A Beginner’s Introduction
to Heuristic Search Planning
2. Planning Formalisms (and Heuristic Search)

Malte Helmert    Gabriele Röger

AAA1 2015 Tutorial

January 25, 2015
Transition Systems & Search
Example: Blocks World
Example: Blocks World
Example: Blocks World
Example: Blocks World
Transition Systems

Definition (Transition system)

A transition system (or state space) is a 6-tuple \( T = \langle S, s_0, S_\star, A, cost, T \rangle \) with

- \( S \) finite set of states
- \( s_0 \in S \) initial state
- \( S_\star \subseteq S \) set of goal states
- \( A \) finite set of actions
- \( cost : A \rightarrow \mathbb{R}_0^+ \) action costs
- \( T \subseteq S \times A \times S \) transition relation
  - deterministic in \( \langle s, a \rangle \):
    - for each \( \langle s, a \rangle \) at most one transition \( \langle s, a, s' \rangle \in T \)
**Plan**

**Definition (Plan)**

A **plan** for a transition system is a **sequence of actions** occurring as labels on a **path from the initial state to a goal state**.

The **cost** of a plan $\langle a_1, \ldots, a_n \rangle$ is $\sum_{i=1}^{n} cost(a_i)$.

A plan is **optimal** if it has minimal cost.
Definition (Plan)

A plan for a transition system is a sequence of actions occurring as labels on a path from the initial state to a goal state.

The cost of a plan $\langle a_1, \ldots, a_n \rangle$ is $\sum_{i=1}^{n} \text{cost}(a_i)$.

A plan is optimal if it has minimal cost.
Automated Planning

Definition (Optimal Planning)
Given an encoding of a transition system, find an optimal plan.

Definition (Satisficing Planning)
Given an encoding of a transition system, find a (not necessarily optimal) plan.

Cheaper plans are better solutions.
Search in a Nutshell
Search in a Nutshell
Search in a Nutshell

Which open node should we select for expansion?
Search in a Nutshell
Search in a Nutshell

Which open node should we select for expansion?
Heuristic Search

- Prioritize open nodes with **heuristic**

- **Heuristic**
  - estimates cost of path from state to closest goal state
  - \( h : S \rightarrow \mathbb{R}_0^+ \)

- Search algorithms differ in how they exploit the heuristic:
  - \( h \): heuristic estimate of state
  - \( g \): cost of path from initial state to open node
  - **Greedy best-first search**: expand node with minimum \( h \)
  - **A* algorithm**: expand node with minimum \( g + h \)
Transition Systems as Input Formalism?

Definition (Transition system)

A transition system (or state space) is a 6-tuple
\[ T = \langle S, s_0, S_*, A, \text{cost}, T \rangle \] with ...
Transition Systems as Input Formalism?

Definition (Transition system)

A transition system (or state space) is a 6-tuple $T = \langle S, s_0, S_x, A, cost, T \rangle$ with ...

$n$ blocks: more than $n!$ states
Definition (Transition system)

A transition system (or state space) is a 6-tuple \( T = \langle S, s_0, S_*, A, cost, T \rangle \) with . . .

\( n \) blocks: more than \( n! \) states

Heuristics require structure
Transition Systems as Input Formalism?

Definition (Transition system)

A transition system (or state space) is a 6-tuple \( T = \langle S, s_0, S_*, A, \text{cost}, T \rangle \) with …

\( n \) blocks: more than \( n! \) states

Heuristics require structure

not suitable as input formalism for planning systems
Planning Formalism in Theory
Propositional STRIPS

- Most basic common planning formalism
- States and actions specified in terms of propositional state variables
Propositional STRIPS

- **Most basic** common planning formalism
- **States and actions** specified in terms of propositional state variables
- **State**: set of state variables
  - $v \in s$: variable $v$ is **true** in state $s$
  - $v \notin s$: variable $v$ is **false** in state $s$
Propositional STRIPS

- Most basic common planning formalism
- States and actions specified in terms of propositional state variables
- State: set of state variables
  - $v \in s$: variable $v$ is true in state $s$
  - $v \notin s$: variable $v$ is false in state $s$
- Actions have preconditions, add effects and delete effects
  - action is applicable in state $s$ if all preconditions are true in $s$
  - add effects become true in successor state
  - delete effects become false in successor state
    (except if also included in add effects)
Propositional STRIPS: Planning Task

Definition (Propositional STRIPS planning task)

A propositional STRIPS planning task is a 4-tuple \( \Pi = \langle V, I, G, A \rangle \) with the following components:

- \( V \): finite set of state variables
- \( I \subseteq V \): initial state
- \( G \subseteq V \): set of goal variables
- \( A \): finite set of actions (or operators), where each action \( a \in A \) has the following components:
  - \( pre(a) \subseteq V \): preconditions
  - \( add(a) \subseteq V \): add effects
  - \( del(a) \subseteq V \): del effects
  - \( cost(a) \in \mathbb{R}_0^+ \): action cost

Remark: Actions costs are an extension of traditional STRIPS.
Propositional STRIPS: Semantics

Definition (transition system induced by a STRIPS planning task)

Let $\Pi = \langle V, I, G, A \rangle$ be a (propositional) STRIPS planning task. Task $\Pi$ induces the transition system $\langle S, s_0, S_\ast, A, cost, T \rangle$:

- **states**: $S = 2^V$ (= power set of $V$)
- **initial state**: $s_0 = I$
- **goal states**: $s \in S_\ast$ iff $G \subseteq s$
- **actions**: actions $A$ of $\Pi$
- **action costs**: $cost$ defined as in $\Pi$
- **transitions**: $\langle s, a, s' \rangle \in T$ iff
  - $\text{pre}(a) \subseteq s$, and
  - $s' = (s \setminus \text{del}(a)) \cup \text{add}(a)$
Example: Blocks World in Propositional STRIPS

\[ \Pi = \langle V, I, G, A \rangle \text{ with:} \]

- \( V = \{ \text{on}_{R,G}, \text{on}_{R,B}, \text{on}_{G,R}, \text{on}_{G,B}, \text{on}_{B,R}, \text{on}_{B,G}, \) 
  \text{on-table}_{R}, \text{on-table}_{G}, \text{on-table}_{B}, \) 
  \text{clear}_{R}, \text{clear}_{G}, \text{clear}_{B} \} \)

- \( I = \{ \text{on}_{R,B}, \text{on}_{B,G}, \text{on-table}_{G}, \text{clear}_{R} \} \)

- \( G = \{ \text{on}_{G,R} \} \)

- \( A = \{ \text{move}_{R,B,G}, \text{move}_{R,G,B}, \text{move}_{B,R,G}, \) 
  \text{move}_{B,G,R}, \text{move}_{G,R,B}, \text{move}_{G,B,R}, \) 
  \text{to-table}_{R,B}, \text{to-table}_{R,G}, \text{to-table}_{B,R}, \) 
  \text{to-table}_{B,G}, \text{to-table}_{G,R}, \text{to-table}_{G,B}, \) 
  \text{from-table}_{R,B}, \text{from-table}_{R,G}, \text{from-table}_{B,R}, \) 
  \text{from-table}_{B,G}, \text{from-table}_{G,R}, \text{from-table}_{G,B} \} \)

...
Example: Blocks World in Propositional STRIPS

Example

move actions encode movements of a block from one block onto another

For example:

- \( \text{pre}(\text{move}_{R,B,G}) = \{\text{on}_{R,B}, \text{clear}_R, \text{clear}_G\} \)
- \( \text{add}(\text{move}_{R,B,G}) = \{\text{on}_{R,G}, \text{clear}_B\} \)
- \( \text{del}(\text{move}_{R,B,G}) = \{\text{on}_{R,B}, \text{clear}_G\} \)
- \( \text{cost}(\text{move}_{R,B,G}) = 1 \)
Example: Blocks World in Propositional STRIPS

**Example**

*to-table* actions encode movements of a block from a block to the table

For example:

- $pre(to-table_{R,B}) = \{on_{R,B}, clear_R\}$
- $add(to-table_{R,B}) = \{on-table_{R}, clear_B\}$
- $del(to-table_{R,B}) = \{on_{R,B}\}$
- $cost(to-table_{R,B}) = 1$
Example: Blocks World in Propositional STRIPS

Example

to-table actions encode movements of a block from a block to the table

For example:

- \( \text{pre}(\text{to-table}_{R,B}) = \{\text{on}_{R,B}, \text{clear}_R\} \)
- \( \text{add}(\text{to-table}_{R,B}) = \{\text{on-table}_R, \text{clear}_B\} \)
- \( \text{del}(\text{to-table}_{R,B}) = \{\text{on}_{R,B}\} \)
- \( \text{cost}(\text{to-table}_{R,B}) = 1 \)

from-table actions encode the inverse action (movements of blocks from table onto block).
SAS$^+$ Formalism

- similar to propositional STRIPS but **state variables** have a (possibly non-binary) **finite domain**
- often more natural formulation than with STRIPS
- **State**: variable assignment
SAS+ Formalism

- similar to propositional STRIPS but state variables have a (possibly non-binary) finite domain
- often more natural formulation than with STRIPS
- **State**: variable assignment
- **Preconditions and goal**: partial variable assignment
  
  **Example**: \( \{v_1 \mapsto a, v_3 \mapsto b\} \) as precondition (or goal)
  
  - If it holds for state \( s \) that \( s(v_1) = a \) and \( s(v_3) = b \), then the action is applicable (or \( s \) is a goal state).
  - Other variable values are irrelevant.
SAS⁺ Formalism

- similar to propositional STRIPS but state variables have a (possibly non-binary) finite domain
- often more natural formulation than with STRIPS
- State: variable assignment
- Preconditions and goal: partial variable assignment
  - Example: \{v_1 \rightarrow a, v_3 \rightarrow b\} as precondition (or goal)
    - If it holds for state s that \(s(v_1) = a\) and \(s(v_3) = b\), then the action is applicable (or s is a goal state).
    - Other variable values are irrelevant.
- Effects: partial variable assignment
  - Example: Effect \{v_1 \rightarrow b, v_2 \rightarrow c\}
    - For successor state \(s'\) it holds that \(s'(v_1) = b\) and \(s'(v_2) = c\).
    - All other variable values stay unchanged.
**SAS\(^+\) Planning Task**

Definition (SAS\(^+\) planning task)

A SAS\(^+\) planning task is a 5-tuple \(\Pi = \langle V, s_0, s_\star, A \rangle\) with the following components:

- \(V\): finite set of state variables \(v\), each with finite domain \(\text{dom}(v)\),
- \(s_0\): variable assignment defining the initial state
- \(s_\star\): partial variable assignment defining the goal
- \(A\): finite set of actions (or operators), where each action \(a \in A\) has the following components:
  - Preconditions \(\text{pre}(a)\): partial variable assignment
  - Effects \(\text{eff}(a)\): partial variable assignment
  - Cost \(\text{cost}(a)\): non-negative real number
Example: Blocks World in SAS+

\[ \Pi = \langle V, s_0, s_*, A \rangle \text{ with:} \]

- \( V = \{ on_R, on_G, on_B, clear_R, clear_G, clear_B \} \) with
  - \( \text{dom}(on_X) = \{ R, G, B, \text{Table} \} \setminus \{ X \} \) and
  - \( \text{dom}(clear_X) = \{ T, F \} \) for \( X \in \{ R, G, B \} \)
- \( s_0 = \{ on_R \leftrightarrow B, on_G \leftrightarrow \text{Table}, on_B \leftrightarrow G, \)
  - \( clear_R \leftrightarrow T, clear_G \leftrightarrow F, clear_B \leftrightarrow F \} \)
- \( s_* = \{ on_G \leftrightarrow R \} \)
- \( A = \text{same action labels as in STRIPS example} \)
Example: Blocks World in SAS$^+$

**Example**

*move* actions encode movements of a block from a block onto another

For example:

- $\text{pre}(\text{move}_{R,B,G}) = \{\text{on}_R \leftrightarrow B, \text{clear}_R \leftrightarrow T, \text{clear}_G \leftrightarrow T\}$
- $\text{eff}(\text{move}_{R,B,G}) = \{\text{on}_R \leftrightarrow G, \text{clear}_B \leftrightarrow T, \text{clear}_G \leftrightarrow F\}$
- $\text{cost}(\text{move}_{R,B,G}) = 1$
Other Formalisms

Extensions of these formalisms include additional features, e.g.

- Propositional formulas in conditions
- Conditional effects
- Derived predicates
- Schematic representation with first-order formulas in conditions and all-quantified effects
- ...
Planner Input Language PDDL
PDDL

- **Planning Domain Definition Language**
- Input language of most planning systems
- Used by the International Planning Competitions
- Several requirements denote different language fragments
- Some fragments beyond classical planning
- Supports *parameterized, schematic definition* of operators
Internal Planner Format

Most planners transform the PDDL input into an internal format.

Fast Downward: SAS+ (+ some extensions)

Hands-On

$ cd hands-on
$ ./fd --translate tile/puzzle.pddl \
tile/puzzle01.pddl
$ less output.sas
$ less fd-internal/search/global_operator.h
Summary
Summary

- **classical planning**: path finding in large deterministic transition systems
- **optimal planning**: only optimal plans are solutions
- **satisficing planning**: any plan is a solution but cheaper plans are better
- **best-first search**: guided by heuristics
- **heuristics**: estimate cost to reach closest goal state
- **planning formalisms**: implicit and structured specification of transition systems
- **research papers**: mostly propositional STRIPS or SAS⁺
- **PDDL**: standard input language for planning systems