Planning and Optimization

A4. Planning Tasks

Malte Helmert and Gabriele Röger

Universität Basel

October 3, 2016

Introduction

State Variables

How to specify huge transition systems
without enumerating the states?

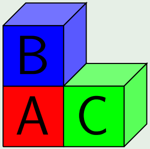
- represent different aspects of the world
  in terms of different Boolean state variables
- treat state variables as atomic propositions
  $\rightsquigarrow$ a state is a valuation of state variables
- n state variables induce 2^n states
  $\rightsquigarrow$ exponentially more compact than "flat" representations

Example: $O(n^2)$ variables suffice for blocks world with n blocks

Blocks World State with Boolean State Variables

Example

$$s(A\text{-}on\text{-}B) = \mathbf{F}$$
$$s(A\text{-}on\text{-}C) = \mathbf{F}$$
$$s(A\text{-}on\text{-}table) = \mathbf{T}$$
$$s(B\text{-}on\text{-}A) = \mathbf{T}$$
$$s(B\text{-}on\text{-}C) = \mathbf{F}$$
$$s(B\text{-}on\text{-}table) = \mathbf{F}$$
$$s(C\text{-}on\text{-}A) = \mathbf{F}$$
$$s(C\text{-}on\text{-}B) = \mathbf{F}$$
$$s(C\text{-}on\text{-}table) = \mathbf{T}$$



Boolean State Variables

Problem:

- How to succinctly represent transitions and goal states?

Idea: Use logical formulas to describe sets of states

- state variables: atomic propositions
- states: all valuations of the state variables
- goal states: defined by a logical formula
- transitions: defined by operators (see following section)

Operators

Syntax of Operators

Definition (Operator)

An operator o over state variables V is an object with three properties:

- a precondition $pre(o)$, a logical formula over V
- an effect $eff(o)$ over V, defined on the following slides
- a cost $cost(o) \in \mathbb{R}_0^+$

Notes:

- Operators are also called actions.
- Operators are often written as triples $\langle pre(o), eff(o), cost(o)\rangle$.
- This can be abbreviated to pairs $\langle pre(o), eff(o)\rangle$ when the cost of the operator is irrelevant.

Operators: Intuition

Intuition for operators o:

- The operator precondition describes the set of states in which a transition labeled with o can be taken.
- The operator effect describes how taking such a transition changes the state.
- The operator cost describes the cost of taking a transition labeled with o.

Syntax of Effects

Definition (Effect)

Effects over state variables V are inductively defined as follows:

- If $v \in V$ is a state variable, then v and $\neg v$ are effects (atomic effect).
- If $e_1, \dots, e_n$ are effects, then $(e_1 \wedge \dots \wedge e_n)$ is an effect (conjunctive effect).
  The special case with $n = 0$ is the empty effect $\top$.
- If χ is a logical formula and e is an effect, then $(\chi \rhd e)$ is an effect (conditional effect).

Parentheses can be omitted when this does not cause ambiguity.

Effects: Intuition

Intuition for effects:

- Atomic effects v and $\neg v$ can be understood as assignments "$v := \mathbf{T}$" and "$v := \mathbf{F}$".
- A conjunctive effect $e = (e_1 \wedge \cdots \wedge e_n)$ means that all subeffects $e_1, \ldots, e_n$ take place simultaneously.
- A conditional effect $e = (\chi \rhd e')$ means that subeffect e' takes place iff χ is true in the state where e takes place.

Semantics of Effects

Definition (Update Set for an Effect)

For all effects e and states s, the update set of e in s, written $[e]_s$, is defined as the following set of literals:

- $[v]_s = \{v\}$ and $[\neg v]_s = \{\neg v\}$ for atomic effects v, $\neg v$
- $[(e_1 \wedge \cdots \wedge e_n)]_s = [e_1]_s \cup \cdots \cup [e_n]_s$
- $[(\chi \rhd e)]_s = \begin{cases} [e]_s & \text{if } s \models \chi \\ \emptyset & \text{otherwise} \end{cases}$

Semantics of Operators

Definition (Applicable, Resulting State)

Let V be a set of state variables.
Let s be a state over V, and let o be an operator over V.

Operator o is applicable in s if $s \models pre(o)$.

If o is applicable in s, the resulting state of applying o in s, written $s[\![o]\!]$, is the state s' defined as follows:

$$s'(v) = \begin{cases} \mathbf{T} & \text{for all } v \in V \text{ with } v \in [\mathit{eff}(o)]_s \\ \mathbf{F} & \text{for all } v \in V \text{ with } \neg v \in [\mathit{eff}(o)]_s \text{ and } v \notin [\mathit{eff}(o)]_s \\ s(v) & \text{for all other } v \in V \end{cases}$$

Semantics of Operators

Definition (Applicable, Resulting State)

Let V be a set of state variables.
Let s be a state over V, and let o be an operator over V.

Operator o is applicable in s if $s \models pre(o)$.

If o is applicable in s, the resulting state of applying o in s, written $s[\![o]\!]$, is the state s' defined as follows:

$$s'(v) = \begin{cases} \mathbf{T} & \text{for all } v \in V \text{ with } v \in [\mathit{eff}(o)]_s \\ \mathbf{F} & \text{for all } v \in V \text{ with } \neg v \in [\mathit{eff}(o)]_s \text{ and } v \notin [\mathit{eff}(o)]_s \\ s(v) & \text{for all other } v \in V \end{cases}$$

Add-after-Delete Semantics

Note:

- The definition implies that if a variable is simultaneously "added" (set to **T**) and "deleted" (set to **F**), the value **T** takes precedence.
- This is called add-after-delete semantics.
- This detail of semantics is somewhat arbitrary, and other definitions are sometimes used.

Applying Operators: Example

Example

Consider the operator $o = \langle a, \neg a \wedge (\neg c \rhd \neg b)\rangle$
and the state $s = \{a \mapsto \mathbf{T}, b \mapsto \mathbf{T}, c \mapsto \mathbf{T}, d \mapsto \mathbf{T}\}$.

The operator o is applicable in s because $s \models a$.

The update set of $\mathit{eff}(o)$ in s is
$[\mathit{eff}(o)]_s = [\neg a]_s \cup [\neg c \rhd \neg b]_s = \{\neg a\} \cup \emptyset = \{\neg a\}$.

The resulting state of applying o in s is the state
$\{a \mapsto \mathbf{F}, b \mapsto \mathbf{T}, c \mapsto \mathbf{T}, d \mapsto \mathbf{T}\}$.

Example Operators: Blocks World

Example (Blocks World Operators)

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

Then blocks world operators can be modeled as:

- $\langle \textit{A-clear} \land \textit{A-on-T} \land \textit{B-clear},\ \textit{A-on-B} \land \neg \textit{A-on-T} \land \neg \textit{B-clear} \rangle$
- $\langle \textit{A-clear} \land \textit{A-on-T} \land \textit{C-clear},\ \textit{A-on-C} \land \neg \textit{A-on-T} \land \neg \textit{C-clear} \rangle$
- $\langle \textit{A-clear} \land \textit{A-on-B},\ \textit{A-on-T} \land \neg \textit{A-on-B} \land \textit{B-clear} \rangle$
- $\langle \textit{A-clear} \land \textit{A-on-C},\ \textit{A-on-T} \land \neg \textit{A-on-C} \land \textit{C-clear} \rangle$
- $\langle \textit{A-clear} \land \textit{A-on-B} \land \textit{C-clear},\ \textit{A-on-C} \land \neg \textit{A-on-B} \land \textit{B-clear} \land \neg \textit{C-clear} \rangle$
- $\langle \textit{A-clear} \land \textit{A-on-C} \land \textit{B-clear},\ \textit{A-on-B} \land \neg \textit{A-on-C} \land \textit{C-clear} \land \neg \textit{B-clear} \rangle$
- ...

Example Operator: 4-Bit Counter

Example (Incrementing a 4-Bit Counter)

Operator to increment a 4-bit number $b_3 b_2 b_1 b_0$ represented by 4 state variables $b_0, \dots, b_3$:

precondition:

$$\neg b_0 \lor \neg b_1 \lor \neg b_2 \lor \neg b_3$$

effect:

$$\begin{gathered}(\neg b_0 \rhd b_0) \land \\ ((\neg b_1 \land b_0) \rhd (b_1 \land \neg b_0)) \land \\ ((\neg b_2 \land b_1 \land b_0) \rhd (b_2 \land \neg b_1 \land \neg b_0)) \land \\ ((\neg b_3 \land b_2 \land b_1 \land b_0) \rhd (b_3 \land \neg b_2 \land \neg b_1 \land \neg b_0))\end{gathered}$$

Planning Tasks

Planning Tasks

Definition (Planning Task)

A planning task is a 4-tuple $\Pi = \langle V, I, O, \gamma \rangle$ where

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

Mapping Planning Tasks to Transition Systems

Definition (Transition System Induced by a Planning Task)

The planning task $\Pi = \langle V, I, O, \gamma \rangle$ induces the transition system $\mathcal{T}(\Pi) = \langle S, L, c, T, s_0, S_\star \rangle$, where

- S is the set of all valuations of V,
- L is the set of operators O,
- $c(o) = cost(o)$ for all operators $o \in O$,
- $T = \{\langle s, o, s' \rangle \mid s \in S,\ o \text{ applicable in } s,\ s' = s[\![o]\!]\}$,
- $s_0 = I$, and
- $S_\star = \{s \in S \mid s \models \gamma\}$.

Planning Tasks: Terminology

- Terminology for transitions systems is also applied to the planning tasks Π that induce them.
- For example, when we speak of the states of Π, we mean the states of $\mathcal{T}(\Pi)$.
- A sequence of operators that forms a solution of $\mathcal{T}(\Pi)$ is called a plan of Π.

Satisficing and Optimal Planning

By planning, we mean the following two algorithmic problems:

Definition (Satisficing Planning)

Given: a planning task Π
Output: a plan for Π, or **unsolvable** if no plan for Π exists

Definition (Optimal Planning)

Given: a planning task Π
Output: a plan for Π with minimal cost among all plans for Π, or **unsolvable** if no plan for Π exists

Summary

Summary

- Planning tasks compactly represent transition systems and are suitable as inputs for planning algorithms.
- Planning tasks are based on concepts from propositional logic, enhanced to model state change.
- States of planning tasks are propositional valuations.
- Operators of planning tasks describe in which situations (precondition) and how (effect) the state of the world can be changed, and at which cost.
- In satisficing planning, we must find a solution for a planning task (or show that no solution exists).
- In optimal planning, we must additionally guarantee that generated solutions are of minimal cost.