Representation for Classical Planning

B(E)4M36PUI – Artificial Intelligence Planning

Antonín Komenda

AIC, FEE, CTU

February 27, 2023

Outline

1. Representations
2. Transition System
3. Planning Formalisms
4. Planning Domain Definition Language
5. Compactness

Representations

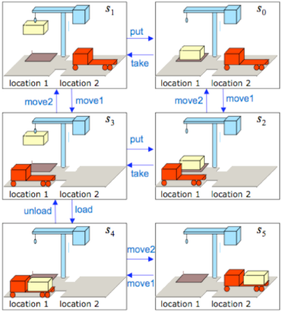
- Transition System (DG[1])
  - states (nodes)
  - transitions (edges)
- Formalism (STRIPS[2], FDR[3] $\cong$ SAS+[4])
  - facts (forming states)
  - actions (set of transitions, preconditions, effects)
  - initial state, goal state(s)
- Language (PDDL[5], NDDL, MA-PDDL, ...)
  - predicates (parametrized facts)
  - operators (parameterized actions)
  - types (optional), objects, functions

[1] Directed Graph
[2] Stanford Research Institute Problem Solver – today used only as formalism
[3] Finite Domain Representation
[4] Simplified Action Structures
[5] Planning Domain Definition Language

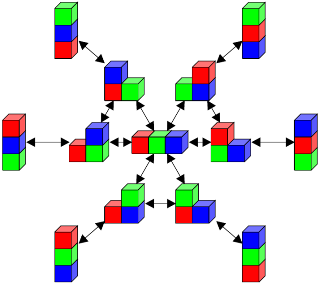
Transition System

Intuition

- model of the planning problem (modeling → description, algorithms)



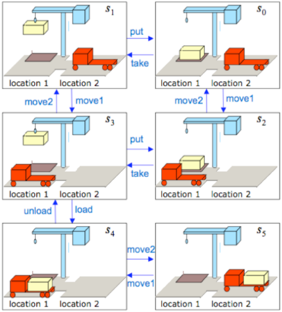
,

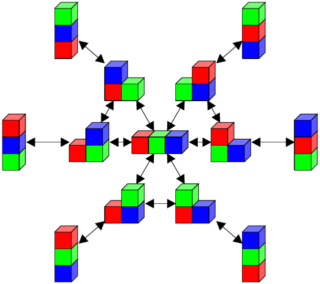


Transition System

Intuition

- modeling a sequential decision problem of modifying a world
- state space of a planning task + transitions between the states
- nodes: describing states (configurations) of the world
- edges: describe transitions between states (modifications of the world)

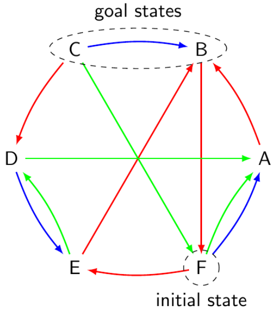




Transition System

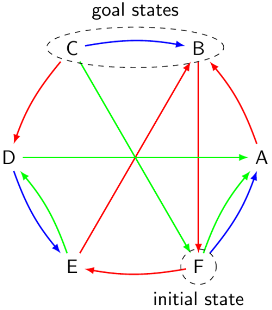
Formally

- transition system is a DG with labeled edges $\Sigma = \langle N, E, L, c \rangle$
- N is a finite set of graph nodes representing the states of the state space
- E is a finite set of edges $e \in E$, between two nodes $e \subseteq N \times N$ representing transitions between two states
- L is a finite set of action labels (one action can label more transitions)
- $c : N \to \mathbb{R}^{0+}$ is a cost function (unit costs planning iff $c(n) \mapsto 1$)



Transition System
Example

- transition system is a DG with labeled edges $\Sigma = \langle N, E, L, c \rangle$
- nodes represent states $N \in \{\mathtt{A}, \ldots, \mathtt{F}\}$, labels actions $L \in \{\mathtt{red}, \mathtt{green}, \mathtt{blue}\}$, edges transitions $E \in \{(\mathtt{C} \overset{\mathtt{blue},1}{\rightarrow} \mathtt{B}), (\mathtt{E} \overset{\mathtt{red},1}{\rightarrow} \mathtt{B}), (\mathtt{A} \overset{\mathtt{red},2}{\rightarrow} \mathtt{B}), \ldots\}$; notation $\overset{l \in L, c(.)}{\rightarrow}$



Transition System

Formally

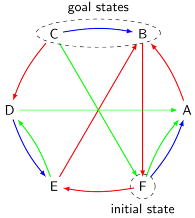
- given a state represented as a node n and an action label l, action application is denoted as nln', where n' is the successor state node
- the application is deterministic there is only one l as outgoing e out of each n
- action with l is applicable in n iff there is a outgoing e with l
- $n' = \mathrm{app}_l(n)$ – application function (non-injective, non-surjective)
- $\mathrm{app}_l : (N' \in 2^N) \to N$ – action application is not necessarily defined for each n
- deterministic planning: a sequence $\sigma = (l_1, l_2, \ldots, l_k)$ of labeled actions $l_1, \ldots, l_k \in L$ and state nodes $n_0, ..., n_k$ (the execution of σ) is a path in the transition system iff:
  1. n_0 is a designated initial state node
  2. $n_i = \mathrm{app}_{l_i}(n_{i-1})$ for every $i \in \{1, ..., k\}$, and
  3. n_k is a goal state.
- equivalently expressed as

$$n_k = \mathrm{app}_{l_k}(\mathrm{app}_{l_{k-1}}(\ldots \mathrm{app}_{l_1}(n_0) \ldots))$$

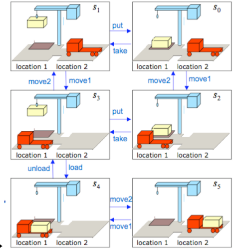
STRIPS/FDR Formalisms

Intuition

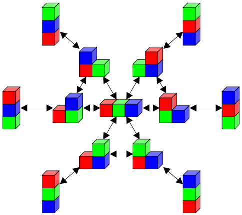
- inner structure of the states (factorization of the state)
- factors $\sim$ facts holding in particular states of the world
- facts about the world
- e.g., "blue block is on the table", "blue block is on the green block", "truck #1 is at location #2", "crane #1 is empty", ...
- subset of zeroth-order (propositional) logic



$\rightsquigarrow$



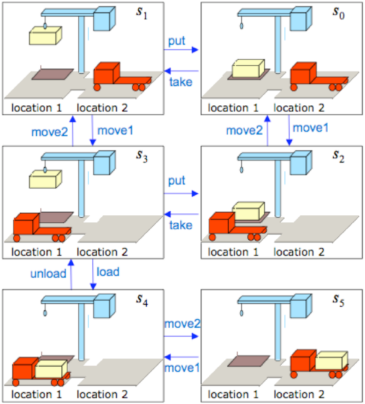
,



STRIPS/FDR Formalisms

Intuition

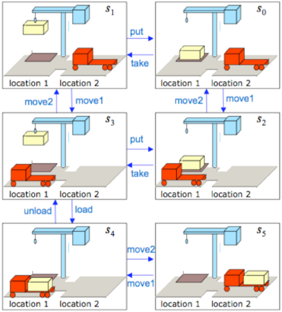
- a planning problem with 1 truck, 1 crane, 2 locations, and 1 crate
- states: $\{s_0, \ldots, s_5\}$
- actions: $\{\texttt{put}, \texttt{take}, \texttt{move1}, \texttt{move2}, \texttt{load}, \texttt{unload}\}$



STRIPS/FDR Formalisms

Intuition

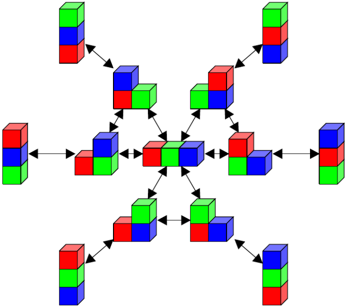
- transition system of the Truck-Crane planning problem
- states $\{\mathtt{s}_0, \ldots, \mathtt{s}_5\}$ describing configuration of the three blocks (nodes in respective Σ)
- actions $\{\mathtt{put}, \mathtt{take}, \mathtt{move1}, \mathtt{move2}, \mathtt{load}, \mathtt{unload}\}$ describing modifications in the world for particular states ($\mathtt{put}$ is one action, but represents 2 transitions!)



STRIPS/FDR Formalisms

Intuition

- transition system of the Blocksworld problem with 3 blocks
- states: describing configuration of the three blocks (on the table and on each other)
- actions: describe stacking and unstacking of each one block clear from the top



STRIPS/FDR Formalisms

Formally (STRIPS)

A planning problem in STRIPS is defined as a tuple $\Pi = \langle F, A, c, s_I, G \rangle$

- finite set of facts $f \in F$
  - facts define a set of all states as $S = 2^F$ (all subsets of F)
  - a particular state is defined as $s \in S$
- finite set of actions $a \in A$, $a = \langle pre(a), add(a), del(a), c(a) \rangle$
  - preconditions: $pre(a) \subseteq F$ (a is applicable in s iff $pre(a) \subseteq s$)
  - additions: $add(a) \subseteq F$ (facts added to s after application)
  - deletions: $del(a) \subseteq F$ (facts removed from s after application
  - action cost: $c : A \to \mathbb{R}^{0+}$
  - application of an action a in state s resulting into state s' is defined:

$$s' = (s \setminus del(a)) \cup add(a)$$

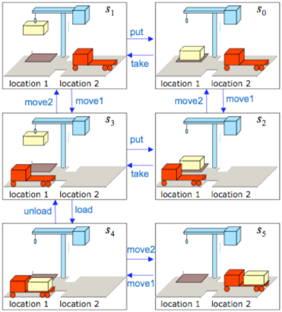
- initial state: $s_I \in S$
- goal condition: $G \subseteq F$, a state s_G is a goal state iff $G \subseteq s_G$

AI CENTER FEE CTU

STRIPS/FDR Formalisms

Example (STRIPS)

- the Truck-Crane problem in STRIPS
- facts: $\{\texttt{crate1-in-truck1}, \texttt{crate1-at-location1}, \ldots\}$
- actions: $\{\texttt{put}, \texttt{take}, \ldots\}$, e.g.:
  - $\mathit{pre}(\texttt{put}) = \{\texttt{crate1-heldby-crane1}, \texttt{location1-platform-empty}\}$
  - $\mathit{add}(\texttt{put}) = \{\texttt{crate1-at-location1}\}$
  - $\mathit{del}(\texttt{put}) = \{\texttt{crate1-heldby-crane1}, \texttt{location1-platform-empty}\}$



STRIPS/FDR Formalisms

Induction of a transition system from STRIPS

- (nodes = states) $\leftarrow$ facts: $n = s$ (defined via facts F)
- edges $\leftarrow$ (labels = actions): $l = a$ (inducing one or more edges e)
- analogous cost function c
- $nln' = sas'$ (transition, action application)
- $\mathrm{app}_l(n) = \mathrm{app}_a(s)$
- $\sigma = \pi$ (path in transition system from the initial state s_I to a goal state $s_k \supseteq G$ is a solution to the STRIPS planning problem, i.e. a plan $\pi = (a_1, \ldots, a_k)$)
- plan π induces a state-action sequence: $s_0, a_1, s_1, \ldots, a_k, s_k$, where $s_I = s_0, s_k \supseteq G$
- equivalently expressed as

$$G \subseteq \mathrm{app}_{a_k}(\mathrm{app}_{a_{k-1}}(\ldots \mathrm{app}_{a_1}(s_I) \ldots))$$

AI CENTER
FEE CTU

STRIPS/FDR Formalisms

Why STRIPS?!?

- STRIPS actions are particularly simple, yet expressive enough to capture general planning problems.
- In particular, STRIPS planning is no easier than general planning problems.
- Many algorithms in the planning literature are easier to present in terns of STRIPS.
- STRIPS states can be represented as binary vectors of size $|F|$
  - each fact either holds in the state (binary value $\top$)
  - or does not hold (binary value $\perp$)

STRIPS/FDR Formalisms

Formally (FDR)

A planning problem in FDR is defined as a tuple $\Pi = \langle V, A, c, s_I, G \rangle$

- variables $v \in V$
  - each variable has a finite domain $dom(v)$ of possible values $dom(v) = \{\mathtt{val}_1, \mathtt{val}_2, \ldots, \mathtt{val}_{dom(v)}\}$
  - a state s is defined as a complete assignment of a value from $dom(v)$ to each variable $v \in V$, denoted as $val_s(v) \mapsto \mathtt{val}_i$
  - a partial assignment over $V' \subseteq V$ defines values only for variables $v \in V'$
- actions $a \in A$, defined as $a = \langle pre(a), eff(a), c(a) \rangle$
  - preconditions: $pre(a)$ is a partial assignment (a is applicable in s iff $\forall v \in pre(a) : val_{pre(a)}(v) = val_s(v)$)
  - effects: $eff(a)$ is a partial assignment (values changed by a)
  - action cost: $c : A \mapsto \mathbb{R}^{0+}$
  - application of an action a in state s resulting into state s' is defined:

$$\forall v \in V : val_{s'}(v) = \begin{cases} val_{eff(a)}(v) & v \in eff(a) \\ val_s(v) & otherwise \end{cases}$$

AI CENTER FEE CTU

STRIPS/FDR Formalisms

Formally (FDR)

A planning problem in FDR is defined as a tuple $\Pi = \langle V, A, c, s_I, G \rangle$

- initial state: $s_I \in S$, where the set of all states $S = dom(v_1) \times dom(v_2) \times \cdots \times dom(v_{|V|})$ for all $v \in V$
- goal condition: is a partial assignment G and a state s_g is a goal iff $\forall v \in G : val_{s_g}(v) = val_G(v)$

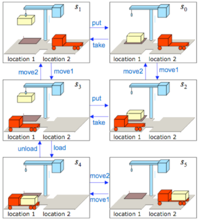
FDR vs STRIPS:

- plan, state-action sequence, expressivity the same as STRIPS
- FDR variable/value pairs $\approx$ STRIPS facts
- usually more efficient implementation
- how many STRIPS facts we need to encode a FDR variable?

STRIPS/FDR Formalisms

Example (FDR)

- the Truck-Crane problem in FDR
- variables: $\{\texttt{crate1}, \texttt{truck1-at}, \texttt{truck1-empty}, \texttt{crane1-empty}, \texttt{location1-platform-empty}\}$
  - $dom(\texttt{crate1}) = \{\texttt{heldby-crane1}, \texttt{on-truck1}, \texttt{at-location1}\}$
  - $dom(\texttt{truck1-at}) = \{\texttt{location1}, \texttt{location2}\}$
  - $dom(\texttt{truck1-empty}) = \{\texttt{true}, \texttt{false}\}$, $dom(\texttt{crane1-empty}) =, \ldots$
- actions: $\{\texttt{put}, \texttt{take}, \ldots\}$, e.g.:
  - $val_{pre(\texttt{put})}(\texttt{crate1}) = \texttt{heldby-crane1}$
  - $val_{pre(\texttt{put})}(\texttt{location1-platform-empty}) = \texttt{true}$
  - $val_{eff(\texttt{put})}(\texttt{crate1}) = \texttt{at-location1}$
  - $val_{eff(\texttt{put})}(\texttt{location1-platform-empty}) = \texttt{false}$



Planning Domain Definition Language (PDDL)

Intuition

- declarative language (high-level principles similar to Prolog)
- subset of first-order (predicate) logic
- s-expressions (LISP syntax)
- domain definition $\times$ instance definition
- easy to describe (exponentially) many STRIPS/FDR facts and actions
- lifted representation (parametrized facts/actions)
- various versions (temporal, continuous variables, negative preconditions, disjunctive goals, ...)
- various extensions (multi-agent, probabilistics, ...)
- de-facto standard language for all automated planners

AI CENTER FEE CTU

Planning Domain Definition Language (PDDL)

Syntax Basics

Domain definition:

```
(define (domain <domain name>)
  (:predicates <predicate-list>)
  (:action <action-details>)
)
```

Problem definition file:

```
(define (problem <title>)
  (:domain <domain-name>)
  (:objects <object-list>)
  (:init <predicates>)
  (:goal <predicates>)
)
```

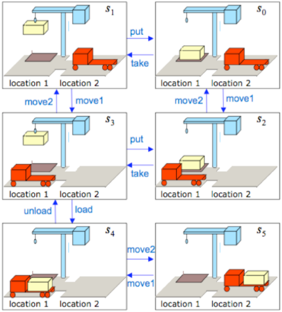
AI CENTER FEE CTU

Planning Domain Definition Language (PDDL)

Planning Domain

Predicates – parametrized facts:

```
(:predicates
  (truck ?truck) (crane ?crane) (crate ?crate) (loc ?loc)
  (at-location ?obj ?loc) (on ?crate ?truck)
  (platform-at-location ?loc)
  (heldby ?crate ?crane) (empty ?obj) ;one empty type
  (road-between ?location1 ?location2))
```



Planning Domain Definition Language (PDDL)

Planning Domain

Operators (aka actions in PDDL):

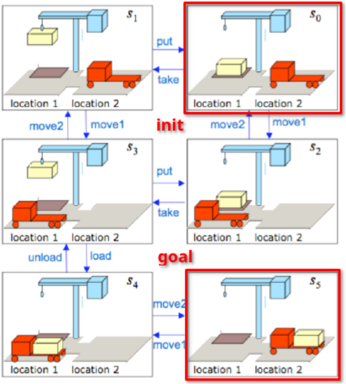
```
(:actions
  (:action move ;one PDDL move for STRIPS move1 and move2
    :parameters (?truck ?from-loc ?to-loc)
    :precondition (and
      (truck ?truck) (loc ?from-loc) (loc ?to-loc) ;types
      (road-between ?from-loc ?to-loc)
      (at-location ?truck ?from-loc)
    :effect (and (not (at-location ?truck ?from-loc))
      (at-location ?truck ?to-loc))
  )
  ...
)
```

AI CENTER
FEE CTU

Planning Domain Definition Language (PDDL)

Planning Problem (Instance)

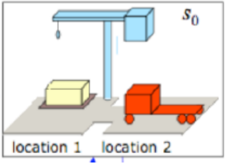
```
(define (problem 1-truck-1-crane-2-locations-1-crate)
  (:domain truck-crane)
  (:objects truck1 crane1 location1 location2 crate1)
  (:init <predicates>)
  (:goal <predicates>))
```



Planning Domain Definition Language (PDDL)

Planning Problem (Instance)

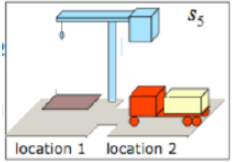
```
(:init
  (truck truck1) (crane crane1) (crate crate1)
  (loc location1) (loc location2)
  (road-between location1 location2)
  (road-between location2 location1)
  (platform-at-location location1) ;red are constant
  (at-location truck1 location2)
  (at-location crane1 location1)
  (at-location crate1 location1)
  (empty crane1) (empty truck1))
```



Planning Domain Definition Language (PDDL)

Planning Problem (Instance)

```
(:goal
  (and ;conjunctive goal facts
    (at-location truck1 location2)
    (on crate1 truck1)
  )
)
```



Planning Domain Definition Language (PDDL)

Grounding of PDDL to STRIPS/FDR

- predicates → facts/variables
- operators → actions
- naive approach (enumeration of all parameter permutations[6]):
  - enumeration of all facts from predicates using objects
  - enumeration of all actions from operators using objects
- some permutations are forbidden (by object types or preconditions)
- some states are unreachable from the initial state (there is no path from init to them in the induced transition system)
- some actions are inapplicable on any path from the initial state (there is no reachable state fulfilling all preconditions of such action)
- if we can detect that a state is a dead-end we do not need to ground following predicates/actions
- some facts are mutually exclusive (mutex) → grounding to FDR

[6] All permutations are with repetition.

Compact Representation

Exponential blowups

- PDDL

  $\downarrow$ exponential blowup #1 (parameter permutations)
- STRIPS/FDR

  $\downarrow$ exponential blowup #2 (fact permutations)
- Transition System (DG)

Parameter permutations in operators:

- analogy in all possible parametrization of a function
- $D_1 \times D_2 \times \cdots \times D_n$, where n is the number of parameters
- $|D_1||D_2| \cdots |D_n| \geq 2^n$ (assuming $|D_i| \geq 2 : i \in \{1, \ldots, n\}$)

Fact permutations in states:

- analogy in compact representation of numbers in prefix notation
- STRIPS $\sim$ digits 1,0; FDR $\sim$ digits of variable domains
- recall $S = 2^F$, i.e., $|S| = 2^{|F|}$