

# Automated Planning with Incomplete Open World Models\*

Mikhail Soutchanski

Toronto Metropolitan (formerly Ryerson) University,  
245 Church St, ENG281, Toronto, Ontario, M5B 2K3, Canada  
soutchanski (at) protonmail.com  
<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. 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, 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 is associated with infinitely many countable first order models. We show a proof-of-the-concept planner can sometimes solve the BPP problem by using a domain-independent heuristic that guides search over sequences of actions. We discuss the differences between BPP and the previously explored formulations of the planning problem.

## 1 Introduction

Instead of solving an existing problem, we propose a new interesting planning problem formulated in a practical fragment of first order logic, show it can be solved sometimes with a minor adaptation of an existing heuristic, and argue a new domain independent heuristic is needed to solve efficiently other planning instances from a broad class of benchmarks. To simplify presentation our focus is on informal explanations and on providing 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 (Soutchanski 2000, 2001).

As an example, consider a variation of Logistic benchmark 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, the lower-case letters  $x, y$  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.  
 $\forall x \forall y. x = B_2 \wedge y = T_2 \rightarrow in(x, y)$   
 $\forall x \forall y. x = B_2 \wedge y \neq T_2 \rightarrow \neg in(x, y)$
- 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 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 basics of first order logic, e.g., from one of the textbooks (Ben-Ari 2012; Enderton 2001; Schöning 2008; Shoenfield 2018).

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.

\*This is a version of the paper that appears as (Soutchanski and Liu 2025). It includes more formal details and proof sketches.

## 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 (McCarthy 1963; McCarthy and Hayes 1969) to capture common sense reasoning about the actions and events and it was subsequently refined by Reiter (Reiter 2001) who introduced the **basic action theories** (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 (Levesque, Pirri, and Reiter 1998; Lin 2008).

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  $\sigma, \sigma'$  denote situation terms and  $A_i(\bar{x})$ , or  $\alpha, \alpha_1, \alpha_2, \alpha'$ , represent action functions and action terms, respectively. The shorthand  $do([\alpha_1, \dots, \alpha_n], S_0)$  represents the situation  $do(\alpha_n, do(\dots, do(\alpha_1, S_0) \dots))$  resulting from the execution of actions  $\alpha_1, \dots, \alpha_n$  in  $S_0$ . The relation  $\sigma \sqsubseteq \sigma'$  between situations terms  $\sigma$  and  $\sigma'$  means that  $\sigma$  is an initial subsequence of  $\sigma'$ . 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  $\psi$  mention only situation  $s$  as their situation argument, and  $\psi(s)$  has no quantifiers over the situation variables.

The basic action theory  $\mathcal{D}$  (Reiter 1991, 2001) 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  $\Sigma$  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) (Fox and Long 2003; Gerevini et al. 2009; Haslum et al. 2019) 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 all the effects of actions on a fluent  $on(x, z, s)$  and an additional precondition axiom for action  $move-b-to-b(x, y, z)$ .

```
 $\forall x, y, z, s. poss(move-b-to-b(x, y, z), s) \leftrightarrow$   

 $clear(x, s) \wedge clear(z, s) \wedge on(x, y, s) \wedge x \neq z$   

/* SSA for fluent  $on(x, z, s)$  */  

 $\forall x, z, a, s. on(x, z, do(a, s)) \leftrightarrow$   

 $\exists y(a = move-b-to-b(x, y, z)) \vee a = move-t-to-b(x, z) \vee$   

 $on(x, z, s) \wedge \neg \exists y(a = move-b-to-b(x, z, y)) \wedge$   

 $\neg(a = move-b-to-t(x, z))$ .
```

This SSA says that block  $x$  is on block  $z$  in situation  $do(a, s)$  that results from doing action  $a$  in 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 (Fikes and Nilsson 1971) fragment of PDDL can be specified in the situation calculus (Lin and Reiter 1997). For our illustrative purposes here, it is sufficient to say that PDDL and SC are based on similar underlying 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 (Nebel 2000), it was shown that there is an exact translation between the ADL (Pednault 1994) 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 (Eyerich et al. 2006; Röger, Helmert, and Nebel 2008).

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 (Fox and Long 2003; Röger, Helmert, and Nebel 2008). 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 (2016). 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  $\mathcal{D}$  (Reiter 2001). 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 (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 (Levesque 1998). 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.** (Cook and Pitassi 2022) *Let  $\Phi$  be any set of formulas in a first order logic language that includes  $=$ . Then  $\Phi$  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 language includes any constants.

Moreover, following (Levesque 1998),  $\mathcal{D}_{S_0}$  is formulated in a standard first order logic language with a countably infinite set of (object) constants  $\{C_1, C_2, \dots\}$ , 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 **unique name axioms** (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 (Levesque 1998) 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 (Levesque 1998) 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  $\rho$  is a fluent literal whose arguments are distinct variables. Recall that the *domain closure assumption* (DCA) for objects (Reiter 1980) 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  $C_1, C_2, \dots, C_K$ , and for any object variable  $x$ ,  $\forall x$  is understood as  $\forall x(x = C_1 \vee x = C_2 \vee \dots \vee x = C_K)$ . 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; we use them instead of symbols  $C_i, C_j$  to improve readability.

$\forall p((p = Ken \vee p = Bob \vee p = Sue) \rightarrow person(p))$   
 $\forall t((t = T_1 \vee t = T_2 \vee t = T_3) \rightarrow tray(t))$   
 $\forall t((t = T_1 \vee t = T_2 \vee t = T_3) \rightarrow \neg person(t))$   
 $\forall p((p = Ken \vee p = Bob \vee p = Sue) \rightarrow \neg tray(p))$   
 $\forall p((p = Ken \vee p = Bob \vee p = Sue) \rightarrow \neg served(p, S_0))$   
 $\forall t, l, r((t = T_1 \wedge l = 0 \wedge r = 100) \rightarrow available(t, l, r, S_0))$   
 $\forall t, l, r((t = T_2 \wedge l = 10 \wedge r = 40) \rightarrow \neg available(t, l, r, S_0))$ .

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  $\phi$  is an arbitrary first order formula (that can only mention  $S_0$ ), since the validity problem is undecidable in first order logic. Therefore, Levesque (Levesque 1998) 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  $\phi$  and a proper  $\Gamma$ . 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  $\Gamma$  is proper, then for every  $\phi$  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

$\mathcal{D}_{ap}$  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 action function. In this paper, we consider a special case, when  $\Pi_A(\bar{x}, s)$  is an extended conjunctive query, e.g., see (Chandra and Merlin 1977; Abiteboul, Hull, and Vianu 1995). An extended conjunctive query (ECQ) is of the form  $\exists \bar{x} \phi(\bar{x}, \bar{y})$ , where  $\phi$  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  $\phi$ . 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 (Soutchanski and Liu 2025), 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 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  $\sigma$  where the goal formula  $\exists p_1, p_2, n(serve(p_1, \sigma) \wedge served(p_2, \sigma) \wedge p_1 \neq p_2 \wedge angleSize(p_1, n, \sigma) \wedge angleSize(p_2, n, \sigma))$  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 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  $\sigma$ , where the goal formula holds. Notice the goal formula is an extended conjunctive query.

### 3.3 Foundational Axioms

We are ready to discuss Reiter's foundational axioms  $\Sigma$ :

$$\begin{aligned}
 &\forall a_1 \forall a_2 \forall s_1 \forall s_2. do(a_1, s_1) = do(a_2, s_2) \rightarrow a_1 = a_2 \wedge s_1 = s_2 \\
 &\neg(s \sqsubset S_0), \\
 &\forall a \forall s \forall s'. s \sqsubset do(a, s') \leftrightarrow (s \sqsubset s' \vee s = s'), \\
 &\forall P. (P(S_0) \wedge \forall a \forall s (P(s) \rightarrow P(do(a, s)))) \rightarrow \forall s (P(s)).
 \end{aligned}$$

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  $\Sigma$  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.

### 3.4 Successor State Axioms (SSA)

Let  $\mathcal{D}_{ss}$  be a set of successor state axioms (SSA): one axiom per fluent. We formulate them using generic notation:  $\gamma_F^+(\bar{x}, a, s)$  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  $\gamma_F$ '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 introduce *weakly context free* (WCF) successor state axioms in this paper; a rigorous definition is provided in

(Soutchanski and Liu 2025). Intuition is that they may have only special situation independent contexts  $\phi(\bar{x}, \bar{z})$ . Their purpose is different from *local effect* actions where the action arguments  $\bar{u}$  include fluent's object arguments  $\bar{x}$ ,  $\bar{x} \subseteq \bar{u}$ , e.g., see (Liu and Lakemeyer 2009). In contrast, 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:

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

where  $n = (l + r)$  is a function (semantic attachment as in (Weyhrauch 1980)) 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.

### 3.5 Progression for Weekly Context Free SSAs

There are two main reasoning mechanisms in SC. One of them relies on the regression operator (Waldinger 1977; Reiter 1991), and another mechanism called progression is responsible for reasoning forward, where after each ground action  $\alpha$  the initial theory  $\mathcal{D}_{S_0}$  is updated to a new theory  $\mathcal{D}_{S_\alpha}$  (Lin and Reiter 1997). We focus on progression

Let  $\alpha$  be a ground action, and let  $S_\alpha$  denote the situation term  $\text{do}(\alpha, S_0)$ . A progression  $\mathcal{D}_{S_\alpha}$  of  $\mathcal{D}_{S_0}$  in response to  $\alpha$  is a set of sentences uniform in  $S_\alpha$  that accounts for an update due to  $\alpha$ . Thus, computing progression requires forgetting all facts that are no longer true, and computing new values to fluents that changed after executing  $\alpha$ .  $\mathcal{D}_{S_\alpha}$  is defined up to logical equivalence in model-theoretical terms, see (Lin and Reiter 1997) for formal details. It turns out that queries about  $S_\alpha$  can be answered from  $\mathcal{D}_{S_\alpha}$  alone, i.e.,  $\Sigma$  and other axioms from  $\mathcal{D}$  are *not* needed:

**Theorem.** (Lin and Reiter 1997) *If  $\mathcal{D}_{S_\alpha}$  is a progression of the initial theory to  $S_\alpha$ , then for any sentence  $\phi$  uniform in  $S_\alpha$ ,  $\mathcal{D} \models \phi$  iff  $\mathcal{D}_{S_\alpha} \models \phi$ .*

As explained in (Lin and Reiter 1997), this “theorem in-

forms us that  $\mathcal{D}_{S_\alpha}$  is a strongest postcondition of the precondition  $\mathcal{D}_{S_0}$  wrt the action  $\alpha$ ”. Our queries  $\phi$  can be precondition formulas and a goal formula.

In general, progression  $\mathcal{D}_{S_\alpha}$  is defined in second-order logic (Lin and Reiter 1997). 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 (Liu and Levesque 2005; Liu and Lakemeyer 2009) one can show that the progression of  $\mathcal{D}_{S_0}$  wrt  $\alpha$ ,  $\mathcal{P}(\mathcal{D}_{S_0}, \alpha)$ , 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 (Petrick and Bacchus 2004). Let  $\vec{C}$  be a tuple of constants taken from  $\{C_1, C_2, \dots\}$ . 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.** (Soutchanski and Liu 2025) *Let  $\mathcal{D}$  be weakly context-free,  $\mathcal{D}_{S_0}$  be FGP, and  $\alpha$  be a ground action. Then the progression of  $\mathcal{D}_{S_0}$  wrt  $\alpha$ ,  $\mathcal{P}(\mathcal{D}_{S_0}, \alpha)$  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  $\alpha$ ,  $\gamma_F^-(\alpha)$  are tuples for which fluent becomes false due to  $\alpha$ .  $\mathcal{P}(\Gamma, \alpha)$  remains a FO formula.

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  $\alpha$ .

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 bounded proper planning (BPP) problem that (1) requires to find a ground situation  $\sigma$  such that the goal formula at  $\sigma$  is true wrt *all* models for a given problem instance and (2) includes an upper bound  $N$  on the plan length, i.e., the number of actions in situations.

**Definition.** (Soutchanski and Liu 2025) *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  $\sigma$  with length  $\leq N$  s.t.*

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

where *executable*( $\sigma$ ) means that ground actions in  $\sigma$  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  $N$ .

## 4 Solving Bounded Proper Planning (BPP) Problem

The BPP problem differs from previously explored planning problems since there are infinitely many countable 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) (Hoffmann and Nebel 2001; Bryce and Kambhampati 2007). 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 (Soutchanski and Young 2023; Soutchanski and Liu 2025). 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 is sound and complete; it terminates because there are only finitely many ground situations with bounded length. 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 *avored actions* proposed in (McDermott 1996, 1999), the *helpful actions* proposed in (Hoffmann and Nebel 2001) and generalized to *preferred*

*operators* in the Fast Downward planner (Helmert 2006; Helmert et. al. 2022). As demonstrated experimentally in (Richter and Helmert 2009), 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.

| 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-p5, p6-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).

We have collected data for the generalized Countdown benchmark, a variation of the well-known competition from the British TV show (Soutchanski 2025; Soutchanski and Liu 2025). 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 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 usually faster than A\* (see the 2nd row). Surprisingly, the number of expanded states was close to optimal. Our random planning instances and the domain encoding are publicly available as (Soutchanski 2025).

## 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 in this paper. Recall we consider only deterministic actions.

Moreover, we solve the single planning problem for goals that are conjunctive queries with  $\exists$ -quantifiers over object variables, but except for (Francès and Geffner 2016; Funkquist, Ståhlberg, and Geffner 2024), previous work considered mostly conjunctions of ground fluents as goals.

It is instructive to compare our BPP with the definition of correctness of a generalized FSA plan given in (Hu and Levesque 2010). 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 (Hu and Levesque 2010) 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*  $\sigma$  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  $N$ .

(Savelli 2006) discussed Reiter’s formulation of the planning problem (Reiter 1991) wrt an incomplete initial theory and provided a proof based on the properties of goal regression in SC (Pirri and Reiter 1999) and on the compactness theorem for first order logic. His work is interesting from a theoretical perspective, but it cannot serve as a foundation for an efficiently implementable planner.

Helmert (Helmert 2002) 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. (Gigante and Scala 2023; Bonassi, Percassi, and Scala 2025). In a general case, numeric planning goes beyond DCA. Our proposal is different since we consider a BPP problem with *incomplete* knowledge, without DCA, and each FO model of progression in BPP is infinite in contrast to numeric planning, where each state is a finite set.

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 (Coles et al. 2013). For this rea-

son, recent research on numeric planning explores other heuristics, alternative reductions and approaches, e.g., see (Cardellini and Giunchiglia 2025; Chen and Thiébaux 2024; Illanes and McIlraith 2017; Kuroiwa et al. 2022; Piacentini et al. 2018; Scala et al. 2020a,b; Scala and Bonassi 2025). 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 conformant planners previously developed require DCA (Hoffmann and Brafman 2006; Palacios and Geffner 2009; Grastien and Scala 2020). The planner in (Hoffmann and Brafman 2006) was actually inspired by situation calculus, and it does search over sequences of actions, but it works only at a propositional level. (Finzi, Pirri, and Reiter 2000) does open-world planning, but they require DCA, see details in (Reiter 2001).

As mentioned before, (Röger, Helmert, and Nebel 2008) 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 (Nebel 2000) that maps  $\mathcal{D}_{S_0}$  to an instance of ADL classical planning in PDDL 2.1 (Level 1). 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 (Röger 2014). 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 and Bacchus 2002, 2004; Petrick 2006) 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.

(Corrêa et al. 2024) 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 (Corrêa et al. 2024). Our BPP problem is

more general, since each state has a model with an infinite universe: we consider incomplete knowledge. Our semantics is standard for FO logic (Enderton 2001; Shoenfield 2018).

Note that in contrast to (De Giacomo, Lespérance, and Patrizi 2016), 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.

(Soutchanski and Young 2023) 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 (Helmert 2006; Helmert et. al. 2022) 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 (Liu and Lakemeyer 2009; Vassos, Sardina, and Levesque 2009; Vassos and Patrizi 2013). They did not consider  $\mathcal{D}_{S_0}$  as a proper theory, and did not attempt planning. (Liu and Levesque 2005; Liu 2005) 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  $\omega$ -Herbrand interpretation to provide a purely declarative semantics for logic programming (Plaza 1990, 1992). His proposal is different from Levesque’s who focused on proper theories in his (1998) paper.

The case of open-world planning is explored in (Borgwardt et al. 2021, 2022). They work with state constraints that are not explored in our approach. However, they restrict the arities of fluents to use description logics ontologies, but 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 (De Giacomo, Lespérance, and Levesque 2011). Additionally, it is interesting to consider acyclic state constraints, since they are realistic and important (McIlraith 2000).

## 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. Our planner has a proof-of-the-concept implementation that demonstrates feasibility.

## 7 Acknowledgments

Thanks to Yongmei Liu for discussions in Fall 2025. This paper is continuation of our joint work in (Soutchanski and Liu 2025). Our planner and its implementation are based on (Soutchanski and Young 2023). 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.

## References

- Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley. ISBN 0-201-53771-0.
- Ben-Ari, M. 2012. *Mathematical Logic for Computer Science*. Springer, 3rd edition.
- Bonassi, L.; Percassi, F.; and Scala, E. 2025. Towards Practical Classical Planning Compilations of Numeric Planning. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence AAAI-2025*, 26472–26480.
- Borgwardt, S.; Hoffmann, J.; Kovtunova, A.; Krötzsch, M.; Nebel, B.; and Steinmetz, M. 2022. Expressivity of Planning with Horn Description Logic Ontologies. In *36th AAAI Conference on Artificial Intelligence, AAAI-2022*, 5503–5511.
- Borgwardt, S.; Hoffmann, J.; Kovtunova, A.; and Steinmetz, M. 2021. Making DL-Lite Planning Practical. In *Proc. of the 18th Intern. Conf. on Principles of Knowledge Representation and Reasoning*, 641–645.
- Bryce, D.; and Kambhampati, S. 2007. Planning Graph Based Reachability Heuristics. *AI Mag.*, 28(1): 47–83.
- Cardellini, M.; and Giunchiglia, E. 2025. Pushing the Envelope in Numeric Pattern Planning. In *KR-25*, 762–771.
- Chandra, A.; and Merlin, P. 1977. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *9th Annual ACM Symp. on Theory of Computing (STOC)*, 77–90.
- Chen, D. Z.; and Thiébaux, S. 2024. Novelty Heuristics, Multi-Queue Search, and Portfolios for Numeric Planning. In *Proceedings of the 17th International Symposium on Combinatorial Search (SoCS 2024)*, 203–207.
- Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2013. A Hybrid LP-RPG Heuristic for Modelling Numeric Resource Flows in Planning. *J. Artif. Intell. Res.*, 46: 343–412.
- Cook, S. A.; and Pitassi, T. 2022. Herbrand Theorem, Equality, and Compactness (Course Notes). <https://www.cs.columbia.edu/~toni/Courses/Logic2022/Notes/page39.pdf>.
- Corrêa, A. B.; Giacomo, G. D.; Helmert, M.; and Rubin, S. 2024. Planning with Object Creation. In *34th International Conference on Automated Planning and Scheduling, ICAPS 2024*, 104–113.
- De Giacomo, G.; Lespérance, Y.; and Levesque, H. J. 2011. Efficient Reasoning in Proper Knowledge Bases with Unknown Individuals. In *22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 827–832.
- De Giacomo, G.; Lespérance, Y.; and Patrizi, F. 2016. Bounded situation calculus action theories. *Artif. Intell.*, 237: 172–203.
- Enderton, H. 2001. *A Mathematical Introduction to Logic*. 2nd edit.: Harcourt Press.

- Eyerich, P.; Nebel, B.; Lakemeyer, G.; and Claßen, J. 2006. Golog and PDDL: What is the Relative Expressivness. In *Proceedings of the 2006 International Symposium on Practical Cognitive Agents and Robots*. ACM. ISBN 978-1-74052-130-7.
- Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4): 189–208.
- Finzi, A.; Pirri, F.; and Reiter, R. 2000. Open World Planning in the Situation Calculus. In *Proceedings of the 7th AAAI*, 754–760.
- Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Intell. Res.*, 20: 61–124.
- Francès, G.; and Geffner, H. 2016.  $\exists$ -STRIPS: Existential Quantification in Planning and Constraint Satisfaction. In Kambhampati, S., ed., *25th International Joint Conference on Artificial Intelligence, IJCAI-2016*, 3082–3088.
- Fuentetaja, R.; and de la Rosa, T. 2016. Compiling Irrelevant Objects to Counters. Special Case of Creation Planning. *AI Commun.*, 29(3): 435–467.
- Funkquist, M.; Ståhlberg, S.; and Geffner, H. 2024. Learning to Ground Existentially Quantified Goals. In *21st Intern. Conf. on Principles of Knowledge Represent. and Reasoning*, 856 – 866.
- Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artif. Intell.*, 173(5-6): 619–668.
- Gigante, N.; and Scala, E. 2023. On the Compilability of Bounded Numeric Planning. In *32nd International Joint Conference on Artificial Intelligence IJCAI-2023*, 5341–5349.
- Grastien, A.; and Scala, E. 2020. CPCES: A planning framework to solve conformant planning problems through a counterexample guided refinement. *Artif. Intell.*, 284: 103271.
- Haslum, P.; Lipovetzky, N.; Magazzeni, D.; and Muise, C. 2019. *An Introduction to the Planning Domain Definition Language*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.
- Helmert, M. 2002. Decidability and Undecidability Results for Planning with Numerical State Variables. In *Sixth International Conference on Artificial Intelligence Planning Systems AIPS-2002*, 44–53.
- Helmert, M. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.*, 26: 191–246.
- Helmert et. al. 2022. Fast Downward at Github. <https://github.com/aibasel/downward>. Accessed: 2022-11-17.
- Hoffmann, J.; and Brafman, R. 2006. Conformant Planning via Heuristic Forward Search: A New Approach. *Artificial Intelligence*, 170(6–7): 507–541.
- Hoffmann, J.; and Nebel, B. 2001. The FF System: Fast Plan Generation Through Heuristic Search. *J. Artif. Intell. Res.*, 14: 253–302.
- Hu, Y.; and Levesque, H. 2010. A Correctness Result for Reasoning about One-Dimensional Planning Problems. In *12th Intern. Conf on Principles of Knowledge Representation and Reasoning*.
- Illanes, L.; and McIlraith, S. A. 2017. Numeric Planning via Abstraction and Policy Guided Search. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence IJCAI*, 4338–4345.
- Kuroiwa, R.; Shleyfman, A.; Piacentini, C.; Castro, M. P.; and Beck, J. C. 2022. The LM-Cut Heuristic Family for Optimal Numeric Planning with Simple Conditions. *J. Artif. Intell. Res.*, 75: 1477–1548.
- Levesque, H.; Pirri, F.; and Reiter, R. 1998. Foundations for the Situation Calculus. *Linköping Electronic Articles in Computer and Information Science*. Available at: <http://www.ep.liu.se/ea/cis/1998/018/>, vol. 3, N 18.
- Levesque, H. J. 1998. A Completeness Result for Reasoning with Incomplete First-Order KBs. In *KR-1998*, 14–23.
- Lin, F. 2008. Situation Calculus. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, 649–669. Elsevier.
- Lin, F.; and Reiter, R. 1997. How to Progress a Database. *Artificial Intelligence*, 92: 131–167.
- Liu, Y. 2005. *Tractable Reasoning in Incomplete First-Order Knowledge Bases*. Ph.D. thesis, Department of Computer Science, University of Toronto.
- Liu, Y.; and Lakemeyer, G. 2009. On First-Order Definability and Computability of Progression for Local-Effect Actions and Beyond. In *21st IJCAI-2009*, 860–866.
- Liu, Y.; and Levesque, H. J. 2005. Tractable Reasoning with Incomplete First-Order Knowledge in Dynamic Systems with Context-Dependent Actions. In *19th International Joint Conference on Artificial Intelligence, IJCAI-2005*, 522–527.
- McCarthy, J. 1963. Situations, Actions and Causal Laws. Technical Report Memo 2, Stanford University AI Laboratory, Stanford, CA. Reprinted in Marvin Minsky, editor, *Semantic Information Processing*. MIT Press, 1968.
- McCarthy, J.; and Hayes, P. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Meltzer, B.; and Michie, D., eds., *Machine Intelligence*, volume 4, 463–502. Edinburgh Univ. Press.
- McDermott, D. V. 1996. A Heuristic Estimator for Means-Ends Analysis in Planning. In *Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-1996)*, 142–149.
- McDermott, D. V. 1999. Using Regression-Match Graphs to Control Search in Planning. *Artif. Intell.*, 109(1-2): 111–159.
- McIlraith, S. A. 2000. Integrating Actions and State Constraints: A Closed-form Solution to the Ramification Problem (sometimes). *Artif. Intell.*, 116(1-2): 87–121.
- Nebel, B. 2000. On the Compilability and Expressive Power of Propositional Planning Formalisms. *J. Artif. Intell. Res.*, 12: 271–315.

- Palacios, H.; and Geffner, H. 2009. Compiling Uncertainty Away in Conformant Planning Problems with Bounded Width. *J. Artif. Intell. Res.*, 35: 623–675.
- Pednault, E. P. D. 1994. ADL and the State-Transition Model of Action. *J. of Logic and Comput.*, 4(5): 467–512.
- Petrack, R. 2006. *A Knowledge-Level Approach for Effective Acting, Sensing, and Planning*. Ph.D. thesis, Department of Computer Science, University of Toronto.
- Petrack, R. P. A.; and Bacchus, F. 2002. A Knowledge-Based Approach to Planning with Incomplete Information and Sensing. In *Proceedings of the 6th Intern. Conf. on Artificial Intelligence Planning Systems (ICAPS)*, 212–222.
- Petrack, R. P. A.; and Bacchus, F. 2004. 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)*, 2–11.
- Piacentini, C.; Castro, M. P.; Ciré, A. A.; and Beck, J. C. 2018. Linear and Integer Programming-Based Heuristics for Cost-Optimal Numeric Planning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, 6254–6261.
- Pirri, F.; and Reiter, R. 1999. Some contributions to the metatheory of the situation calculus. *Journal of the ACM (JACM)*, 46(3): 325–361.
- Plaza, J. A. 1990. *Fully Declarative Programming With Logic - Mathematical Foundations*. Ph.D. thesis, City University of New York.
- Plaza, J. A. 1992. Operators on Lattices of omega-Herbrand Interpretations. In Nerode, A.; and Taitslin, M. A., eds., *Logical Foundations of Computer Science Tver'92, 2nd International Symposium*, volume 620 of *Lecture Notes in Computer Science*, 358–369. Springer.
- Reiter, R. 1980. Equality and Domain Closure in First-Order Databases. *J. ACM*, 27(2): 235–249.
- Reiter, R. 1991. 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*, 359–380. San Diego: Academic Press.
- Reiter, R. 2001. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT, <http://cognet.mit.edu/book/knowledge-action>.
- Richter, S.; and Helmert, M. 2009. Preferred Operators and Deferred Evaluation in Satisficing Planning. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009*, 273–280.
- Röger, G.; Helmert, M.; and Nebel, B. 2008. On the Relative Expressiveness of ADL and Golog: The Last Piece in the Puzzle. In *11th Intern. Confer., KR-2008*, 544–550.
- Röger, G. 2014. *Planning Techniques and the Action Language Golog, Dissertation*. Ph.D. thesis, Technische Fakultät Albert-Ludwigs-Universität, Freiburg, Germany.
- Savelli, F. 2006. Existential assertions and quantum levels on the tree of the situation calculus. *Artif. Intell.*, 170(6-7): 643–652.
- Scala, E.; and Bonassi, L. 2025. On Using Lazy Greedy Best-First Search with Subgoal Relaxation in Numeric Planning Problems. In *Proc. of the 35th Intern. Conf. on Autom. Planning and Sched. (ICAPS-2025)*, 245–249.
- Scala, E.; Haslum, P.; Thiébaux, S.; and Ramírez, M. 2020a. Subgoal Relaxation Techniques for Satisficing and Optimal Numeric Planning. *J. Artif. Intell. Res.*, 68: 691–752.
- Scala, E.; Saetti, A.; Serina, I.; and Gerevini, A. E. 2020b. Search-Guidance Mechanisms for Numeric Planning Through Subgoal Relaxation. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling ICAPS*, 226–234.
- Schöning, U. 2008. *Logic for Computer Scientists*. Springer/Birkhäuser.
- Shoenfield, J. 2018. *Mathematical Logic*. A.K.Peters/ CRC Press.
- Soutchanski, M. 2000. An On-line Decision-Theoretic Golog Interpreter. In Lakemeyer, G., ed., *The Second International Cognitive Robotics Workshop held in conjunction with 14th European Conference on Artificial Intelligence ECAI-2000*, 117–124. Berlin, Germany: available at <http://www.cs.torontomu.ca/mes/publications/cr.ps>.
- Soutchanski, M. 2001. A correspondence between two different solutions to the projection task with sensing. In *The 5th Symposium on Logical Formalizations of Commonsense Reasoning*, 235–242. Courant Institute of Mathematical Sciences, New York University, New York, USA, May 20–22. <http://www.cs.torontomu.ca/mes/publications/cs2001.pdf>
- Soutchanski, M. 2025. The Countdown Planning Benchmark. PROLOG code of randomly generated instances available at <https://www.cs.torontomu.ca/mes/publications/Countdown.zip>
- Soutchanski, M.; and Liu, Y. 2025. 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*. <https://arxiv.org/abs/2508.02697>
- Soutchanski, M.; and Young, R. 2023. 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*. [https://ceur-ws.org/Vol-3585/paper10\\_RCRA6.pdf](https://ceur-ws.org/Vol-3585/paper10_RCRA6.pdf)
- Vassos, S.; and Patrizi, F. 2013. A Classification of First-Order Progressable Action Theories in Situation Calculus. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013*, 1132–1138.
- Vassos, S.; Sardina, S.; and Levesque, H. 2009. Progressing Basic Action Theories with Non-Local Effect Actions. In *International Symposium on Commonsense Reasoning*.
- Waldinger, R. 1977. Achieving Several Goals Simultaneously. In *Machine Intelligence*, volume 8, 94–136. Edinburgh, Scotland: Ellis Horwood.
- Weyhrauch, R. W. 1980. Prolegomena to a Theory of Mechanized Formal Reasoning. *Artif. Intell.*, 13(1-2): 133–170.