Making DL-Lite Planning Practical (Extended Abstract)*

Stefan Borgwardt[1], Jörg Hoffmann[2], Alisa Kovtunova[1], and Marcel Steinmetz[2]

[1] Institute of Theoretical Computer Science, TU Dresden, Germany
`firstname.lastname@tu-dresden.de`
[2] Saarland University, Saarland Informatics Campus, Germany
`lastname@cs.uni-saarland.de`

AI planning is a well-investigated framework for describing the evolution of system states through actions [6]. Action preconditions are first-order (FO) formulas, which are evaluated over states, i.e. finite sets of facts. Action effects either add or remove facts from the current state. Formulas are evaluated using a closed-domain, closed-world semantics, i.e. the domain is fixed a priori and facts that are not contained in a state are assumed to be false. The aim is to find a *plan*, i.e. a sequence of (grounded) actions that satisfy a goal formula.

Knowledge representation formalisms are a natural way to introduce global constraints on permissible states; however, they usually interpret FO formulas over arbitrary models, instead of just one model with a fixed domain. *DL-Lite Explicit-Input Knowledge and Action Bases (eKABs)* [5] were proposed to combine classical planning with state axioms formulated in DL-Lite [4], while allowing an open domain and interpreting action preconditions under open-world semantics. Due to FO rewritability of DL-Lite, eKABs can be translated into the classical planning language PDDL. In theory, this allows the use of off-the-shelf planning systems. However, an initial evaluation [5, 13] using the Fast Downward (FD) planning platform [7] showed poor performance on a simple hand-crafted domain, with the planner being unable to solve even trivial problem instances. In our experiments, the problem also appeared in Fast Forward (FF) [9].

Planning Formalisms. A *PDDL task* is a tuple $\Pi = \langle \mathcal{P}, \mathcal{A}, \mathcal{O}, I, G\rangle$ with finite sets of *predicate symbols* $\mathcal{P}$, *action schemas* $\mathcal{A}$, *objects* $\mathcal{O}$, the *initial state* I, and the *goal* G. I is a set of facts that are true initially; all facts not in I are considered false. G is a closed FO formula over $\mathcal{P}$ and $\mathcal{O}$. Every $a \in \mathcal{A}$ is a triple $\langle \overrightarrow{x_a}, \mathsf{pre}_a, \mathsf{eff}_a\rangle$ with *parameters* $\overrightarrow{x_a}$, *precondition* pre_a, and a set of *effects* eff_a. pre_a is an FO formula over $\mathcal{P}$ and $\mathcal{O}$ with free variables from $\overrightarrow{x_a}$. An effect $e \in \mathsf{eff}_a$ is a tuple $\langle \overrightarrow{y_e}, \mathsf{cond}_e, \mathsf{add}_e, \mathsf{del}_e\rangle$ where $\overrightarrow{y_e}$ are variables, cond_e is the *effect condition*, an FO formula over $\mathcal{P}$ and $\mathcal{O}$ with variables from $\overrightarrow{x_a} \cup \overrightarrow{y_e}$, the *add list* add_e is a set of atoms with free variables from $\overrightarrow{x_a} \cup \overrightarrow{y_e}$, and the *delete list* del_e is a set of such negated atoms. For example, the action $\mathfrak{A} = \langle x, \mathsf{Emp}(x), \langle \emptyset, \top, \{\mathsf{SoDev}(x)\}, \emptyset\rangle\rangle$ promotes any employee to a software developer.

* Abstract of a short paper accepted at KR 2021. This work is supported by the DFG grant 389792660 (TRR 248) (see `https://perspicuous-computing.science`). Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

A *DL-Lite eKAB* $\langle \mathcal{P}, \mathcal{A}, \mathcal{O}, \mathcal{T}, I, G\rangle$[1] [5] extends a PDDL task by a DL-Lite TBox $\mathcal{T}$, a possibly infinite set of objects $\mathcal{O}$, the fact that states use the open-world assumption and must be consistent with $\mathcal{T}$, and a modified syntax for conditions. Action and effect preconditions and the goal are specified as *ECQs* [2], which are FO formulas with *conjunctive queries (CQs)* as atoms. The CQs are evaluated over $\mathcal{T}$ using open-world semantics, but their results (i.e. certain answers) are interpreted under an epistemic semantics to combine open- and closed-world conditions. For example, consider the TBox $\{\mathsf{Emp} \sqsubseteq \exists \mathsf{worksFor}\}$, state $\{\mathsf{Emp}(a)\}$, and the conditions $\phi_1(x) = \neg[\exists y.\mathsf{worksFor}(x, y)]$ and $\phi_2(x) = \neg\exists y.[\mathsf{worksFor}(x, y)]$, where the conjunctive queries are indicated by square brackets. Then ϕ_1 does not apply to object a, because all employees are known to work for some department (even if the specific department is unknown), but $\phi_2(a)$ is true since no particular y is known for which $\mathsf{worksFor}(a, y)$ holds. Moreover, the promoting action $\mathfrak{A}$ is not applicable to object b in state $\{\mathsf{ElEng}(b)\}$ considering the TBox $\mathcal{T} = \{\mathsf{ElEng} \sqsubseteq \mathsf{Emp}, \mathsf{ElEng} \sqsubseteq \neg\mathsf{SoDev}\}$ since it would cause an inconsistency.

In [5], a translation from *state-bounded* DL-Lite eKABs to equivalent PDDL tasks is presented. It is based on (i) a bound on the number of objects in each state [3, 5], (ii) a translation of ECQs into first-oder formulas under closed-world semantics [2, 5, 12], and (iii) an additional predicate and action that checks consistency of a state. The translation (ii) effectively compiles the TBox into the ECQs, thereby simulating open-world query answering by a closed-world formula. For example, in $\mathfrak{A}$ the precondition $\mathsf{Emp}(x)$ becomes $\mathsf{Emp}(x) \vee \mathsf{ElEng}(x)$ to simulate the fact that electronic engineers are employees ($\mathsf{ElEng} \sqsubseteq \mathsf{Emp}$). Step (iii) uses a formula that describes every possible inconsistent state. The set of all axioms involving negation, e.g. $\mathsf{ElEng} \sqsubseteq \neg\mathsf{SoDev}$, is first reformulated into a disjunction of CQs of the form $\mathsf{ElEng}(x) \wedge \mathsf{SoDev}(x)$, which describe basic inconsistent situations. By applying the translation (ii) to these CQs, the resulting formula includes all possible ways in which inconsistencies can be generated.

Our Contribution. We find that the bottleneck lies in the DNF transformations used to compile PDDL tasks into the planners' internal (grounded) representations. These naïve transformations are applied in-situ on complex formulas, causing a worst-case exponential blow-up on non-DNF input. Although the disjunctions of CQs generated by the above translation are in DNF, they are often nested in more complex ECQ conditions. Here we investigate two PDDL pre-compilations that enable polynomial DNF transformations also for such complex formulas.

The first pre-compilation (Ne), proposed by Nebel [11], uses auxiliary predicates to represent each (already grounded) complex sub-formula ϕ: P_ϕ represents the truth value of ϕ in the current state; and P'_ϕ is designed to be true iff ϕ has already been evaluated. Additional ground actions are introduced to determine the value of P_ϕ provided that P'_ψ is true for all sub-formulas ψ of ϕ. This evaluation is repeated after every regular action, which increases the plan length.

We therefore propose a second pre-compilation (DP) that avoids this overhead by employing PDDL *derived predicates* [8, 14], which specify a rule-based update

[1] The syntax is slightly adapted for compatibility with PDDL.

of auxiliary predicates that is applied at every planning state. Every complex subformula ϕ is replaced by a new predicate P_ϕ that obeys the update rule $P_\phi \leftarrow \phi$. This results in a set of rules that is equivalent to a non-recursive, and therefore stratified, Datalog program with negation [1].

Table 1. Per-domain aggregated statistics. (a) Number of instances solved within resource limits. (b) Number of instances that passed the planners' PDDL pre-processing.

		(a) # solved						(b) # PDDL processed					
		FF			FD			FF			FD		
Domain	#	O	Ne	DP	O	Ne	DP	O	Ne	DP	O	Ne	DP
Robot	20	**20**	1	**20**	4	1	**20**	**20**	**20**	**20**	4	**20**	**20**
TaskAssign	20	1	1	15	3	1	**20**	1	10	15	3	10	**20**
Cats	20	14	14	**20**	14	11	**20**	14	**20**	**20**	14	**20**	**20**
Elevator	20	12	0	**20**	**20**	0	**20**	12	**20**	**20**	**20**	**20**	**20**
TPSA	15	7	5	5	**14**	4	5	7	5	5	**14**	4	5
VTA	15	**15**	6	**15**	**15**	4	13	**15**	6	**15**	**15**	4	13
VTA-Roles	15	5	4	5	**15**	0	5	6	4	5	**15**	0	5
Assembly	30	0	0	24	**30**	0	**30**	9	10	24	**30**	4	**30**
GridPlacement	20	5	1	17	6	2	**20**	5	**20**	**20**	6	**20**	**20**
Miconic	30	9	3	**13**	9	2	9	11	27	19	14	18	**30**
$\sum$	205	88	35	154	130	25	**162**	100	142	163	135	120	**183**

Experiments. We compare the pre-compilations against the original PDDL files (O) using FF and FD 20.06. We developed a set of benchmarks, comprising the previous eKAB tasks [5,13] (Robot, TaskAssign), new eKAB domains (Cats, Elevator), and benchmarks adapted from prior work on planning with propositional background ontologies [10] (TPSA, VTA). We also investigate existing PDDL benchmark domains with minor modifications (Assembly, Miconic), as our pre-compilation may be useful on any planning domain with complex preconditions, and a showcase domain (GridPlacement) enforcing challenging DNF transformations. The benchmarks[2] and pre-compilers[3] are available online.

As indicated by the O columns in Table 1, PDDL processing indeed constitutes the main bottleneck: In almost every unsolved instance, the planners failed already during PDDL processing. Part (b) shows that DP reduces the overhead of PDDL input handling and thereby increases overall performance in nearly every domain. TPSA and VTA are the only exceptions were the pre-compilation turned out detrimental. Besides an absence of complex conditions, in these domains the actions assign predicates to previously unbound objects. Therefore, by increasing the number of objects, the grounding and translation sizes grow drastically.

In contrast, the number of instances that could be solved by FF and FD after the Ne pre-compilation drops substantially in every domain. Apart from a blow-up of the file size, the Ne pre-compiler obfuscates the planning task's original structure, which leads both planners' searches into serious troubles.

In future work, we plan to design tools tailored to eKABs over DL-Lite and more expressive ontology languages, leveraging e.g. PDDL derived predicates.

[2] https://gitlab.perspicuous-computing.science/m.steinmetz/pddl-dllite-benchmarks.git

[3] https://gitlab.perspicuous-computing.science/a.kovtunova/moreflags2.git

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: EQL-Lite: Effective first-order query processing in description logics. In: Veloso, M.M. (ed.) Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI'07). pp. 274–279 (2007), https://www.ijcai.org/Abstract/07/042
3. Calvanese, D., De Giacomo, G., Montali, M., Patrizi, F.: Verification and synthesis in description logic based dynamic systems. In: Faber, W., Lembo, D. (eds.) Proc. of the 7th Int. Conf. on Web Reasoning and Rule Systems (RR'13). pp. 50–64. Lecture Notes in Computer Science, Springer (2013). https://doi.org/10.1007/978-3-642-39666-3_5
4. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Dl-Lite: Tractable description logics for ontologies. In: Veloso, M.M., Kambhampati, S. (eds.) Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference. pp. 602–607. AAAI Press / The MIT Press (2005), http://www.aaai.org/Library/AAAI/2005/aaai05-094.php
5. Calvanese, D., Montali, M., Patrizi, F., Stawowy, M.: Plan synthesis for knowledge and action bases. In: Kambhampati, S. (ed.) Proc. of the 25th Int. Joint Conf. on Artificial Intelligence (IJCAI'16). pp. 1022–1029. AAAI Press (2016), https://www.ijcai.org/Abstract/16/149
6. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and Practice. Morgan Kaufmann (2004)
7. Helmert, M.: The fast downward planning system. Journal of Artificial Intelligence Research **26**, 191–246 (2006). https://doi.org/10.1613/jair.1705, https://doi.org/10.1613/jair.1705
8. Hoffmann, J., Edelkamp, S.: The deterministic part of IPC-4: An overview. Journal of Artificial Intelligence Research **24**, 519–579 (2005). https://doi.org/10.1613/jair.1677
9. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research **14**, 253–302 (2001). https://doi.org/10.1613/jair.855
10. Hoffmann, J., Weber, I., Scicluna, J., Kacmarek, T., Ankolekar, A.: Combining scalability and expressivity in the automatic composition of semantic web services. In: 8th International Conference on Web Engineering (ICWE'08) (2008). https://doi.org/10.1109/ICWE.2008.8
11. Nebel, B.: On the compilability and expressive power of propositional planning formalisms. Journal of Artificial Intelligence Research **12**, 271–315 (2000). https://doi.org/10.1613/jair.735
12. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. Journal on Data Semantics **X**, 133–173 (2008). https://doi.org/10.1007/978-3-540-77688-8_5
13. Stawowy, M.: Plan Synthesis in Explicit-input Knowledge and Action Bases. Ph.D. thesis, IMT School for Advanced Studies Lucca, Italy (2016). https://doi.org/10.6092/imtlucca/e-theses/208
14. Thiebaux, S., Hoffmann, J., Nebel, B.: In defense of PDDL axioms. Artificial Intelligence **168**(1–2), 38–69 (2005). https://doi.org/10.1016/j.artint.2005.05.004