftrightarrow)$
Definition (propositional valuation)

A valuation of propositions \( A \) is a function \( v : A \to \{0, 1\} \).

Define the notation \( v \models \varphi \) (\( v \) satisfies \( \varphi \); \( v \) is a model of \( \varphi \); \( \varphi \) is true under \( v \)) for valuations \( v \) and formulae \( \varphi \) by

- \( v \models T \)
- \( v \not\models \bot \)
- \( v \models a \) iff \( v(a) = 1 \), for \( a \in A \).
- \( v \models \neg\varphi \) iff \( v \not\models \varphi \)
- \( v \models \varphi \lor \psi \) iff \( v \models \varphi \) or \( v \models \psi \)
- \( v \models \varphi \land \psi \) iff \( v \models \varphi \) and \( v \models \psi \)
Propositional logic terminology

- A propositional formula $\varphi$ is **satisfiable** if there is at least one valuation $v$ so that $v \models \varphi$.
- Otherwise it is **unsatisfiable**.
- A propositional formula $\varphi$ is **valid** or a **tautology** if $v \models \varphi$ for all valuations $v$.
- A propositional formula $\psi$ is a **logical consequence** of a propositional formula $\varphi$, written $\varphi \models \psi$, if $v \models \psi$ for all valuations $v$ with $v \models \varphi$.
- Two propositional formulae $\varphi$ and $\psi$ are **logically equivalent**, written $\varphi \equiv \psi$, if $\varphi \models \psi$ and $\psi \models \varphi$.

**Question:** How to phrase these in terms of **models**?
Propositional logic terminology (ctd.)

- A propositional formula that is a proposition $a$ or a negated proposition $\neg a$ for some $a \in A$ is a literal.
- A formula that is a disjunction of literals is a clause. This includes unit clauses $l$ consisting of a single literal, and the empty clause $\bot$ consisting of zero literals.

**Normal forms:** NNF, CNF, DNF
Transitions for state sets described by propositions $A$ can be concisely represented as operators or actions $\langle \chi, e \rangle$ where

- the **precondition** $\chi$ is a propositional formula over $A$ describing the set of states in which the transition can be taken (states in which a transition starts), and

- the **effect** $e$ describes how the resulting successor states are obtained from the state where the transitions is taken (where the transition goes).
Example: blocks world operators

Blocks world operators

To model blocks world operators conveniently, we use auxiliary state variables $A$-clear, $B$-clear, and $C$-clear to denote that there is nothing on top of a given block.

Then blocks world operators can be modeled as:

- $\langle A$-clear $\land A$-on-$T \land B$-clear, $A$-on-$B \land \neg A$-on-$T \land \neg B$-clear $\rangle$
- $\langle A$-clear $\land A$-on-$T \land C$-clear, $A$-on-$C \land \neg A$-on-$T \land \neg C$-clear $\rangle$
- $\langle A$-clear $\land A$-on-$B$, $A$-on-$T \land \neg A$-on-$B \land B$-clear $\rangle$
- $\langle A$-clear $\land A$-on-$C$, $A$-on-$T \land \neg A$-on-$C \land C$-clear $\rangle$
- $\langle A$-clear $\land A$-on-$B \land C$-clear, $A$-on-$C \land \neg A$-on-$B \land B$-clear $\land \neg C$-clear $\rangle$
- $\langle A$-clear $\land A$-on-$C \land B$-clear, $A$-on-$B \land \neg A$-on-$C \land C$-clear $\land \neg B$-clear $\rangle$
- ...
Effects (for deterministic operators)

Definition (effects)

(Deterministic) effects are recursively defined as follows:

- If $a \in A$ is a state variable, then $a$ and $\neg a$ are effects (atomic effect).
- If $e_1, \ldots, e_n$ are effects, then $e_1 \land \cdots \land e_n$ is an effect (conjunctive effect).
  The special case with $n = 0$ is the empty effect $\top$.
- If $\chi$ is a propositional formula and $e$ is an effect, then $\chi \triangleright e$ is an effect (conditional effect).

Atomic effects $a$ and $\neg a$ are best understood as assignments $a := 1$ and $a := 0$, respectively.
Effect example

$\chi \triangleright e$ means that change $e$ takes place if $\chi$ is true in the current state.

Example

Increment 4-bit number $b_3 b_2 b_1 b_0$ represented as four state variables $b_0, \ldots, b_3$:

$$
(\neg b_0 \triangleright b_0) \land \\
((\neg b_1 \land b_0) \triangleright (b_1 \land \neg b_0)) \land \\
((\neg b_2 \land b_1 \land b_0) \triangleright (b_2 \land \neg b_1 \land \neg b_0)) \land \\
((\neg b_3 \land b_2 \land b_1 \land b_0) \triangleright (b_3 \land \neg b_2 \land \neg b_1 \land \neg b_0))
$$
Operator semantics

Definition (changes caused by an operator)

For each effect $e$ and state $s$, we define the change set of $e$ in $s$, written $[e]_s$, as the following set of literals:

- $[a]_s = \{a\}$ and $[-a]_s = \{-a\}$ for atomic effects $a$, $-a$
- $[e_1 \land \cdots \land e_n]_s = [e_1]_s \cup \cdots \cup [e_n]_s$
- $[\chi \triangleright e]_s = [e]_s$ if $s \models \chi$ and $[\chi \triangleright e]_s = \emptyset$ otherwise

Definition (applicable operators)

Operator $\langle \chi, e \rangle$ is applicable in a state $s$ iff $s \models \chi$ and $[e]_s$ is consistent (i.e., does not contain two complementary literals).
Operator semantics (ctd.)

Definition (successor state)
The successor state $\text{app}_o(s)$ of $s$ with respect to operator $o = \langle \chi, e \rangle$ is the state $s'$ with $s' \models [e]_s$ and $s'(v) = s(v)$ for all state variables $v$ not mentioned in $[e]_s$. This is defined only if $o$ is applicable in $s$.

Example
Consider the operator $\langle a, \neg a \land (\neg c \triangleright \neg b) \rangle$ and the state $s = \{a \mapsto 1, b \mapsto 1, c \mapsto 1, d \mapsto 1\}$. The operator is applicable because $s \models a$ and $[\neg a \land (\neg c \triangleright \neg b)]_s = \{\neg a\}$ is consistent. Applying the operator results in the successor state $\text{app}_{\langle a, \neg a \land (\neg c \triangleright \neg b) \rangle}(s) = \{a \mapsto 0, b \mapsto 1, c \mapsto 1, d \mapsto 1\}$. 
A deterministic planning task is a 4-tuple $\Pi = (A, I, O, \gamma)$ where

- $A$ is a finite set of state variables (propositions),
- $I$ is a valuation over $A$ called the initial state,
- $O$ is a finite set of operators over $A$, and
- $\gamma$ is a formula over $A$ called the goal.

Notes:

- In this course, we usually omit the word “deterministic” since all our tasks are deterministic.
- In the general literature “planning task” often refers to broader problem classes, e.g. including nondeterminism.
Definition (induced transition system of a planning task)

Every planning task $\Pi = \langle A, I, O, \gamma \rangle$ induces a corresponding deterministic transition system $T(\Pi) = \langle S, L, T, s_0, S_\star \rangle$:

- $S$ is the set of all valuations of $A$,
- $L$ is the set of operators $O$,
- $T = \{ \langle s, o, s' \rangle \mid s \in S, \ o \text{ applicable in } s, \ s' = app_o(s) \}$,
- $s_0 = I$, and
- $S_\star = \{ s \in S \mid s \models \gamma \}$
Planning tasks: terminology

- Terminology for transitions systems is also applied to the planning tasks that induce them.
- For example, when we speak of the states of $\Pi$, we mean the states of $T(\Pi)$.
- A sequence of operators that forms a goal path of $T(\Pi)$ is called a plan of $\Pi$. 
By **planning**, we mean the following two algorithmic problems:

<table>
<thead>
<tr>
<th>Definition (satisficing planning)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Given:</strong> a planning task $\Pi$</td>
</tr>
<tr>
<td><strong>Output:</strong> a plan for $\Pi$, or <strong>unsolvable</strong> if no plan for $\Pi$ exists</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition (optimal planning)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Given:</strong> a planning task $\Pi$</td>
</tr>
<tr>
<td><strong>Output:</strong> a plan for $\Pi$ with minimal length among all plans for $\Pi$, or <strong>unsolvable</strong> if no plan for $\Pi$ exists</td>
</tr>
</tbody>
</table>
Summary

- **Transition systems** are a kind of directed graph (typically huge) that encode how the state of the world can change.

- **Planning tasks** are compact representations for transition systems, suitable as input for planning algorithms.

- Planning tasks are based on concepts from propositional logic, suitably enhanced to model state change.

- **States** of planning tasks are propositional valuations.

- **Operators** of planning tasks describe *when* (precondition) and *how* (effect) to change the current state of the world.

- In **satisficing planning**, we must find a solution to planning tasks (or show that no solution exists).

- In **optimal planning**, we must additionally guarantee that generated solutions are of the shortest possible length.