Counting and Reasoning with Plans

David Speck[1], **Markus Hecher**[2,3], **Daniel Gnad**[4], **Johannes K. Fichte**[4], **Augusto B. Corrêa**[1,5]
[1]University of Basel, Switzerland
[2]Univ. Artois, CNRS, UMR 8188, Centre de Recherche en Informatique de Lens (CRIL), F-62300 Lens, France
[3]CSAIL, Massachusetts Institute of Technology, United States
[4]Linköping University, Sweden
[5]University of Oxford, United Kingdom
davidjakob.speck@unibas.ch, hecher@mit.edu, {daniel.gnad,johannes.fichte}@liu.se, augusto.blaascorrea@chch.ox.ac.uk

Abstract

Classical planning asks for a sequence of operators reaching a given goal. While the most common case is to compute a plan, many scenarios require more than that. However, quantitative reasoning on the plan space remains mostly unexplored. A fundamental problem is to *count plans*, which relates to the conditional probability on the plan space. Indeed, qualitative and quantitative approaches are well-established in various other areas of automated reasoning.

We present the first study to quantitative and qualitative reasoning on the plan space. In particular, we focus on polynomially bounded plans. On the theoretical side, we study its complexity, which gives rise to rich reasoning modes. Since counting is hard in general, we introduce the easier notion of facets, which enables understanding the significance of operators. On the practical side, we implement quantitative reasoning for planning. Thereby, we transform a planning task into a propositional formula and use knowledge compilation to count different plans. This framework scales well to large plan spaces, while enabling rich reasoning capabilities such as learning pruning functions and explainable planning.

Introduction

The overarching objective of classical planning is to find a plan, i.e., a sequence of operators, that transforms the current state into a goal state. While in some scenarios a single plan is sufficient, in others, it may not be clear which plan is preferable based on the description of the planning task. To address this, solvers like top-k or top-quality planners have been developed to enumerate the k shortest plans or all plans up to a certain length bound allowing for post hoc consideration of the plan space and selection (Katz et al. 2018; Katz and Sohrabi 2020; Speck, Mattmüller, and Nebel 2020; von Tschammer, Mattmüller, and Speck 2022; Chakraborti et al. 2024). Although this paradigm has been successfully applied in practical areas such as malware detection (Boddy et al. 2005) and scenario planning for risk management (Sohrabi et al. 2018), it remains an indirect method for reasoning about the plan space of a planning task.

Considering fundamental problems in computer science, such as the propositional satisfiability problem (SAT), answer set programming (ASP), and constraint satisfaction problems (CSP), more directed reasoning schemes exist that are anchored around counting. The most prominent and canonical counting problem is #SAT, also called *model counting*, which asks to compute the number of models of a formula. While #SAT is considered computationally harder than asking whether a single model exists (SAT), it also allows for automated reasoning about the solution space (Darwiche 2001; Darwiche and Marquis 2002). Recent competitions illustrate that, despite high computational complexity, state-of-the-art solvers are effective in practice (Fichte, Hecher, and Hamiti 2021). Due favorable reasoning power and vast applications, counting techniques have been extended to other fields (Aziz et al. 2015; Fichte et al. 2017; Hahn et al. 2022; Eiter, Hecher, and Kiesel 2024).

In this paper, we bridge the gap between model counting and classical planning by introducing a new framework for reasoning and analyzing plan space. To do so, we consider all plans for a given planning task with polynomially bounded length, consistent with the approach used in top-quality planning (Katz and Sohrabi 2020).

Contributions Our main contributions are as follows:

1. We introduce a *taxonomy of counting and reasoning problems for classical planning* with polynomially bounded plan lengths and establish the computational complexity of these problems.
2. We identify a class of reasoning problems on the plan space, called *facet reasoning*, that are as hard as polynomially bounded planning and thus can be solved more efficiently than counting problems.
3. We present a practical tool, `Planalyst`, that builds on existing planning and knowledge compilation techniques to answer plan-space reasoning queries and demonstrate its practical feasibility.

In more detail, on the theoretical side, we formally define a taxonomy of counting and reasoning problems for planning and analyze the computational complexity of these problems. Among other results, we show that the problem of probabilistic reasoning about the plan space such as determining how many plans contain a given operator is $\mathrm{C}^{\mathrm{P}}_{=}$-complete, which is considered computationally harder than counting the number of plans, known to be #P-complete (Speck, Mattmüller, and Nebel 2020). We also introduce the

notion of *facet reasoning* in the context of planning, which has origins in computational complexity (Papadimitriou and Yannakakis 1982) and is well studied in ASP (Alrabbaa, Rudolph, and Schweizer 2018; Fichte, Gaggl, and Rusovac 2022). We show that facet reasoning in planning is NP-complete, and thus probably much simpler than counting the number of plans. This theoretical result is significant because it allows more efficient answers to complex reasoning queries about the plan space, such as identifying which operators can complement a given partial plan and which provide more flexibility for further complementation.

On the practical side, we present a solution to the studied counting and reasoning problems by transforming a planning task into a propositional formula, where satisfying assignments correspond one-to-one to plans, followed by subsequent knowledge compilation into a d-DNNF (Darwiche and Marquis 2002). We implement this as a tool called `Planalyst`, which builds on existing tools from planning (Rintanen 2014) and knowledge compilation (Lagniez and Marquis 2017; Sundermann et al. 2024) and thus readily allows plan counting and automated reasoning in plan space. Empirically, we compare `Planalyst` to state-of-the-art top-quality planners on the computationally challenging problem of counting plans, and show that our tool performs favorably, especially when the plan space is large and reasoning over trillions of plans is critical. Finally, by constructing a d-DNNF, our approach not only supports plan counting, but can also answer reasoning questions such as conditional probability, faceted reasoning, and unbiased uniform plan sampling, all through efficient d-DNNF queries.

Related Work

Darwiche and Marquis (2002) detailed the theoretical capabilities and limitations of normal forms in knowledge compilation. Established propositional knowledge compilers are `c2d` (Darwiche 1999) and `d4` (Lagniez and Marquis 2017), new developments are extensions of `SharpSAT-TD` (Kiesel and Eiter 2023). Incremental and approximate counting has been considered for ASP (Kabir et al. 2022; Fichte et al. 2024). In SAT and ASP, advanced enumeration techniques have also been studied (Masina, Spallitta, and Sebastiani 2023; Spallitta, Sebastiani, and Biere 2024; Gebser, Kaufmann, and Schaub 2009; Alviano et al. 2023), which can be beneficial for counting if the number of solutions is sufficiently low or when (partial) solutions need to be materialized. Exact uniform sampling using knowledge compilation has also been implemented (Lai, Meel, and Yap 2021). Model counting has been applied to probabilistic planning in the past (Domshlak and Hoffmann 2007). In classical planning and grounding, Corrêa et al. (2023) argued that grounding is infeasible for some domains if the number of operators in a planning task is too high. Therefore, they manually employed model counting, but did not develop extended reasoning techniques or counting tools for planning. Fine-grained reasoning modes and facets have been studied for ASP (Alrabbaa, Rudolph, and Schweizer 2018; Fichte, Gaggl, and Rusovac 2022; Fichte, Hecher, and Nadeem 2022; Rusovac et al. 2024; Eiter et al. 2024) and significance notions based on facets (Böhl, Gaggl, and Rusovac 2023).

Preliminaries

We assume that the reader is familiar with basics of propositional logic (Kleine Büning and Lettmann 1999) and computational complexity (Papadimitriou 1994). Below, we follow standard definitions (Bylander 1994; Speck, Mattmüller, and Nebel 2020) to summarize basic notations for planning.

Basics For an integer i, we define $[i] \coloneqq \{0, 1, \ldots, i\}$. We abbreviate the *domain* of a function $f : \mathcal{D} \to \mathcal{R}$ by $\mathrm{dom}(f)$. By $f^{-1} : \mathcal{R} \to \mathcal{D}$ we denote the inverse function $f^{-1} \coloneqq \{f(d) \to d \mid d \in \mathrm{dom}(f)\}$ of function f, if it exists. Let $\sigma = \langle s_1, s_2, \ldots, s_\ell \rangle$ be a sequence, then we write $s \in \sigma$ if $s = s_i$ for some $1 \leq i \leq \ell$ and $\bigtriangledown(\sigma)$ the set of elements that occur in σ, i.e., $\bigtriangledown(\sigma) \coloneqq \{s \mid s \in \sigma\}$. For a propositional formula F, we abbreviate by $\mathrm{vars}(F)$ the variables that occur in F and by $\mathrm{Mod}(F)$ the set of all models of F and the number of models by $\#(F) \coloneqq |\mathrm{Mod}(F)|$.

Computational Complexity We follow standard terminology in computational complexity (Papadimitriou 1994) and the Polynomial Hierarchy (PH) (Stockmeyer and Meyer 1973; Stockmeyer 1976; Wrathall 1976). The complexity class D^P captures the (independent) combination of an NP and a coNP problem, i.e., $\mathsf{D}^\mathsf{P} \coloneqq \{L_1 \cap L_2 \mid L_1 \in \mathsf{NP}, L_2 \in \mathsf{coNP}\}$ (Papadimitriou and Yannakakis 1982). Class PP (Gill 1977) refers to those decision problems that can be characterized by a nondeterministic Turing machine, such that the positive instances are those where at least $1/2$ of the machine's paths are accepting. Counting class $\#\mathsf{P}$ captures counting problems that can be solved by counting the number of accepting paths of a nondeterministic Turing machine (Valiant 1979). Class $\mathsf{C}^\mathsf{P}_=$ (Fenner et al. 1999) refers to decision problems that can be characterized via nondeterministic Turing machines where positive instances are those with the same number of accepting and rejecting paths.

Classical Planning A *planning task* is a tuple $\Pi = \langle \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{A}$ is a finite set of propositional *state variables*. A *(partial) state* s is a total (partial) mapping $s : \mathcal{A} \to \{0, 1\}$. For a state s and a partial state p, we write $s \models p$ if s *satisfies* p, more formally, $p^{-1}(0) \subseteq s^{-1}(0)$ and $p^{-1}(1) \subseteq s^{-1}(1)$. $\mathcal{O}$ is a finite set of *operators*, where each operator is a tuple $o = \langle \mathrm{pre}_o, \mathrm{eff}_o \rangle$ of partial states, called *preconditions* and *effects*. An operator $o \in \mathcal{O}$ is *applicable* in a state s if $s \models \mathrm{pre}_o$. *Applying* operator o to state s, $s[\![o]\!]$ for short, yields state s', where $s'(a) \coloneqq \mathrm{eff}_o(a)$, if $a \in \mathrm{dom}(\mathrm{eff}_o)$ and $s'(a) \coloneqq s(a)$, otherwise. Finally, $\mathcal{I}$ is the *initial state* of Π and $\mathcal{G}$ a partial state called *goal condition*. A state s_* is a *goal state* if $s_* \models \mathcal{G}$. Let Π be a planning task. A *plan* $\pi = \langle o_0, \ldots, o_{n-1} \rangle$ is a sequence of applicable operators that *generates* a sequence of states $s_0, \ldots, s_n$, where $s_0 = \mathcal{I}$, s_n is a goal state, and $s_{i+1} = s_i[\![o_i]\!]$ for every $i \in [n-1]$. Furthermore, we let $\pi(i) \coloneqq o_i$ and denote by $|\pi|$ the *length* of a plan π. We denote the set of all plans by $\mathrm{Plans}(\Pi)$ and the set of all plans of length at most ℓ by $\mathrm{Plans}_\ell(\Pi)$ and call it occasionally *plan space* as done in the literature (Russell and Norvig 1995).

A plan π is *optimal* if there is no plan $\pi' \in \mathrm{Plans}(\Pi)$ where $|\pi'| < |\pi|$. The notion naturally extends to bounded-length plans. Deciding or counting plans is computationally

Name	Given	Task	Compl.	Ref.
POLY-BOUNDED-PLAN-EXIST	Π, ℓ	$\pi \in \mathrm{Plans}_\ell(\Pi)$	NP-c	[1]
POLY-BRAVE-PLAN-EXIST	Π, ℓ, o	$\exists \pi \in \mathrm{Plans}_\ell(\Pi) : o \in \pi$	NP-c	Lem. 6
POLY-CAUTIOUS-PLAN-EXIST	Π, ℓ, o	$\forall \pi \in \mathrm{Plans}_\ell(\Pi) : o \in \pi$	coNP-c	Lem. 6
POLY-BOUNDED-TOP-K-EXIST	Π, ℓ	$\lvert \mathrm{Plans}_\ell \rvert \geq k$	PP-h	[2]
#POLY-BOUNDED-PLAN	Π, ℓ	$\lvert \mathrm{Plans}_\ell \rvert$	#P-c	[2]
POLY-PROBABILISTIC-REASON	Π, ℓ, Q, p	$\mathbb{P}_\ell[\Pi, Q] = p$	$\mathrm{C}^{\mathrm{P}}_{=}$-c	Thm. 9
FACETREASON	Π, ℓ, o	$o \in \mathcal{F}_\ell(\Pi)$	NP-c	Thm. 10
ATLEAST-K-FACETS	Π, ℓ, k	$\lvert \mathcal{F}_\ell(\Pi) \rvert \geq k$	NP-c	Lem. 11
ATMOST-K-FACETS	Π, ℓ, k	$\lvert \mathcal{F}_\ell(\Pi) \rvert \leq k$	coNP-c	Cor. 12
EXACT-K-FACETS	Π, ℓ, k	$\lvert \mathcal{F}_\ell(\Pi) \rvert = k$	D^{P}-c	Thm. 13

Table 1: *Computational Complexity of Qualitative and Quantitative Reasoning Problems.* We let Π be a planning task, $\ell \in \mathbb{N}_0$ with $\ell \leq \mathit{poly}(\Pi)$, $o \in \mathcal{O}$, $k \in \mathbb{N}_o$, $0 \leq p \leq 1$, and Q a query. [1]: (Bylander 1994), [2]: (Speck, Mattmüller, and Nebel 2020).

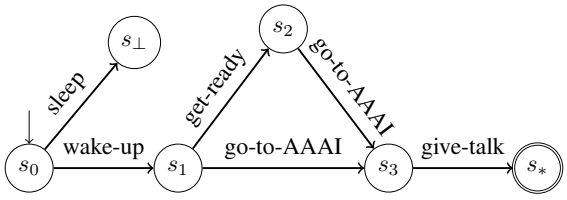

Figure 1: State space of our running example task Π_1. The initial state is denoted by s_0; the goal state is denoted by s_*.

hard. More precisely, the BOUNDED-PLAN-EXIST problem, which asks to decide whether there exists a plan of length at most ℓ, is PSPACE-complete (Bylander 1994). The #BOUNDED-PLAN problem, which asks to output the number of plans of length at most ℓ, remains PSPACE-complete (Speck, Mattmüller, and Nebel 2020). We say that a plan is *polynomially bounded* if we restrict the length to be polynomial in the instance size, i.e., the length ℓ of Π is bounded by $\ell \leq \|\Pi\|^c$ for some constant c, where $\|\Pi\|$ is the encoding size of Π. For a planning problem $\mathbb{P}$ with input ℓ that bounds the length of a plan, we abbreviate by POLY-$\mathbb{P}$ the problem $\mathbb{P}$ where ℓ is polynomially bounded. Then, the complexity drops. POLY-BOUNDED-PLAN-EXIST is NP-complete (Bylander 1994) and #POLY-BOUNDED-PLAN is #P-complete, and the decision problem POLY-BOUNDED-TOP-K-EXIST is PP-hard, which asks to decide, given in addition an integer k, whether there are at least k different plans of length up to ℓ (Speck, Mattmüller, and Nebel 2020).

Example 1 (Running Example)**.** *Consider a planning task Π_1 consisting of a scenario with a slightly chaotic researcher, who has to wake up and give a talk at AAAI. Depending on how late they are, they can go straight to the talk without any preparation. However, they could also spend time getting ready. Less pleasant to the audience, they could also continue sleeping and not give the talk at all. Figure 1 illustrates the state space. The initial state is s_0, and the single goal state is s_*. The labels in each edge identify the operator being applied. We can easily identify two plans:*

(i) wake-up; get-ready; go-to-AAAI; give-talk.
(ii) wake-up; go-to-AAAI; give-talk.

Plan (i) has length 4*, while Plan (ii) has length* 3*. Observe that action* sleep *does not appear in any plan.*

Landmarks A *fact landmark* is a state variable that occurs in every plan (Porteous, Sebastia, and Hoffmann 2001). An *operator landmark* is an operator that occurs in every plan (Richter, Helmert, and Westphal 2008; Karpas and Domshlak 2009). We can extend these notions to *bounded landmarks* where we assume bounded length ℓ.

Example 2. *Consider planning task Π_1 from Example 1. We observe that* wake-up*,* go-to-AAAI*, and* give-talk *are operator landmarks.*

Planning as Satisfiability (SAT) Let $\Pi = \langle \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ be a planning task and $\ell > 0$ an integer to bound the length of a potential plan. We can employ a standard technique to encode finding a plan into a propositional formula and ask for its satisfiability (SAT) (Kautz and Selman 1992; Rintanen 2012). In more detail, we can construct a formula $F^{\text{plan}}_{\leq \ell}[\Pi]$ whose models are in one-to-one correspondence with the ℓ-bounded plans of Π. For space reasons, we present only the core idea. The variables are as follows: $\mathrm{vars}(F^{\text{plan}}_{\leq \ell}) = \{a^i \mid a \in \mathcal{A}, i \in [\ell]\} \cup \{o^i \mid o \in \mathcal{O}, i \in [\ell]\}$. Variable a^i indicates the value of state variable a at the i-th step of the plan. Hence, if $M \in \mathrm{Mod}(F^{\text{plan}}_{\leq \ell}[\Pi])$ and $a^\ell \in M$, then state variable a has value 1 after applying operators $o^0, \ldots, o^{\ell-1}$ to the initial state. We assume *sequential encodings*, where the following constraints hold.

1. a set of clauses encoding the value of each state variable at the initial state;
2. a set of clauses encoding the value of each state variable in the goal condition;
3. a set of clauses guaranteeing that no two operators are chosen at the same step; and
4. a set of clauses guaranteeing the consistency of state variables after an operator is applied. If o^i is true and the effect of operator o makes a true, then a^{i+1} must be true.

Since plans might be shorter than ℓ, we move "unused" steps to the end using the formula $\bigwedge_{i \in [\ell]} (\bigwedge_{o \in \mathcal{O}} \neg o^i \rightarrow$

$\bigwedge_{o\in\mathcal{O}} \neg o^{i+1}$), which encodes that if no operator was assigned at step i, then no operator can be assigned at step $i+1$. Thereby, we obtain a one-to-one mapping between models of $F^{\text{plan}}_{\leq\ell}[\Pi]$ and l-bounded plans for the task.

From Qualitative to Quantitative Reasoning

Classical planning aims at finding one plan or enumerating certain plans. But what if we want plans that contain a certain operator, or to count the number of possible plans given certain assumptions, or if we want to identify the frequency of an operator among all possible plans? Currently, there is no unified reasoning tool to deal with these types of questions. We introduce more detailed qualitative and quantitative reasoning modes for planning and analyze its complexity. We start with two extreme reasoning modes that consider whether an operator is part of some or all plans.

Definition 3. *Let $\Pi = \langle \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{G}\rangle$ be a planning task, $o \in \mathcal{O}$ an operator, and ℓ an integer. We define the*

- brave *operator by $\mathcal{BO}_\ell(\Pi) \coloneqq \bigcup_{\pi\in\text{Plans}_\ell(\Pi)} \nabla(\pi)$ and*
- cautious *operator by $\mathcal{CO}_\ell(\Pi) \coloneqq \bigcap_{\pi\in\text{Plans}_\ell(\Pi)} \nabla(\pi)$.*

The problem POLY-BRAVE-PLAN-EXIST *asks to decide whether $o \in \mathcal{BO}_\ell(\Pi)$. The problem* POLY-CAUTIOUS-PLAN-EXIST *asks to decide whether $o \in \mathcal{CO}_\ell(\Pi)$.*

Note that we use $\nabla(\cdot)$ to convert sequences into sets, as we aim only for an operator occurring at any time-point.

Remark 4. *Our definition of cautious operators is similar to* operator landmarks *(Zhu and Givan 2003), but for plans with up to a given bounded length.*

Example 5. *Consider task Π_1 from Example 1 and Plans (i) and (ii). Furthermore, let $\ell = 4$. Then, the brave and cautious operators of our task are the following:*

$$\mathcal{BO}_\ell(\Pi_1) = \{\text{wake-up, get-ready, go-to-AAAI, give-talk}\},$$
$$\mathcal{CO}_\ell(\Pi_1) = \{\text{wake-up, go-to-AAAI, give-talk}\}.$$

Operator get-ready *is brave but not cautious, as it appears in Plan (i) but not in Plan (ii). Operator* sleep *is neither brave nor cautious, as it does not appear in any plan.*

Lemma 6 ($\star$[1])**.** *The problem* POLY-BRAVE-PLAN-EXIST *is* NP*-complete and the problem* POLY-CAUTIOUS-PLAN-EXIST *is* coNP*-complete.*

To find brave operators in practice, we can employ a standard SAT (Audemard and Simon 2018) or ASP solver (Gebser et al. 2011, 2014; Alviano et al. 2015). For cautious operators, we can employ a dedicated backbone solver (Biere, Froleyks, and Wang 2023) or again ASP solvers.

Probability Reasoning

Both problems POLY-BRAVE-PLAN-EXIST and POLY-CAUTIOUS-PLAN-EXIST give rise to extreme reasoning modes on plans. Cautious reasoning is quite strict and so unlikely to hold in general. Brave reasoning is too general and permissive, and thus quite weak in practice. Figure 2 illustrates the two reasoning modes and a more fine-grained

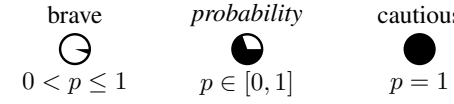

Figure 2: Quantitative reasoning is a fine-grained reasoning mode between brave and cautious reasoning. It asks whether a literal matches $\geq p\cdot 100\%$ of the plans for planning task Π.

mode, which we introduce below. This new mode asks whether the conditional probability of an operator is above a given threshold. It generalizes the known POLY-BOUNDED-TOP-K-EXIST planning problem, which only asks whether at least k plans exists. The crucial ingredient is counting the number of possible plans and relating them to the number of possible plans which contain a given operator. More formally: Let $\Pi = \langle \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{G}\rangle$ be a planning task, o be an operator. We abbreviate the set of all plans of Π containing o by $\text{Plans}_\ell(\Pi, o) \coloneqq \{\pi \mid \pi \in \text{Plans}_\ell(\Pi), o \in \pi\}$. Then, we define the *conditional probability* of o in plans of Π by $\mathbb{P}_\ell[\Pi, o] \coloneqq \frac{|\text{Plans}_\ell(\Pi,o)|}{\max(1,|\text{Plans}_\ell(\Pi)|)}$. Note that the usage of $\max$ prevents division by zero in case of no possible plan. Analogously, we can talk about operator o in position i by replacing $o \in \pi$ with $o = \pi(i)$. With the help of conditional probability, we can define a fine-grained reasoning mode.

To be more flexible, we define *a query* Q as a propositional formula in conjunctive normal form (CNF) and assume its meaning as expected. We let Q contain variables corresponding to the set $\mathcal{A}$ of state variables, the set $\mathcal{O}$ of operators, as well as of states and operators in position i (similar to $F^{\text{plan}}_{\leq\ell}$). Let $\pi \in \text{Plans}_\ell(\Pi)$ be a plan with $\pi = \langle o_0, \ldots, o_{n-1}\rangle$ that generates sequence $s_0, \ldots, s_n$. π *satisfies* a variable $v \in \mathcal{A}$ if there is some $i \in [\ell]$ such that $s_i(v) = 1$; *satisfies* an operator $o \in \mathcal{O}$ if there is some $i \in [\ell]$ such that $\pi(i) = o$, analogously for fixed time-points i. Then, π satisfies $\neg v$ if π does not satisfy v. A plan π satisfies a clause C in Q, if π satisfies one of its literals; π satisfies Q, denoted $\pi \models Q$, if it satisfies every clause in Q. We define $\text{Plans}_\ell(\Pi, Q) \coloneqq \{\pi \mid \pi \in \text{Plans}_\ell(\Pi), \pi \models Q\}$.

Definition 7 (Probability Reasoning)**.** *Let $\Pi = \langle \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{G}\rangle$ be a planning task, $\ell > 0$ be an integer, Q be a query, and $0\leq p\leq 1$ with $p \in \mathbb{Q}$. Then,* probability reasoning *on Q asks if $\mathbb{P}_\ell[\Pi, Q] = p$, where $\mathbb{P}_\ell[\Pi, Q] \coloneqq \frac{|\text{Plans}_\ell(\Pi,Q)|}{\max(1,|\text{Plans}_\ell(\Pi)|)}$.*

Example 8 (Probability Reasoning)**.** *Again, consider planning task Π_1 from Example 1 and let $\ell = 4$. Take the following probability reasoning queries: (i) $\mathbb{P}_\ell[\Pi_1, \text{wake-up}] = 1$, (ii) $\mathbb{P}_\ell[\Pi_1, \text{get-ready}] = 0.5$, and (iii) $\mathbb{P}_\ell[\Pi_1, \text{sleep}] = 0$. Reasoning (i) illustrates that the researcher must always use operator* wake-up *to reach a goal; (ii) indicates that* get-ready *occurs in half of the plans; (iii) allows us to conclude that no plan uses operator* sleep*. More complex queries might ask for the probability of a plan containing both* wake-up *and* sleep*, or at least one of them:*

$$\mathbb{P}_\ell[\Pi_1, \text{wake-up} \wedge \text{sleep}] = 0,$$
$$\mathbb{P}_\ell[\Pi_1, \text{wake-up} \vee \text{sleep}] = 1.$$

Probability reasoning can be achieved by counting twice, which is computationally hard. In more detail, we obtain:

[1]We prove statements marked by "$\star$" in the extended version.

Theorem 9 ($\star$). *The problem* POLY-PROBABILISTIC-REASON *is* $\mathrm{C}^{\mathsf{P}}_{=}$*-complete.*

Faceted Reasoning

Above, we introduced three different reasoning modes, namely brave, probability, cautious reasoning. Unfortunately the most precise reasoning mode —the probability mode— is the computational most expensive one and requires to count plans. Therefore, we turn our attention to reasoning that is less hard than probabilistic reasoning and allows us still to filter plans and quantify uncertainty among plans. We call this reasoning *faceted reasoning* following terminology from combinatorics (Papadimitriou and Yannakakis 1982) and ASP (Alrabbaa, Rudolph, and Schweizer 2018). At the heart of these tasks is a combination of brave and cautious reasoning. These are particularly useful if we want to develop plans gradually/incrementally to see at a given time point, which operators are still possible or have the biggest effect. We focus on operators that belong to some (brave) but not to all plans (cautious).

More formally, for a planning task Π and an integer ℓ, we let $\mathcal{F}^{+}_{\ell}(\Pi) \coloneqq \mathcal{BO}_{\ell}(\Pi) \setminus \mathcal{CO}_{\ell}(\Pi)$ and call the elements of $\mathcal{F}^{+}_{\ell}(\Pi)$ *inclusive facets*. In addition, we distinguish *excluding facets* $\mathcal{F}^{-}_{\ell}(\Pi)$, which indicate that operators are not part of a plan. More formally, we let $\mathcal{F}^{-}_{\ell} \coloneqq \{\neg o \mid o \in \mathcal{F}^{+}(\Pi)\}$ and define the set $\mathcal{F}_{\ell}(\Pi)$ of all facets by $\mathcal{F}_{\ell}(\Pi) \coloneqq \mathcal{F}^{+}_{\ell}(\Pi) \cup \mathcal{F}^{-}_{\ell}(\Pi)$. Interestingly, a facet $p \in \{o, \neg o\}$ is directly related to *uncertainty*, since the operator o can either be included in or be excluded from a plan. When we *enforce* that a *facet* $p \in \{o, \neg o\}$ is present in a plan, which we abbreviate by $\Pi[p]$, we immediately reduce uncertainty on operators among the plans. Based on this understanding, we define the notion of *significance* for a planning task $\Pi = \langle \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ and an operator $o \in \mathcal{O}$:

$$\mathbb{S}_{\ell}(\Pi, o) \coloneqq \frac{|\mathcal{F}_{\ell}(\Pi)| - |\mathcal{F}_{\ell}(\Pi[o])|}{|\mathcal{F}_{\ell}(\Pi)|}.$$

Note that the notion of significance is particularly interesting when we already have a prefix $\omega_k = \langle o_0, \ldots, o_k \rangle$ and are interested in plans that complete the prefix. Here, facets can assist in understanding which operator is the most significant for the next step or some step in the future. Furthermore, we can include state variables into significance notations without effect on the complexity. We omit these cases from the presentation due to space constraints and readability of our introduced notion.

Computational Aspects of Facets

Next, we study the computational complexity for problems related to facets. We limit ourselves to including facets, assume the case where an operator occurs in some step, and we omit prefixes in the following. These restrictions have only a negligible effect on the complexity. We start with a natural reasoning problem: The FACETREASON problem asks, given a planning task Π and an operator $o \in \mathcal{O}$, to decide whether $o \in \mathcal{F}(\Pi)$. We start with a lower and upper bound on the FACETREASON problem.

Theorem 10 ($\star$). *Let* Π *be a planning task and* $o \in \mathcal{O}$*. The problem* FACETREASON *is* NP*-complete.*

Next, we look into counting facets and first observe that the number of facets is bound by $0 \leq |\mathcal{F}(\Pi)| \leq |\mathcal{O}|$ for a planning task Π. Therefore, we consider a parameterized version by taking a bound k on the number of facets as input. Then, the problem EXACT-K-FACETS asks, given a planning task Π and an integer k, to decide whether $|\mathcal{F}(\Pi)| = k$. Before, we look into upper and lower bounds by the problems ATLEAST-K-FACETS and ATMOST-K-FACETS, which ask whether $|\mathcal{F}(\Pi)| \geq k$ and $|\mathcal{F}(\Pi)| \leq k$, respectively.

Lemma 11 ($\star$). *Let* Π *be a planning task, and* $\ell \in \mathbb{N}$*,* $k \in \mathbb{N}_0$ *be integers.* ATLEAST-K-FACETS *is* NP*-complete.*

Corollary 12 ($\star$). *Let* Π *be a planning task,* $\ell \in \mathbb{N}$*,* $k \in \mathbb{N}_0$*. Then, the problem* ATMOST-K-FACETS *is* coNP*-complete.*

Both results together yield D^{P}-completeness.

Theorem 13 ($\star$). *Let* Π *be a program, and* $\ell \in \mathbb{N}$*,* $k \in \mathbb{N}_0$ *be integers. The problem* EXACT-K-FACETS *is* D^{P}*-complete.*

Discussion: Applications of Plan Reasoning

Our new reasoning modes offer a rich framework to query the solution space of planning tasks. In Remark 4, we discussed the connection between landmarks and cautious reasoning. Similarly, with brave and cautious reasoning it is easy to answer questions such as "does operator o appear on any plan?", or "does partial state p occur on any trajectory?"

The expressiveness of the queries goes way beyond and can be leveraged in many existing planning techniques. For example, determining the set of operators that are always or never part of a plan is important for learning pruning functions (Gnad et al. 2019). We can generalize these more global queries to reason about operators being only (never) applied in states that satisfy certain conditions, which is essential for learning policies (Krajnanský et al. 2014; Bonet and Geffner 2015). Furthermore, brave and cautious reasoning can be helpful for model debugging, offering a convenient tool to find out if an operator expected to occur in a plan does in fact never appear (Lin, Grastien, and Bercher 2023; Gragera et al. 2023). In over-subscription planning (Smith 2004), we can determine the achievability of soft goals or compute the achievable maximum set of soft goals by answering multiple queries. This can be utilized in explainable planning, providing reasons for the absence of solutions that achieve the desired set of soft goals (Eifler et al. 2020; Krarup et al. 2021). We can even generalize the notion of soft goals to desired state atoms that are achieved *along* a plan, but which might no longer hold in the goal.

With faceted reasoning, we are able to answer plan-space queries without actually counting the number of solutions. This reduces the complexity of answering queries to NP-completeness, making reasoning much more practically usable. What makes facet reasoning particularly interesting is that it allows to efficiently answer conditional queries, such as "if I want operator o to occur at step k, how much choice is left for the remaining operators?". Similar to previous

work in ASP, facet reasoning allows for an interactive querying mode in which users can gain insights about the particular solution space of a planning task (Fichte, Gaggl, and Rusovac 2022). For tasks with a large set of plans that cannot possibly be navigated manually, facets offer the possibility to systematically navigate the solution space, narrowing down the set of plans by committing to desired operators. The `Planalyst` tool, which we describe in more detail in the next section, enables this form of interactive exploration in the context of classical planning.

Empirical Evaluation

We implemented our reasoning framework for classical planning as a tool called `Planalyst`. Therefore, we transform planning tasks into SAT formulas based on the `Madagascar` planner (Rintanen 2011, 2014). To efficiently carry out counting, we use `d4` (Lagniez and Marquis 2017; Audemard, Lagniez, and Miceli 2022), which compiles (potentially large) formulas into a specialized normal form called *d-DNNF* (Darwiche and Marquis 2002), enabling fast reasoning. Finally, we reason over the plan space via counting queries using the `ddnnife` reasoner (Sundermann et al. 2024), which works in poly-time on d-DNNFs.

Experimental Setup

We focus on solving #BOUNDED-PLAN, i.e., counting the number of plans, which is the computationally hardest problem studied above. This allows us to address all reasoning questions discussed, including computing conditional probabilities. For each task of the benchmark set, we defined an upper bound by collecting known bounds from `planning.domains` (Muise 2016) and running winning planners from the most recent International Planning Competitions (IPC) (Taitler et al. 2024). In the experiments, we count plans of length up to a multiplicative factor $c \in \{1.0, 1.1, 1.2, 1.3, 1.4, 1.5\}$ of the collected upper bounds. We consider two different configurations for our approach: `Count`, which only counts the number of plans, and `Enum`, which additionally enumerates all plans, resulting in a novel top-quality planner for classical planning with unit operator costs. For comparison, we have chosen two top-quality planners, `K*` (Katz et al. 2018) and `SymK` (Speck, Mattmüller, and Nebel 2020), both of which can be readily used to count the number of plans as they enumerate them, and both of which are considered to scale well to large numbers of plans. We ran both baseline planners in their recommended configurations[2]: `K*`, which implements orbit-space search (Katz and Lee 2023) with the landmark-cut heuristic (Helmert and Domshlak 2009), and `SymK`, which implements a variant of bidirectional symbolic search (Torralba et al. 2017). For enumeration approaches (`K*`, `SymK`, `Enum`), we let these solvers enumerate the plans only internally to avoid writing billions (or more) of plans to the disk. All experiments ran on Intel Xeon Silver 4114 processors running at 2.2 GHz. We used a time limit of 30 minutes and a memory limit of 6 GiB per task. Our benchmarks include all optimal planning domains from IPCs 1998-2023 with unit operator costs and without conditional effects or axioms. Source code, benchmarks, and data are available online (Speck et al. 2024).

[2]We disabled a default optimization that removes operators causally irrelevant to the goal, as it prunes valid plans.

Length Bound	Coverage				#Plans		
	`K*`	`SymK`	**`Enum`**	**`Count`**	Max	Mean	Median
$\times$ 1.0	**351**	309	253	335	$>10^{15}$	$>10^{13}$	$>10^{2}$
$\times$ 1.1	289	231	182	**300**	$>10^{15}$	$>10^{13}$	$>10^{4}$
$\times$ 1.2	212	173	130	**251**	$>10^{15}$	$>10^{13}$	$>10^{5}$
$\times$ 1.3	177	135	101	**210**	$>10^{18}$	$>10^{15}$	$>10^{5}$
$\times$ 1.4	142	112	77	**189**	$>10^{21}$	$>10^{18}$	$>10^{6}$
$\times$ 1.5	112	91	61	**170**	$>10^{21}$	$>10^{18}$	$>10^{6}$

Table 2: (Left): Coverage, i.e., the number of tasks where the number of plans within a multiplicative factor of a length bound was found by `K*`, `SymK`, and our SAT-based approaches, `Count` and `Enum`. `Count` only counts plans, while `Enum` additionally enumerates them. (Right): Statistics on the number of plans in the benchmark set, considering the length bound determined by the four solvers.

Overall Performance

Table 2 (left) compares the coverage, i.e., the number of tasks for which different approaches can determine the number of plans, for different multiplicative length bounds. `K*` has the best coverage for a length bound of 1.0. Our enumeration approach, `Enum`, ranks overall last, although being able to solve a notable number of tasks by first creating a d-DNNF, followed by a subsequent enumeration query for all models, and finally mapping them to actual plans. For the 1.0 bound, our counting approach `Count` performs worse than `K*`, but has better coverage than the `SymK` planner. When considering higher length bounds, the counting approach, `Count`, has the highest coverage. The gap between `Count` and the other approaches gets larger as the length bound increases. This can be explained by the increasing number of plans, see Table 2 (right), where enumeration becomes less feasible due to the large plan space. This highlights the usefulness of our approach for sampling or reasoning in tasks with huge plan spaces. For example, in scenarios where end-users want to understand the plan space, enumerating over a sextillion (10^{21}) different plans is infeasible, but counting them (and using the related reasoning) is possible. Moreover, a decent performance with larger bounds gives us more flexibility for problems where a good bound is not easily available but an over-approximation is, e.g., using a non-admissible heuristic to come up with a bound.

Domain-Wise Performance

Table 3 shows a domain-wise comparison of the different approaches for the two extreme bounds in our experiments, 1.0 and 1.5. For both bounds, the performance differs a lot depending on the domain. Our SAT-based approach performs particularly well in the blocksworld and psr-small domains in both cases. In blocksworld, the largest task that we could still solve had $1.5 \cdot 10^9$ plans, while in psr-small the largest

Domains	Bound: ×1				Bound: ×1.5			
	K*	SymK	Enum	Count	K*	SymK	Enum	Count
airport (49)	7	7	7	**11**	7	7	6	**11**
barman (14)	**3**	0	0	0	**0**	**0**	**0**	**0**
blocks (35)	28	31	29	**33**	9	8	7	**15**
childsnack (20)	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
depot (22)	**4**	2	2	3	0	0	0	**1**
driverlog (20)	**10**	8	6	8	1	1	1	**2**
freecell (80)	**15**	13	5	5	**0**	**0**	**0**	**0**
grid (5)	**2**	**2**	1	1	**1**	0	0	**1**
gripper (20)	**3**	2	2	**3**	1	1	0	**2**
hiking (20)	4	3	1	**7**	0	0	0	**1**
logistics (63)	9	6	4	**13**	1	1	0	**3**
miconic (150)	**39**	35	31	**39**	14	13	10	**24**
movie (30)	2	2	0	**30**	0	0	0	**30**
mprime (35)	22	20	22	**23**	**12**	7	2	9
mystery (19)	**16**	14	14	15	**11**	8	7	9
nomystery (20)	**14**	13	8	8	**5**	2	1	4
organic (16)	**7**	**7**	0	0	**7**	**7**	0	0
parking (40)	**3**	1	0	0	**0**	**0**	**0**	**0**
pipes-nt (46)	**16**	11	10	12	2	1	1	**3**
pipe-t (45)	**9**	7	5	8	**2**	1	1	**2**
psr-small (50)	46	44	41	**48**	14	14	8	**24**
quantum (20)	**10**	8	9	9	**2**	1	1	**2**
rovers (40)	**4**	**4**	**4**	**4**	0	0	0	**4**
satellite (36)	5	5	5	**6**	**1**	**1**	0	**1**
snake (20)	**6**	5	1	1	**2**	0	0	0
storage (29)	**16**	15	12	12	**7**	6	5	**7**
termes (20)	5	**6**	2	2	**0**	**0**	**0**	**0**
tidybot (40)	**20**	10	4	5	**1**	**1**	**1**	**1**
tpp (30)	**5**	4	4	**5**	3	3	3	**4**
visitall (40)	12	**16**	**16**	**16**	5	5	5	**6**
zenotravel (20)	**9**	8	8	8	**4**	3	2	**4**
Sum (1094)	**351**	309	253	335	112	91	61	**170**

Table 3: Coverage per domain, i.e., number of tasks per domain where the number of plans within a factor 1.0 or 1.5 of a cost bound was found by K*, SymK, and our SAT-based approaches, Count and Enum. Count only counts plans, while Enum outputs each plan.

solved task had $8.9 \cdot 10^{12}$. In contrast, K* could only count up to a 10 million plans in these domains.

The SAT-based approach is less effective in other domains. One reason is that they are less specialized than heuristic and symbolic search approaches to optimal planning. Among other factors, the sequential encoding is not concise enough for some tasks and bounds (e.g., airport), or the grounding algorithm of Madagascar is inferior to those of other planners built on top of the FastDownward grounder (Helmert 2006, 2009), making it impossible to ground certain tasks (e.g., organic-synthesis). It would be interesting to evaluate how other encodings perform (Rintanen 2012), but that brings the additional problem of losing the one-to-one correspondence between plans and SAT models.

For 1.5, counting is more feasible than enumeration in many domains: as the number of plans increases, enumeration becomes less practical. Counting works for many reasoning tasks, e.g., those based on conditional probabilities.

Beyond Counting

As illustrated above, our Planalyst tool effectively counts plans by compiling into a d-DNNF and performing a counting query. This method can not only answer conditional probability questions, such as the quantity of an operator in plans, but also addresses other reasoning questions more directly and efficiently through d-DNNF queries using ddnnife (Sundermann et al. 2024). Consider reasoning questions about the plan space of a given planning task, while respecting a cost bound. Given the d-DNNF representing the plan space, questions about brave and cautious operators can be answered directly, even without traversing the entire d-DNNF, when the number of plans is known (Sundermann et al. 2024). This can be achieved by traversing the literal nodes of the d-DNNF and collecting the backbone variables, i.e., the variables that are always true (core) or false (dead). In addition, given the d-DNNF, it is possible to uniformly sample plans without enumerating the full set by d-DNNF traversing with ddnnife. This allows to address planning biases when selecting plans (Paredes et al. 2024; Frank et al. 2024) and thus collect unbiased training data for different learning approaches (Shen, Trevizan, and Thiébaux 2020; Areces et al. 2023; Chen, Thiébaux, and Trevizan 2024; Bachor and Behnke 2024). We omit empirical results for these queries, as their overhead is negligible once the d-DNNF is constructed. Our experiments with the Count configuration of Planalyst have shown that this construction is feasible for many planning tasks.

Conclusion and Future Work

We count plans and reason in the solution space, which is orthogonal to previous works in planning (Katz et al. 2018; Speck, Mattmüller, and Nebel 2020; Katz and Sohrabi 2020). Moreover, we reason about the plan space in the form of queries and introduce faceted reasoning to planning allowing for questions on the significance of operators. Although faceted reasoning is computationally hard (NP-c), it is, under standard theoretical assumptions, significantly more efficient than counting the number of plans (#P-c). Finally, we present our new reasoning tool, Planalyst, which can count the number of plans assuming fixed given length. It also supports different plan space queries. In general, Planalyst is competitive with state-of-the-art top-k planners and outperforms all other methods when the plan space is too large, i.e., more than 10 million plans.

In the future, we plan to integrate Planalyst into other pipelines, such as goal recognition (Mirsky, Keren, and Geib 2021), grounding via learning (Gnad et al. 2019), and task rewriting (Areces et al. 2014; Elahi and Rintanen 2024), using counting and facet reasoning for guidance. Interesting topics for considerations could be to deal with inconsistencies (Ulbricht 2019) and certifying results (Alviano et al. 2019; Fichte, Hecher, and Roland 2022) as well as explaining reasoning behind decisions (Cabalar, Fandinno, and Muñiz 2020). We will study how our framework extends to other encodings, such as parallel operator encodings (Rintanen 2012) or lifted encodings (Höller and Behnke 2022).

Acknowledgements

Authors are ordered in reverse alphabetical order. David Speck was funded by the Swiss National Science Foundation (SNSF) as part of the project "Unifying the Theory and Algorithms of Factored State-Space Search" (UTA). Hecher was supported by the Austrian Science Fund (FWF), grants J 4656 and P 32830, the Society for Research Funding in Lower Austria (GFF, Gesellschaft für Forschungsförderung NÖ), grant ExzF-0004, as well as the Vienna Science and Technology Fund (WWTF), grant ICT19-065. The work has been carried out while Hecher visited the Simons Institute at UC Berkeley. Fichte was funded by ELLIIT funded by the Swedish government.

References

Alrabbaa, C.; Rudolph, S.; and Schweizer, L. 2018. Faceted Answer-Set Navigation. In *RuleML+RR 2018*.

Alviano, M.; Dodaro, C.; Fichte, J. K.; Hecher, M.; Philipp, T.; and Rath, J. 2019. Inconsistency Proofs for ASP: The ASP - DRUPE Format. *Theory Pract. Log. Program.*, 19(5-6).

Alviano, M.; Dodaro, C.; Fiorentino, S.; Previti, A.; and Ricca, F. 2023. ASP and subset minimality: Enumeration, cautious reasoning and MUSes. *AIJ*, 320.

Alviano, M.; Dodaro, C.; Leone, N.; and Ricca, F. 2015. Advances in WASP. In *LPNMR 2015*.

Areces, C.; Bustos, F.; Dominguez, M. A.; and Hoffmann, J. 2014. Optimizing Planning Domains by Automatic Action Schema Splitting. In *ICAPS 2014*.

Areces, F.; Ocampo, B.; Areces, C.; Domínguez, M.; and Gnad, D. 2023. Partial Grounding in Planning using Small Language Models. In *ICAPS 2023 Workshop on Knowledge Engineering for Planning and Scheduling*.

Audemard, G.; Lagniez, J.; and Miceli, M. 2022. A New Exact Solver for (Weighted) Max#SAT. In *SAT 2022*, 28.

Audemard, G.; and Simon, L. 2018. On the Glucose SAT Solver. *Int. J. Artif. Intell. Tools*, 27(1): 27.

Aziz, R. A.; Chu, G.; Muise, C.; and Stuckey, P. 2015. Stable Model Counting and Its Application in Probabilistic Logic Programming. In *AAAI 2015*.

Bachor, P.; and Behnke, G. 2024. Learning Planning Domains from Non-redundant Fully-Observed Traces: Theoretical Foundations and Complexity Analysis. In *AAAI 2024*.

Biere, A.; Froleyks, N.; and Wang, W. 2023. CadiBack: Extracting Backbones with CaDiCaL. In *SAT 2023*,3.

Boddy, M.; Gohde, J.; Haigh, T.; and Harp, S. 2005. Course of Action Generation for Cyber Security Using Classical Planning. In *ICAPS 2005*.

Böhl, E.; Gaggl, S. A.; and Rusovac, D. 2023. Representative Answer Sets: Collecting Something of Everything. In *ECAI 2023*.

Bonet, B.; and Geffner, H. 2015. Policies that Generalize: Solving Many Planning Problems with the Same Policy. In *IJCAI 2015*.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *AIJ*, 69(1–2).

Cabalar, P.; Fandinno, J.; and Muñiz, B. 2020. A System for Explainable Answer Set Programming. *Electronic Proceedings in Theoretical Computer Science*, 325.

Chakraborti, T.; Kang, J.; Fuggitti, F.; Katz, M.; and Sohrabi, S. 2024. Interactive Plan Selection Using Linear Temporal Logic, Disjunctive Action Landmarks, and Natural Language Instruction. In *AAAI 2024*.

Chen, D. Z.; Thiébaux, S.; and Trevizan, F. 2024. Learning Domain-Independent Heuristics for Grounded and Lifted Planning. In *AAAI 2024*.

Corrêa, A. B.; Hecher, M.; Helmert, M.; Longo, D. M.; Pommerening, F.; and Woltran, S. 2023. Grounding Planning Tasks Using Tree Decompositions and Iterated Solving. In *ICAPS 2023*.

Darwiche, A. 1999. Compiling Knowledge into Decomposable Negation Normal Form. In *IJCAI 1999*.

Darwiche, A. 2001. Decomposable Negation Normal Form. *JACM*, 48(4).

Darwiche, A.; and Marquis, P. 2002. A Knowledge Compilation Map. *JAIR*, 17.

Domshlak, C.; and Hoffmann, J. 2007. Probabilistic Planning via Heuristic Forward Search and Weighted Model Counting. *JAIR*, 30.

Eifler, R.; Cashmore, M.; Hoffmann, J.; Magazzeni, D.; and Steinmetz, M. 2020. A New Approach to Plan-Space Explanation: Analyzing Plan-Property Dependencies in Oversubscription Planning. In *AAAI 2020*.

Eiter, T.; Fichte, J. K.; Hecher, M.; and Woltran, S. 2024. Epistemic Logic Programs: Non-Ground and Counting Complexity. In *IJCAI 2024*.

Eiter, T.; Hecher, M.; and Kiesel, R. 2024. aspmc: New frontiers of algebraic answer set counting. *AIJ*, 330.

Elahi, M.; and Rintanen, J. 2024. Optimizing the Optimization of Planning Domains by Automatic Action Schema Splitting. In *AAAI 2024*.

Fenner, S. A.; Green, F.; Homer, S.; and Pruim, R. 1999. Determining Acceptance Possibility for a Quantum Computation is Hard for the Polynomial Hierarchy. *ECCC*, TR99-003.

Fichte, J. K.; Gaggl, S. A.; Hecher, M.; and Rusovac, D. 2024. IASCAR: Incremental Answer Set Counting by Anytime Refinement. *Theory Pract. Log. Program.*, 24(2).

Fichte, J. K.; Gaggl, S. A.; and Rusovac, D. 2022. Rushing and Strolling among Answer Sets – Navigation Made Easy. In *AAAI 2022*.

Fichte, J. K.; Hecher, M.; and Hamiti, F. 2021. The Model Counting Competition 2020. *ACM Journal of Experimental Algorithmics*, 26(13).

Fichte, J. K.; Hecher, M.; Morak, M.; and Woltran, S. 2017. Answer Set Solving with Bounded Treewidth Revisited. In *LPNMR 2017*.

Fichte, J. K.; Hecher, M.; and Nadeem, M. A. 2022. Plausibility Reasoning via Projected Answer Set Counting - A Hybrid Approach. In *IJCAI 2022*.

Fichte, J. K.; Hecher, M.; and Roland, V. 2022. Proofs for Propositional Model Counting. In *SAT 2022*, 30.

Frank, J.; Paredes, A.; Benton, J.; and Muise, C. 2024. Bias in Planning Algorithms. In *ICAPS Workshop on Reliable Data-Driven Planning and Scheduling (RDDPS)*.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2014. Clingo = ASP + Control: Preliminary Report. arXiv:1405.3694 [cs.PL].

Gebser, M.; Kaminski, R.; König, A.; and Schaub, T. 2011. Advances in gringo Series 3. In *LPNMR 2011*.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2009. Solution Enumeration for Projected Boolean Search Problems. In *CPAIOR 2009*.

Gill, J. 1977. Computational Complexity of Probabilistic Turing Machines. *SICOMP*, 6(4).

Gnad, D.; Torralba, Á.; Domínguez, M. A.; Areces, C.; and Bustos, F. 2019. Learning How to Ground a Plan – Partial Grounding in Classical Planning. In *AAAI 2019*.

Gragera, A.; Fuentetaja, R.; Olaya, Á. G.; and Fernández, F. 2023. A Planning Approach to Repair Domains with Incomplete Action Effects. In *ICAPS 2023*.

Hahn, S.; Janhunen, T.; Kaminski, R.; Romero, J.; Rühling, N.; and Schaub, T. 2022. Plingo: A System for Probabilistic Reasoning in Clingo Based on LP $^{\text{MLN}}$. In *RuleML+RR 2022*.

Helmert, M. 2006. The Fast Downward Planning System. *JAIR*, 26.

Helmert, M. 2009. Concise Finite-Domain Representations for PDDL Planning Tasks. *AIJ*, 173.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *ICAPS 2009*.

Höller, D.; and Behnke, G. 2022. Encoding Lifted Classical Planning in Propositional Logic. In *ICAPS 2022*.

Kabir, M.; Everardo, F. O.; Shukla, A. K.; Hecher, M.; Fichte, J. K.; and Meel, K. S. 2022. ApproxASP - a Scalable Approximate Answer Set Counter. In *AAAI 2022*.

Karpas, E.; and Domshlak, C. 2009. Cost-Optimal Planning with Landmarks. In *IJCAI 2009*.

Katz, M.; and Lee, J. 2023. K* Search Over Orbit Space for Top-k Planning. In *IJCAI 2023*.

Katz, M.; and Sohrabi, S. 2020. Reshaping Diverse Planning. In *AAAI 2020*.

Katz, M.; Sohrabi, S.; Udrea, O.; and Winterer, D. 2018. A Novel Iterative Approach to Top-k Planning. In *ICAPS 2018*.

Kautz, H.; and Selman, B. 1992. Planning as Satisfiability. In *ECAI 1992*.

Kiesel, R.; and Eiter, T. 2023. Knowledge Compilation and More with SharpSAT-TD. In *KR 2023*.

Kleine Büning, H.; and Lettmann, T. 1999. *Propositional logic – deduction and algorithms*, volume 48 of *Cambridge tracts in theoretical computer science*. Cambridge University Press.

Krajnanský, M.; Hoffmann, J.; Buffet, O.; and Fern, A. 2014. Learning Pruning Rules for Heuristic Search Planning. In *ECAI 2014*.

Krarup, B.; Krivic, S.; Magazzeni, D.; Long, D.; Cashmore, M.; and Smith, D. E. 2021. Contrastive Explanations of Plans through Model Restrictions. *JAIR*, 72.

Lagniez, J.; and Marquis, P. 2017. An Improved Decision-DNNF Compiler. In *IJCAI 2017*.

Lai, Y.; Meel, K. S.; and Yap, R. H. C. 2021. The Power of Literal Equivalence in Model Counting. In *AAAI 2021*.

Lin, S.; Grastien, A.; and Bercher, P. 2023. Towards Automated Modeling Assistance: An Efficient Approach for Repairing Flawed Planning Domains. In *AAAI 2023*.

Masina, G.; Spallitta, G.; and Sebastiani, R. 2023. On CNF Conversion for Disjoint SAT Enumeration. In *SAT 2023*, 15.

Mirsky, R.; Keren, S.; and Geib, C. W. 2021. *Introduction to Symbolic Plan and Goal Recognition*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Muise, C. 2016. Planning.Domains. In *ICAPS 2016 System Demonstrations and Exhibits*.

Papadimitriou, C. H. 1994. *Computational Complexity*. Addison-Wesley.

Papadimitriou, C. H.; and Yannakakis, M. 1982. The Complexity of Facets (and Some Facets of Complexity). In *STOC 1982*.

Paredes, A.; Frank, J.; Benton, J.; and Muise, C. 2024. Planning Bias: Planning as a Source of Sampling Bias. In *ICAPS Workshop on Reliable Data-Driven Planning and Scheduling (RDDPS)*.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the Extraction, Ordering, and Usage of Landmarks in Planning. In *ECP 2001*.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks Revisited. In *AAAI 2008*.

Rintanen, J. 2011. Madagascar: Scalable Planning with SAT. In *IPC 2011 Planner Abstracts*.

Rintanen, J. 2012. Planning as Satisfiability: Heuristics. *AIJ*, 193.

Rintanen, J. 2014. Madagascar: Scalable Planning with SAT. In *IPC-8 Planner Abstracts*.

Rusovac, D.; Hecher, M.; Gebser, M.; Gaggl, S. A.; and Fichte, J. K. 2024. Navigating and Querying Answer Sets: How Hard Is It Really and Why? In *KR 2024*.

Russell, S.; and Norvig, P. 1995. *Artificial Intelligence — A Modern Approach*. Prentice Hall.

Shen, W.; Trevizan, F.; and Thiébaux, S. 2020. Learning Domain-Independent Planning Heuristics with Hypergraph Networks. In *ICAPS 2020*.

Smith, D. E. 2004. Choosing Objectives in Over-Subscription Planning. In *ICAPS 2004*.

Sohrabi, S.; Riabov, A. V.; Katz, M.; and Udrea, O. 2018. An AI Planning Solution to Scenario Generation for Enterprise Risk Management. In *AAAI 2018*.

Spallitta, G.; Sebastiani, R.; and Biere, A. 2024. Disjoint Partial Enumeration without Blocking Clauses. In *AAAI 2024*.

Speck, D.; Hecher, M.; Gnad˙ D.; Fichte, J. K.; and Corrêa, A. B. 2024. Code, benchmarks and data for the AAAI 2025 paper "Counting and Reasoning with Plans". https://doi.org/10.5281/zenodo.14499686.

Speck, D.; Mattmüller, R.; and Nebel, B. 2020. Symbolic Top-k Planning. In *AAAI 2020*.

Stockmeyer, L. J. 1976. The Polynomial-Time Hierarchy. *Theoretical Computer Science*, 3(1).

Stockmeyer, L. J.; and Meyer, A. R. 1973. Word problems requiring exponential time. In *STOC 1973*.

Sundermann, C.; Raab, H.; Hess, T.; Thüm, T.; and Schaefer, I. 2024. Reusing d-DNNFs for Efficient Feature-Model Counting. *ACM Trans. Softw. Eng. Methodol.*

Taitler, A.; Alford, R.; Espasa, J.; Behnke, G.; Fišer, D.; Gimelfarb, M.; Pommerening, F.; Sanner, S.; Scala, E.; Schreiber, D.; Segovia-Aguas, J.; and Seipp, J. 2024. The 2023 International Planning Competition. *AI Magazine*.

Torralba, Á.; Alcázar, V.; Kissmann, P.; and Edelkamp, S. 2017. Efficient Symbolic Search for Cost-optimal Planning. *AIJ*, 242.

Ulbricht, M. 2019. *Understanding Inconsistency – A Contribution to the Field of Non-monotonic Reasoning.* Ph.D. thesis, Universität Leipzig.

Valiant, L. G. 1979. The Complexity of Computing the Permanent. *Theoretical Computer Science*, 8.

von Tschammer, J.; Mattmüller, R.; and Speck, D. 2022. Loopless Top-K Planning. In *ICAPS 2022*.

Wrathall, C. 1976. Complete Sets and the Polynomial-Time Hierarchy. *Theoretical Computer Science*, 3(1).

Zhu, L.; and Givan, R. 2003. Landmark Extraction via Planning Graph Propagation. In *ICAPS 2003 Doctoral Consortium*.