Classical Planning Algorithms

2. Planning Formalisms and Models

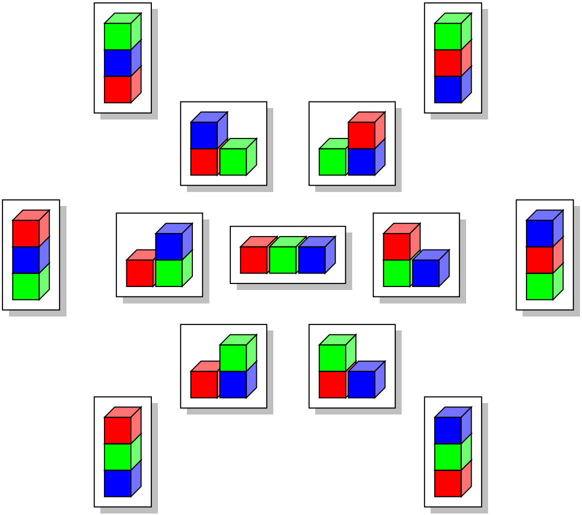
Malte Helmert

ICAPS 2018 Summer School

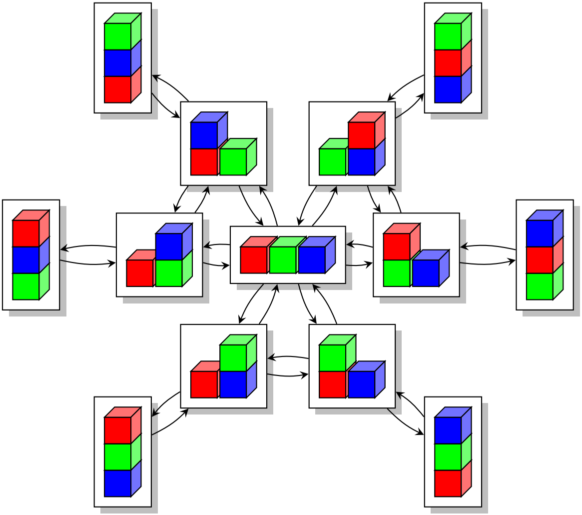
June 21, 2018

Transition Systems

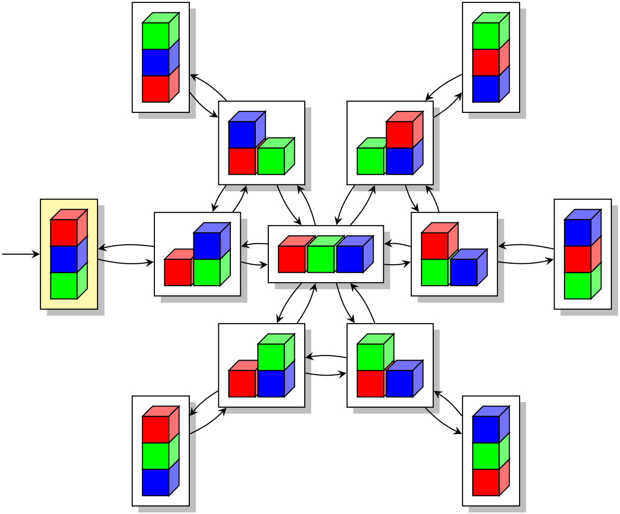
Example: Blocks World



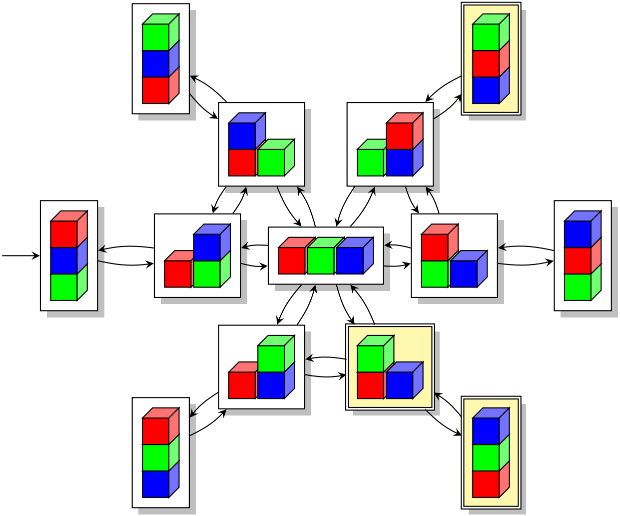
Example: Blocks World



Example: Blocks World



Example: Blocks World



Transition Systems

Definition (Transition System)

A transition system (or state space) is a 6-tuple $\mathcal{T} = \langle S, s_0, S_\star, A, cost, T \rangle$ 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
- $cost : A \to \mathbb{R}_0^+$ 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 $\langle s, a, s' \rangle \in T$

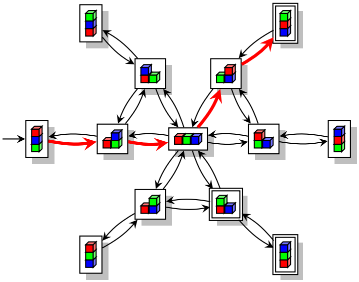
Plans

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.



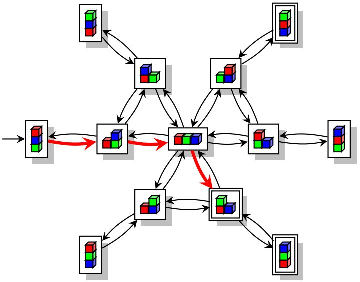
Plans

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.



Classical Planning

Definition (Optimal Classical Planning)

Given an encoding of a transition system, find an optimal plan.

Definition (Satisficing Classical Planning)

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

Cheaper plans are better solutions.

Transition Systems as Input Formalism?

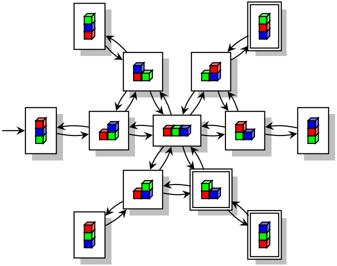
Definition (Transition System)

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

Transition Systems as Input Formalism?

Definition (Transition System)

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

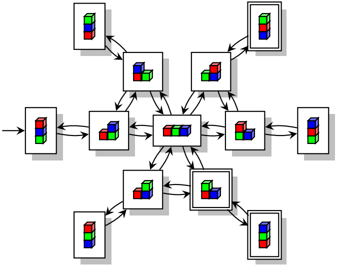


n blocks: more than $n!$ states

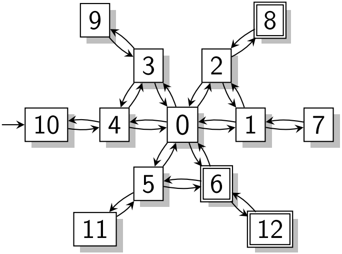
Transition Systems as Input Formalism?

Definition (Transition System)

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



n blocks: more than $n!$ states

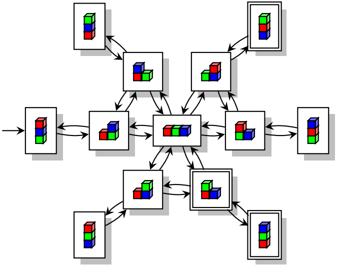


heuristics require structure

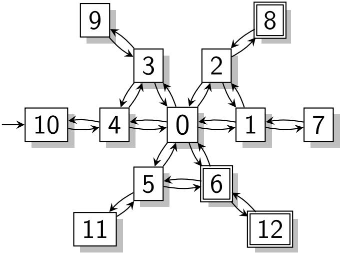
Transition Systems as Input Formalism?

Definition (Transition System)

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



n blocks: more than *n*! states



heuristics require structure

not suitable as input formalism for planning systems

Planning Formalisms 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 (unless also included in add effects)

Propositional STRIPS: Planning Tasks

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
  - $del(a) \subseteq V$: delete effects
  - $cost(a) \in \mathbb{R}_0^+$: action cost

Remark: action 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 Π induces the transition system $\langle S, s_0, S_\star, A, cost, T \rangle$:

- states: $S = 2^V$ ($=$ power set of V)
- initial state: $s_0 = I$
- goal states: $s \in S_\star$ iff $G \subseteq s$
- actions: actions A of Π
- action costs: $cost$ defined as in Π
- transitions: $\langle s, a, s' \rangle \in T$ iff
  - $pre(a) \subseteq s$, and
  - $s' = (s \setminus del(a)) \cup add(a)$

Example: Blocks World in Propositional STRIPS

Example

$\Pi = \langle V, I, G, A \rangle$ with:

- $V = \{on_{R,G}, on_{R,B}, on_{G,R}, on_{G,B}, on_{B,R}, on_{B,G},$
  $on\text{-}table_R, on\text{-}table_G, on\text{-}table_B,$
  $clear_R, clear_G, clear_B\}$
- $I = \{on_{R,B}, on_{B,G}, on\text{-}table_G, clear_R\}$
- $G = \{on_{G,R}\}$
- $A = \{move_{R,B,G}, move_{R,G,B}, move_{B,R,G},$
  $move_{B,G,R}, move_{G,R,B}, move_{G,B,R},$
  $to\text{-}table_{R,B}, to\text{-}table_{R,G}, to\text{-}table_{B,R},$
  $to\text{-}table_{B,G}, to\text{-}table_{G,R}, to\text{-}table_{G,B},$
  $from\text{-}table_{R,B}, from\text{-}table_{R,G}, from\text{-}table_{B,R},$
  $from\text{-}table_{B,G}, from\text{-}table_{G,R}, from\text{-}table_{G,B}\}$

. . .

Example: Blocks World in Propositional STRIPS

Example

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

Example:

- $pre(move_{R,B,G}) = \{on_{R,B}, clear_R, clear_G\}$
- $add(move_{R,B,G}) = \{on_{R,G}, clear_B\}$
- $del(move_{R,B,G}) = \{on_{R,B}, clear_G\}$
- $cost(move_{R,B,G}) = 1$

skip: *to-table* and *from-table* actions

SAS$^+$ Formalism

- similar to propositional STRIPS but state variables may have an arbitrary (possibly non-binary) finite domain
- often more natural formulation than with STRIPS

SAS$^+$ Formalism

- similar to propositional STRIPS but state variables may have an arbitrary (possibly non-binary) finite domain
- often more natural formulation than with STRIPS
- state: variable assignment
- preconditions and goal: partial variable assignments

  Example: $\{v_1 \mapsto a, v_3 \mapsto b\}$ as precondition (or goal)
  - If state s satisfies $s(v_1) = a$ and $s(v_3) = b$, then the action is applicable (or s is a goal state).
  - Values of other variables are irrelevant.
- effects: partial variable assignment

  Example: effect $\{v_1 \mapsto b, v_2 \mapsto c\}$
  - Successor state s' satisfies $s'(v_1) = b$ and $s'(v_2) = c$.
  - All other variables remain unchanged.

SAS$^+$ Planning Tasks

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 $dom(v)$,
- 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 $pre(a)$: partial variable assignment
  - effects $eff(a)$: partial variable assignment
  - cost $cost(a)$: non-negative real number

Example: Blocks World in SAS$^+$

Example

$\Pi = \langle V, s_0, s_\star, A \rangle$ with:

- $V = \{on_R, on_G, on_B, clear_R, clear_G, clear_B\}$ with
  $dom(on_X) = \{R, G, B, \text{Table}\} \setminus \{X\}$ and
  $dom(clear_X) = \{T, F\}$ for all $X \in \{R, G, B\}$
- $s_0 = \{on_R \mapsto B, on_G \mapsto \text{Table}, on_B \mapsto G,$
  $clear_R \mapsto T, clear_G \mapsto F, clear_B \mapsto F\}$
- $s_\star = \{on_G \mapsto R\}$
- A = 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:

- $pre(move_{R,B,G}) = \{on_R \mapsto B, clear_R \mapsto T, clear_G \mapsto T\}$
- $eff(move_{R,B,G}) = \{on_R \mapsto G, clear_B \mapsto T, clear_G \mapsto F\}$
- $cost(move_{R,B,G}) = 1$

skip: *to-table* and *from-table* actions

Other Formalisms

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

- propositional formulas in conditions
- conditional effects
- derived predicates
- schematic representations with first-order formulas in conditions and universally quantified effects
- ...

Planner Input Language PDDL

PDDL

PDDL

- Planning Domain Definition Language
- input language of most planning systems
- used by the International Planning Competitions
- `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 /vagrant/lectures/classical/demo
$ ./fd --translate \
      tile/puzzle.pddl tile/puzzle01.pddl
$ less output.sas
$ ./fd --translate \
      tile/puzzle.pddl tile/puzzle01.pddl \
      --translate-options --dump-task
$ less output.dump
```

Summary

Summary

- classical planning: path finding in very large deterministic transition systems
  - optimal planning: only optimal plans are solutions
  - satisficing planning: any plan is a solution, but cheaper plans are preferred
- planning formalisms: factored declarative specification languages for transition systems
  - research papers: mostly propositional STRIPS or SAS^+
  - PDDL: standard input language for planning systems