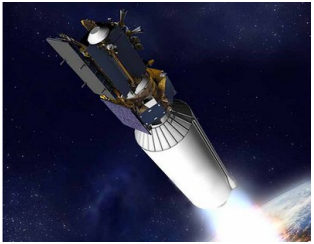
Stefan Borgwardt, Jörg Hoffmann, Alisa Kovtunova, Marcel Steinmetz
Technische Universität Dresden & Saarland University

Making DL-Lite Planning Practical

(short paper at KR'21) // DL workshop, Bratislava, Slovakia, 20th September, 2021

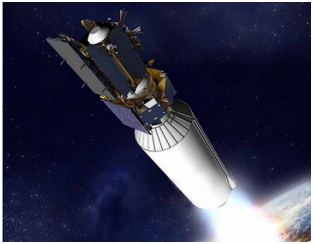
Planning and Reasoning



AI planning

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

Planning and Reasoning



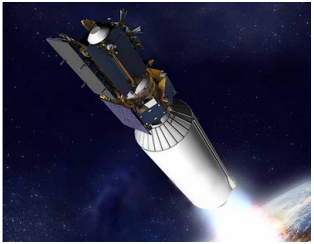
AI planning

- 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

Making eKABs practical with the help of AI planning researchers

AI planning

- actions that modif[illegible]tates
- fixed dom[illegible]
- closed [illegible]d 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 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))
) ) ) )
```

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 ...)
```

Fixed domain

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

Initial state (closed-world)

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

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 ...)
```

Fixed domain

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

Initial state (closed-world)

FO-formulas evaluated on states

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

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)))
) ) )
```

Effects operate on states

```
(:action move-up
  ...
```

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

Fixed domain

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

Initial state (closed-world)

FO-formulas evaluated on states

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

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)
  ...
)
...
```

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)
    )
    ...
```

Effects operate on ABoxes

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

Fixed active domain

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

Initial ABox (open-world)

Explicit-Input Knowledge and Action Bases (eKABs)

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

(*DL-Lite*$_{\mathcal{F}}$) ontology

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

Fixed active domain

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

Initial ABox (open-world)

```
(: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)
    )
    ...
```

Effects operate on ABoxes

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)
    )
    ...
```

(*DL-Lite*$_{\mathcal{F}}$) ontology

Epistemic conjunctive queries (ECQs)

Effects operate on ABoxes

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

Fixed active domain

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

Initial ABox (open-world)

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)
    )
    ...
```

(*DL-Lite*$_{\mathcal{F}}$) ontology

Epistemic conjunctive queries (ECQs)

Effects operate on ABoxes

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

Fixed active domain

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

Initial ABox (open-world)

atom = minimal-knowledge operator + CQ

Rewriting ECQs into PDDL

```
(isA boarded passenger)
(isA served passenger)
(isA (exists origin) passenger)
(isA (exists destin) passenger)
```

```
(mko (passenger ?p))
```

$\xrightarrow{\text{FO-rewriting}}$

```
(or
  (passenger ?p)
  (boarded ?p)
  (served ?p)
  (exists (?x)
    (origin ?p ?x)
  )
  (exists (?x)
    (destin ?p ?x)
  )
)
```

Rewriting ECQs into PDDL

```
(isA boarded passenger)
(isA served passenger)
(isA (exists origin) passenger)
(isA (exists destin) passenger)
```

```
(mko (passenger ?p))
```

$\xrightarrow{\text{FO-rewriting}}$

```
(or
  (passenger ?p)
  (boarded ?p)
  (served ?p)
  (exists (?x)
    (origin ?p ?x)
  )
  (exists (?x)
    (destin ?p ?x)
  )
)
```

can result in large unions of CQs (UCQs)

Performance

eKAB rewriting

↓

not practical

↑

state-of-the-art planners

Performance

eKAB rewriting

↓

not practical → planning experts

Jörg Hoffmann
Marcel Steinmetz

↑

state-of-the-art planners

bottleneck: pre-processing

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

Eliminating Nested Subformulas in PDDL

$$\boxed{\Phi = (\texttt{or}\ \Phi_1\ \Phi_2)} \longmapsto \boxed{\mathrm{P}_\Phi}$$

Eliminating Nested Subformulas in PDDL

$$\boxed{\Phi = (\text{or } \Phi_1\ \Phi_2)} \longmapsto \boxed{\mathrm{P}_\Phi}$$

(Nebel 2000)

```
(:action evaluate-Φ-1
  :precondition (and P'_{Φ1} P_{Φ1}))
  :effect (and P'_Φ P_Φ))
)
(:action evaluate-Φ-2
  :precondition (and P'_{Φ2} P_{Φ2}))
  :effect (and P'_Φ P_Φ))
)
(:action evaluate-Φ-neg
  :precondition
    (and P'_{Φ1} P'_{Φ2} (not P_{Φ1}) (not P_{Φ2})))
  :effect (and P'_Φ (not P_Φ)))
)
```

Eliminating Nested Subformulas in PDDL

$$\boxed{\Phi = (\text{or } \Phi_1\ \Phi_2)} \longmapsto \boxed{P_\Phi}$$

(Nebel 2000)

```
(:action evaluate-Φ-1
  :precondition (and P'_{Φ1} P_{Φ1}))
  :effect (and P'_Φ P_Φ))
)
(:action evaluate-Φ-2
  :precondition (and P'_{Φ2} P_{Φ2}))
  :effect (and P'_Φ P_Φ))
)
(:action evaluate-Φ-neg
  :precondition
    (and P'_{Φ1} P'_{Φ2} (not P_{Φ1}) (not P_{Φ2})))
  :effect (and P'_Φ (not P_Φ)))
)
```

"manual" re-evaluation of P_Φ after each normal action by resetting P'_Φ

Eliminating Nested Subformulas in PDDL

$$\boxed{\Phi = (\text{or } \Phi_1\ \Phi_2)} \longmapsto \boxed{P_\Phi}$$

(Nebel 2000)

```
(:action evaluate-Φ-1
  :precondition (and P'_Φ1 P_Φ1))
  :effect (and P'_Φ P_Φ))
)
(:action evaluate-Φ-2
  :precondition (and P'_Φ2 P_Φ2))
  :effect (and P'_Φ P_Φ))
)
(:action evaluate-Φ-neg
  :precondition
    (and P'_Φ1 P'_Φ2 (not P_Φ1) (not P_Φ2)))
  :effect (and P'_Φ (not P_Φ)))
)
```

"manual" re-evaluation of P_Φ after each normal action by resetting P'_Φ

New approach

(:derived P_Φ (or Φ_1 Φ_2))

Eliminating Nested Subformulas in PDDL

$$\boxed{\Phi = (\text{or } \Phi_1\ \Phi_2)} \longmapsto \boxed{P_\Phi}$$

(Nebel 2000)

```
(:action evaluate-Φ-1
  :precondition (and P'_{Φ1} P_{Φ1}))
  :effect (and P'_Φ P_Φ))
)
(:action evaluate-Φ-2
  :precondition (and P'_{Φ2} P_{Φ2}))
  :effect (and P'_Φ P_Φ))
)
(:action evaluate-Φ-neg
  :precondition
    (and P'_{Φ1} P'_{Φ2} (not P_{Φ1}) (not P_{Φ2})))
  :effect (and P'_Φ (not P_Φ)))
)
```

"manual" re-evaluation of P_Φ after each normal action by resetting P'_Φ

New approach

(:derived P_Φ (or Φ_1 Φ_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

Domain	#	FF O	FF Ne	FF DP	FD O	FD Ne	FD DP
Robot	20	**20**	1	**20**	4	1	**20**
TaskAssign	20	1	1	**15**	3	1	**20**
Cats	20	14	14	**20**	14	11	**20**
Elevator	20	12	0	**20**	**20**	0	**20**
TPSA	15	**7**	5	5	**14**	4	5
VTA	15	**15**	6	**15**	**15**	4	13
VTA-Roles	15	**5**	4	**5**	**15**	0	5
Assembly	30	0	0	**24**	**30**	0	**30**
GridPlacement	20	5	1	**17**	6	2	**20**
Miconic	30	9	3	**13**	**9**	2	**9**
$\sum$	205	88	35	**154**	130	25	**162**

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

Domain	#	FF O	FF Ne	FF DP	FD O	FD Ne	FD DP
Robot	20	**20**	1	**20**	4	1	**20**
TaskAssign	20	1	1	**15**	3	1	**20**
Cats	20	14	14	**20**	14	11	**20**
Elevator	20	12	0	**20**	**20**	0	**20**
TPSA	15	**7**	5	5	**14**	4	5
VTA	15	**15**	6	**15**	**15**	4	13
VTA-Roles	15	**5**	4	**5**	**15**	0	5
Assembly	30	0	0	**24**	**30**	0	**30**
GridPlacement	20	5	1	**17**	6	2	**20**
Miconic	30	9	3	**13**	**9**	2	**9**
$\sum$	205	88	35	**154**	130	25	**162**

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

Domain	#	FF O	FF Ne	FF DP	FD O	FD Ne	FD DP
Robot	20	**20**	1	**20**	4	1	**20**
TaskAssign	20	1	1	**15**	3	1	**20**
Cats	20	14	14	**20**	14	11	**20**
Elevator	20	12	0	**20**	**20**	0	**20**
TPSA	15	**7**	5	5	**14**	4	5
VTA	15	**15**	6	**15**	**15**	4	13
VTA-Roles	15	**5**	4	**5**	**15**	0	5
Assembly	30	0	0	**24**	**30**	0	**30**
GridPlacement	20	5	1	**17**	6	2	**20**
Miconic	30	9	3	**13**	**9**	2	**9**
$\sum$	205	88	35	**154**	130	25	**162**

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

The derived predicates improve performance on most benchmarks.

Summary

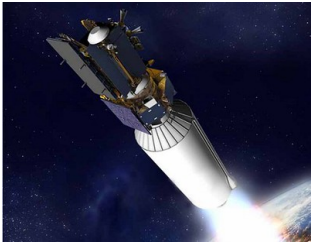
- Simple pre-processing for eKAB PDDL encodings
- Increased number of solved tasks
- Can even help standard planning benchmarks

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!

Borgwardt, Stefan, Jörg Hoffmann, Alisa Kovtunova, and Marcel Steinmetz (2021). "Making DL-Lite Planning Practical". In: *Proc. of the 18th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'21)*. Short paper. To appear.

Calvanese, Diego, Marco Montali, Fabio Patrizi, and Michele Stawowy (2016). "Plan Synthesis for Knowledge and Action Bases". In: *Proc. of the 25th Int. Joint Conf. on Artificial Intelligence (IJCAI'16)*, pages 1022–1029. URL: https://www.ijcai.org/Abstract/16/149.

Helmert, Malte (2006). "The Fast Downward Planning System". In: *Journal of Artificial Intelligence Research* 26, pages 191–246. DOI: 10.1613/jair.1705.

Hoffmann, Jörg and Bernhard Nebel (2001). "The FF Planning System: Fast Plan Generation Through Heuristic Search". In: *Journal of Artificial Intelligence Research* 14, pages 253–302. DOI: 10.1613/jair.855.

Hoffmann, Jörg, Ingo Weber, James Scicluna, Tomasz Kacmarek, and Anupriya Ankolekar (2008). "Combining Scalability and Expressivity in the Automatic Composition of Semantic Web Services". In: *8th International Conference on Web Engineering (ICWE'08)*. DOI: 10.1109/ICWE.2008.8.

Nebel, Bernhard (2000). "On the Compilability and Expressive Power of Propositional Planning Formalisms". In: *Journal of Artificial Intelligence Research* 12, pages 271–315. DOI: 10.1613/jair.735.