A Beginner's Introduction to Heuristic Search Planning

2. Planning Formalisms (and Heuristic Search)

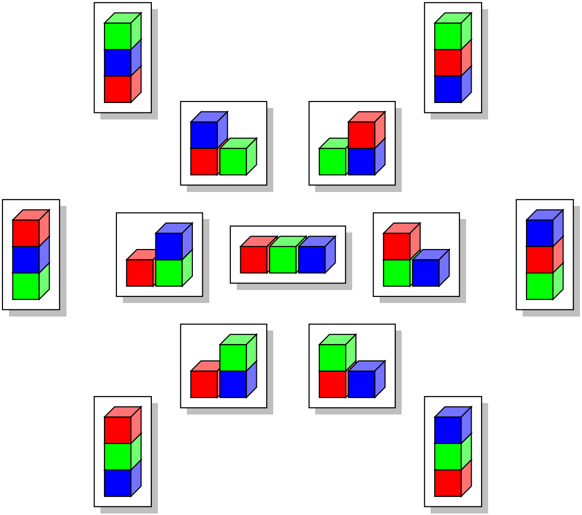
Malte Helmert   Gabriele Röger

AAAI 2015 Tutorial

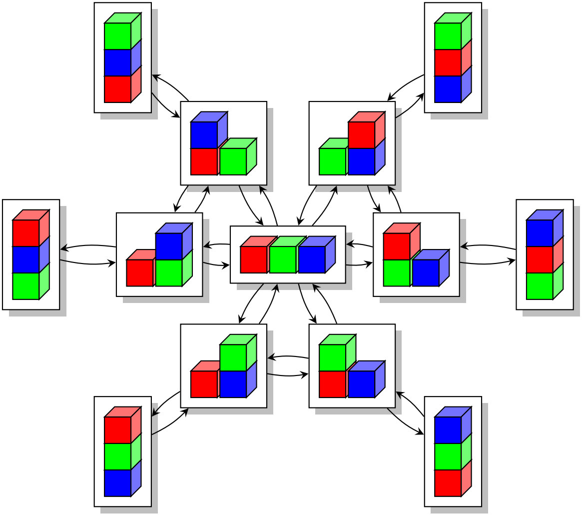
January 25, 2015

Transition Systems & Search

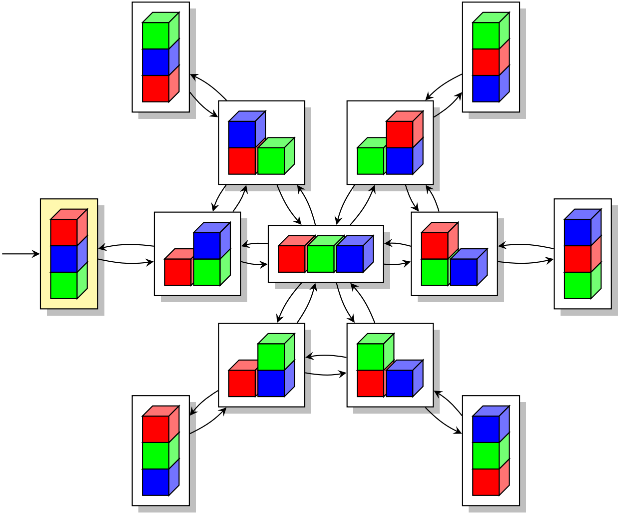
Example: Blocks World



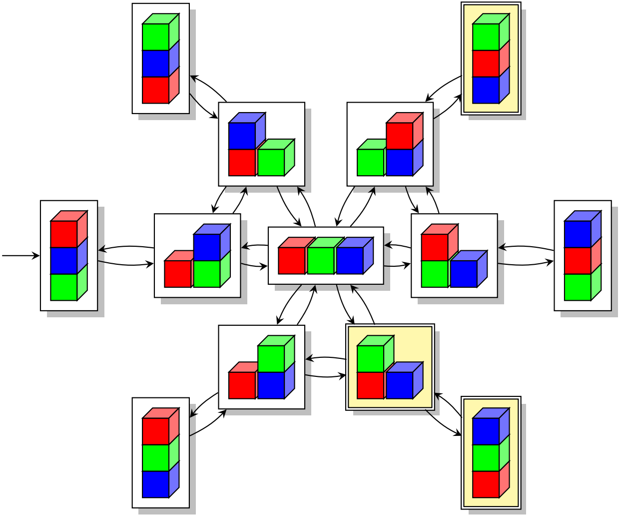
Example: Blocks World



Example: Blocks World



Example: Blocks World



Transition Systems

Definition (Transition system)

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

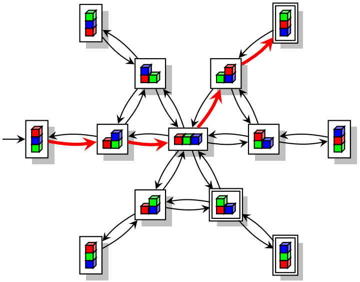
Plan

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.



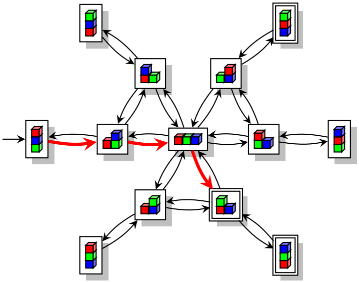
Plan

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 $\langle a_1, \dots, a_n \rangle$ is $\sum_{i=1}^{n} cost(a_i)$.

A plan is optimal if it has minimal cost.



Automated Planning

Definition (Optimal Planning)

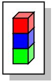
Given an encoding of a transition system, find an optimal plan.

Definition (Satisficing Planning)

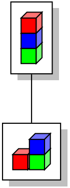
Given an encoding of a transition system, find a (not necessarily optimal) plan.

Cheaper plans are better solutions.

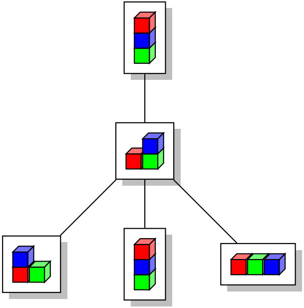
Search in a Nutshell



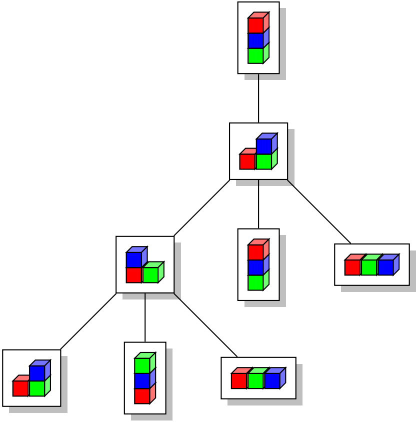
Search in a Nutshell



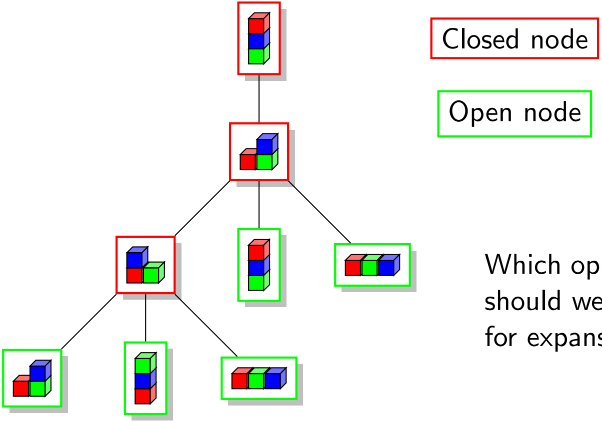
Search in a Nutshell



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?

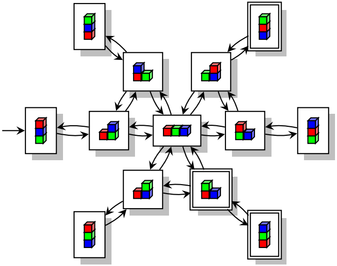
Definition (Transition system)

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

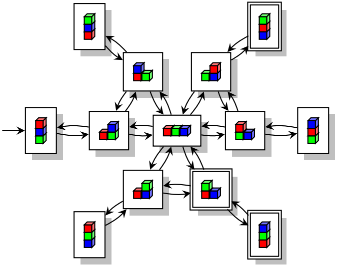


n blocks: more than $n!$ states

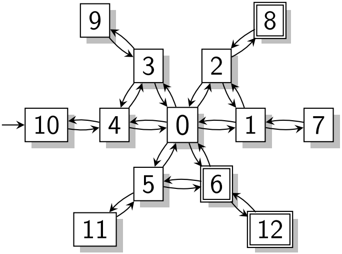
Transition Systems as Input Formalism?

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

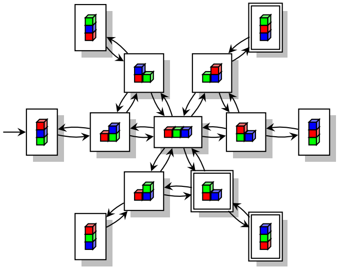


Heuristics require structure

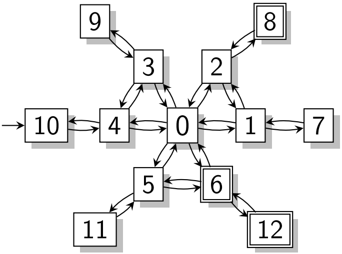
Transition Systems as Input Formalism?

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 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 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 a block onto another

For example:

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

Example: Blocks World in Propositional STRIPS

Example

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

For example:

- $pre(\textit{to-table}_{R,B}) = \{on_{R,B}, clear_R\}$
- $add(\textit{to-table}_{R,B}) = \{\textit{on-table}_R, clear_B\}$
- $del(\textit{to-table}_{R,B}) = \{on_{R,B}\}$
- $cost(\textit{to-table}_{R,B}) = 1$

Example: Blocks World in Propositional STRIPS

Example

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

For example:

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

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: $\{v_1 \mapsto a, v_3 \mapsto b\}$ 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.
- Effects: partial variable assignment

  Example: Effect $\{v_1 \mapsto b, v_2 \mapsto c\}$
  - 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 $dom(v)$,
- 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 $X \in \{R, G, B\}$
- $s_0 = \{on_R \mapsto B, on_G \mapsto \text{Table}, on_B \mapsto G,$
  $clear_R \mapsto T, clear_G \mapsto F, clear_B \mapsto F\}$
- $s_\star = \{on_G \mapsto R\}$
- 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:

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

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

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