Operator Pruning using Lifted Mutex Groups via Compilation on Lifted Level

Daniel Fišer

Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
danfis@danfis.cz
Representations of Classical Planning Tasks

PDDL – lifted representation

- Types: (:types vehicle location)
- Objects: (:objects car1 car2 - vehicle A B C - location)
- Predicates: (at ?x - vehicle ?y - location)
- Action: (move ?v - vehicle ?f - location ?t - location)
- Initial state: $\psi_I$, Goal $\psi_G$

STRIPS – ground representation

- Facts: $\mathcal{F} = \{(at \ car1 \ A), (at \ car1 \ B), \ldots\}$
- State: $s = \{(at \ car1 \ A), \ldots\} \subseteq \mathcal{F}$
- Operators $\mathcal{O}$, $o = \langle pre(o) \subseteq \mathcal{F}, add(o) \subseteq \mathcal{F}, del(o) \subseteq \mathcal{F} \rangle$
- Initial state $I \subseteq \mathcal{F}$, Goal $G \subseteq \mathcal{F}$. 
Mutex Groups in STRIPS

Mutex Group (in STRIPS)
A set of facts \( M \subseteq F \) is a **mutex group** if \( |M \cap s| \leq 1 \) for every reachable state \( s \).

Fact-Alternating Mutex Group (in STRIPS)
A set of facts \( M \subseteq F \) is a **fam-group** if \( |M \cap I| \leq 1 \) and \( |M \cap \text{add}(o)| \leq |M \cap \text{pre}(o) \cap \text{del}(o)| \) for every operator \( o \in \mathcal{O} \).

Barman: Example fam-groups
- \( \text{handempty}(\text{hand}_1), \text{holding}(\text{hand}_1, \text{shot}_1), \text{holding}(\text{hand}_1, \text{shaker}_1) \)
- \( \text{contains}(\text{shot}_1, \text{cocktail}_1), \text{clean}(\text{shot}_1), \text{used}(\text{shot}_1, \text{cocktail}_1), \text{used}(\text{shot}_1, \text{ingredient}_1), \text{used}(\text{shot}_1, \text{ingredient}_2) \)
Facts from a mutex group are pairwise mutex, i.e., they cannot appear together in any state.

**Barman: Example Pruning of Unreachable Operators**

**Mutex group:**
- `handempty(hand_1)`, `holding(hand_1, shot_1)`, `holding(hand_1, shaker_1)`

**Operator** `fill-shot(shot_1, i, hand_1, hand_1, d)`

pre(o) = \{`handempty(hand_1)`, `holding(hand_1, shot_1)`, ...\}
Facts from a mutex group are pairwise mutex, i.e., they cannot appear together in any state.

Barman: Example Pruning of Unreachable Operators

Mutex group:

- `handempty(hand_1), holding(hand_1, shot_1), holding(hand_1, shaker_1)`

Operator `fill-shot(shot_1, i, hand_1, hand_1, d) o`:

`pre(o) = {handempty(hand_1), holding(hand_1, shot_1), ...}`
Fam-groups can detect dead-end operators: Let $o$ be an operator, let $M$ be a fam-group. If $M \cap G \neq \emptyset$ and $M \cap \text{pre}(o) \cap \text{del}(o) \neq \emptyset$ and $M \cap \text{add}(o) = \emptyset$, then $o$ is a dead-end operator. (F & Komenda, 2018)

Barman: Example Pruning of Dead-End Operators

Fam-group $M$: contains($\text{shot}_1, \text{cocktail}_1$), clean($\text{shot}_1$), used($\text{shot}_1, \text{cocktail}_1$),...

Goal $G = \{\text{contains}(\text{shot}_1, \text{cocktail}_1)\}$.

Operator empty-shot($\text{hand}_1, \text{shot}_1, \text{cocktail}_1$) $o$:
pre($o$) = \{holding($\text{hand}_1, \text{shot}_1$), contains($\text{shot}_1, \text{cocktail}_1$)\}
del($o$) = \{contains($\text{shot}_1, \text{cocktail}_1$)\}
add($o$) = \{empty($\text{shot}_1$)\}
Fam-groups can detect dead-end operators: Let $o$ be an operator, let $M$ be a fam-group. If $M \cap G \neq \emptyset$ and $M \cap \text{pre}(o) \cap \text{del}(o) \neq \emptyset$ and $M \cap \text{add}(o) = \emptyset$, then $o$ is a dead-end operator. (F & Komenda, 2018)

Barman: Example Pruning of Dead-End Operators

Fam-group $M$:
\text{contains}(\text{shot}_1, \text{cocktail}_1), \text{clean}(\text{shot}_1), \text{used}(\text{shot}_1, \text{cocktail}_1), \ldots$

Goal $G = \{\text{contains}(\text{shot}_1, \text{cocktail}_1)\}$.

Operator $\text{empty-shot}(\text{hand}_1, \text{shot}_1, \text{cocktail}_1)$ $o$:
\text{pre}(o) = \{\text{holding}(\text{hand}_1, \text{shot}_1), \text{contains}(\text{shot}_1, \text{cocktail}_1)\}
\text{del}(o) = \{\text{contains}(\text{shot}_1, \text{cocktail}_1)\}
\text{add}(o) = \{\text{empty}(\text{shot}_1)\}$
Fam-groups can detect dead-end operators: Let \( o \) be an operator, let \( M \) be a fam-group. If \( M \cap G \neq \emptyset \) and \( M \cap \text{pre}(o) \cap \text{del}(o) \neq \emptyset \) and \( M \cap \text{add}(o) = \emptyset \), then \( o \) is a dead-end operator. (F & Komenda, 2018)

Barman: Example Pruning of Dead-End Operators

Fam-group \( M \): 
\( \text{contains}(\text{shot}_1, \text{cocktail}_1), \text{clean}(\text{shot}_1), \text{used}(\text{shot}_1, \text{cocktail}_1), \ldots \)

Goal \( G = \{ \text{contains}(\text{shot}_1, \text{cocktail}_1) \} \).

Operator \( \text{empty-shot}(\text{hand}_1, \text{shot}_1, \text{cocktail}_1) \) \( o \): 
\( \text{pre}(o) = \{ \text{holding}(\text{hand}_1, \text{shot}_1), \text{contains}(\text{shot}_1, \text{cocktail}_1) \} \)
\( \text{del}(o) = \{ \text{contains}(\text{shot}_1, \text{cocktail}_1) \} \)
\( \text{add}(o) = \{ \text{empty}(\text{shot}_1) \} \)
Fam-groups can detect dead-end operators: Let \( o \) be an operator, let \( M \) be a fam-group. If \( M \cap G \neq \emptyset \) and \( M \cap \text{pre}(o) \cap \text{del}(o) \neq \emptyset \) and \( M \cap \text{add}(o) = \emptyset \), then \( o \) is a dead-end operator. (F & Komenda, 2018)

Barman: Example Pruning of Dead-End Operators

Fam-group \( M \):
contains(\( \text{shot}_1, \text{cocktail}_1 \)), clean(\( \text{shot}_1 \)), used(\( \text{shot}_1, \text{cocktail}_1 \)), ...

Goal \( G = \{ \text{contains}(\text{shot}_1, \text{cocktail}_1) \} \).

Operator empty-shot(\( \text{hand}_1, \text{shot}_1, \text{cocktail}_1 \)) \( o \):
pre(\( o \)) = \{ holding(\( \text{hand}_1, \text{shot}_1 \), contains(\( \text{shot}_1, \text{cocktail}_1 \)) \}
del(\( o \)) = \{ contains(\( \text{shot}_1, \text{cocktail}_1 \)) \}
add(\( o \)) = \{ \text{empty}(\text{shot}_1) \}
Lifted (Fact-Alternating) Mutex Groups

Lifted fam-group:

- handempty($v : \text{hand}$), holding($v : \text{hand}$, $c : \text{container}$)
  where $v$ is \textbf{fixed} variable, $c$ is \textbf{counted} variable.

Corresponding ground fam-groups:

- handempty($\text{hand}_1$), holding($\text{hand}_1$, $\text{shot}_1$), holding($\text{hand}_1$, $\text{shaker}_1$)
- handempty($\text{hand}_2$), holding($\text{hand}_2$, $\text{shot}_1$), holding($\text{hand}_2$, $\text{shaker}_1$)
Lifted (Fact-Alternating) Mutex Groups

Lifted fam-group:

- handempty($v : \text{hand}$), holding($v : \text{hand}, c : \text{container}$)
  where $v$ is **fixed** variable, $c$ is **counted** variable.

Corresponding ground fam-groups:

- handempty($\text{hand}_1$), holding($\text{hand}_1, \text{shot}_1$), holding($\text{hand}_1, \text{shaker}_1$)
- handempty($\text{hand}_2$), holding($\text{hand}_2, \text{shot}_1$), holding($\text{hand}_2, \text{shaker}_1$)
Lifted (Fact-Alternating) Mutex Groups

Lifted fam-group:

- handempty($v : \text{hand}$), holding($v : \text{hand}, c : \text{conatiner}$)
  where $v$ is **fixed** variable, $c$ is **counted** variable.

Corresponding ground fam-groups:

- handempty($\text{hand}_1$), holding($\text{hand}_1, \text{shot}_1$), holding($\text{hand}_1, \text{shaker}_1$)
- handempty($\text{hand}_2$), holding($\text{hand}_2, \text{shot}_1$), holding($\text{hand}_2, \text{shaker}_1$)
Lifted (Fact-Alternating) Mutex Groups

Lifted fam-group:

- handempty($v : \text{hand}$), holding($v : \text{hand}, c : \text{container}$)
  where $v$ is \textbf{fixed} variable, $c$ is \textbf{counted} variable.

Corresponding ground fam-groups:

- handempty($\text{hand}_1$), holding($\text{hand}_1, \text{shot}_1$), holding($\text{hand}_1, \text{shaker}_1$)
- handempty($\text{hand}_2$), holding($\text{hand}_2, \text{shot}_1$), holding($\text{hand}_2, \text{shaker}_1$)
### Unifier

A **substitution** $\sigma$ is a function mapping variables and objects to variables and objects so that (i) it acts as identity on objects, and (ii) mapping from variables must respect types.

Given a set of atoms $A$, a substitution $\sigma$ is call a **unifier for** $A$ if $\sigma(A)$ is a singleton.

---

| Operator Pruning using Lifted Mutex Groups | 7 / 15 |
Unifier

A substitution $\sigma$ is a function mapping variables and objects to variables and objects so that (i) it acts as identity on objects, and (ii) mapping from variables must respect types. Given a set of atoms $A$, a substitution $\sigma$ is called a unifier for $A$ if $\sigma(A)$ is a singleton.

Pruning of Unreachable Operator

- **fam-group**: `handempty(v : hand), holding(v : hand, c : container)`
- **Action**: `fill-shot(s : shot, i : ingredient, hand_1, hand_1, d : dispenser)`
  
  $\text{pre}(a) = \{\text{holding(hand}_1, s), \text{handempty(hand}_1), \ldots\}$

- There is a unifier $\sigma$: $\sigma(v) = \text{hand}_1, \sigma(c) = s$ for both
  
  $\{\text{holding(v, c), holding(hand}_1, s}\}$ and
  
  $\{\text{handempty(v), handempty(hand}_1)\}$. 

---

Pruning with Lifted Fam-Groups
A substitution $\sigma$ is a function mapping variables and objects to variables and objects so that (i) it acts as identity on objects, and (ii) mapping from variables must respect types.

Given a set of atoms $A$, a substitution $\sigma$ is called a unifier for $A$ if $\sigma(A)$ is a singleton.

Pruning of Unreachable Operator

- fam-group: $\text{handempty}(v : \text{hand}), \text{holding}(v : \text{hand}, c : \text{conatiner})$
- Action: $\text{fill-shot}(s : \text{shot}, i : \text{ingredient}, \text{hand}_1, \text{hand}_1, d : \text{dispenser})$
  \[ \text{pre}(a) = \{\text{holding}(\text{hand}_1, s), \text{handempty}(\text{hand}_1), \ldots\} \]
- There is a unifier $\sigma$: $\sigma(v) = \text{hand}_1, \sigma(c) = s$ for both
  $\{\text{holding}(v, c), \text{holding}(\text{hand}_1, s)\}$ and $\{\text{handempty}(v), \text{handempty}(\text{hand}_1)\}$. 
A substitution $\sigma$ is a function mapping variables and objects to variables and objects so that (i) it acts as identity on objects, and (ii) mapping from variables must respect types.

Given a set of atoms $A$, a substitution $\sigma$ is call a **unifier for $A$** if $\sigma(A)$ is a singleton.

**Pruning of Unreachable Operator**

- **fam-group:** $\text{handempty}(v : \text{hand}), \text{holding}(v : \text{hand}, c : \text{container})$
- **Action:** $\text{fill-shot}(s : \text{shot}, i : \text{ingredient}, \text{hand}_1, \text{hand}_1, d : \text{dispenser})$
  
  $\text{pre}(a) = \{\text{holding(\text{hand}_1, s), handempty(\text{hand}_1), \ldots}\}$

- There is a unifier $\sigma$: $\sigma(v) = \text{hand}_1, \sigma(c) = s$ for both
  
  $\{\text{holding}(v, c), \text{holding(\text{hand}_1, s)}\}$ and

  $\{\text{handempty}(v), \text{handempty(\text{hand}_1)}\}$. 
Unifier

A substitution $\sigma$ is a function mapping variables and objects to variables and objects so that (i) it acts as identity on objects, and (ii) mapping from variables must respect types.

Given a set of atoms $A$, a substitution $\sigma$ is called a **unifier for $A$** if $\sigma(A)$ is a singleton.

Pruning of Unreachable Operator

- **fam-group**: handempty($v : \text{hand}$), holding($v : \text{hand}, c : \text{container}$)
- **Action**: fill-shot($s : \text{shot}, i : \text{ingredient}, \text{hand}_1, \text{hand}_1, d : \text{dispenser}$)

\[
\text{pre}(a) = \{\text{holding}(\text{hand}_1, s), \text{handempty}(\text{hand}_1), \ldots\}
\]

- There is a unifier $\sigma$: $\sigma(v) = \text{hand}_1, \sigma(c) = s$ for both
  \{holding($v, c$), holding($\text{hand}_1, s$)\} and
  \{handempty($v$), handempty($\text{hand}_1$)\}.
Pruning with Lifted Fam-Groups

Unifier

A substitution $\sigma$ is a function mapping variables and objects to variables and objects so that (i) it acts as identity on objects, and (ii) mapping from variables must respect types. Given a set of atoms $A$, a substitution $\sigma$ is called a unifier for $A$ if $\sigma(A)$ is a singleton.

Pruning of Unreachable Operator

- **fam-group**: `handempty(v : hand), holding(v : hand, c : container)`
- **Action**: `fill-shot(s : shot, i : ingredient, hand1, hand1, d : dispenser)`
  
  $\text{pre}(a) = \{\text{holding(hand1, s)}, \text{handempty(hand1)}, \ldots\}$

- There is a unifier $\sigma$: $\sigma(v) = \text{hand1}, \sigma(c) = s$ for both
  \n  $\{\text{holding}(v, c), \text{holding(hand1, s)}\}$ and
  $\{\text{handempty}(v), \text{handempty(hand1)}\}$. 

### Experimental Evaluation: Percentage of Pruned Operators


<table>
<thead>
<tr>
<th>domain</th>
<th>#ps</th>
<th>unreach</th>
<th>+dead-end</th>
</tr>
</thead>
<tbody>
<tr>
<td>agricultura</td>
<td>20</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>barman*</td>
<td>74</td>
<td>15.42</td>
<td>45.95</td>
</tr>
<tr>
<td>citycar*</td>
<td>40</td>
<td>0.00</td>
<td>1.61</td>
</tr>
<tr>
<td>flashfill</td>
<td>18</td>
<td>0.10</td>
<td>0.10</td>
</tr>
<tr>
<td>floortile*</td>
<td>70</td>
<td>0.00</td>
<td>22.79</td>
</tr>
<tr>
<td>organic-synthesis*</td>
<td>7</td>
<td>11.44</td>
<td>23.11</td>
</tr>
<tr>
<td>parcprinter*</td>
<td>30</td>
<td>0.00</td>
<td>40.83</td>
</tr>
<tr>
<td>parking</td>
<td>80</td>
<td>3.09</td>
<td>3.09</td>
</tr>
<tr>
<td>scanalyzer</td>
<td>30</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>spider</td>
<td>15</td>
<td>3.47</td>
<td>3.47</td>
</tr>
<tr>
<td>trucks*</td>
<td>30</td>
<td>0.00</td>
<td>82.14</td>
</tr>
<tr>
<td>woodworking*</td>
<td>30</td>
<td>0.00</td>
<td>10.08</td>
</tr>
<tr>
<td>overall from above</td>
<td>444</td>
<td>3.52</td>
<td>21.52</td>
</tr>
<tr>
<td>overall</td>
<td>1247</td>
<td>1.25</td>
<td>7.66</td>
</tr>
</tbody>
</table>
### Percentage of Pruned Operators (IPC 2006–2018) (AAAI’20)

<table>
<thead>
<tr>
<th>domain</th>
<th>#ps</th>
<th>unreach</th>
<th>+dead-end</th>
</tr>
</thead>
<tbody>
<tr>
<td>agricola</td>
<td>20</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>barman*</td>
<td>74</td>
<td>15.42</td>
<td>45.95</td>
</tr>
<tr>
<td>citycar*</td>
<td>40</td>
<td>0.00</td>
<td>1.61</td>
</tr>
<tr>
<td>flashfill</td>
<td>18</td>
<td>0.10</td>
<td>0.10</td>
</tr>
<tr>
<td>floortile*</td>
<td>70</td>
<td>0.00</td>
<td>22.79</td>
</tr>
<tr>
<td>organic-synthesis*</td>
<td>7</td>
<td>11.44</td>
<td>23.11</td>
</tr>
<tr>
<td>parcprinter*</td>
<td>30</td>
<td>0.00</td>
<td>40.83</td>
</tr>
<tr>
<td>parking</td>
<td>80</td>
<td>3.09</td>
<td>3.09</td>
</tr>
<tr>
<td>scanalyzer</td>
<td>30</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>spider</td>
<td>15</td>
<td>3.47</td>
<td>3.47</td>
</tr>
<tr>
<td>trucks*</td>
<td>30</td>
<td>0.00</td>
<td>82.14</td>
</tr>
<tr>
<td>woodworking*</td>
<td>30</td>
<td>0.00</td>
<td>10.08</td>
</tr>
<tr>
<td>overall from above</td>
<td>444</td>
<td>3.52</td>
<td>21.52</td>
</tr>
<tr>
<td>overall</td>
<td>1247</td>
<td>1.25</td>
<td>7.66</td>
</tr>
</tbody>
</table>

We need to modify the grounding algorithm.
Unifier as a Formula

Given a substitution $\sigma$ and a set of variables $V \subset \mathcal{V}$, we define:

$$
\Phi_{\sigma,V}^{\text{var}} = \bigwedge_{\{v,w\} \in X} (v = w),
$$

(1)

where $X = \{\{v, w\} \subseteq V \mid v \neq w, \sigma v = \sigma w\}$;

$$
\Phi_{\sigma,V}^{\text{var-obj}} = \bigwedge_{v \in Y} (v = \sigma v),
$$

(2)

where $Y = \{v \in V \mid \sigma v \in \mathcal{B}\}$;

$$
\Phi_{\sigma,V}^{\text{subtype}} = \bigwedge_{v \in Z} \left( \bigvee_{o \in \mathcal{D}(\tau_{\text{var}}(\sigma v))} (v = o) \right),
$$

(3)

where $Z = \{v \in V \mid \sigma v \notin \mathcal{B}, \tau_{\text{var}}(\sigma v) \neq \tau_{\text{var}}(v)\}$; and

$$
\Phi_{\sigma,V}^{\text{unifier}} = \Phi_{\sigma,V}^{\text{var}} \land \Phi_{\sigma,V}^{\text{var-obj}} \land \Phi_{\sigma,V}^{\text{subtype}}.
$$

(4)
Unifier as a Formula

Given a substitution $\sigma$ and a set of variables $V \subset \mathcal{V}$, we define:

$$\Phi_{\sigma,V}^{\text{var}} = \bigwedge_{\{v,w\} \in X} (v = w),$$

where $X = \{\{v, w\} \subseteq V \mid v \neq w, \sigma v = \sigma w\}$;

$$\Phi_{\sigma,V}^{\text{var-obj}} = \bigwedge_{v \in Y} (v = \sigma v),$$

where $Y = \{v \in V \mid \sigma v \in \mathcal{B}\}$;

$$\Phi_{\sigma,V}^{\text{subtype}} = \bigwedge_{v \in Z} \left( \bigvee_{o \in \mathcal{D}(\tau_{\text{var}}(\sigma v))} (v = o) \right),$$

where $Z = \{v \in V \mid \sigma v \notin \mathcal{B}, \tau_{\text{var}}(\sigma v) \neq \tau_{\text{var}}(v)\}$; and

$$\Phi_{\sigma,V}^{\text{unifier}} = \Phi_{\sigma,V}^{\text{var}} \land \Phi_{\sigma,V}^{\text{var-obj}} \land \Phi_{\sigma,V}^{\text{subtype}}.$$

Formula $\Phi_{\sigma,V}^{\text{unifier}}$ captures unifier $\sigma$ perfectly.
Lifted fam-group:
\[ M = \text{handempty}(v : \text{hand}), \text{holding}(v : \text{hand}, c : \text{container}) \]

**Action** fill-shot\((s : \text{shot}, i : \text{ingredient}, h_1 : \text{hand}, h_2 : \text{hand}, d : \text{dispenser})\)

pre\((o) = \{\text{handempty}(h_1), \text{holding}(h_2, s), \ldots\} \)
Pruning via Compilation

Lifted fam-group:
\[ M = \text{handempty}(v : \text{hand}), \text{holding}(v : \text{hand}, c : \text{conatiner}) \]

Action \( \text{fill-shot}(s : \text{shot}, i : \text{ingredient}, h_1 : \text{hand}, h_2 : \text{hand}, d : \text{dispenser}) \)
\( \text{pre}(o) = \{ \text{handempty}(h_1), \text{holding}(h_2, s), \ldots \} \)

- Unifier \( \sigma_1, \sigma_1(v) = \sigma_1(h_1) = x \), for
  \( \{ \text{handempty}(h_1), \text{handempty}(v) \} \)
Pruning via Compilation

Lifted fam-group:
\[ M = \text{handempty}(v:\text{hand}), \text{holding}(v:\text{hand}, c:\text{container}) \]

Action \text{fill-shot}(s:\text{shot}, i:\text{ingredient}, h_1:\text{hand}, h_2:\text{hand}, d:\text{dispenser})

\[ \text{pre}(o) = \{\text{handempty}(h_1), \text{holding}(h_2, s), \ldots\} \]

- Unifier \( \sigma_1 \), \( \sigma_1(v) = \sigma_1(h_1) = x \), for \( \{\text{handempty}(h_1), \text{handempty}(v)\} \)
- Unifier \( \sigma_2 \), \( \sigma_2(x) = \sigma_2(h_2) = y \), \( \sigma_2(c) = \sigma_2(s) = z \), for \( \sigma_1\{\text{holding}(h_2, s), \text{holding}(v, c)\} = \{\text{holding}(h_2, s), \text{holding}(x, c)\} \)
Pruning via Compilation

Lifted fam-group:
\[ M = \text{handempty}(v: \text{hand}), \text{holding}(v: \text{hand}, c: \text{container}) \]

Action \( \text{fill-shot}(s: \text{shot}, i: \text{ingredient}, h_1: \text{hand}, h_2: \text{hand}, d: \text{dispenser}) \)
\[ \text{pre}(o) = \{ \text{handempty}(h_1), \text{holding}(h_2, s), \ldots \} \]

- Unifier \( \sigma_1, \sigma_1(v) = \sigma_1(h_1) = x, \) for
  \[ \{ \text{handempty}(h_1), \text{handempty}(v) \} \]
- Unifier \( \sigma_2, \sigma_2(x) = \sigma_2(h_2) = y, \sigma_2(c) = \sigma_2(s) = z, \) for
  \[ \sigma_1\{ \text{holding}(h_2, s), \text{holding}(v, c) \} = \{ \text{holding}(h_2, s), \text{holding}(x, c) \} \]
- Therefore \( \text{fill-shot}(s, i, h_1, h_2, d) \) is recognized as unreachable with \( M \)
  if and only if \( \Phi_{\text{unifier}}^{\sigma_1,\{h_1\}} \wedge \Phi_{\text{unifier}}^{\sigma_2\sigma_1,\{h_2,s\}} \) is true.
Pruning via Compilation

Lifted fam-group:
\[ M = \text{handempty}(v : \text{hand}), \text{holding}(v : \text{hand}, c : \text{conatiner}) \]

Action \text{fill-shot}(s : \text{shot}, i : \text{ingredient}, h_1 : \text{hand}, h_2 : \text{hand}, d : \text{dispenser})
\[ \text{pre}(o) = \{\text{handempty}(h_1), \text{holding}(h_2, s), \ldots\} \]

- Unifier \( \sigma_1, \sigma_1(v) = \sigma_1(h_1) = x, \) for \( \{\text{handempty}(h_1), \text{handempty}(v)\} \)
- Unifier \( \sigma_2, \sigma_2(x) = \sigma_2(h_2) = y, \sigma_2(c) = \sigma_2(s) = z, \) for \( \sigma_1\{\text{holding}(h_2, s), \text{holding}(v, c)\} = \{\text{holding}(h_2, s), \text{holding}(x, c)\} \)
- Therefore \text{fill-shot}(s, i, h_1, h_2, d) is recognized as unreachable with \( M \) if and only if \(\Phi_{\sigma_1,\{h_1\}}^{\text{unifier}} \land \Phi_{\sigma_2\sigma_1,\{h_2, s\}}^{\text{unifier}}\) is true.
- Therefore, we can prune \text{fill-shot} by extending its preconditions with \(\neg(\Phi_{\sigma_1,\{h_1\}}^{\text{unifier}} \land \Phi_{\sigma_2\sigma_1,\{h_2, s\}}^{\text{unifier}}) = (h_1 \neq h_2)\).
The pruning power is exactly the same as before, but we do not need to modify the grounding algorithm.

The pruning automatically carries to other methods, because it is directly encoded in the PDDL task. For example, successor generator for lifted planning.

There is, however, a price to pay: The resulting formulas can incur an exponential blow-up when normalized to conjunctions.
Experimental Results

- un: unreachability, de: dead-ends
- 56 (HTG + Optimal and satisficing IPC) domains, 3 464 tasks
- 26 domains and 1 485 tasks affected by un
- 10 domains and 544 tasks affected by de

Figure: Number of normalized actions.
Experimental Results

Figure: Translation time in seconds.
Experimental Results

<table>
<thead>
<tr>
<th>domain</th>
<th>blind base</th>
<th>blind de</th>
<th>hmax base</th>
<th>hmax de</th>
<th>hadd base</th>
<th>hadd de</th>
<th>lz-hadd base</th>
<th>lz-hadd de</th>
</tr>
</thead>
<tbody>
<tr>
<td>airport (50)</td>
<td>16 17</td>
<td>16 16</td>
<td>22 22</td>
<td>21 21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>barman (74)</td>
<td>4 4</td>
<td>4 4</td>
<td>4 4</td>
<td>6 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>floortile (80)</td>
<td>2 10</td>
<td>2 2</td>
<td>14 14</td>
<td>13 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parcprinter (70)</td>
<td>14 18</td>
<td>20 20</td>
<td>49 57</td>
<td>49 57</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trucks (30)</td>
<td>2 5</td>
<td>3 4</td>
<td>8 9</td>
<td>9 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>woodworking (98)</td>
<td>12 13</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>others (142)</td>
<td>46 46</td>
<td>42 42</td>
<td>111 111</td>
<td>115 115</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Σ (544)</td>
<td>96 113</td>
<td>87 88</td>
<td>208 217</td>
<td>213 225</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table: Number of solved tasks.

Figure: Number of dead-end states.
Figure: Number of expanded states (before the last $f$-layer for blind and hmax; all for hadd and lz-hadd).