Proper Open World Automated Planning

Mikhail Soutchanski[1]

[1] Toronto Metropolitan (formerly Ryerson) University, 245 Church St, ENG281, Toronto, Ontario, M5B 2K3, Canada

Abstract

In classical planning and conformant planning, it is implicitly 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. To address this challenge, we formulate the bounded proper planning (BPP) problem in first-order logic as follows: assume an initial incomplete theory is a finite consistent set of fluent literals, require that preconditions are safe extensions of conjunctive queries, 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. This builds on the fact that there are easy mappings between PDDL action schemas and situation calculus action theories, but an incomplete initial state cannot be specified in PDDL as a finite set of fluent literals without DCA. It turns out there is a light-weight implementation. We show a proof-of-the-concept lifted planner can sometimes solve the BPP problem by using a domain-independent heuristic that guides search for a plan. We argue that BPP problem is different from previously proposed numeric, generalized, classical and conformant planning problems.

1. Introduction

Instead of solving an existing “classical" planning problem defined in the 1970s [1], we propose a new interesting planning problem that can be more relevant for modern applications of automated planning. We formulate it in a practical fragment of first order logic (FOL). We show it can be sometimes solved with a minor adaptation of an existing heuristic, and mention a new domain independent heuristic is needed to solve efficiently planning instances from other benchmarks. To simplify presentation our focus is on informal explanations and on preliminary experimental data to illustrate our approach.

We study deterministic 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. We formulate a planning problem in a first order (FO) language, since this allows us to leave unmentioned the specific details that are not known. For simplicity, we do not consider sensing actions but they can be included as in [2, 3].

As an example, consider a new variation of Logistic [4] benchmark, but without DCA. Let predicate $at(x, l)$ mean that object x is at location l, The objects can be boxes or trucks. Let predicate $in(b, t)$ mean that box b is inside truck t. If a box is inside a truck located at L, then the box itself is not at that location L, but it can be unloaded from the truck, and then it will be at the same location L. Suppose an initial state is only partially known. Constants start with an upper-case letter; x, y are variables.

- Truck T_1 contains only box B_1 and nothing else:
  $\forall x \forall y.\ x = B_1 \wedge y = T_1 \rightarrow in(x, y)$
  $\forall x \forall y.\ x = B_1 \wedge y \neq T_1 \rightarrow \neg in(x, y)$
  $\forall x \forall y.\ x \neq B_1 \wedge y = T_1 \rightarrow \neg in(x, y)$
- Truck T_2 contains box B_2 and possibly other boxes.
  $\forall x \forall y.\ x = B_2 \wedge y = T_2 \rightarrow in(x, y)$
  $\forall x \forall y.\ x = B_2 \wedge y \neq T_2 \rightarrow \neg in(x, y)$

8th International Workshop on Artificial Intelligence and fOrmal VERification, Logic, Automata, and sYnthesis. OVERLAY 2026 @ FLoC, July 18–19, Lisbon, Portugal

soutchanski(at)protonmail.com (M. Soutchanski)
https://www.cs.torontomu.ca/mes (M. Soutchanski)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-09

CEUR-WS.org/Vol-4324/paper2.pdf

- Box B_3 is not inside any of the trucks
  $\forall x \forall y.\ x = B_3 \rightarrow \neg in(x, y)$
- Only box B_4 and B_5 can be inside truck T_3
  $\forall x \forall y.\ x \neq B_4 \wedge x \neq B_5 \wedge y = T_3 \rightarrow \neg in(x, y)$.
- Truck T_2 is initially at location L_7
  $\forall x \forall y.\ x = T_2 \wedge y = L_7 \rightarrow at(x, y)$.
- Unique Name Assumption: B_1,B_2,B_3,B_4, T_1, T_2, T_3, L_7 are distinct, e.g., $B_1 \neq L_7 \wedge T_1 \neq T_2$, etc.

There are actions that load and unload boxes from trucks, and trucks can drive between different locations. If the goal formula is that box B_2 must be at the same location L_7, then there is an obvious one-step plan that unloads box B_2 from truck T_2, and thereby solves this planning instance despite *incomplete* knowledge about boxes and trucks. In our simple example, finitely many implications may have infinite FOL models, since there are *no* axioms saying that there are no other *unnamed* boxes and trucks except of those mentioned above. This example illustrates that an instance of the planning problem can be specified incompletely, but nevertheless, it may have a simple solution. Subsequently, we assume that the readers know first order logic, e.g., from one of the textbooks [5, 6, 7, 8].

This paper is structured as follows. In Section 2 we review briefly the situation calculus and Reiter's basic action theories. In Section 3 we consider a special case of an action theory that is suitable for defining the class of planning problems (with incomplete knowledge) that we would like to solve. In the next Section 4 we consider our planner and experimental data collected from our preliminary implementation. Finally, in Section 5 we discuss the previous related work and then conclude.

2. The Situation Calculus

The readers who are familiar with the situation calculus can skip this section. The situation calculus (SC) is a logical approach to representation and reasoning about actions and their effects. It was introduced in [9, 10] to capture common sense reasoning about the actions and events, and it was subsequently refined by Reiter [11] who introduced the **b**asic **a**ction **t**heories (BATs). Unlike the notion of state that is common in model-based planning, SC is based on the situation, namely a sequence of actions, which is a concise symbolic representation and a convenient proxy for the state in the cases where all actions are deterministic [12, 13].

We use variables s, s', s_1, s_2 for situations, variables a, a' for actions, and $\bar{x}, \bar{y}$ for tuples of object variables. The constant S_0 represents the initial situation, and the successor function $do : action \times situation \mapsto situation$, e.g., $do(a, s)$, denotes situation that results from doing action a in previous situation s. The terms σ, σ' denote situation terms and $A_i(\bar{x})$, or $\alpha, \alpha_1, \alpha_2, \alpha'$, represent action functions and action terms, respectively. The shorthand $do([\alpha_1, \cdots, \alpha_n], S_0))$ represents the situation $do(\alpha_n, do(\cdots, do(\alpha_1, S_0) \cdots)$ resulting from the execution of actions $\alpha_1, \cdots, \alpha_n$ in S_0. The relation $\sigma \sqsubset \sigma'$ between situations terms σ and σ' means that σ is an initial subsequence of σ'. Any predicate symbol $F(\bar{x}, s)$ with exactly one situation argument s and possibly a tuple of object arguments $\bar{x}$ is called a (relational) *fluent*. Without loss of generality, we consider only relational fluents. There is a special predicate $poss(a, s)$; it means that it is possible to execute action a in situation s. A first-order logic formula $\psi(s)$ composed of fluents, equalities, and situation-independent predicates is called *uniform in* s if all fluents in ψ mention only situation s as their situation argument, and $\psi(s)$ has no quantifiers over the situation variables.

The basic action theory $\mathcal{D}$ [14, 11] 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 (one axiom per fluent), $\mathcal{D}_{ap}$ specify action preconditions (using predicate $poss(a, s)$, i.e., action a is possible in situation s, one axiom per action), $\mathcal{D}_{S_0}$ 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) saying that differently named actions and objects are actually different.

In the case of deterministic planning, it helps compare the differences how a planning domain can be represented in the Planning Domain Definition Language (PDDL) [15, 16, 17] vs how it is represented

in a BAT. For example, consider the well known Blocks World. It can be formulated in PDDL with a single action $move(x, y, z)$, or with the following 3 different actions: *move-b-to-b*(x,y,z) move block x from the block y to another block z, *move-t-to-b*(x, y) move block x from the table to the block y, *move-b-to-t*(x, y) move block x from the block y to the table. As usual, $clear(x, s)$ means that there is nothing on the top of block x, and $on(x, y, s)$ means that block x is immediately on the top of another block y. To illustrate the syntactic differences consider the following.

```
(: action          move-b-to-b
                   : parameters (?bX ?bY ?bZ)
 : precondition    (and (clear ?bX) (clear ?bZ)
                   (on ?bX ?bY)  (not (= ?bY ?bZ)))
 : effect        (and (not (clear ?bZ)) (not (on ?bX ?bY))
                   (on ?bX ?bZ)  (clear ?bY))
  )
```

In SC, there is a successor state axiom (SSA) that specifies the effects of all actions on a fluent $on(x, z, s)$ and an additional precondition axiom for action *move-b-to-b*(x,y,z).

$$\forall x, y, z, s.\ \ poss(\textit{move-b-to-b}(x,y,z), s) \leftrightarrow\ \ clear(x,s) \wedge clear(z,s) \wedge on(x, y, s) \wedge x \neq z$$

$$\forall x, z, a, s.\ \ on(x, z, do(a, s)) \leftrightarrow\ \ \exists y(a = \textit{move-b-to-b}(x, y, z)) \vee a = \textit{move-t-to-b}(x, z) \vee$$
$$on(x, z, s) \wedge \neg\exists y(a = \textit{move-b-to-b}(x, z, y)\ )\ \wedge\ \neg(a = \textit{move-b-to-t}(x, z)\ ).$$

This SSA says that block x is on block z in situation $do(a, s)$ that results from doing arbitrary action a in any previous situation s iff the last action a is either *move-b-to-b*(x, y, z), or *move-t-to-b*(x, z), otherwise, if block x was already on block z in situation s, and the last action a did not move x to another block y and did not move x to the table.

As we see, PDDL is action-centered while Situation Calculus is fluent-centered and this leads to syntactic differences. It turns out that declarative semantics of STRIPS [18, 1] fragment of PDDL can be specified in the situation calculus [19]. For our illustrative purposes, it is sufficient to say that PDDL and SC are based on similar intuitions about actions and their effects. Therefore, it is easy to write computer programs that translate between PDDL and SC representations. From a more formal perspective, using a *compilation schemes* technique introduced in [20], it was shown that there is an exact translation between the ADL [21] fragment of PDDL, known as PDDL 2.1 (Level 1), and restricted BATs, where the initial theory is a complete logical theory (data-base), so that for each fluent there is a finite set of tuples, where it is initially true, and there are a few other syntactic restrictions on BATs [22, 23].

However, there is an important difference between the two languages. Namely, in SC one can consider a more general class of action theories where the number of participating objects can be left unspecified and initial knowledge $\mathcal{D}_{S_0}$ can be left incomplete because $\mathcal{D}_{S_0}$ can be formulated in a simple fragment of first order logic, and therefore some (unknown) details can be left unmentioned. This is not allowed in PDDL [15, 23]. This important distinction implies that one can formulate a new interesting planning problem in a practical fragment of first order logic, as explained in the next section.

3. Proper Basic Action Theories (BATs)

As an example, we consider a new variation of the planning problem proposed by Fuentetaja and De La Rosa ([24]). 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. It is realistic to solve this problem even if the total number of people, and the number of trays and slices are *not* given. For simplicity, assume that the diameters of all pizzas are the same. Each continuous pizza piece $\langle l, r\rangle$ is characterized with the two angles: the left angle l wrt a fixed chosen axis, and the right angle r 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 pair $\langle l, r\rangle$, 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 $\langle l, r\rangle$ from a tray t, and $cutHalf(t, l, r)$, on a tray t cut a slice $\langle l, r\rangle$ into two equal halves that remain on the same tray t. The first half has its angles from l to $0.5 \cdot (l + r)$; in the second half angles vary between $0.5 \cdot (l + r)$ and r. This action destroys a previously available slice, and produces two smaller slices.

We consider a special, restricted, but useful form of the basic action theory $\mathcal{D}$ [11]. We illustrate all classes of axioms with our example. (For brevity, variables $\bar{x}, a, s$ are implicitly $\forall$-quantified).

3.1. Initial Axioms

$\mathcal{D}_{S_0}$ is a set of first-order 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 [25]. 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 true initially and there are some other facts that are initially false. $\mathcal{D}_{S_0}$ generalizes databases (and $ABox$ in description logics) 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 the following well-known Equality Theorem holds.

Theorem 1. *[26] Let Φ be any set of formulas in a first order logic language that includes =. Then Φ is satisfiable (in a structure that interprets = as true equality) iff $\Phi \cup \mathcal{E}$ is weakly satisfiable,*

that is $\Phi \cup \mathcal{E}$ is satisfied by some weak structure (that treats = as any binary predicate symbol) under some object assignment function. Equality Theorem is true for models of any cardinality, e.g., it holds for countable models. Note that it holds no matter whether a FOL language includes any constants.

Moreover, following [25], $\mathcal{D}_{S_0}$ 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. Recall that the names starting with capital letters denote constants. These constants satisfy a set of equality axioms $\mathcal{E}$ and the set of **u**nique **n**ame **a**xioms (UNA): formulas $\{C_i \neq C_j \mid i \neq j\}$. Informally speaking, the purpose of these (extra) constants is to supply enough entities to answer correctly quantified queries, since as proved in [25] it is not sufficient to consider only constants mentioned in a query or in a given set of fluent literals (knowledge base). These constants stand for the objects that are not named in the initial theory and in the query; we can informally call them *unnamed* objects. Intuitively, countably many constants are needed to answer correctly queries with arbitrary many quantifiers.

However, from Theorem 4 in [25] follows that the constants mentioned neither in the initial theory, nor in a query are indistinguishable for reasoning purposes, and one can pickup any *one* extra constant as a representative when answering a query with single quantifier, or any m constants to answer correctly a query with m quantifiers. This is based on simple intuition that all those constants which occur neither in $\mathcal{D}_{S_0}$ nor in a query behave identically wrt queries. Consequently, if there are only n queries that include at most q quantifiers, then answering these queries correctly wrt the current knowledge base requires at most $n \cdot q$ extra constants. Since actions can change fluents, after doing an action, answering the same precondition query correctly at run time may need new extra constants. But if we impose a bound on the plan length, then only finitely many unnamed objects may need be considered.

Therefore, if there is an integer bound on the plan length, every action can create at most finitely many objects, and, as usual, the BAT $\mathcal{D}$ includes only finitely many axioms, then only finitely many queries about finitely many objects may ever occur in the process of planning, and the planning algorithm may ever need only finitely many constants. This observation is important for implementation purposes. Notice the subtle detail about unnamed objects. They cannot be identified before a plan has been computed. Thus, they cannot be found in advance before search for a plan starts.

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 (with $\wedge$, $\vee$, $\neg$) 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 [27, 28] 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. In contrast, PDDL has a built-in DCA.

For example, we consider the following proper initial theory where the constants start with an upper-case letter. In $\mathcal{D}_{S_0}$ we use names distinct from symbols C_i, C_j since they have different purpose.
$\forall p\big((p{=}Ken \vee p{=}Bob \vee p{=}Sue) \rightarrow person(p)\big)$
$\forall t\big((t{=}T_1 \vee t{=}T_2 \vee t{=}T_3) \rightarrow tray(t)\big)$
$\forall t\big((t{=}T_1 \vee t{=}T_2 \vee t{=}T_3) \rightarrow \neg person(t)\big)$
$\forall p\big((p{=}Ken \vee p{=}Bob \vee p{=}Sue) \rightarrow \neg tray(p)\big)$
$\forall p\big((p{=}Ken \vee p{=}Bob \vee p{=}Sue) \rightarrow \neg served(p,S_0)\big)$
$\forall t,l,r\big((t{=}T_1 \wedge l{=}0 \wedge r{=}100) \rightarrow available(t,l,r,S_0)\big)$
$\forall t,l,r\big((t{=}T_2 \wedge l{=}10 \wedge r{=}40) \rightarrow \neg available(t,l,r,S_0)\big)$.
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 10 and 40, 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 other trays, if any. For people mentioned in $\mathcal{D}_{S_0}$, it is known they were not initially served, but nothing is known about other people not named in $\mathcal{D}_{S_0}$. Thus, every logical model of $\mathcal{D}_{S_0}$ includes facts mentioned above, and a combination of other facts.

Notice that the problem whether $\mathcal{D}_{S_0} \cup \mathcal{E} \models \phi$ is undecidable even if $\mathcal{D}_{S_0}$ is empty, if ϕ is an arbitrary first order formula that can only mention S_0, since the validity problem is undecidable in first order logic. Therefore, Levesque [25] proposed a specialized reasoning procedure $V[\Gamma, \phi]$ that returns one of three values 0 (known false), 1 (known true), or $\frac{1}{2}$ (unknown) depending on the query ϕ and a proper Γ. Subsequently, we assume that reasoning is done using his procedure V. In addition, he proved that V is sound wrt $\models$, and it is complete for a broad class of queries in a so-called normal form $\mathcal{NF}$: if Γ is proper, then for every ϕ in $\mathcal{NF}$, $\Gamma \cup \mathcal{E} \models \phi$ iff $V[\Gamma, \phi] = 1$; and $\Gamma \cup \mathcal{E} \models \neg\phi$ iff $V[\Gamma, \phi] = 0$. For example, V is complete if a query formula has no negations, or has limited occurrences of negations. This is important for our purposes, since we consider precondition formulas for which V is both sound and complete. Note when searching for what specific instantiated action to execute next, it should be possible wrt all first order models.

3.2. Precondition Axioms

$\mathcal{D}_{\mathbf{ap}}$ is a set of action precondition axioms, one per action, of the form $poss(A(\bar{x}), s) \leftrightarrow \Pi_A(\bar{x}, s)$, 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 action function. In this paper, we consider a special case, when $\Pi_A(\bar{x}, s)$ is an extended conjunctive query, e.g., see [29, 30]. An extended conjunctive query (ECQ) is of the form $\exists\vec{x}\phi(\vec{x}, \vec{y})$, 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 precondition axioms for 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 on a tray t there is a slice available such that its angles are 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 $\langle l, r\rangle$ available on a tray t in s and its right angle r is greater than its left angle l. As mentioned in a formal version of this paper [31], ECQ

queries can retrieve at most finitely many distinct tuples in any situation s, and therefore each ground situation has at most finitely many successors. This is important for our implementation: see below.

It is easy to see that $[cutHalf(T_1, 0, 100), serve(T_1, Bob, 0, 50.0), serve(T_1, Sue, 50.0, 100.0)]$ are the actions 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. Our goal formula is an extended conjunctive query. However, if $\mathcal{D}_{S_0}$ does not include 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 10 and 40 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.

3.3. Successor State Axioms (SSA)

Let $\mathcal{D}_{\mathbf{ss}}$ be a set of successor state axioms (SSA): one axiom per fluent. We formulate them using generic notation: formula $\gamma_F^+(\bar{x}, a, s)$ for the case when fluent F becomes true (the "+" superscript), and $\gamma_F^-(\bar{x}, a, s)$ when the fluent that was previously true becomes false (hence, the "−" superscript).

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

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 any. 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. If $\bar{x} \subseteq \bar{u}$, then we say action $A(\bar{u})$ has local effect on F, since its arguments $\bar{u}$ determine the tuple where F changes.

We introduce *weakly context free* (WCF) successor state axioms in this paper; a rigorous definition is provided in [31]. Intuition is that they may have only special *situation independent* contexts $\phi(\bar{x}, \bar{z})$. Their purpose is different from *local effect* axioms where action's arguments $\bar{u}$ include fluent's object arguments $\bar{x}$, $\bar{x} \subseteq \bar{u}$, e.g., see [32]. In contrast to local effect actions, in WCF axioms, some of the fluent arguments are *not* directly determined from the action itself, but are computed using *situation independent functions* with fixed interpretation. We allow both context-free local effect and weakly context free SSAs. The SSAs in our example are the following:

$$\begin{aligned}
&served(t,p,do(a,s)) \leftrightarrow \quad \exists l \exists r(\, a = serve(t,p,l,r)) \ \vee \ served(t,p,s),\\
&angleSize(p,n,do(a,s)) \leftrightarrow \ \exists l \exists r\big(\, a = serve(t,p,l,r) \ \wedge \ n = (l+r)\big) \ \vee angleSize(p,n,s),\\
&available(t,l',r',do(a,s)) \leftrightarrow \ \exists l \exists r\big(a = cutHalf(t,l,r) \wedge l' = l \wedge r' = 0.5 \cdot (l+r)\big) \vee\\
&\qquad\qquad\qquad\qquad \exists l \exists r\big(a = cutHalf(t,l,r) \wedge l' = 0.5 \cdot (l+r) \wedge r' = r\big) \vee\\
&\qquad\qquad\qquad\quad available(t,l',r',s) \wedge a \neq cutHalf(t,l',r') \wedge \neg\exists p(a = serve(t,p,l',r')),
\end{aligned}$$

where $n = (l + r)$ is a function (semantic attachment as in [33]) 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, $0.5 \cdot (l + r)$ is a situation-independent 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 $0.5(l + r)$, and another slice with the angles between $0.5(l + r)$ and r.

In the pizza example, the functions used to compute new objects are numeric, but they can be any situation independent computable functions. For example, attaching a binary tree to a leaf node of another binary tree produces a new tree, while an action that concatenates two strings produces a new string that includes all symbols from the input strings.

3.4. Foundational Axioms

We are ready to discuss Reiter's foundational axioms Σ for situations, where $\sqsubset$ means 'before'. They are based on intuition that situations are sequences of actions, except of the initial situation S_0.

$$\forall a_1 \forall a_2 \forall s_1 \forall s_2 . do(a_1, s_1) = do(a_2, s_2) \rightarrow a_1 = a_2 \wedge s_1 = s_2,$$

$$\forall s. \ \neg(s \sqsubset S_0),$$

$\forall a \forall s \forall s'.\ s \sqsubset do(a, s') \leftrightarrow (s \sqsubset s' \vee s = s')$,

$\forall P.\ \big(P(S_0) \wedge \forall a \forall s(P(s) \rightarrow P(do(a, s)))\big) \rightarrow \forall s(P(s))$.

The last second-order axiom limits the sort *situation* to the smallest set containing S_0 that is closed under the application of do to an action and a situation. These axioms say that the set of situations is really a tree; there are no cycles, and no merging. These axioms Σ are domain independent. They are *not* actually needed in any reasonable implementation that maintains situations as lists or as sequences, since it will naturally satisfy these foundational axioms.

3.5. Progression for Weekly Context Free SSAs

There are two main reasoning mechanisms in SC. One of them relies on the regression operator [34, 14], and another mechanism called progression is responsible for reasoning forward, where after each ground action α the initial theory $\mathcal{D}_{S_0}$ is updated to a new theory $\mathcal{D}_{S_\alpha}$ [19]. We focus on progression.

Let α be a ground action, and let S_α denote the situation term $do(\alpha, S_0)$. A progression $\mathcal{D}_{S_\alpha}$ of $\mathcal{D}_{S_0}$ in response to α is a set of sentences uniform in S_α that accounts for an update due to α. Thus, computing progression requires forgetting all facts that are no longer true, and computing new values to fluents that changed after executing α. $\mathcal{D}_{S_\alpha}$ is defined up to logical equivalence in model-theoretical terms, see [19] for formal details. We will not present the formal definition of progression, but we cite the following important property of progression. It turns out that queries about S_α can be answered from $\mathcal{D}_{S_\alpha}$ alone, i.e., Σ and other axioms from $\mathcal{D}$ are *not* needed:

Theorem 2. *[19] If $\mathcal{D}_{S_\alpha}$ is a progression of the initial theory to S_α, then for any sentence ϕ uniform in S_α, $\mathcal{D} \models \phi$ iff $\mathcal{D}_{S_\alpha} \models \phi$.*

As explained in [19], this "theorem informs us that $\mathcal{D}_{S_\alpha}$ is a strongest postcondition of the precondition $\mathcal{D}_{S_0}$ wrt the action α". Our queries ϕ can be precondition formulas and a goal formula.

In general, progression $\mathcal{D}_{S_\alpha}$ is defined in second-order logic [19]. 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 and local effect context free SSAs (call them both WCF, for short), generalizing the results from [35, 32] one can show that the progression of $\mathcal{D}_{S_0}$ wrt α, $\mathcal{P}(\mathcal{D}_{S_0}, \alpha)$, remains in first order logic. More specifically, it remains a FGP theory, and moreover, it can be efficiently computed. We introduce convenient abbreviations motivated by notation from [36]. Let $\vec{C}$ be any tuple of constants mentioned in $\mathcal{D}_{S_0}$. For any fluent P, let KP denote $\{\vec{C} \mid P(\vec{C}, S_0) \in \mathcal{D}_{S_0}\}$, the set of tuples where P is known to be true and $K\neg P$ denote $\{\vec{C} \mid \neg P(\vec{C}, S_0) \in \mathcal{D}_{S_0}\}$, the set of tuples where P is false.

Theorem 3. *[31] Let $\mathcal{D}$ be weakly context-free, $\mathcal{D}_{S_0}$ be FGP, and α be a ground action. Then the progression of $\mathcal{D}_{S_0}$ wrt α, $\mathcal{P}(\mathcal{D}_{S_0}, \alpha)$ can be computed for each fluent F as follows:*

$$\begin{aligned} KF &:= KF - \gamma^-_F(\alpha) \cup \gamma^+_F(\alpha), \\ K\neg F &:= K\neg F - \gamma^+_F(\alpha) \cup \gamma^-_F(\alpha). \end{aligned}$$

Note the set $\gamma^+_F(\alpha)$ includes tuples for which fluent becomes true thanks to α, $\gamma^-_F(\alpha)$ are tuples for which fluent becomes false due to α. $\mathcal{P}(\Gamma, \alpha)$ remains a FGP formula, i.e., its syntactic form is preserved.

Informally speaking, for those new constants (not mentioned 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 $\mathcal{D}_{S_0}$, but are no longer mentioned in progression, one can say that they represent objects destroyed by α.

If the goal formula is ECQ, then 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.

We address the conceptual limitations of the previous definitions of what is planning by introducing a new proper planning problem that requires to find a ground situation σ such that the goal formula at σ is true wrt *all* models for a given instance of a proper BAT. If there is an upper bound N on the plan length, i.e., on the number of actions in σ, then this is the bounded proper planning (BPP) problem.

Definition 1. *[31] A bounded proper planning (BPP) problem is a triple $P = \langle \mathcal{D}, G(s), N \rangle$, where $\mathcal{D}$ is a proper BAT, $G(s)$ is a goal formula, which is an extended conjunctive query uniform in s and without other free variables, and $N \geq 0$ is a bound. A solution to P is a ground σ with length $\leq N$ s.t.*

$$\mathcal{D} \cup \mathcal{E} \models executable(\sigma) \wedge G(\sigma),$$

where $executable(\sigma)$ means that ground actions in σ are consecutively possible.

Note that one might wish to consider a similar but unbounded proper planning (UPP) problem using a proper BAT introduced above and without the upper bound N on the plan length.

4. Solving Bounded Proper Planning (BPP) Problem

The BPP problem differs from previously explored planning problems since there are infinitely many 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 in $\mathcal{D}_{S_0}$ may no longer occur in $\mathcal{D}_{S_\alpha}$ (objects were destroyed).

It turns out that the BPP problem can sometimes be solved using an improved version of the well-known domain independent heuristic developed for the Fast Forward planner (FF) [37, 38]. Our algorithms and implementation are variation of the usual heuristic search with our improved version of FF adapted to the case when search is done over situations (sequences of actions). The interested readers can find our algorithms in pseudo-code and other implementation details in [39, 31]. The latter paper argues that the A^*-based planner is sound and complete. In contrast to the usual FF heuristic that operates on the grounded transition system, our adaptation computes each action layer at *run-time* from the precondition ECQ queries. Consecutively, it computes each fluent layer using progression to find which new fluents become true. As usual, due to delete-relaxation, there is no need to compute which fluents become false. After computing the heuristic value for an evaluated action the planning graph is discarded and removed from memory including all fluent and action layers.

The key idea of the planning algorithm is that search is actually organized over situations (sequences of actions) that serve as convenient symbolic proxies for FGP theories and their infinite models. Our planner keeps in memory situations only, and recomputes progressions at run time from the initial theory $\mathcal{D}_{S_0}$. An alternative implementation could also keep progressions in a priority queue. The proposed algorithm terminates because there are only finitely many ground situations with bounded length, if the number of created objects is finite and $\mathcal{D}_{S_0}$ mentions only finitely many constants. It relies on the fact that for each situation there are finitely many actions possible. The planning algorithm is *lifted*, since possible actions are determined at run-time when expanding the current situation to compute its successors. Similarly, inside heuristic, preconditions for actions are evaluated at run-time, as mentioned above. In all experiments below, we set the upper bound N to 100.

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 relevant 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 [40, 41], the *helpful actions* proposed in [37] and generalized to *preferred operators* in the Fast Downward planner [42, 43]. As demonstrated experimentally in [44], an additional priority queue for preferred operators is beneficial. We chose 50% : 50%. Note that “useful" situations can be misleading due to delete relaxation inside heuristic.

We have collected data for the generalized Countdown benchmark, a variation of the well-known competition from the British TV show [45, 31]. There are at least 6 counters that can hold any nonnegative integers, but the total number of counters is not fixed (no DCA). Initially, our program assigns a

p1G	p1A	p2G	p2A	p3G	p3A	p4G	p4A	p5G	p5A
5	151	9317	5	5	60	3	3	—	249
14.6	89.7	66.0	13.1	14.8	22.4	14.5	14.5	—	59.8

p6G	p6A	p7G	p7A	p8G	p8A	p9G	p9A	p10G	p10A
5	67	5	9	9317	5	9317	5	10147	5
12.0	20.9	9.1	64.7	69.2	16.2	65.8	13.3	425.4	11.2

Table 1
Addition problems p1-p10: Number of situations expanded and time (sec) rounded to 1 digit after ".".

p1G	p1A	p2G	p2A	p3G	p3A	p4G	p4A	p5G	p5A
5	5	5	5	5	5	5	5	5	5
14.5	14.6	12.0	12.0	14.8	15.0	14.9	14.8	14.9	15.0

p6G	p6A	p7G	p7A	p8G	p8A	p9G	p9A	p10G	p10A
5	5	5	6	5	5	5	5	5	6
12.0	12.5	9.1	30.8	15.0	15.0	12.1	12.1	10.4	39.7

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

randomly generated integer from 0 to 100 to each counter. There are 2 possible actions: either addition or multiplication, for simplicity. 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 usually expanded more situations than A* (see the 1st row). However, greedy search was sometimes faster than A* (see the 2nd row). Surprisingly, the number of expanded states was close to optimal, since apparently our adaptation of FF is very informative. Our random planning instances and the domain encoding are publicly available as [45].

5. Discussion and Related Work

There is little work on planning without DCA. The previous research on generalized planning focused on iterative plans with sensing actions, programs or learning algorithms that support the search for a *policy*, a mapping from states to actions, that works at once for a *set of planning instances*. We are interested in computing a sequential plan for a single problem instance under the open world assumption without DCA. We do not consider generalized planning. Recall we consider only deterministic actions.

Moreover, we solve the planning problem for goals that are conjunctive queries with ∃-quantifiers over object variables, but except for [46, 47], goal formulas must be conjunctions of ground fluents.

It is instructive to compare our BPP with the definition of correctness of a generalized FSA plan given in [48]. A plan axiomatized by FSA is correct iff $\mathcal{D} \cup FSA \models \exists s.T^*(Q_0, S_0, Q_F, s) \wedge G[s]$, where Q_0, Q_F are constants that denote the initial and the final states of a FSA plan, $G(s)$ is a goal formula, and T^* is the reflexive transitive closure of the one-step transitions in the FSA plan. Recall that in [48] there is DCA for all objects except that there is a special numeric fluent that varies over integers, and a finite set of actions can affect this one fluent. Their initial theory (that is not proper) does not specify an initial value of this numeric fluent in S_0, and therefore a plan must be produced for any of its infinitely many values. Notice that, in contrast to BPP, for every model of their $\mathcal{D} \cup FSA$ *there is situation* s that encodes a plan, while we search for a *ground situation* σ that serves as a plan for *all*

models of a single given planning instance specified in the BAT $\mathcal{D}$ that includes proper $\mathcal{D}_{S_0}$. Hu and Levesque (2010) prove that plans generated in their restricted setting are actually correct. Our planner is sound, and it is complete thanks to an upper bound N, see the details in [31].

In [49], Savelli discussed Reiter's formulation of the planning problem [14] wrt an incomplete initial theory and provided a proof based on the properties of goal regression in SC [50] and on the compactness theorem for first order logic. His work is interesting from a theoretical perspective, but it hardly can serve as a foundation for an efficiently implementable planner. Nevertheless, the recent paper [51] demonstrates that goal regression can be used to design an efficient generalized planner.

Helmert [52] provides a comprehensive classification of the numeric planning formalisms, demonstrates the cases where the planning problem is undecidable, and explores the reductions between numeric planning formalisms. The numeric planning problems where the range of values is finite can be reduced to classical planning with DCA; see, e.g. [53, 54]. In a general case, numeric planning goes beyond DCA. Our proposal is different since we consider a BPP problem with *incomplete* knowledge, without DCA.

It is well-known that FF-inspired heuristics based on delete relaxation and value accumulation are inadequate for realistic numeric planning benchmarks related to resources and exchange, e.g., see [55]. For this reason, recent research on numeric planning explores other heuristics, alternative reductions and approaches, e.g., see [56, 57, 58, 59, 60, 61, 62, 63]. Similarly, a challenge of how to develop informative heuristics that guide search for a solution to BPP problem is important for us. This research direction remains important future work.

To our knowledge, there are no other heuristic planners that can solve problems without the DCA given *incomplete* initial theory. The previously developed conformant planners require DCA [64, 65, 66]. The planner in [64] was actually inspired by situation calculus, and it does search over sequences of actions, but it works only at a propositional level. In [67], Finzi, Pirri and Reiter explore open-world planning, but they require DCA, see details in [11].

As mentioned before, [23] proved that there are easy exact translations between the ADL fragment of PDDL and the restricted BATs. One of the restrictions requires that an initial theory $\mathcal{D}_{S_0}$ must enumerate for each fluent finitely many object tuples, where the fluent is initially true. This is needed to develop an easy compilation scheme [20] that maps $\mathcal{D}_{S_0}$ to an instance of ADL classical planning in PDDL 2.1 (Level 1), and therefore the initial theory $\mathcal{D}_{S_0}$ must be a complete theory where all fluents are specified. The authors show that if instead of a complete initial database, $\mathcal{D}_{S_0}$ would be incomplete about at least one unary fluent or situation independent predicate, then this modification would lead to a significant jump in the computational complexity of the planning problem. Also, they show that leaving out DCA, while keeping the initial theory $\mathcal{D}_{S_0}$ a complete database, does not affect the planning problem, since in the presence of a complete initial theory $\mathcal{D}_{S_0}$, all unnamed objects are identical. Therefore, it is possible to take finitely many extra constants to account for the longest finite quantifier prefix in a BAT that ranges over arbitrary many objects. The detailed proofs and discussion are provided in [68]. Their analysis is purely theoretical. They did not consider the task how to design and implement a planner that can use a domain independent heuristic to search for a plan without DCA, when an initial theory $\mathcal{D}_{S_0}$ is *incomplete*.

Petrick [69, 36, 70] presented a knowledge-level approach to conditional planning with sensing actions under incomplete information (without CWA). Their approach is based on a first-order language of the situation calculus. They did not discuss planning without DCA. Our work is different since we rely on progression, we require $\mathcal{D}_{S_0}$ be a proper theory, we consider planning when objects can be destroyed or created, and we control search for a plan with a domain-independent heuristic.

The recent paper [71] considers planning with object creation as 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 [71]. Our BPP problem is more general, since we consider incomplete knowledge. Our semantics is standard for first order logic [6, 8].

Note that in contrast to [72], we do not require that the number of objects where fluent holds must 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.

A lifted deductive planner based on the situation calculus was developed in [39], but their implementation required both DCA and CWA. Their planner was competitive with Fast Downward [42, 43] in terms of IPC scores based on the number of visited states and the length of the plan. They focused on classical planning benchmarks with a small number of objects. Their paper includes a detailed review of previous work on deductive planning.

Several publications discuss when progression can be formulated in FO logic, e.g., see [32, 73, 74]. They did not consider $\mathcal{D}_{S_0}$ as a proper theory, and did not attempt planning. Liu [35, 75] considered $\mathcal{D}_{S_0}$ as a proper theory in SC and used progression for a simple case of local effect SSAs, but they did not consider planning.

Jan Plaza proposed to extend the Herbrand universe with countably many new individual constants and consider ω-Herbrand interpretation to provide a purely declarative semantics for logic programming [76, 77]. His proposal is different from Levesque's who focused on proper theories in his (1998) paper.

Another case of open-world planning is explored in [78, 79]. They work with state constraints that are not explored in our approach. However, they restrict the arities of fluents to use description logics ontologies. In our approach, fluents and actions can have any finite arity.

Future work may consider the case where the identity of some objects is not known, and for this reason $\mathcal{D}_{S_0}$ may include $\exists$-quantifiers over objects; they can be replaced with Skolem constants. This case was discussed for proper KBs in [80]. Additionally, it is interesting to consider acyclic state constraints, since they are realistic and important as argued in [81].

6. Conclusion

The main contribution of this paper is a new interesting planning problem formulated in a restricted but useful fragment of first order logic that has a light-weight implementation. We discussed a few examples that show why this problem is actually interesting. The main constituents of our approach are the following: (1) proper initial theory, (2) preconditions for actions and goal are ECQ formulas, (3) there are syntactic restrictions on SSAs to guarantee that progression remains a FO proper theory, (4) heuristic search can be used sometimes to compute a plan. In addition, we developed a new class of basic action theories with incomplete knowledge where progression can be efficiently computed. We discussed other previously proposed formulations of the planning problem, and argued why our formulation is different. Our planner has a proof-of-the-concept implementation that demonstrates feasibility. The most important future work has to focus on developing new domain independent heuristics.

7. Acknowledgments

Thanks to Yongmei Liu for discussions in Fall 2025. This paper is continuation of our joint work in [31], where more formal details are provided. Our planner and its implementation are based on [39]. Thanks to the Natural Sciences and Engineering Research Council (NSERC) of Canada for partial funding of this research and to the reviewers of the preliminary version of this paper for comments.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] R. Fikes, STRIPS, a retrospective, Artif. Intell. 59 (1993) 227–232. URL: https://doi.org/10.1016/0004-3702(93)90190-M. doi:10.1016/0004-3702(93)90190-M.

[2] M. Soutchanski, An On-line Decision-Theoretic Golog Interpreter, in: G. Lakemeyer (Ed.), The Second International Cognitive Robotics Workshop held in conjunction with 14th European Conference on Artificial Intelligence (ECAI-2000), Available at http://www.cs.torontomu.ca/mes/publications/cr.ps, Berlin, Germany, 2000, pp. 117–124.
[3] M. Soutchanski, A correspondence between two different solutions to the projection task with sensing, in: The 5th Symposium on Logical Formalizations of Commonsense Reasoning, Courant Institute of Mathematical Sciences, New York University, New York, USA, May 20-22, 2001, pp. 235–242. http://www.cs.torontomu.ca/mes/publications/cs2001.pdf.
[4] D. V. McDermott, The 1998 AI planning systems competition, AI Mag. 21 (2000) 35–55. URL: https://doi.org/10.1609/aimag.v21i2.1506. doi:10.1609/AIMAG.V21I2.1506.
[5] M. Ben-Ari, Mathematical Logic for Computer Science, 3rd ed., Springer, 2012.
[6] H. Enderton, A Mathematical Introduction to Logic, Harcourt Press, 2nd edit., 2001.
[7] U. Schöning, Logic for Computer Scientists, Springer/Birkhäuser, 2008. doi:https://doi.org/10.1007/978-0-8176-4763-6.
[8] J. Shoenfield, Mathematical Logic, A.K.Peters/ CRC Press, 2018. doi:https://doi.org/10.1201/9780203749456.
[9] J. McCarthy, Situations, Actions and Causal Laws, Technical Report Memo 2, Stanford University AI Laboratory, Stanford, CA, 1963. URL: https://stacks.stanford.edu/file/druid:kf190cg0706/SC1041_SAIL_AIM_002.pdf, reprinted in Marvin Minsky, editor, Semantic Information Processing, MIT Press, 1968.
[10] J. McCarthy, P. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, in: B. Meltzer, D. Michie (Eds.), Machine Intelligence, volume 4, Edinburgh Univ. Press, 1969, pp. 463–502. URL: http://jmc.stanford.edu/articles/mcchay69.html.
[11] R. Reiter, Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems, MIT, http://cognet.mit.edu/book/knowledge-action, 2001. URL: http://www.cs.toronto.edu/cogrobo/kia/index.html.
[12] H. Levesque, F. Pirri, R. Reiter, Foundations for the situation calculus, Linköping Electronic Articles: Electronic Transactions on AI https://ep.liu.se/ej/etai/fulltext/98005.pdf vol. 3, N 18 (1998).
[13] F. Lin, Situation calculus, in: Handbook of Knowledge Representation, volume 3 of *Foundations of Artificial Intelligence*, 2008, pp. 649–669. URL: https://doi.org/10.1016/S1574-6526(07)03016-7.
[14] R. Reiter, The Frame Problem in the Situation Calculus: A Simple Solution (sometimes) and a Completeness Result for Goal Regression, in: V. Lifschitz (Ed.), AI and Mathematical Theory of Computation: Papers in Honor of John McCarthy, Academic Press, 1991, pp. 359–380. URL: http://www.cs.toronto.edu/kr/publications/simple.pdf.
[15] M. Fox, D. Long, PDDL2.1: an extension to PDDL for expressing temporal planning domains, J. Artif. Intell. Res. 20 (2003) 61–124. URL: https://doi.org/10.1613/jair.1129. doi:10.1613/JAIR.1129.
[16] A. Gerevini, P. Haslum, D. Long, A. Saetti, Y. Dimopoulos, Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners, Artif. Intell. 173 (2009) 619–668. URL: https://doi.org/10.1016/j.artint.2008.10.012. doi:10.1016/J.ARTINT.2008.10.012.
[17] P. Haslum, N. Lipovetzky, D. Magazzeni, C. Muise, An Introduction to the Planning Domain Definition Language, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2019. URL: https://doi.org/10.2200/S00900ED2V01Y201902AIM042.
[18] R. E. Fikes, N. J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, Artificial Intelligence 2 (1971) 189–208. doi:10.1016/0004-3702(71)90010-5.
[19] F. Lin, R. Reiter, How to Progress a Database, Artificial Intelligence 92 (1997) 131–167.
[20] B. Nebel, On the compilability and expressive power of propositional planning formalisms, J. Artif. Intell. Res. 12 (2000) 271–315. URL: https://doi.org/10.1613/jair.735. doi:10.1613/JAIR.735.
[21] E. P. D. Pednault, ADL and the State-Transition Model of Action, J. of Logic and Comput. 4 (1994) 467–512. URL: https://doi.org/10.1093/logcom/4.5.467.
[22] P. Eyerich, B. Nebel, G. Lakemeyer, J. Claßen, Golog and PDDL: What is the Relative Expressivness, in: Proceedings of the 2006 International Symposium on Practical Cognitive Agents and Robots,

ACM, 2006. URL: https://dl.acm.org/doi/10.1145/1232425.1232440.
[23] G. Röger, M. Helmert, B. Nebel, On the Relative Expressiveness of ADL and Golog: The Last Piece in the Puzzle, in: 11th Intern. Confer., KR-2008, 2008, pp. 544–550. URL: http://www.aaai.org/Library/KR/2008/kr08-053.php.
[24] R. Fuentetaja, T. de la Rosa, Compiling irrelevant objects to counters. special case of creation planning, AI Commun. 29 (2016) 435–467. URL: https://doi.org/10.3233/AIC-150692.
[25] H. J. Levesque, A Completeness Result for Reasoning with Incomplete First-Order KBs, in: KR-1998, 1998, pp. 14–23. URL: https://www.cs.toronto.edu/~hector/Papers/Kr98.pdf.
[26] S. A. Cook, T. Pitassi, Herbrand Theorem, Equality, and Compactness (Course Notes), https://www.cs.columbia.edu/~toni/Courses/Logic2022/Notes/page39.pdf, 2022.
[27] R. Reiter, Equality and Domain Closure in First-Order Databases, J. ACM 27 (1980) 235–249. URL: https://doi.org/10.1145/322186.322189.
[28] R. Reiter, Towards a logical reconstruction of relational database theory, in: On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages, Book resulting from the Intervale Workshop 1982, Springer, 1982, pp. 191–233.
[29] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995. URL: http://webdam.inria.fr/Alice/.
[30] A. Chandra, P. Merlin, Optimal implementation of conjunctive queries in relational data bases, in: 9th Annual ACM Symp. on Theory of Computing (STOC), 1977, pp. 77–90. URL: https://doi.org/10.1145/800105.803397.
[31] M. Soutchanski, Y. Liu, 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] Y. Liu, G. Lakemeyer, On First-Order Definability and Computability of Progression for Local-Effect Actions and Beyond, in: 21st IJCAI-2009, 2009, pp. 860–866. URL: http://ijcai.org/papers09/Papers/IJCAI09-147.pdf.
[33] R. W. Weyhrauch, Prolegomena to a theory of mechanized formal reasoning, Artif. Intell. 13 (1980) 133–170. URL: https://doi.org/10.1016/0004-3702(80)90015-6.
[34] R. Waldinger, Achieving Several Goals Simultaneously, in: Machine Intelligence, volume 8, 1977, pp. 94–136. URL: https://www.sri.com/wp-content/uploads/2021/12/763.pdf, https://www.sri.com/publication/artificial-intelligence-pubs/achieving-several-goals-simultaneously/.
[35] Y. Liu, H. J. Levesque, Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions, in: 19th International Joint Conference on Artificial Intelligence, IJCAI-2005, 2005, pp. 522–527. URL: http://ijcai.org/Proceedings/05/Papers/0706.pdf.
[36] R. P. A. Petrick, F. Bacchus, Extending the knowledge-based approach to planning with incomplete information and sensing, in: Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS 2004), 2004, pp. 2–11. URL: https://www.cs.toronto.edu/~fbacchus/Papers/PetrickBacchusICAPS2004.pdf.
[37] J. Hoffmann, B. Nebel, The FF System: Fast Plan Generation Through Heuristic Search, J. Artif. Intell. Res. 14 (2001) 253–302. URL: https://doi.org/10.1613/jair.855.
[38] D. Bryce, S. Kambhampati, Planning Graph Based Reachability Heuristics, AI Mag. 28 (2007) 47–83. URL: https://doi.org/10.1609/aimag.v28i1.2028.
[39] M. Soutchanski, R. Young, Planning as theorem proving with heuristics, in: Proceedings of the Italian Workshop on Planning and Scheduling, co-located with AIxIA-2023, volume 3585 of *CEUR Workshop Proceedings*, 2023. https://ceur-ws.org/Vol-3585/paper10_RCRA6.pdf.
[40] D. V. McDermott, A heuristic estimator for means-ends analysis in planning, in: Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-1996), 1996, pp. 142–149. URL: http://www.aaai.org/Papers/AIPS/1996/AIPS96-018.pdf.
[41] D. V. McDermott, Using regression-match graphs to control search in planning, Artif. Intell. 109 (1999) 111–159. URL: https://doi.org/10.1016/S0004-3702(99)00010-7.
[42] M. Helmert, The Fast Downward Planning System, J. Artif. Intell. Res. 26 (2006) 191–246. URL:

https://doi.org/10.1613/jair.1705.
[43] Helmert et. al., Fast Downward at Github, https://github.com/aibasel/downward, 2022. Accessed: 2022-11-17.
[44] S. Richter, M. Helmert, Preferred operators and deferred evaluation in satisficing planning, in: Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, 2009, pp. 273–280. URL: http://aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/view/700.
[45] M. Soutchanski, The Countdown Planning Benchmark, 2025. PROLOG code of randomly generated instances at https://www.cs.torontomu.ca/mes/publications/Countdown.zip.
[46] G. Francès, H. Geffner, ∃-STRIPS: Existential quantification in planning and constraint satisfaction, in: S. Kambhampati (Ed.), 25th International Joint Conference on Artificial Intelligence, IJCAI-2016, 2016, pp. 3082–3088. URL: http://www.ijcai.org/Abstract/16/437.
[47] M. Funkquist, S. Ståhlberg, H. Geffner, Learning to ground existentially quantified goals, in: 21st Intern. Conf. on Principles of Knowledge Represent. and Reasoning, 2024, pp. 856 – 866. URL: https://doi.org/10.24963/kr.2024/80.
[48] Y. Hu, H. Levesque, A correctness result for reasoning about one-dimensional planning problems, in: 12th Intern. Conf on Principles of Knowledge Representation and Reasoning, 2010. URL: http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1270.
[49] F. Savelli, Existential assertions and quantum levels on the tree of the situation calculus, Artif. Intell. 170 (2006) 643–652. URL: https://doi.org/10.1016/j.artint.2006.01.002. doi:10.1016/J.ARTINT.2006.01.002.
[50] F. Pirri, R. Reiter, Some contributions to the metatheory of the situation calculus, Journal of the ACM (JACM) 46 (1999) 325–361. URL: http://www.cs.toronto.edu/kr/publications/metatheory.pdf.
[51] D. Z. Chen, T. Hofmann, T. Q. Klassen, S. A. McIlraith, Satisficing and optimal generalised planning via goal regression, in: 40th AAAI Conference on Artificial Intelligence, 2026, pp. 36197–36206. URL: https://doi.org/10.1609/aaai.v40i43.40938.
[52] M. Helmert, Decidability and undecidability results for planning with numerical state variables, in: Sixth International Conference on Artificial Intelligence Planning Systems AIPS-2002, 2002, pp. 44–53. URL: http://www.aaai.org/Library/AIPS/2002/aips02-005.php.
[53] N. Gigante, E. Scala, On the compilability of bounded numeric planning, in: 32nd International Joint Conference on Artificial Intelligence IJCAI-2023, 2023, pp. 5341–5349. URL: https://doi.org/10.24963/ijcai.2023/593.
[54] L. Bonassi, F. Percassi, E. Scala, Towards practical classical planning compilations of numeric planning, in: Proceedings of the 39th AAAI Conference on Artificial Intelligence AAAI-2025, 2025, pp. 26472–26480. URL: https://doi.org/10.1609/aaai.v39i25.34847.
[55] A. J. Coles, A. Coles, M. Fox, D. Long, A hybrid LP-RPG heuristic for modelling numeric resource flows in planning, J. Artif. Intell. Res. 46 (2013) 343–412. URL: https://doi.org/10.1613/jair.3788. doi:10.1613/JAIR.3788.
[56] M. Cardellini, E. Giunchiglia, Pushing the envelope in numeric pattern planning, in: KR-25, 2025, pp. 762–771. URL: https://proceedings.kr.org/2025/73/kr2025-0073-cardellini-et-al.pdf.
[57] D. Z. Chen, S. Thiébaux, Novelty heuristics, multi-queue search, and portfolios for numeric planning, in: Proceedings of the 17th International Symposium on Combinatorial Search (SoCS 2024), 2024, pp. 203–207. URL: https://arxiv.org/abs/2404.05235v2.
[58] L. Illanes, S. A. McIlraith, Numeric planning via abstraction and policy guided search, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence IJCAI, 2017, pp. 4338–4345. URL: https://doi.org/10.24963/ijcai.2017/606. doi:10.24963/IJCAI.2017/606.
[59] R. Kuroiwa, A. Shleyfman, C. Piacentini, M. P. Castro, J. C. Beck, The lm-cut heuristic family for optimal numeric planning with simple conditions, J. Artif. Intell. Res. 75 (2022) 1477–1548. URL: https://doi.org/10.1613/jair.1.14034. doi:10.1613/JAIR.1.14034.
[60] C. Piacentini, M. P. Castro, A. A. Ciré, J. C. Beck, Linear and integer programming-based heuristics for cost-optimal numeric planning, in: Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18), 2018, pp. 6254–6261. URL: https://doi.org/10.1609/aaai.v32i1.12082. doi:10.1609/AAAI.V32I1.12082.

[61] E. Scala, P. Haslum, S. Thiébaux, M. Ramírez, Subgoaling techniques for satisficing and optimal numeric planning, J. Artif. Intell. Res. 68 (2020) 691–752. URL: https://doi.org/10.1613/jair.1.11875. doi:10.1613/JAIR.1.11875.
[62] E. Scala, A. Saetti, I. Serina, A. E. Gerevini, Search-guidance mechanisms for numeric planning through subgoaling relaxation, in: Proceedings of the 30th International Conference on Automated Planning and Scheduling ICAPS, 2020, pp. 226–234. URL: https://ojs.aaai.org/index.php/ICAPS/article/view/6665.
[63] E. Scala, L. Bonassi, On using lazy greedy best-first search with subgoaling relaxation in numeric planning problems, in: Proc. of the 35th Intern. Conf. on Autom. Planning and Sched. (ICAPS-2025), 2025, pp. 245–249.
[64] J. Hoffmann, R. Brafman, Conformant planning via heuristic forward search: A new approach, Artificial Intelligence 170 (2006) 507–541. URL: https://www.sciencedirect.com/science/article/pii/S0004370206000087.
[65] H. Palacios, H. Geffner, Compiling uncertainty away in conformant planning problems with bounded width, J. Artif. Intell. Res. 35 (2009) 623–675. URL: https://doi.org/10.1613/jair.2708.
[66] A. Grastien, E. Scala, CPCES: A planning framework to solve conformant planning problems through a counterexample guided refinement, Artif. Intell. 284 (2020) 103271. URL: https://doi.org/10.1016/j.artint.2020.103271.
[67] A. Finzi, F. Pirri, R. Reiter, Open world planning in the situation calculus, in: Proceedings of the 7th AAAI, 2000, pp. 754–760. URL: http://www.cs.toronto.edu/cogrobo/Papers/openworld-aaai00.ps.gz.
[68] G. Röger, Planning Techniques and the Action Language Golog, Dissertation, Ph.D. thesis, Technische Fakultät Albert-Ludwigs-Universität, Freiburg, Germany, 2014. URL: https://ai.dmi.unibas.ch/papers/roeger-dissertation.pdf.
[69] R. P. A. Petrick, F. Bacchus, A knowledge-based approach to planning with incomplete information and sensing, in: Proceedings of the 6th Intern. Conf. on Artificial Intelligence Planning Systems (ICAPS), 2002, pp. 212–222.
[70] R. Petrick, A Knowledge-Level Approach for Effective Acting, Sensing, and Planning, Ph.D. thesis, Department of Computer Science, University of Toronto, 2006. URL: https://tspace.library.utoronto.ca/handle/1807/121667. doi:https://hdl.handle.net/1807/121667.
[71] A. B. Corrêa, G. D. Giacomo, M. Helmert, S. Rubin, Planning with object creation, in: 34th International Conference on Automated Planning and Scheduling, ICAPS 2024, 2024, pp. 104–113. URL: https://doi.org/10.1609/icaps.v34i1.31466.
[72] G. De Giacomo, Y. Lespérance, F. Patrizi, Bounded situation calculus action theories, Artif. Intell. 237 (2016) 172–203. URL: https://doi.org/10.1016/j.artint.2016.04.006.
[73] S. Vassos, S. Sardina, H. Levesque, Progressing basic action theories with non-local effect actions, in: International Symposium on Commonsense Reasoning, 2009. URL: https://commonsensereasoning.org/2009/papers/commonsense2009paper22.pdf.
[74] S. Vassos, F. Patrizi, A classification of first-order progressable action theories in situation calculus, in: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013, 2013, pp. 1132–1138. URL: https://www.ijcai.org/Proceedings/13/Papers/171.pdf.
[75] Y. Liu, Tractable Reasoning in Incomplete First-Order Knowledge Bases, Ph.D. thesis, Department of Computer Science, University of Toronto, 2005.
[76] J. A. Plaza, Fully Declarative Programming With Logic - Mathematical Foundations, Ph.D. thesis, City University of New York, 1990.
[77] J. A. Plaza, Operators on lattices of omega-herbrand interpretations, in: A. Nerode, M. A. Taitslin (Eds.), Logical Foundations of Computer Science Tver'92, 2nd International Symposium, volume 620 of *Lecture Notes in Computer Science*, Springer, 1992, pp. 358–369. URL: https://doi.org/10.1007/BFb0023889. doi:10.1007/BFB0023889.
[78] S. Borgwardt, J. Hoffmann, A. Kovtunova, M. Steinmetz, Making DL-Lite Planning Practical, in: Proc. of the 18th Intern. Conf. on Principles of Knowledge Representation and Reasoning, 2021, pp. 641–645. URL: https://doi.org/10.24963/kr.2021/61.
[79] S. Borgwardt, J. Hoffmann, A. Kovtunova, M. Krötzsch, B. Nebel, M. Steinmetz, Expressivity

of Planning with Horn Description Logic Ontologies, in: 36th AAAI Conference on Artificial Intelligence, AAAI-2022, 2022, pp. 5503–5511. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20489.

[80] G. De Giacomo, Y. Lespérance, H. J. Levesque, Efficient reasoning in proper knowledge bases with unknown individuals, in: 22nd International Joint Conference on Artificial Intelligence (IJCAI), 2011, pp. 827–832. URL: https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-144.

[81] S. A. McIlraith, Integrating Actions and State Constraints: A Closed-form Solution to the Ramification Problem (sometimes), Artif. Intell. 116 (2000) 87–121. URL: http://dx.doi.org/10.1016/S0004-3702(99)00087-9. doi:https://doi.org/10.1016/S0004-3702(99)00087-9.