Stefan Borgwardt, Jörg Hoffmann, Alisa Kovtunova, Marcel Steinmetz
Technische Universität Dresden & Saarland University

Making DL-Lite Planning Practical
Planning and Reasoning

AI planning

- actions that modify abstract states
- fixed domain
- closed-world assumption
Planning and Reasoning

AI planning
- actions that modify abstract states
- fixed domain
- closed-world assumption

+ Static reasoning
- global state constraints
- high-level background knowledge
- open-world assumption
Planning and Reasoning

AI planning
• actions that modify abstract states
• fixed domain
• closed-world assumption

+ Static reasoning
• global state constraints
• high-level background knowledge
• open-world assumption

extended-input Knowledge and Action Bases (eKABs)
(Calvanese, Montali, Patrizi, and Stawowy 2016)
Planning and Reasoning

AI planning
- actions that modify abstract states
- fixed domains
- closed-world assumption

Extended-input Knowledge and Action Bases (eKABs)

Static reasoning
- global state constraints
- high-level background knowledge
- open-world assumption

Making eKABs practical with the help of AI planning researchers

(Calvanese, Montali, Patrizi, and Stawowy 2016)
Planning Domain Definition Language (PDDL) 2.1 - Lift Control

(:action stop
  :parameters (?p ?f)
  :precondition (and (lift_at ?f) (passenger ?p))
  :effect (and
    (when
      (and (origin ?p ?f) (not (served ?p)))
      (boarded ?p)
    )
    (when
      (and (destin ?p ?f) (boarded ?p))
      (and (served ?p) (not (boarded ?p)))
    )
  )
)

(:action move-up

(:objects pa pb ... fa fb ...)

(:init
  (passenger pa)
  (origin pa fe)
  (destin pa fa)
  ...
  (lift_at fa)
)

(:goal (not (exists (?x)
    (and
      (passenger ?x)
      (not (served ?x))
    )
  )
)

Initial state
(closed-world)

Fixed domain
FO-formulas evaluated on states
Effects operate on states

Making DL-Lite Planning Practical
Technische Universität Dresden & Saarland University // © Stefan Borgwardt
DL workshop, Bratislava, Slovakia, 20th September, 2021
Slide 2 of 9
Planning Domain Definition Language (PDDL) 2.1 - Lift Control

(:action stop
  :parameters (?p ?f)
  :precondition (and (lift_at ?f) (passenger ?p))
  :effect (and
    (when
      (and (origin ?p ?f) (not (served ?p)))
      (boarded ?p)
    )
    (when
      (and (destin ?p ?f) (boarded ?p))
      (and (served ?p) (not (boarded ?p)))
    )
  )
)

(:action move-up
  ...
)

(:objects pa pb ... fa fb ...)

(:init
  (passenger pa)
  (origin pa fe)
  (destin pa fa)
  ...
  (lift_at fa)
)

(:goal (not (exists (?x)
    (and
      (passenger ?x)
      (not (served ?x))
    ))))

Initial state (closed-world)

Fixed domain
Planning Domain Definition Language (PDDL) 2.1 - Lift Control

(:action stop
 :parameters (?p ?f)
 :precondition (and (lift_at ?f) (passenger ?p))
 :effect (and
 (when
 (and (origin ?p ?f) (not (served ?p)))
 (boarded ?p)
 )
 (when
 (and (destin ?p ?f) (boarded ?p))
 (and (served ?p) (not (boarded ?p)))
 )
 )
)

(:action move-up ...

(:objects pa pb ... fa fb ...)

(:init
 (passenger pa)
 (origin pa fe)
 (destin pa fa)
 ...)

(:goal (not
 (exists (?x)
 (passenger ?x)
 (not (served ?x))
 )))

Initial state (closed-world)

FO-formulas evaluated on states

Fixed domain
Planning Domain Definition Language (PDDL) 2.1 - Lift Control

(:action stop
 :parameters (?p ?f)
 :precondition (and (lift_at ?f) (passenger ?p))
 :effect (and
 (when
   (and (origin ?p ?f) (not (served ?p)))
   (boarded ?p)
 )
 (when
   (and (destin ?p ?f) (boarded ?p))
   (and (served ?p) (not (boarded ?p)))
 )
 ) )

(:action move-up
 ...

(:objects pa pb ... fa fb ...)

(:init
 (passenger pa)
 (origin pa fe)
 (destin pa fa)
 ...
 (lift_at fa)
 )

(:goal (not (exists (?x)
   (passenger ?x)
   (not (served ?x))
 )))

Initial state (closed-world)

Fixed domain

Effects operate on states

FO-formulas evaluated on states
Explicit-Input Knowledge and Action Bases (eKABs)

(:axioms
  (isA passenger (not floor))
  (isA (exists origin) passenger)
  (isA (exists (inverse origin)) floor)
  ...
)

(:action stop
  :parameters (?p ?f)
  :precondition (and (mko (lift_at ?f)) (mko (passenger ?p)))
  :effect (and
    (when
      (and (mko (origin ?p ?f)) (not (mko (served ?p))))
      (boarded ?p)
    )
  )
Explicit-Input Knowledge and Action Bases (eKABs)

(:axioms
  (isA passenger (not floor))
  (isA (exists origin) passenger)
  (isA (exists (inverse origin)) floor)
  ...
)

(:action stop
  :parameters (?p ?f)
  :precondition (and (mko (lift_at ?f)) (mko (passenger ?p)))
  :effect (and
    (when
      (and (mko (origin ?p ?f)) (not (mko (served ?p))))
      (boarded ?p)
    )
  )
  ...
)

(:objects pa pb ... fa fb ...)

(:init
  (origin pa fe)
  ...
)

Effects operate on ABoxes

Fixed active domain

Initial ABox (open-world)
Explicit-Input Knowledge and Action Bases (eKABs)

(:axioms (DL-Lite$_F$) ontology
  (isA passenger (not floor))
  (isA (exists origin) passenger)
  (isA (exists (inverse origin)) floor)
  ...
)

(:action stop
  :parameters (?p ?f)
  :precondition (and (mko (lift_at ?f)) (mko (passenger ?p)))
  :effect (and
    (when
      (and (mko (origin ?p ?f)) (not (mko (served ?p))))
      (boarded ?p)
    )
  )
)

(:objects pa pb ... fa fb ...)

(:init
  (origin pa fe)
  ...
)

Fixed active domain

Initial ABox (open-world)

Effects operate on ABoxes
Explicit-Input Knowledge and Action Bases (eKABs)

\[
\text{:axioms (DL-Lite\text{$_F$}) ontology}
\]

\[
\text{(isA passenger (not floor))}
\]

\[
\text{(isA (exists origin) passenger)}
\]

\[
\text{(isA (exists (inverse origin)) floor)}
\]

\[
\text{...}
\]

\[
\text{:objects pa pb ... fa fb ...)
\]

\[
\text{(:init origin pa fe)}
\]

\[
\text{Fixed active domain}
\]

\[
\text{Initial ABox (open-world)}
\]

\[
\text{:action stop}
\]

\[
\text{:parameters (?p ?f)}
\]

\[
\text{:precondition (and (mko (lift_at ?f)) (mko (passenger ?p)))}
\]

\[
\text{:effect (and (when (and (mko (origin ?p ?f)) (not (mko (served ?p)))) (boarded ?p) ) Effects operate on ABoxes
\]

\[
\text{...}
\]

\[
\text{Epistemic conjunctive queries (ECQs)}
\]
Explicit-Input Knowledge and Action Bases (eKABs)

(:axioms (DL-Lite) ontology
 (isA passenger (not floor))
 (isA (exists origin) passenger)
 (isA (exists (inverse origin)) floor)
 ...)

(:action stop
 :parameters (?p ?f)
 :precondition (and (mko (lift_at ?f)) (mko (passenger ?p)))
 :effect (and
 (when
 (and (mko (origin ?p ?f)) (not (mko (served ?p))))
 (boarded ?p)
 )
 )
 ...)

(:objects pa pb ... fa fb ...)

Fixed active domain

Initial ABox (open-world)

atom = minimal-knowledge operator + CQ

Epistemic conjunctive queries (ECQs)

Effects operate on ABoxes

Initial ABox

Fixed active domain

atom = minimal-knowledge operator + CQ

Epistemic conjunctive queries (ECQs)

Effects operate on ABoxes
Rewriting ECQs into PDDL

\[(\text{mko (passenger ?p)})\] (isA boarded passenger)
(isA served passenger)
(isA (exists origin) passenger)
(isA (exists destin) passenger)

\[\text{FO-rewriting}\]

\[(\text{or})\]
(passenger ?p)
(boarded ?p)
(served ?p)
(exists (?x)
(origin ?p ?x)
)
(exists (?x)
(destin ?p ?x)
)

(\text{or})
Rewriting ECQs into PDDL

\[(mko \text{ (passenger } ?p))\]  
\[(\text{isA boarded passenger})\]  
\[(\text{isA served passenger})\]  
\[(\text{isA (exists origin) passenger})\]  
\[(\text{isA (exists destin) passenger})\]  

\[\text{FO-rewriting}\]

\[(\text{or})\]
\[(\text{passenger } ?p)\]
\[(\text{boarded } ?p)\]
\[(\text{served } ?p)\]
\[(\text{exists } (?x)\]
\[(\text{origin } ?p ?x)\]
\[
(\text{destin } ?p ?x)\)
\]

\[\text{can result in large unions of CQs (UCQs)}\]
Performance

eKAB rewriting

not practical

state-of-the-art planners
Performance

eKAB rewriting

not practical

bottleneck: pre-processing
- FO-formulas transformed into ground DNF using naive transformations
- grounded UCQs are in DNF, but nested in other formulas

planning experts
Jörg Hoffmann
Marcel Steinmetz

state-of-the-art planners
Eliminating Nested Subformulas in PDDL

$$\Phi = (\text{or } \Phi_1 \Phi_2) \mapsto P_\Phi$$
Eliminating Nested Subformulas in PDDL

\[ \Phi = (\text{or } \Phi_1 \Phi_2) \mapsto P_\Phi \]

(Nebel 2000)

(:action evaluate-\(\Phi\)-1
  :precondition (and \(P'_{\Phi_1} P_{\Phi_1}\))
  :effect (and \(P_{\Phi} P_{\Phi}\))
)

(:action evaluate-\(\Phi\)-2
  :precondition (and \(P'_{\Phi_2} P_{\Phi_2}\))
  :effect (and \(P_{\Phi} P_{\Phi}\))
)

(:action evaluate-\(\Phi\)-neg
  :precondition
    (and \(P'_{\Phi_1} P'_{\Phi_2} (\text{not } P_{\Phi_1}) (\text{not } P_{\Phi_2})\))
  :effect (and \(P_{\Phi} (\text{not } P_{\Phi})\))
)
Eliminating Nested Subformulas in PDDL

\[ \Phi = (\text{or } \Phi_1 \Phi_2) \mapsto P_\Phi \]

(Nebel 2000)

(:action evaluate-\(-\Phi\)-1
  :precondition (and \(P'_\Phi \Phi_1 \))
  :effect (and \(P'_\Phi P_\Phi \))
)

(:action evaluate-\(-\Phi\)-2
  :precondition (and \(P'_\Phi \Phi_2 \))
  :effect (and \(P'_\Phi P_\Phi \))
)

(:action evaluate-\(-\Phi\)-neg
  :precondition
    (and \(P'_\Phi \Phi_1 \) \(P'_\Phi \Phi_2 \) (not \(P_\Phi_1 \)) (not \(P_\Phi_2 \))
  :effect (and \(P'_\Phi (\text{not } P_\Phi) \)))
)

"manual" re-evaluation of \(P_\Phi\) after each normal action by resetting \(P'_\Phi\)
Eliminating Nested Subformulas in PDDL

\[
\Phi = (\text{or } \Phi_1 \Phi_2) \mapsto P_\Phi
\]

(Nebel 2000)

(:action evaluate-\(\Phi-1\)
  :precondition (and \(P'_\Phi_1 P_\Phi\))
  :effect (and \(P'_\Phi P_\Phi\)) )

(:action evaluate-\(\Phi-2\)
  :precondition (and \(P'_\Phi_2 P_\Phi_2\))
  :effect (and \(P'_\Phi P_\Phi\)) )

(:action evaluate-\(\Phi\)-neg
  :precondition
    (and \(P'_\Phi_1 P'_\Phi_2 (\text{not } P_\Phi_1) (\text{not } P_\Phi_2)\))
  :effect (and \(P'_\Phi (\text{not } P_\Phi)\)) )

“manual” re-evaluation of \(P_\Phi\) after each normal action by resetting \(P'_\Phi\)

New approach

(:derived \(P_\Phi (\text{or } \Phi_1 \Phi_2)\) )
Eliminating Nested Subformulas in PDDL

\[ \Phi = (or \, \Phi_1 \, \Phi_2) \rightarrow P_{\Phi} \]

(Nebel 2000)

(:action evaluate-\(\Phi\)-1
  :precondition (and \(P'_{\Phi_1} \, P_{\Phi_1}\))
  :effect (and \(P'_{\Phi} \, P_{\Phi}\))
)

(:action evaluate-\(\Phi\)-2
  :precondition (and \(P'_{\Phi_2} \, P_{\Phi_2}\))
  :effect (and \(P'_{\Phi} \, P_{\Phi}\))
)

(:action evaluate-\(\Phi\)-neg
  :precondition
    (and \(P'_{\Phi_1} \, P'_{\Phi_2} \, (not \, P_{\Phi_1}) \, (not \, P_{\Phi_2})\))
  :effect (and \(P'_{\Phi} \, (not \, P_{\Phi})\))
)

“manual” re-evaluation of \(P_{\Phi}\) after each normal action by resetting \(P'_{\Phi}\)

New approach

(:derived \(P_{\Phi}\) (or \(\Phi_1 \, \Phi_2\))

automatic evaluation via “derived predicates” (PDDL 2.1), similar to Datalog
Benchmarks

Robot, TaskAssign
(Calvanese, Montali, Patrizi, and Stawowy 2016)

Cats, Elevator
inspired by planning benchmarks

TPSA, VTA, VTA-Roles
(Hoffmann, Weber, Scicluna, Kacmarek, and Ankolekar 2008)

Assembly, Miconic
(modified) planning benchmarks with complex conditions

GridPlacement
artificial benchmark designed with huge CNFs
Results

<table>
<thead>
<tr>
<th>Domain</th>
<th>#</th>
<th>FF O</th>
<th>Ne</th>
<th>DP</th>
<th>FD O</th>
<th>Ne</th>
<th>DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot</td>
<td>20</td>
<td>20</td>
<td>1</td>
<td>20</td>
<td>4</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>TaskAssign</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>3</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Cats</td>
<td>20</td>
<td>14</td>
<td>14</td>
<td>20</td>
<td>14</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>Elevator</td>
<td>20</td>
<td>12</td>
<td>0</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>TPSA</td>
<td>15</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>14</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>VTA</td>
<td>15</td>
<td>15</td>
<td>6</td>
<td>15</td>
<td>15</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>VTA-Roles</td>
<td>15</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>15</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Assembly</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>30</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>GridPlacement</td>
<td>20</td>
<td>5</td>
<td>1</td>
<td>17</td>
<td>6</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>Miconic</td>
<td>30</td>
<td>9</td>
<td>3</td>
<td>13</td>
<td>9</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>∑</td>
<td>205</td>
<td>88</td>
<td>35</td>
<td>154</td>
<td>130</td>
<td>25</td>
<td>162</td>
</tr>
</tbody>
</table>

Planners
FF (Hoffmann and Nebel 2001)
FD 20.06 (Helmert 2006)

PDDL variants
Original eKAB PDDL
+ Nebel's encoding
+ Derived Predicates encoding

Metric
# instances solved

Intel Core i5-4590 @3.30GHz,
8GB RAM, 600s timeout
## Results

<table>
<thead>
<tr>
<th>Domain</th>
<th>#</th>
<th>O</th>
<th>Ne</th>
<th>DP</th>
<th>O</th>
<th>Ne</th>
<th>DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot</td>
<td>20</td>
<td>20</td>
<td>1</td>
<td>20</td>
<td>4</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>TaskAssign</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>3</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Cats</td>
<td>20</td>
<td>14</td>
<td>14</td>
<td>20</td>
<td>14</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>Elevator</td>
<td>20</td>
<td>12</td>
<td>0</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>TPSA</td>
<td>15</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>14</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>VTA</td>
<td>15</td>
<td>15</td>
<td>6</td>
<td>15</td>
<td>15</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>VTA-Roles</td>
<td>15</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>15</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Assembly</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>30</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>GridPlacement</td>
<td>20</td>
<td>5</td>
<td>1</td>
<td>17</td>
<td>6</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>Miconic</td>
<td>30</td>
<td>9</td>
<td>3</td>
<td>13</td>
<td>9</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td><strong>∑</strong></td>
<td>205</td>
<td>88</td>
<td>35</td>
<td>154</td>
<td>130</td>
<td>25</td>
<td>162</td>
</tr>
</tbody>
</table>

**Planners**
- FF (Hoffmann and Nebel 2001)
- FD 20.06 (Helmert 2006)

**PDDL variants**
- Original eKAB PDDL
  + Nebel's encoding
  + Derived Predicates encoding

**Metric**
- # instances solved
- Intel Core i5-4590 @3.30GHz, 8GB RAM, 600s timeout

Nebel's encoding helps the pre-processor, but is too complex for the solver.
## Results

<table>
<thead>
<tr>
<th>Domain</th>
<th>#</th>
<th>FF</th>
<th></th>
<th></th>
<th>FD</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot</td>
<td>20</td>
<td>20</td>
<td>1</td>
<td>20</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>TaskAssign</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Cats</td>
<td>20</td>
<td>14</td>
<td>14</td>
<td>20</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>Elevator</td>
<td>20</td>
<td>12</td>
<td>0</td>
<td>20</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>TPSA</td>
<td>15</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>VTA</td>
<td>15</td>
<td>15</td>
<td>6</td>
<td>15</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>VTA-Roles</td>
<td>15</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>Assembly</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>GridPlacement</td>
<td>20</td>
<td>5</td>
<td>1</td>
<td>17</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Miconic</td>
<td>30</td>
<td>9</td>
<td>3</td>
<td>13</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td><strong>∑</strong></td>
<td>205</td>
<td>88</td>
<td>35</td>
<td>154</td>
<td>130</td>
<td>25</td>
</tr>
</tbody>
</table>

The derived predicates improve performance on most benchmarks.

### Planners
- **FF** (Hoffmann and Nebel 2001)
- **FD 20.06** (Helmert 2006)

### PDDL variants
- Original eKAB PDDL
- **Nebel's encoding**
- Derived Predicates encoding

### Metric
- # instances solved

Intel Core i5-4590 @3.30GHz, 8GB RAM, 600s timeout
Summary

• Simple pre-processing for eKAB PDDL encodings
• Increased number of solved tasks
• Can even help standard planning benchmarks
Summary

- Simple pre-processing for eKAB PDDL encodings
- Increased number of solved tasks
- Can even help standard planning benchmarks

Making eKABs even more practical:

- Support more expressive (Horn) DLs
- Encode Datalog rewriting into derived predicates
Thank you!


Picture rights:
2016 Sample Return Robot Challenge (NHQ201609050020) by “NASA HQ PHOTO”, CC BY-NC-ND 2.0
Drone by “ninfaj”, CC BY-NC-ND 2.0
Artist Concept of LCROSS/LRO (NASA, Moon, 6/15/09) by “NASA’s Marshall Space Flight Center”, CC BY-NC-ND 2.0