Optimal Planning and Shortcut Learning: An Unfulfilled Promise

Erez Karpas Carmel Domshlak

Faculty of Industrial Engineering and Management,
Technion — Israel Institute of Technology

May 28, 2013

Outline

STRIPS

- A STRIPS planning problem with action costs is a 5-tuple $\Pi = \langle P, s_0, G, A, C \rangle$
  - P is a set of boolean propositions
  - $s_0 \subseteq P$ is the initial state
  - $G \subseteq P$ is the goal
  - A is a set of actions.
  - Each action is a triple $a = \langle \text{pre}(a), \text{add}(a), \text{del}(a) \rangle$
  - $C : A \rightarrow \mathbb{R}^{0+}$ assigns a cost to each action
- Applying action sequence $\rho = \langle a_0, a_1, \ldots, a_n \rangle$ at state s leads to $s[[\rho]]$
- The cost of action sequence ρ is $\sum_{i=0}^{n} C(a_i)$

Intended Effects

Chicken logic

Why did the chicken cross the road?

Intended Effects

Chicken logic

Why did the chicken cross the road?

To get to the other side

Intended Effects

Chicken logic

Why did the chicken cross the road?

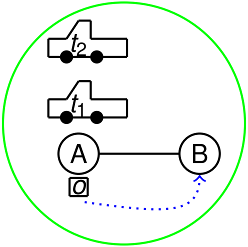
To get to the other side

Observation

Every action along an optimal plan is there for a reason

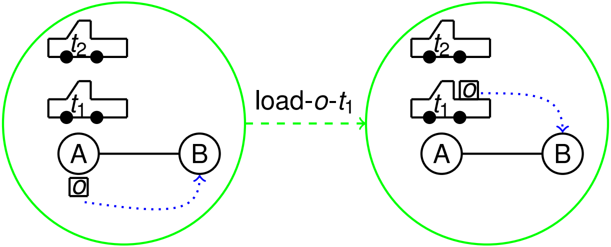
- Achieve a precondition for another action
- Achieve a goal

Intended Effects — Example



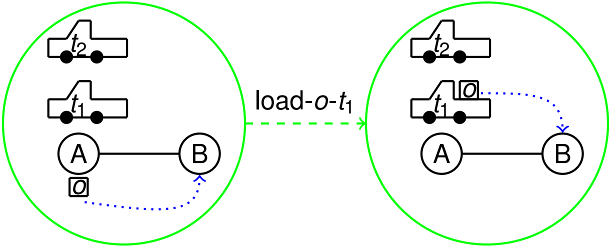
- If ⟨load-o-t_1⟩ is the beginning of an optimal plan, then:
  - There must be a reason for applying load-o-t_1
  - load-o-t_1 achieves o-in-t_1
  - Any continuation of this path to an optimal plan must use some action which requires o-in-t_1

Intended Effects — Example



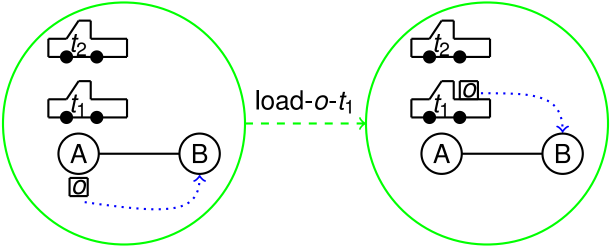
- If $\langle$load-o-$t_1\rangle$ is the beginning of an optimal plan, then:
  - There must be a reason for applying load-o-t_1
  - load-o-t_1 achieves o-in-t_1
  - Any continuation of this path to an optimal plan must use some action which requires o-in-t_1

Intended Effects — Example



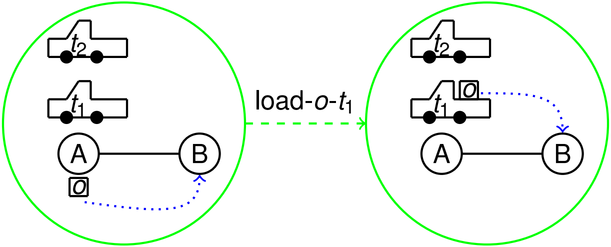
- If $\langle$load-o-$t_1\rangle$ is the beginning of an optimal plan, then:
  - There must be a reason for applying load-o-t_1
  - load-o-t_1 achieves o-in-t_1
  - Any continuation of this path to an optimal plan must use some action which requires o-in-t_1

Intended Effects — Example



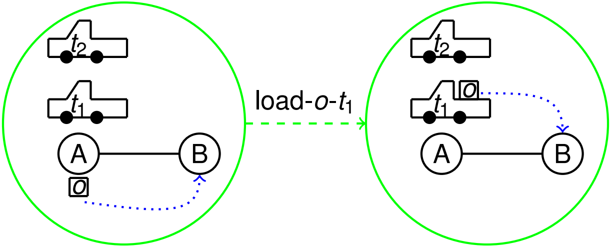
- If $\langle$load-o-$t_1\rangle$ is the beginning of an optimal plan, then:
  - There must be a reason for applying load-o-t_1
  - load-o-t_1 achieves o-in-t_1
  - Any continuation of this path to an optimal plan must use some action which requires o-in-t_1

Intended Effects — Example



- If ⟨load-o-t_1⟩ is the beginning of an optimal plan, then:
  - There must be a reason for applying load-o-t_1
  - load-o-t_1 achieves o-in-t_1
  - Any continuation of this path to an optimal plan must use some action which requires o-in-t_1

Intended Effects — Example



- If $\langle$load-o-$t_1\rangle$ is the beginning of an optimal plan, then:
  - There must be a reason for applying load-o-t_1
  - load-o-t_1 achieves o-in-t_1
  - Any continuation of this path to an optimal plan must use some action which requires o-in-t_1

Intended Effects — Formal Definition

Intended Effects

Given a path $\pi = \langle a_0, a_1, \ldots a_n \rangle$ a set of propositions $X \subseteq s_0 [[\pi]]$ is an intended effect of π iff there exists a path π' such that $\pi \cdot \pi'$ is an optimal plan and π' consumes exactly X, i.e.,
($p \in X$ iff there is a causal link $\langle a_i, p, a_j \rangle$ in $\pi \cdot \pi'$, with $a_i \in \pi$ and $a_j \in \pi'$).

- $\mathrm{IE}(\pi)$ — the set of all intended effect of π

Intended Effects: Complexity

Hard to Find Exactly

It is P-SPACE Hard to find the intended effects of path π.

Sound Approximation

We can use supersets of $\mathrm{IE}(\pi)$ to derive constraints about any continuation of π.

Shortcuts and Approximate Intended Effects

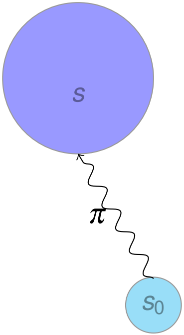
Intuition

X can not be an intended effect of π if there is a cheaper way to achieve X

Shortcuts and Approximate Intended Effects

Intuition

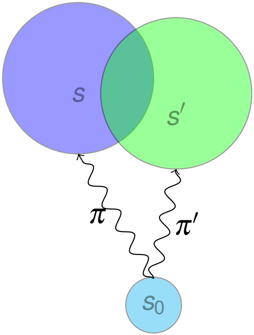
X can not be an intended effect of π if there is a cheaper way to achieve X



Shortcuts and Approximate Intended Effects

Intuition

X can not be an intended effect of π if there is a cheaper way to achieve X

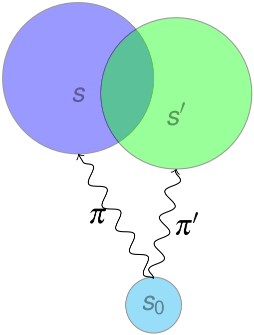


$C(\pi') < C(\pi)$

Shortcuts and Approximate Intended Effects

Intuition

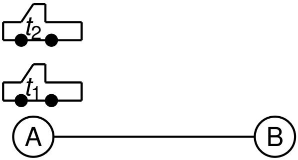
X can not be an intended effect of π if there is a cheaper way to achieve X



$C(\pi') < C(\pi)$

Any continuation of π into an optimal plan must use some fact in $s \setminus s'$

Shortcuts and Approximate Intended Effects: Example



$$\pi = \langle \qquad \rangle$$

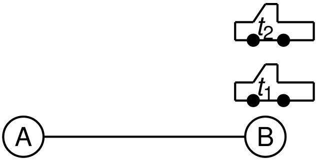
- t_2-at-B can not be an intended effect of π — we must use t_1-at-B
- t_1-at-B can not be an intended effect of π — we must use t_2-at-B

Shortcuts and Approximate Intended Effects: Example

$\pi = \langle$ drive-t_1-A-B $\rangle$

- t_2-at-B can not be an intended effect of π — we must use t_1-at-B
- t_1-at-B can not be an intended effect of π — we must use t_2-at-B

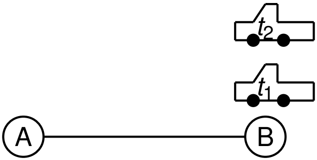
Shortcuts and Approximate Intended Effects: Example



$\pi = \langle$ drive-t_1-A-B ,drive-t_2-A-B $\rangle$

- t_2-at-B can not be an intended effect of π — we must use t_1-at-B
- t_1-at-B can not be an intended effect of π — we must use t_2-at-B

Shortcuts and Approximate Intended Effects: Example

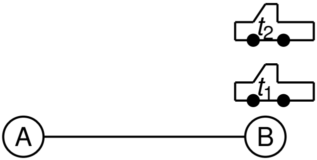


$\pi = \langle$ drive-t_1-A-B ,drive-t_2-A-B $\rangle$

$\pi' = \langle$drive-t_2-A-$B\rangle$

- t_2-at-B can not be an intended effect of π — we must use t_1-at-B
- t_1-at-B can not be an intended effect of π — we must use t_2-at-B

Shortcuts and Approximate Intended Effects: Example

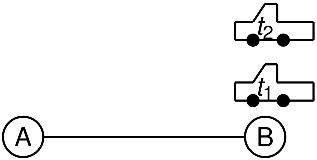


$\pi = \langle$ drive-t_1-A-B ,drive-t_2-A-B $\rangle$

$\pi' = \langle$drive-t_2-A-$B\rangle$

- t_2-at-B can not be an intended effect of π — we must use t_1-at-B
- t_1-at-B can not be an intended effect of π — we must use t_2-at-B

Shortcuts and Approximate Intended Effects: Example



$\pi = \langle$ drive-t_1-A-B ,drive-t_2-A-B $\rangle$

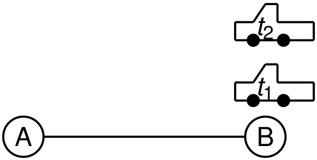
$\pi' = \langle$drive-t_2-A-$B\rangle$

- t_2-at-B can not be an intended effect of π — we must use t_1-at-B

$\pi'' = \langle$drive-t_1-A-$B\rangle$

- t_1-at-B can not be an intended effect of π — we must use t_2-at-B

Shortcuts and Approximate Intended Effects: Example



$\pi = \langle$ drive-t_1-A-B ,drive-t_2-A-B $\rangle$

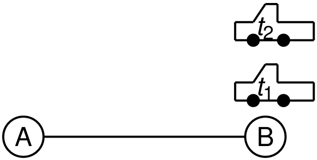
$\pi' = \langle$drive-t_2-A-$B\rangle$

- t_2-at-B can not be an intended effect of π — we must use t_1-at-B

$\pi'' = \langle$drive-t_1-A-$B\rangle$

- t_1-at-B can not be an intended effect of π — we must use t_2-at-B

Shortcuts and Approximate Intended Effects: Example



$\pi = \langle$ drive-t_1-A-B ,drive-t_2-A-B $\rangle$

$\pi' = \langle$drive-t_2-A-$B\rangle$

- t_2-at-B can not be an intended effect of π — we must use t_1-at-B

$\pi'' = \langle$drive-t_1-A-$B\rangle$

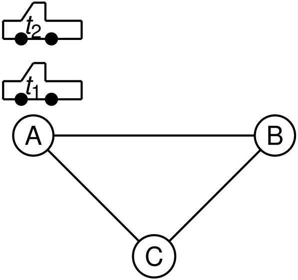
- t_1-at-B can not be an intended effect of π — we must use t_2-at-B

We must use both t_1-at-B and t_2-at-B

Finding Shortcuts

- Where do the shortcuts come from?
- They can be dynamically generated for each path
- Our previous paper used the causal structure of the current path — a graph whose nodes are action occurrences, with an edge from a_i to a_j if there is a causal link where a_i provides some proposition for a_j
- Previous shortcut rules attempted to remove some actions, according to the the causal structure, to obtain a shortcut

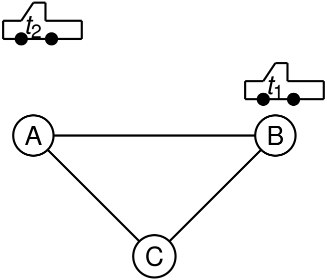
Shortcuts Example



Causal Structure

$\pi = \langle \quad \rangle$

Shortcuts Example

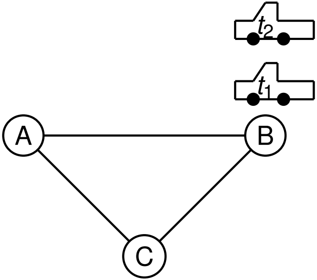


Causal Structure

drive-t_1-A-B

$\pi = \langle$ drive-t_1-A-B $\rangle$

Shortcuts Example



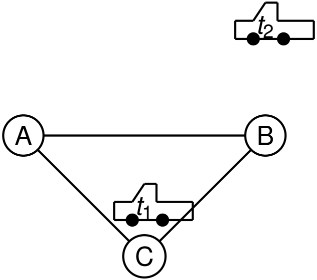
Causal Structure

drive-t_1-A-B

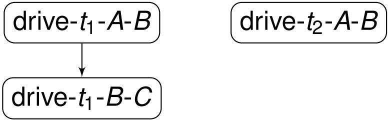
drive-t_2-A-B

$\pi = \langle$ drive-t_1-A-B ,drive-t_2-A-B $\rangle$

Shortcuts Example

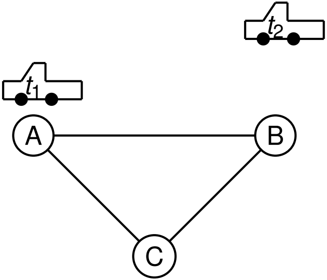


Causal Structure

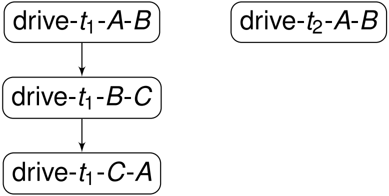


$$\pi = \langle \text{ drive-}t_1\text{-}A\text{-}B \text{ ,drive-}t_2\text{-}A\text{-}B \text{ ,drive-}t_1\text{-}B\text{-}C \qquad\qquad \rangle$$

Shortcuts Example

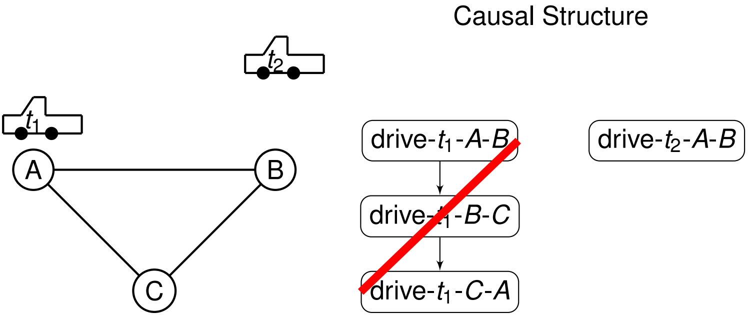


Causal Structure



$\pi = \langle$ drive-t_1-A-B ,drive-t_2-A-B ,drive-t_1-B-C ,drive-t_1-C-A $\rangle$

Shortcuts Example

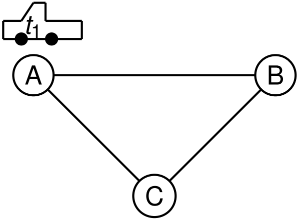


$\pi = \langle$ drive-t_1-A-B ,drive-t_2-A-B ,drive-t_1-B-C ,drive-t_1-C-A $\rangle$

$\pi' = \langle$drive-t_2-A-$B\rangle$

Shortcut Rules that Add Actions

- The previous shortcut rules only remove actions from π

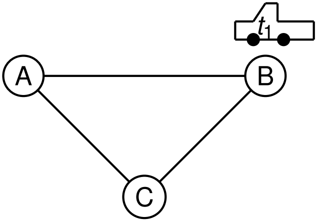


$\pi = \langle \qquad \rangle$

- The previous shortcut rules can not generate the shortcut $\pi' = \langle$drive-t_1-A-$C\rangle$

Shortcut Rules that Add Actions

- The previous shortcut rules only remove actions from π

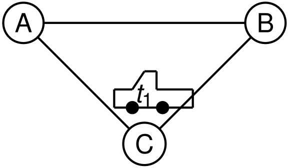


$\pi = \langle$ drive-t_1-A-B $\rangle$

- The previous shortcut rules can not generate the shortcut $\pi' = \langle$drive-t_1-A-$C\rangle$

Shortcut Rules that Add Actions

- The previous shortcut rules only remove actions from π

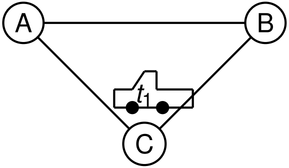


$\pi = \langle$ drive-t_1-A-B ,drive-t_1-B-C $\rangle$

- The previous shortcut rules can not generate the shortcut $\pi' = \langle$drive-t_1-A-$C\rangle$

Shortcut Rules that Add Actions

- The previous shortcut rules only remove actions from π



$\pi = \langle$ drive-t_1-A-B ,drive-t_1-B-C $\rangle$

- The previous shortcut rules can not generate the shortcut $\pi' = \langle$drive-t_1-A-$C\rangle$

Outline

1. Background
2. Learning Shortcut Rules
3. Empirical Evaluation

Learning Shortcut Rules that Add Actions

- We developed techniques for learning shortcut rules that add new actions

Spoiler

These shortcut rules do not improve performance

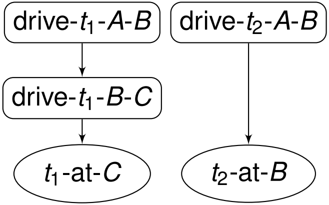
When to Learn

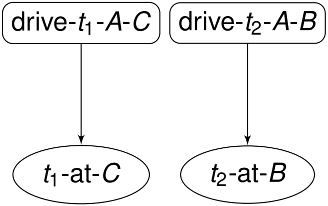
- Recall that a good shortcut π' achieves almost everything the original path π did
- In the extreme: π' achieves everything π achieved
- The search algorithm detects when this happens — when a new path to an existing state is detected
- We learn whenever we have two paths reaching the same state, regardless of whether the new path is cheaper or not

How to Learn

- The input to our learning algorithm is two paths reaching the same state
- Instead of looking at the state as a whole, we look at individual facts, and the causal structure leading to each fact

π	π'
$\langle$ drive-t_1-A-B drive-t_2-A-B drive-t_1-B-C $\rangle$	$\langle$ drive-t_1-A-C drive-t_2-A-B $\rangle$

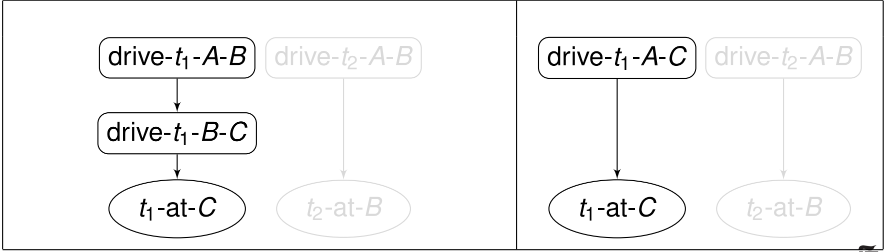




How to Learn

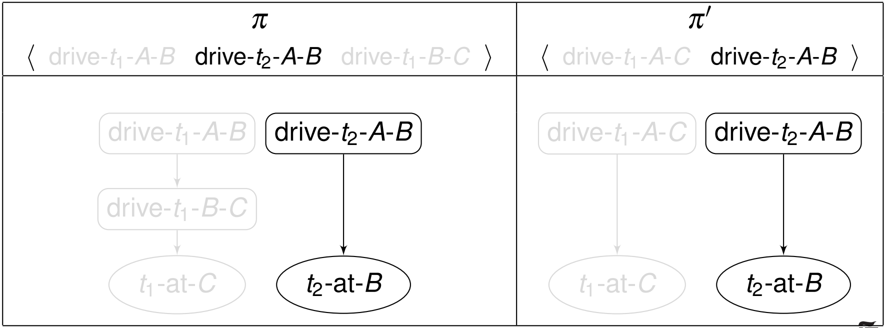
- The input to our learning algorithm is two paths reaching the same state
- Instead of looking at the state as a whole, we look at individual facts, and the causal structure leading to each fact

π	π'
$\langle$ drive-t_1-A-B drive-t_2-A-B drive-t_1-B-C $\rangle$	$\langle$ drive-t_1-A-C drive-t_2-A-B $\rangle$



How to Learn

- The input to our learning algorithm is two paths reaching the same state
- Instead of looking at the state as a whole, we look at individual facts, and the causal structure leading to each fact



Shortcut Rules

- From the pair of partial paths reaching each fact, we learn a new shortcut rule
- Shortcut rules are used when a path π is evaluated, as follows:
  1. Each shortcut rule is checked for applicability
  2. If it is applicable, a set of shortcuts is generated
  3. From each such shortcut, an $\exists$-opt landmark is derived
- Three types of shortcut rules, which differ in the details of these steps

Concrete Shortcut Rule

Rule

$\langle \text{drive-}t_1\text{-}A\text{-}B, \text{drive-}t_1\text{-}B\text{-}C \rangle \leftarrow \langle \text{drive-}t_1\text{-}A\text{-}C \rangle$

Concrete Shortcut Rule

Rule

$\langle$drive-t_1-A-B, drive-t_1-B-$C\rangle \leftarrow \langle$drive-$t_1$-$A$-$C\rangle$

Example 1

$\pi = \langle$drive-t_2-C-A, drive-t_1-A-B, drive-t_1-B-C, drive-t_2-A-$B\rangle$

Concrete Shortcut Rule

Rule

$\langle$drive-t_1-A-B, drive-t_1-B-$C\rangle \leftarrow \langle$drive-$t_1$-$A$-$C\rangle$

Example 1

$\pi = \langle$drive-t_2-C-A, drive-t_1-A-B, drive-t_1-B-C, drive-t_2-A-$B\rangle$

Concrete Shortcut Rule

Rule

$\langle$drive-t_1-A-B, drive-t_1-B-$C\rangle \leftarrow \langle$drive-$t_1$-$A$-$C\rangle$

Example 1

$\pi = \langle$drive-t_2-C-A, drive-t_1-A-B, drive-t_1-B-C, drive-t_2-A-$B\rangle$

$\langle$drive-t_2-C-A, drive-t_1-A-$C\rangle$

Concrete Shortcut Rule

Rule

$\langle$drive-t_1-A-B, drive-t_1-B-$C\rangle \leftarrow \langle$drive-$t_1$-$A$-$C\rangle$

Example 1

$\pi = \langle$drive-t_2-C-A, drive-t_1-A-B, drive-t_1-B-C, drive-t_2-A-$B\rangle$

$\langle$drive-t_2-C-A, drive-t_1-A-$C\rangle$
$\langle$drive-t_2-C-A, drive-t_1-A-C, drive-t_2-A-$B\rangle$

Concrete Shortcut Rule

Rule

$\langle$drive-t_1-A-B, drive-t_1-B-$C\rangle \leftarrow \langle$drive-$t_1$-$A$-$C\rangle$

Example 1

$\pi = \langle$drive-t_2-C-A, drive-t_1-A-B, drive-t_1-B-C, drive-t_2-A-$B\rangle$

$\langle$drive-t_2-C-A, drive-t_1-A-$C\rangle$
$\langle$drive-t_2-C-A, drive-t_1-A-C, drive-t_2-A-$B\rangle$

Example 2

$\pi = \langle$drive-t_2-C-A, drive-t_1-A-B, drive-t_2-A-B, drive-t_1-B-$C\rangle$

Concrete Shortcut Rule

Rule

$\langle$drive-t_1-A-B, drive-t_1-B-$C\rangle \leftarrow \langle$drive-$t_1$-$A$-$C\rangle$

Example 1

$\pi = \langle$drive-t_2-C-A, drive-t_1-A-B, drive-t_1-B-C, drive-t_2-A-$B\rangle$

$\langle$drive-t_2-C-A, drive-t_1-A-$C\rangle$
$\langle$drive-t_2-C-A, drive-t_1-A-C, drive-t_2-A-$B\rangle$

Example 2

$\pi = \langle$drive-t_2-C-A, drive-t_1-A-B, drive-t_2-A-B, drive-t_1-B-$C\rangle$

Concrete Shortcut Rule

Rule

$\langle$drive-t_1-A-B, drive-t_1-B-$C\rangle \leftarrow \langle$drive-$t_1$-$A$-$C\rangle$

Example 1

$\pi = \langle$drive-t_2-C-A, drive-t_1-A-B, drive-t_1-B-C, drive-t_2-A-$B\rangle$

$\langle$drive-t_2-C-A, drive-t_1-A-$C\rangle$
$\langle$drive-t_2-C-A, drive-t_1-A-C, drive-t_2-A-$B\rangle$

Example 2

$\pi = \langle$drive-t_2-C-A, drive-t_1-A-B, drive-t_2-A-B, drive-t_1-B-C $\rangle$

Not applicable

Unordered Shortcut Rule

Rule

$\{\text{drive-}t_1\text{-}A\text{-}B, \text{drive-}t_1\text{-}B\text{-}C\} \leftarrow \{\text{drive-}t_1\text{-}A\text{-}C\}$

Example 1

$\pi = \langle \text{drive-}t_2\text{-}C\text{-}A, \text{drive-}t_1\text{-}A\text{-}B, \text{drive-}t_1\text{-}B\text{-}C, \text{drive-}t_2\text{-}A\text{-}B \rangle$

$\langle \text{drive-}t_2\text{-}C\text{-}A, \text{drive-}t_1\text{-}A\text{-}C \rangle$
$\langle \text{drive-}t_2\text{-}C\text{-}A, \text{drive-}t_1\text{-}A\text{-}C, \text{drive-}t_2\text{-}A\text{-}B \rangle$

Example 2

$\pi = \langle \text{drive-}t_2\text{-}C\text{-}A, \text{drive-}t_1\text{-}A\text{-}B, \text{drive-}t_2\text{-}A\text{-}B, \text{drive-}t_1\text{-}B\text{-}C \rangle$

$\langle \text{drive-}t_2\text{-}C\text{-}A, \text{drive-}t_1\text{-}A\text{-}C \rangle$
$\langle \text{drive-}t_2\text{-}C\text{-}A, \text{drive-}t_1\text{-}A\text{-}C, \text{drive-}t_2\text{-}A\text{-}B \rangle$

Generalized Shortcut Rule

Rule

$\{\text{drive-}T\text{-}X\text{-}Y, \text{drive-}T\text{-}Y\text{-}Z\} \leftarrow \{\text{drive-}T\text{-}X\text{-}Z\}$

Example 1

$\pi = \langle\ \text{drive-}t_1\text{-}B\text{-}C, \text{drive-}t_2\text{-}A\text{-}B, \text{drive-}t_1\text{-}C\text{-}A\rangle$

$\langle\text{drive-}t_1\text{-}B\text{-}A, \text{drive-}t_2\text{-}A\text{-}B\ \rangle$

The Utility Problem

- Checking if a shortcut rule is applicable takes time
- Sometimes, this applicabilty check says the rule is not applicable
- This is the well known utility problem

- We address it by keeping counts of how many times each rule was checked for applicability, and how many times it was applicable
- Low utlity shortcut rules are discarded

Outline

1. Background
2. Learning Shortcut Rules
3. Empirical Evaluation

Coverage

Shortcut Rules	Solved Problems
none	**661**
concrete	609
unordered	585
generalized	487

- None of the shortcut rules solve more problems than no shortcuts in any domain

Expansions

Shortcut Rules	Total Expanded States
none	3785517
concrete	3758042
unordered	**3736052**
generalized	3777860

- The reduction in total number of expanded state for commonly solved problems is about 1%

Discussions

- Possible reasons for failure:
  - Concrete shortcut rules are too strict
  - Unordered and generalized shortcut rules generate too many possible shortcuts, and we only look at some of them
  - The base heuristic ($\exists$-opt and regular landmarks) is already very powerful
- Future work:
  - Exploit the partial order information from the causal structure
  - Smarter ways of applying unordered/generalized shortcut rules
  - Inter-problem learning with generalized shortcut rules
  - Insert your idea here

Discussions

- Possible reasons for failure:
  - Concrete shortcut rules are too strict
  - Unordered and generalized shortcut rules generate too many possible shortcuts, and we only look at some of them
  - The base heuristic ($\exists$-opt and regular landmarks) is already very powerful
- Future work:
  - Exploit the partial order information from the causal structure
  - Smarter ways of applying unordered/generalized shortcut rules
  - Inter-problem learning with generalized shortcut rules
  - Insert your idea here

Thank You