Algorithms and Satisfiability

10. Abstraction Heuristics For Planning

It's a Long Way to the Goal, But How Long Exactly?

Álvaro Torralba

AALBORG UNIVERSITET

Spring 2023

Thanks to Jörg Hoffmann for slide sources

Agenda

Agenda

Agenda

Planning

Ambition:

Write one program (planner) that can solve all sequential decision-making problems.

How do we describe our problem to the planner?

- A *logical description* of the possible states
- A *logical description* of the initial state I
- A *logical description* of the goal condition G
- *logical description* of the set A of actions in terms of preconditions and effects

$\rightarrow$ Solution (plan) $=$ sequence of actions from A, transforming I into a state that satisfies G.

Planning

Ambition:

Write one program (planner) that can solve all sequential decision-making problems.

How do we describe our problem to the planner?

- A *logical description* of the possible states
- A *logical description* of the initial state I
- A *logical description* of the goal condition G
- *logical description* of the set A of actions in terms of preconditions and effects

$\rightarrow$ Solution (plan) $=$ sequence of actions from A, transforming I into a state that satisfies G.

$\rightarrow$Here, we focus on the simplest form of planning: Classical Planning. In the mini-project, we will briefly cover other extensions.

Algorithmic Problems in Planning

[redacted]

Input: A planning task Π.

Output: A plan for Π, or "unsolvable" if no plan for Π exists.

[redacted]

Input: A planning task Π.

Output: An *optimal* plan for Π, or "unsolvable" if no plan for Π exists.

Algorithmic Problems in Planning

[redacted]

Input: A planning task Π.
Output: A plan for Π, or "unsolvable" if no plan for Π exists.

[redacted]

Input: A planning task Π.
Output: An *optimal* plan for Π, or "unsolvable" if no plan for Π exists.

$\rightarrow$ The techniques successful for either one of these are almost disjoint. And satisficing planning is *much* more effective in practice.

Algorithmic Problems in Planning

[redacted]

Input: A planning task Π.
Output: A plan for Π, or "unsolvable" if no plan for Π exists.

[redacted]

Input: A planning task Π.
Output: An *optimal* plan for Π, or "unsolvable" if no plan for Π exists.

$\rightarrow$ The techniques successful for either one of these are almost disjoint. And satisficing planning is *much* more effective in practice.

$\rightarrow$ Programs solving these problems are called (optimal) planners, planning systems, or planning tools.

Classical Planning IPC Overview

- **IPC 2000:** Winner heuristic search.
- **IPC 2002:** Winner heuristic search.
- **IPC 2004:** Winner satisficing heuristic search; optimal compilation to SAT.
- **IPC 2006:** Winner satisficing heuristic search; optimal compilation to SAT.
- **IPC 2008:** Winner satisficing heuristic search; optimal symbolic search.
- **IPC 2011:** Winner satisficing heuristic search; optimal heuristic search.
- **IPC 2014:** Winner satisficing heuristic search; optimal symbolic search.
- **IPC 2018:** Winner satisficing heuristic search; optimal portfolio/symbolic search/heuristic search.

→ This and next lecture focus on planning as heuristic search;
Chapter 12 will focus on compilation to SAT and symbolic search.

Classical Planning IPC Overview

- **IPC 2000:** Winner heuristic search.
- **IPC 2002:** Winner heuristic search.
- **IPC 2004:** Winner satisficing heuristic search; optimal compilation to SAT.
- **IPC 2006:** Winner satisficing heuristic search; optimal compilation to SAT.
- **IPC 2008:** Winner satisficing heuristic search; optimal symbolic search.
- **IPC 2011:** Winner satisficing heuristic search; optimal heuristic search.
- **IPC 2014:** Winner satisficing heuristic search; optimal symbolic search.
- **IPC 2018:** Winner satisficing heuristic search; optimal portfolio/symbolic search/heuristic search.

→ This and next lecture focus on planning as heuristic search; Chapter 12 will focus on compilation to SAT and symbolic search.

→ This is a VERY short summary of the history of the IPC! There are many different categories, and many different awards.

Our Agenda for This Topic

Planning and heuristic search were already introduced in the Machine Intelligence course, as it is a sub-area of Artificial Intelligence. Here, we focus on how to (1) design efficient algorithms that can solve planning tasks in practice; and (2) make use of existing planners by encoding your problems as planning tasks.

- **This Chapter:** How to automatically generate a heuristic function, given planning language input?

  → Focusing on heuristic search as the solution method, this is the main question that needs to be answered.
- **Mini-project:** How to use planners to solve your problems?
- **Chapter 11:** How to solve planning using SAT? How to solve planning using BDDs?

  →Other algorithms to solve planning based on the techniques we have seen before.

Agenda

STRIPS Planning: Syntax

Definition (STRIPS Planning Task). *A STRIPS planning task, short planning task, is a 4-tuple* $\Pi = (P, A, I, G)$ *where:*

- P *is a finite set of facts (aka propositions).*
- A *is a finite set of actions; each* $a \in A$ *is a triple* $a = (pre_a, add_a, del_a)$ *of subsets of* P *referred to as the action's precondition, add list, and delete list respectively; we require that* $add_a \cap del_a = \emptyset$*.*
- $I \subseteq P$ *is the initial state.*
- $G \subseteq P$ *is the goal.*

We will often give each action $a \in A$ *a name (a string), and identify* a *with that name.*

STRIPS Planning: Syntax

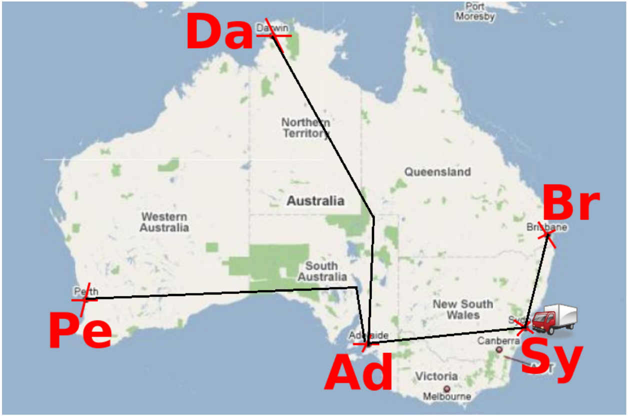
Definition (STRIPS Planning Task). *A STRIPS planning task, short planning task, is a 4-tuple $\Pi = (P, A, I, G)$ where:*

- *P is a finite set of facts (aka propositions).*
- *A is a finite set of actions; each $a \in A$ is a triple $a = (pre_a, add_a, del_a)$ of subsets of P referred to as the action's precondition, add list, and delete list respectively; we require that $add_a \cap del_a = \emptyset$.*
- *$I \subseteq P$ is the initial state.*
- *$G \subseteq P$ is the goal.*

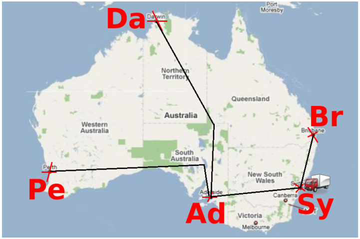
We will often give each action $a \in A$ a name (a string), and identify a with that name.

→We'll see some extensions beyond STRIPS for the mini-project, when we discuss PDDL.

"TSP" in Australia

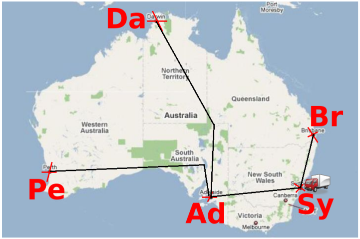


STRIPS Encoding of "TSP"



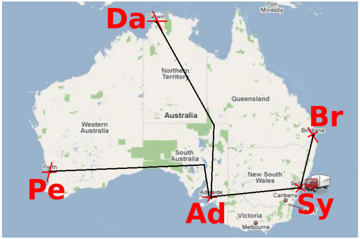
- Facts P: $\{at(x), visited(x) \mid x \in \{Sydney, Adelaide, Brisbane, Perth, Darwin\}\}$.
- Initial state I:

STRIPS Encoding of "TSP"



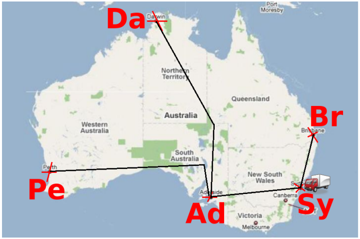
- Facts P: $\{at(x), visited(x) \mid x \in \{Sydney, Adelaide, Brisbane, Perth, Darwin\}\}$.
- Initial state I: $\{at(Sydney), visited(Sydney)\}$.
- Goal G:

STRIPS Encoding of "TSP"



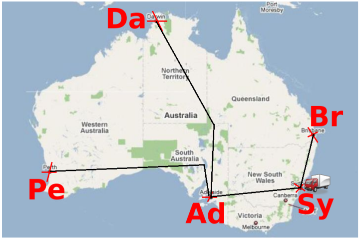
- Facts P: $\{at(x), visited(x) \mid x \in \{Sydney, Adelaide, Brisbane, Perth, Darwin\}\}$.
- Initial state I: $\{at(Sydney), visited(Sydney)\}$.
- Goal G: $\{at(Sydney)\} \cup \{visited(x) \mid x \in \{Sydney, Adelaide, Brisbane, Perth, Darwin\}\}$.
- Actions $a \in A$: $drive(x, y)$ where x, y have a road.
  Precondition pre_a:

STRIPS Encoding of "TSP"



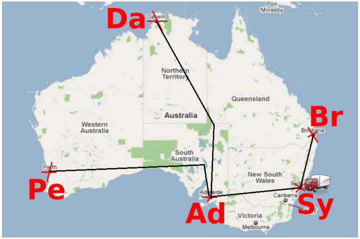
- Facts P: $\{at(x), visited(x) \mid x \in \{Sydney, Adelaide, Brisbane, Perth, Darwin\}\}$.
- Initial state I: $\{at(Sydney), visited(Sydney)\}$.
- Goal G:
  $\{at(Sydney)\} \cup \{visited(x) \mid x \in \{Sydney, Adelaide, Brisbane, Perth, Darwin\}\}$.
- Actions $a \in A$: $drive(x, y)$ where x, y have a road.
  Precondition pre_a: $\{at(x)\}$.
  Add list add_a:

STRIPS Encoding of "TSP"



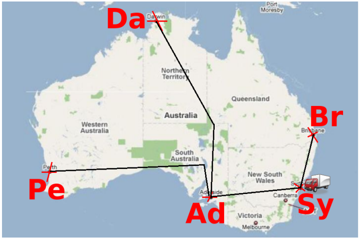
- Facts P: $\{at(x), visited(x) \mid x \in \{Sydney, Adelaide, Brisbane, Perth, Darwin\}\}$.
- Initial state I: $\{at(Sydney), visited(Sydney)\}$.
- Goal G:
  $\{at(Sydney)\} \cup \{visited(x) \mid x \in \{Sydney, Adelaide, Brisbane, Perth, Darwin\}\}$.
- Actions $a \in A$: $drive(x, y)$ where x, y have a road.
  Precondition pre_a: $\{at(x)\}$.
  Add list add_a: $\{at(y), visited(y)\}$.
  Delete list del_a:

STRIPS Encoding of "TSP"



- Facts P: $\{at(x), visited(x) \mid x \in \{Sydney, Adelaide, Brisbane, Perth, Darwin\}\}$.
- Initial state I: $\{at(Sydney), visited(Sydney)\}$.
- Goal G:
  $\{at(Sydney)\} \cup \{visited(x) \mid x \in \{Sydney, Adelaide, Brisbane, Perth, Darwin\}\}$.
- Actions $a \in A$: $drive(x, y)$ where x, y have a road.
  Precondition pre_a: $\{at(x)\}$.
  Add list add_a: $\{at(y), visited(y)\}$.
  Delete list del_a: $\{at(x)\}$.
- Plan:

STRIPS Encoding of "TSP"



- Facts P: $\{at(x), visited(x) \mid x \in \{Sydney, Adelaide, Brisbane, Perth, Darwin\}\}$.
- Initial state I: $\{at(Sydney), visited(Sydney)\}$.
- Goal G:
  $\{at(Sydney)\} \cup \{visited(x) \mid x \in \{Sydney, Adelaide, Brisbane, Perth, Darwin\}\}$.
- Actions $a \in A$: $drive(x, y)$ where x, y have a road.
  Precondition pre_a: $\{at(x)\}$.
  Add list add_a: $\{at(y), visited(y)\}$.
  Delete list del_a: $\{at(x)\}$.
- Plan: $\langle drive(Sydney, Brisbane), drive(Brisbane, Sydney), drive(Sydney, Adelaide), drive(Adelaide, Perth), drive(Perth, Adelaide), drive(Adelaide, Darwin), drive(Darwin, Adelaide), drive(Adelaide, Sydney)\rangle$.

STRIPS Planning: Semantics

Definition (STRIPS State Space). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. The state space of Π is $\Theta_\Pi = (S, A, T, I, S^G)$ where:*

STRIPS Planning: Semantics

Definition (STRIPS State Space). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. The state space of Π is $\Theta_\Pi = (S, A, T, I, S^G)$ where:*

- *The states (also world states) $S = 2^P$ are the subsets of P.*

STRIPS Planning: Semantics

Definition (STRIPS State Space). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. The state space of Π is $\Theta_\Pi = (S, A, T, I, S^G)$ where:*

- *The states (also world states) $S = 2^P$ are the subsets of P.*
- *A is Π's action set.*

STRIPS Planning: Semantics

Definition (STRIPS State Space). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. The state space of Π is $\Theta_\Pi = (S, A, T, I, S^G)$ where:*

- *The states (also world states) $S = 2^P$ are the subsets of P.*
- *A is Π's action set.*
- *The transitions are $T = \{s \xrightarrow{a} s' \mid pre_a \subseteq s, s' = appl(s, a)\}$.*

  If $pre_a \subseteq s$, then a is applicable in s and $appl(s, a) := (s \cup add_a) \setminus del_a$.
  If $pre_a \not\subseteq s$, then $appl(s, a)$ is undefined.

STRIPS Planning: Semantics

Definition (STRIPS State Space). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. The state space of Π is $\Theta_\Pi = (S, A, T, I, S^G)$ where:*

- *The states (also world states) $S = 2^P$ are the subsets of P.*
- *A is Π's action set.*
- *The transitions are $T = \{s \xrightarrow{a} s' \mid pre_a \subseteq s, s' = appl(s, a)\}$.*

  If $pre_a \subseteq s$, then a is applicable in s and $appl(s, a) := (s \cup add_a) \setminus del_a$.
  If $pre_a \not\subseteq s$, then $appl(s, a)$ is undefined.
- *I is Π's initial state.*
- *The goal states $S^G = \{s \in S \mid G \subseteq s\}$ are those that satisfy Π's goal.*

STRIPS Planning: Semantics

Definition (STRIPS State Space). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. The state space of Π is $\Theta_\Pi = (S, A, T, I, S^G)$ where:*

- *The states (also world states) $S = 2^P$ are the subsets of P.*
- *A is Π's action set.*
- *The transitions are $T = \{s \xrightarrow{a} s' \mid pre_a \subseteq s, s' = appl(s, a)\}$.*

  If $pre_a \subseteq s$, then a is applicable in s and $appl(s, a) := (s \cup add_a) \setminus del_a$.
  If $pre_a \not\subseteq s$, then $appl(s, a)$ is undefined.
- *I is Π's initial state.*
- *The goal states $S^G = \{s \in S \mid G \subseteq s\}$ are those that satisfy Π's goal.*

An (optimal) plan for $s \in S$ is an (optimal) solution for s in Θ_Π, i.e., a path from s to some $s' \in S^G$. A solution for I is called a plan for Π. Π is solvable if a plan for Π exists.

STRIPS Planning: Semantics

Definition (STRIPS State Space). *Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. The state space of Π is $\Theta_\Pi = (S, A, T, I, S^G)$ where:*

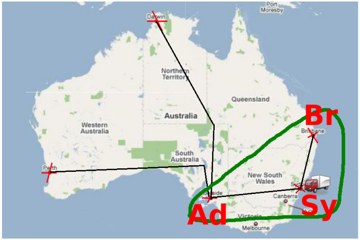
- *The states (also world states) $S = 2^P$ are the subsets of P.*
- *A is Π's action set.*
- *The transitions are $T = \{s \xrightarrow{a} s' \mid pre_a \subseteq s, s' = appl(s, a)\}$.*

  If $pre_a \subseteq s$, then a is applicable in s and $appl(s, a) := (s \cup add_a) \setminus del_a$.
  If $pre_a \not\subseteq s$, then $appl(s, a)$ is undefined.
- *I is Π's initial state.*
- *The goal states $S^G = \{s \in S \mid G \subseteq s\}$ are those that satisfy Π's goal.*

An (optimal) plan for $s \in S$ is an (optimal) solution for s in Θ_Π, i.e., a path from s to some $s' \in S^G$. A solution for I is called a plan for Π. Π is solvable if a plan for Π exists.

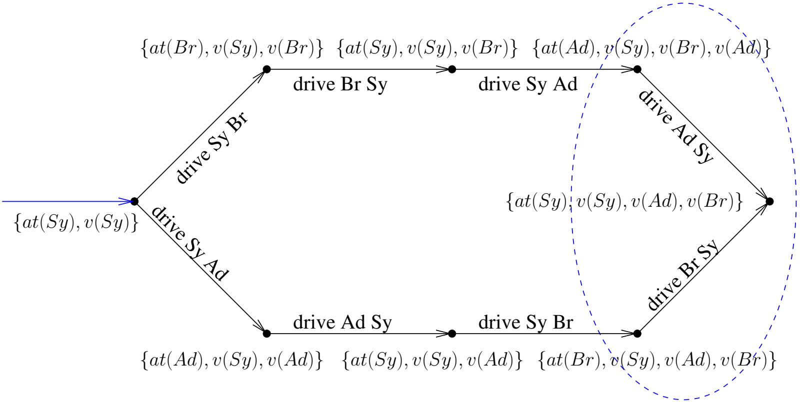
For $\vec{a} = \langle a_1, \dots, a_n \rangle$, $appl(s, \vec{a}) := appl(\dots appl(appl(s, a_1), a_2) \dots, a_n)$ if each a_i is applicable in the respective state; else, $appl(s, \vec{a})$ is undefined.

STRIPS Encoding of Simplified "TSP"

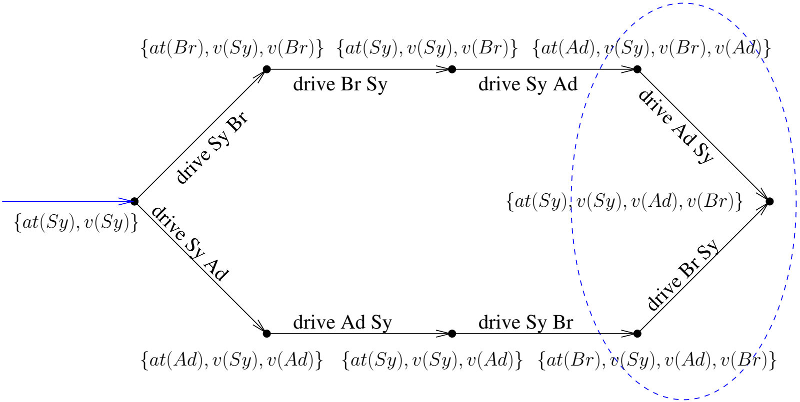


- Facts P: $\{at(x), visited(x) \mid x \in \{Sydney, Adelaide, Brisbane\}\}$.
- Initial state I: $\{at(Sydney), visited(Sydney)\}$.
- Goal G: $\{visited(x) \mid x \in \{Sydney, Adelaide, Brisbane\}\}$. (Note: no "$at(Sydney)$".)
- Actions $a \in A$: $drive(x, y)$ where x, y have a road.
  - Precondition pre_a: $\{at(x)\}$.
  - Add list add_a: $\{at(y), visited(y)\}$.
  - Delete list del_a: $\{at(x)\}$.

STRIPS Encoding of Simplified "TSP": State Space



STRIPS Encoding of Simplified “TSP”: State Space



→ Is this actually the state space?