Bounded Proper Planning

Mikhail Soutchanski$^{[ORCID\ 0000-0003-0227-2424]}$

Toronto Metropolitan (formerly Ryerson) University, Toronto, ON, M5B 2K3, Canada
`mes(at)cs.torontomu.ca`
https://www.cs.torontomu.ca/mes/

Abstract. In classical planning and conformant planning, it is assumed that there are finitely many named objects given in advance, and that only they can participate in actions and in fluents. This is the Domain Closure Assumption (DCA). However, there are realistic open-world deterministic planning problems where the set of initially given objects changes as planning proceeds: new objects are created, and old objects cease to exist. These problems are particularly challenging when knowledge is incomplete. We formulate the bounded proper planning (BPP) problem in first-order logic, assume an initial incomplete theory is a finite consistent set of fluent literals, consider a special form of weakly context free action theories, impose an integer upper bound on the length of the plan, and propose to organize search for a plan over sequences of actions that are grounded at planning time. In contrast to numeric or generalized planning problems, where each state is a finite set, in the BPP each state can be characterized with infinitely many infinitely sized first order models. We show how a planner can solve the BPP problem by using a domain-independent heuristic that guides search over sequences of actions. We discuss the differences between our approach and the formulations of the planning problem explored previously.

1 Proper Basic Action Theories

This is a shorter version of the paper that appears as [31]. Our focus here is on informal explanation of [31] and on providing experimental data to illustrate our approach.

We study planning problems when there are numerical variables, when knowledge is incomplete, and when the actions can create new objects (not mentioned initially) or possibly destroy objects that participated at the previous steps of planning. To the best of our knowledge this direction has not been explored before.

As an example, we consider a new variation of the planning problem proposed in [10]. There are trays with pizza slices. There are people who want pizza. The problem is how to cut some of the available slices and serve pizza to some people so that they will get equally sized slices. For simplicity, assume that the diameters of all pizzas are the same. Each continuous pizza piece is characterized with the two angles: the left angle wrt a fixed chosen axis, and the right angle which is always greater than the left angle. To say that in situation s on $tray_x$ there is a pizza slice with the angles l, r, we use logical fluent $available(tray_x, l, r, s)$. The angles can be any (rational) numbers in the range from 0 to 360. There are situation independent predicates $person(p)$ and $tray(t)$, but there are no upper bounds on the number of people and trays, and for simplicity, it is assumed that each tray holds initially only one slice. In addition, there are fluents $served(p, s)$, a person p was served pizza in s or in a previous situation, and $angleSize(p, n, s)$ meaning that a person p has a slice with size n in situation s, where n is the difference between the right and left angles. There are two actions: $serve(t, p, l, r)$, serve a person p a slice l, r from a tray t, and $cutHalf(t, l, r)$, on a tray t cut a slice l, r into two equal halves that remain on the same tray t. The first half has its angles from l to $0.5 \cdot (r - l)$, and the second half is between $0.5 \cdot (r - l)$ and r.

We consider a special form of the basic action theory (BAT) $\mathcal{D}$ [28]. As usual, it is the conjunction of the following classes of axioms $\mathcal{D} = \Sigma \wedge \mathcal{D}_{ss} \wedge \mathcal{D}_{ap} \wedge \mathcal{D}_{una} \wedge \mathcal{D}_{S_0}$, where Σ are foundational axioms characterizing situations as sequences of actions, $\mathcal{D}_{ss}$ describe effects and non-effects of actions, $\mathcal{D}_{ap}$ specify action preconditions, $\mathcal{D}_{S_0}$ include an incomplete logical theory about what is true initially in the situation S_0, and $\mathcal{D}_{una}$ are the unique name axioms (UNA). (For brevity, variables $\bar{x}, a, s$ are implicitly $\forall$-quantified at front.)

$\mathcal{D}_{S_0}$ is a set of first-order (FO) sentences whose only situation term is the initial situation S_0. The syntactic form of $\mathcal{D}_{S_0}$ is motivated by a *proper KB* introduced in [19]. More specifically, we assume that $\mathcal{D}_{S_0}$ is a consistent finite set of ground fluent literals, i.e., there are some facts that are known to be true initially and there are some other facts that are known to be initially false. $\mathcal{D}_{S_0}$ generalizes databases by allowing incomplete knowledge about some of the elements of the application domain: if some fluent literal is not mentioned, then the Closed World Assumption (CWA) does not apply, and this literal is treated as unknown. In addition, $\mathcal{D}_{S_0}$ includes usual equality axioms $\mathcal{E}$ (reflexivity, symmetry, transitivity, substitution of equals for equals) and therefore Equality Theorem applies [5]. Moreover, as proposed in [19], $\mathcal{D}_{S_0}$ is formulated in a standard first order logic language with a countably infinite set of (object) constants $\{C_1, C_2, \ldots\}$, and no other function symbols. These constants satisfy a set of equality axioms and the set of UNA formulas $\{C_i \neq C_j \mid i \neq j\}$. Informally speaking, the purpose of these constants is to supply enough entities to answer quantified queries, since as proved in [19] it is not sufficient to consider only constants mentioned in a query or in given fluent literals. Another purpose is to provide the names of any objects that may ever be created or destroyed in the process of planning. Using the logically equivalent transformations, our proper $\mathcal{D}_{S_0}$ can be written as a finite set of implications $e \rightarrow \rho$, where e is a quantifier-free formula whose only predicate is equality, and ρ is a fluent literal whose arguments are distinct variables. Recall that the *domain closure assumption* (DCA) for objects [26] means that the domain of interest is finite, the names of all objects in $\mathcal{D}_{S_0}$ are explicitly given as a finite set of constants $C_1, C_2, \ldots, C_K$, and for any object variable x, $\forall x$ is understood as $\forall x(x{=}C_1 \vee x{=}C_2 \vee \ldots \vee x{=}C_K)$. We do not include DCA.

For example, we consider the following proper initial theory where the constants start with an upper-case letter; we use them instead of symbols C_i, C_j to improve readability.

$$\forall p\big((p{=}Jane \vee p{=}Ken \vee p{=}Bob \vee p{=}Sue) \rightarrow person(p)\big)$$
$$\forall t\big((t{=}T_1 \vee t{=}T_2 \vee t{=}T_3) \rightarrow tray(t)\big)$$
$$\forall p\big((p{=}Jane \vee p{=}Ken \vee p{=}Bob \vee p{=}Sue) \rightarrow \neg served(p, S_0)\big)$$
$$\forall t, l, r\big((t{=}T_1 \wedge l{=}0 \wedge r{=}100) \rightarrow available(t, l, r, S_0)\big)$$
$$\forall t, l, r\big((t{=}T_2 \wedge l{=}180 \wedge r{=}360) \rightarrow \neg available(t, l, r, S_0)\big)$$

Since we do not include DCA for objects, there might be infinitely many different infinitely-sized models of $\mathcal{D}_{S_0}$ where the pizza slices have different angles. According to $\mathcal{D}_{S_0}$, it is known that the tray T_1 holds the specific slice with the angles between 0 and 100, the tray T_2 does not have a slice with a size between 180 and 360, but it is not known if T_2 has any other slices, and nothing is known about the pizza slices on the tray T_3, or on any other trays. For people mentioned in $\mathcal{D}_{S_0}$, it is known they were not initially served, but nothing is known about any other people not mentioned in $\mathcal{D}_{S_0}$. Thus, every model of $\mathcal{D}_{S_0}$ includes facts mentioned above, and a combination of other facts. When planning for what specific instantiated action to execute next, note it should be possible wrt all models of $\mathcal{D}_{S_0}$.

$\mathcal{D}_{\mathbf{ap}}$ is a set of action precondition axioms $poss(A(\bar{x}), s) \leftrightarrow \Pi_A(\bar{x}, s)$, where $poss(a, s)$ is a new predicate symbol that means an action a is possible in situation s, $\Pi_A(\bar{x}, s)$ is a formula uniform in s, and A is an k-ary action function. In this paper, we consider a special case, when $\Pi_A(\bar{x}, s)$ is an extended conjunctive query, e.g., see [4, 1]. An extended conjunctive query (ECQ) is of the form $\exists \boldsymbol{x} \phi(\boldsymbol{x}, \boldsymbol{y})$, where ϕ is a conjunction of positive literals, *safe dis-equalities*, that is, dis-equalities ($\neq$) between variables or variables and constants, and *safe comparisons*, that is, arithmetical comparisons ($\leq$, $\geq$) between two variables or variables and constants, such that each dis-equality variable, and each comparison variable appears in at least one positive literal in ϕ. The following are action precondition axioms for actions in our example.

$poss(serve(t,p,l,r),s) \leftrightarrow tray(t) \wedge person(p) \wedge available(t,l,r,s)$,
i.e., action $serve(t,p,l,r)$ is possible in s, if t is a tray, p is a person and there is a slice available on t with the angles between l and r.

$poss(cutHalf(t,l,r),s) \leftrightarrow available(t,l,r,s) \wedge r > l$,
i.e., action $cutHalf(t,l,r)$ is possible in situation s, if there is a slice l,r available on a tray t in s and its right angle r is greater than its left angle l.

It is easy to see that the actions

$$[cutHalf(T_1,0,100), serve(T_1,Bob,0,50.0), serve(T_1,Sue,50.0,100.0)]$$

are consecutively ***possible wrt all models***, including those infinite models which have countably many trays and slices. They result in ground situation σ where the goal formula $\exists p_1,p_2,n(served(p_1,\sigma) \wedge served(p_2,\sigma) \wedge p_1 \neq p_2 \wedge angleSize(p_1,n,\sigma) \wedge angleSize(p_2,n,\sigma))$ holds. However, if $\mathcal{D}_{S_0}$ does not include any statements about fluent $available(t,l,r,S_0)$, or if it includes only the fact that there is no available slice with the angles between 180 and 360 on the tray T_2, then this subtle modification has significant consequences. Namely, there is no sequence of actions ***possible in all models*** that leads to a ground situation σ, where the goal formula holds. Notice the goal formula is an extended conjunctive query.

Let $\mathcal{D}_{\mathbf{ss}}$ be a set of successor state axioms (SSA):

$$F(\bar{x}, do(a,s)) \leftrightarrow \gamma_F^+(\bar{x},a,s) \vee\ F(\bar{x},s) \wedge \neg\gamma_F^-(\bar{x},a,s),$$

where $\bar{x}$ is a tuple of object arguments of the fluent F, and each of the γ_F's is a disjunction of uniform formulas $[\exists\bar{z}].a = A(\bar{u}) \wedge \phi(\bar{x},\bar{z},s)$,where $A(\bar{u})$ is an action with a tuple $\bar{u}$ of object arguments, $\phi(\bar{x},\bar{z},s)$ is a context condition, and $\bar{z} \subseteq \bar{u}$ are optional object arguments. If $\bar{u}$ in an action function $A(\bar{u})$ does not include any z variables, then there is no optional $\exists\bar{z}$ quantifier. We consider Weakly Context Free (WCF) successor state axioms in this paper, see a formal definition in [31]. The SSAs in our example are the following:

$served(t,p,do(a,s)) \leftrightarrow \exists l \exists r(\, a = serve(t,p,l,r))\ \vee served(t,p,s)$
$angleSize(p,n,do(a,s)) \leftrightarrow \exists l \exists r(\, a = serve(t,p,l,r) \wedge n = (r-l)) \vee angleSize(p,n,s)$
$available(t,l',r',do(a,s)) \leftrightarrow \exists l \exists r(\, a = cutHalf(t,l,r) \wedge l' = l \wedge r' = 0.5 \cdot (r-l)) \vee$
$\exists l \exists r(\, a = cutHalf(t,l,r) \wedge l' = 0.5 \cdot (r-l) \wedge r' = r\,) \vee$
$available(t,l,r,s) \wedge a \neq cutHalf(t,l,r) \wedge \neg\exists p(a = serve(t,p,l,r),$

where $n = (r-l)$ is a function (semantic attachment) that computes the new value n for fluent $angleSize$ in the next situation that results from performing action $serve(t,p,l,r)$ in situation s. Similarly, $0.5 \cdot (r-l)$ is a function that computes the new left angle (right angle, respectively) when an action cuts an available slice in half. We say that $cutHalf(t,l,r)$ actions destroy a previously available object, that is a slice with the angles between l and r, and also create two new objects, namely, a new slice with the angles between l and $0.5(r-l)$, and another slice with the angles between $0.5(r-l)$ and r.

There are two main reasoning mechanisms in SC. One of them relies on the regression operator [35, 27], and another mechanism called progression is responsible for the reasoning forward, where after each ground action α the initial theory $\mathcal{D}_{S_0}$ is updated to a new theory $\mathcal{D}_{S_\alpha}$. In this paper, we focus on progression [20].

In general, progression $\mathcal{D}_{S_\alpha}$ is defined in second-order logic [20]. However, in this paper, we consider a special case of proper $\mathcal{D}_{S_0}$ in the form of a finite set of *ground* fluent literals, which we call a finite grounded proper initial theory (FGP) $\mathcal{D}_{S_0}$. In our special case of weakly context free SSAs, generalizing the results from [22] one can show that the progression of $\mathcal{D}_{S_0}$ wrt α, $\mathcal{P}(\mathcal{D}_{S_0},\alpha)$, remains FGP, and it can be efficiently computed. Informally speaking, for those new constants (not in $\mathcal{D}_{S_0}$) which are arguments of a fluent literal that enters

$\mathcal{P}(\mathcal{D}_{S_0}, \alpha)$, one can say that they represent created objects, while for the constants that previously occurred in $\mathcal{D}_{S_0}$ and are no longer mentioned in progression, one can say that they represent objects destroyed by α.

If the goal formula is ECQ, our BAT provides the prerequisites for open-world planning without DCA. We say a *BAT is proper*, if it satisfies all the conditions in this section. The bounded proper planning (BPP) problem includes an upper bound N on the plan length.

2 Solving Bounded Proper Planning (BPP) Problem

The BPP problem differs from previously explored planning problems since there are infinitely many infinitely sized models of $\mathcal{D}_{S_\alpha}$ due to incomplete knowledge. After each step of progression, new constants can appear in $\mathcal{D}_{S_\alpha}$ that never appeared there before (objects were created), and some of the constants that were mentioned previously may no longer belong to $\mathcal{D}_{S_\alpha}$ (objects were destroyed).

It turns out that the BPP problem can be solved using an improved version of the well-known domain independent heuristic developed for the Fast Forward planner (FF) [18, 3]. See the details in [32, 31]. The key idea is that search is actually organized over situations (sequences of actions) that serve as convenient symbolic proxies for FGP theories (and their infinite models). The planning algorithm is lifted, since possible actions are determined at run-time when expanding the current situation to compute successors. In all experiments, we set the upper bound N to 100, as an obvious over-estimate of the plan length.

p1G	p1A	p2G	p2A	p3G	p3A	p4G	p4A	p5G	p5A
5	151	9317	5	5	60	3	3	–	249
14.55	89.69	65.96	13.13	14.83	22.35	14.52	14.52	–	59.76

p6G	p6A	p7G	p7A	p8G	p8A	p9G	p9A	p10G	p10A
5	67	5	9	9317	5	9317	5	10147	5
12.01	20.92	9.13	64.71	69.23	16.25	65.8	13.3	425.43	11.24

Table 1. Addition problems p1-p5, p6-p10: Number of situations expanded and total time (sec)

p1G	p1A	p2G	p2A	p3G	p3A	p4G	p4A	p5G	p5A
5	5	5	5	5	5	5	5	5	5
14.49	14.59	12.00	12.05	14.82	15.00	14.89	14.76	14.91	15.00

p6G	p6A	p7G	p7A	p8G	p8A	p9G	p9A	p10G	p10A
5	5	5	6	5	5	5	5	5	6
11.99	12.53	9.13	30.84	15.00	15.03	12.08	12.09	10.41	39.73

Table 2. Multiplication problems p1-p5, p6-p10 solved using Greedy (G) or A* search (A)

Our planner calls a random number generator to choose between the two priority queues: the queue “all" includes all successors of explored situations, and the queue “useful" includes only situations deemed to be useful at the stage of counting relevent easiest actions in a planning graph when our reachability algorithm back-chains from the final layer with the goal atoms to the first fluent layer that represents a state produced by an evaluated action. The easiest relevant actions from the 1st fluent layer together with situation leading to the 1st fluent layer form situations that are inserted into the “useful" queue with an heuristic value computed for an evaluated action. This is inspired by intuitions similar to the *favored actions* proposed in [23, 24], the *helpful actions* proposed in [18] and generalized to *preferred operators* in the Fast Downward planner [15, 16]. As demonstrated experimentally in [29], an additional priority queue for preferred operators is beneficial. We chose 50% : 50%. Note that “useful" situations can be misleading due to delete relaxation when computing heuristic.

We have collected data for the generalized Countdown benchmark [30, 31]. There are 6 counters that can hold any integers. Initially, our program assigns a randomly generated integer from 0 to 100 to each counter. There are 2 possible actions: either addition or multiplication. Each action stores the result in one of the participating counters, but another counter becomes unavailable. The goal is to produce the target integer in any of the initially available counters. We randomly generated 10 Addition problems, where the target number can be produced by adding the initially available numbers, and 10 Multiplication problems, where the target number is the product of the initial numbers. We run our planner implemented in PROLOG on a desktop computer with an 11th Gen Intel(R) Core(TM) i7-11700K CPU 3.60GHz, single thread, under the ECLiPSe System version 7.0#63 (April 24, 2022), using a 75 MB memory limit. The results are presented in Tables 1 and 2 with averages over 5 runs. The planner can run either Greedy Best First Search (G) or A* search (A). For some reason, multiplication problems are easier than the addition problem, e.g., the addition problem 5 was not solved (out of memory). Greedy search expanded more situations than A* (see the 1st row). However, greedy search was usually faster than A* (see the 2nd row). Our random planning instances and the domain encoding are publicly available at [30].

3 Related work

Helmert [14] provides a comprehensive classification of the numerical planning formalisms, demonstrates the cases where the planning problem is undecidable, and explores the reductions between numerical planning formalisms. The numerical planning problems where the range of values is finite can be reduced to classical planning with DCA; see, e.g. [11, 2]. In a general case, numerical planning goes beyond DCA. Our proposal is different since we consider a BPP problem with *incomplete* knowledge, each model of progression in BPP is infinite in contrast to numerical planning, where each state is a finite set. Our actions can have parameters that range over an *infinite universe*, while this is not allowed in PDDL [13].

Several publications discuss when progression can be formulated in FO logic, e.g., see [21, 34, 33]. They did not consider $\mathcal{D}_{S_0}$ as a proper theory, and did not attempt planning.

Note that in contrast to [8], we do not require that the number of objects where fluent can be bounded for all s. Informally, boundedness of the set of objects that may ever be considered by our planner becomes the consequence of working with a proper BAT and imposing the upper bound on the number of actions.

[32] proposed a lifted deductive planner based on the situation calculus (SC), but their implementation required both DCA and CWA. Their planner was competitive with Fast Downward [15, 16] in terms of IPC scores based on the number of visited states and the length of the plan (over classical planning benchmarks with a small number of objects).

[6] considers an extension of classical planning (the universe of objects is *finite*), with *complete* knowledge, but their semantics is based on an unusual object assignment to variables that can take values outside of the universe. We define created/destroyed objects syntactically, while they are defined semantically in [6]. Our BPP is more general, since we plan over infinite domains and consider incomplete knowledge. Our semantics is standard.

To our knowledge, there are no other heuristic planners that can solve problems without the DCA given *incomplete* initial theory. The conformant planners previously developed require DCA [17, 25, 12]. The planner in [17] was actually inspired by situation calculus, and it does search over sequences of actions, but it works only at a propositional level. [9] does open-world planning, but they require DCA, see details in [28].

Future work may consider the case where $\mathcal{D}_{S_0}$ may include $\exists$-quantifiers over objects; they can be replaced with Skolem constants. This case was discussed for proper KBs in [7].

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases (1995), http://webdam.inria.fr/Alice/
2. Bonassi, L., Percassi, F., Scala, E.: Towards practical classical planning compilations of numeric planning. In: Proceedings of the 39th AAAI Conference on Artificial Intelligence AAAI-2025. pp. 26472–26480 (2025), https://doi.org/10.1609/aaai.v39i25.34847
3. Bryce, D., Kambhampati, S.: Planning Graph Based Reachability Heuristics. AI Mag. **28**(1), 47–83 (2007), https://doi.org/10.1609/aimag.v28i1.2028
4. Chandra, A., Merlin, P.: Optimal implementation of conjunctive queries in relational data bases. In: 9th Annual ACM Symposium on Theory of Computing (STOC). pp. 77–90 (1977), https://doi.org/10.1145/800105.803397
5. Cook, S.A., Pitassi, T.: Herbrand Theorem, Equality, and Compactness (Course Notes). https://www.cs.columbia.edu/∼toni/Courses/Logic2022/Notes/page39.pdf (2022)
6. Corrêa, A.B., Giacomo, G.D., Helmert, M., Rubin, S.: Planning with object creation. In: 34th International Conference on Automated Planning and Scheduling, ICAPS 2024. pp. 104–113 (2024), https://doi.org/10.1609/icaps.v34i1.31466
7. De Giacomo, G., Lespérance, Y., Levesque, H.J.: Efficient reasoning in proper knowledge bases with unknown individuals. In: 22nd International Joint Conference on Artificial Intelligence (IJCAI). pp. 827–832 (2011), https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-144
8. De Giacomo, G., Lespérance, Y., Patrizi, F.: Bounded situation calculus action theories. Artif. Intell. **237**, 172–203 (2016), https://doi.org/10.1016/j.artint.2016.04.006
9. Finzi, A., Pirri, F., Reiter, R.: Open world planning in the situation calculus. In: Proceedings of the 7th AAAI. pp. 754–760 (2000), http://www.cs.toronto.edu/cogrobo/Papers/openworld-aaai00.ps.gz
10. Fuentetaja, R., de la Rosa, T.: Compiling Irrelevant Objects to Counters. Special Case of Creation Planning. AI Commun. **29**(3), 435–467 (2016), https://doi.org/10.3233/AIC-150692
11. Gigante, N., Scala, E.: On the compilability of bounded numeric planning. In: 32nd International Joint Conference on Artificial Intelligence IJCAI-2023. pp. 5341–5349 (2023), https://doi.org/10.24963/ijcai.2023/593
12. Grastien, A., Scala, E.: CPCES: A planning framework to solve conformant planning problems through a counterexample guided refinement. Artif. Intell. **284**, 103271 (2020), https://doi.org/10.1016/j.artint.2020.103271
13. Haslum, P., Lipovetzky, N., Magazzeni, D., Muise, C.: An Introduction to the Planning Domain Definition Language. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2019), https://doi.org/10.2200/S00900ED2V01Y201902AIM042
14. Helmert, M.: Decidability and undecidability results for planning with numerical state variables. In: Sixth International Conference on Artificial Intelligence Planning Systems AIPS-2002. pp. 44–53 (2002), http://www.aaai.org/Library/AIPS/2002/aips02-005.php
15. Helmert, M.: The Fast Downward Planning System. J. Artif. Intell. Res. **26**, 191–246 (2006), https://doi.org/10.1613/jair.1705
16. Helmert et. al.: Fast Downward at Github. https://github.com/aibasel/downward (2022), accessed: 2022-11-17
17. Hoffmann, J., Brafman, R.: Conformant planning via heuristic forward search: A new approach. Artificial Intelligence **170**(6–7), 507–541 (2006)
18. Hoffmann, J., Nebel, B.: The FF System: Fast Plan Generation Through Heuristic Search. J. Artif. Intell. Res. **14**, 253–302 (2001), https://doi.org/10.1613/jair.855
19. Levesque, H.J.: A Completeness Result for Reasoning with Incomplete First-Order KBs. In: KR-1998. pp. 14–23 (1998)
20. Lin, F., Reiter, R.: How to Progress a Database. Artificial Intelligence **92**, 131–167 (1997), http://www.cs.toronto.edu/cogrobo/Papers/progress.pdf

21. Liu, Y., Lakemeyer, G.: On First-Order Definability and Computability of Progression for Local-Effect Actions and Beyond. In: 21st IJCAI-2009. pp. 860–866 (2009), http://ijcai.org/papers09/Papers/IJCAI09-147.pdf
22. Liu, Y., Levesque, H.J.: Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In: 19th International Joint Conference on Artificial Intelligence, IJCAI-2005. pp. 522–527 (2005), http://ijcai.org/Proceedings/05/Papers/0706.pdf
23. McDermott, D.V.: A heuristic estimator for means-ends analysis in planning. In: Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-1996). pp. 142–149 (1996), http://www.aaai.org/Library/AIPS/1996/aips96-018.php
24. McDermott, D.V.: Using regression-match graphs to control search in planning. Artif. Intell. **109**(1-2), 111–159 (1999), https://doi.org/10.1016/S0004-3702(99)00010-7
25. Palacios, H., Geffner, H.: Compiling uncertainty away in conformant planning problems with bounded width. J. Artif. Intell. Res. **35**, 623–675 (2009), https://doi.org/10.1613/jair.2708
26. Reiter, R.: Equality and Domain Closure in First-Order Databases. J. ACM **27**(2), 235–249 (1980), https://doi.org/10.1145/322186.322189
27. Reiter, R.: The Frame Problem in the Situation Calculus: A Simple Solution (sometimes) and a Completeness Result for Goal Regression. In: Lifschitz, V. (ed.) AI and Mathematical Theory of Computation: Papers in Honor of John McCarthy. pp. 359–380. Academic Press, San Diego (1991), http://www.cs.toronto.edu/kr/publications/simple.pdf
28. Reiter, R.: Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems. MIT, http://cognet.mit.edu/book/knowledge-action (2001), http://www.cs.toronto.edu/cogrobo/kia/index.html
29. Richter, S., Helmert, M.: Preferred operators and deferred evaluation in satisficing planning. In: Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009. pp. 273–280 (2009), http://aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/view/700
30. Soutchanski, M.: The Countdown Planning Benchmark (October 2025), https://www.cs.torontomu.ca/mes/publications/Countdown.zip, PROLOG source code of randomly generated instances
31. Soutchanski, M., Liu, Y.: Planning with dynamically changing domains. In: Proceedings of the 1st International Workshop on Trends in Knowledge Representation and Reasoning, co-located with the 34th International Joint Conference on Artificial Intelligence, Montreal, Canada (2025), https://arxiv.org/abs/2508.02697
32. Soutchanski, M., Young, R.: Planning as theorem proving with heuristics. In: Proceedings of the Italian Workshop on Planning and Scheduling, co-located with AIxIA-2023. CEUR Workshop Proceedings, vol. 3585 (2023), https://ceur-ws.org/Vol-3585/paper10_RCRA6.pdf
33. Vassos, S., Patrizi, F.: A classification of first-order progressable action theories in situation calculus. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013. pp. 1132–1138 (2013), http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6646
34. Vassos, S., Sardina, S., Levesque, H.: Progressing basic action theories with non-local effect actions. In: International Symposium on Commonsense Reasoning (2009)
35. Waldinger, R.: Achieving Several Goals Simultaneously. In: Machine Intelligence. vol. 8, pp. 94–136. Ellis Horwood, Edinburgh, Scotland (1977), https://www.sri.com/wp-content/uploads/pdf/763.pdf