




A knowledge compilation perspective on queries and transformations for belief tracking

Alexandre Niveau¹ · Hector Palacios² · Sergej Scheck¹ · Bruno Zanuttini¹ 

Accepted: 17 October 2023

© The Author(s), under exclusive licence to Springer Nature Switzerland AG 2024

Abstract

Nondeterministic planning is the process of computing plans or policies of actions achieving given goals, when there is nondeterministic uncertainty about the initial state and/or the outcomes of actions. This process encompasses many precise computational problems, from classical planning, where there is no uncertainty, to contingent planning, where the agent has access to observations about the current state. Fundamental to these problems is belief tracking, that is, obtaining information about the current state after a history of actions and observations. At an abstract level, belief tracking can be seen as maintaining and querying the current belief state, that is, the set of states consistent with the history. We take a knowledge compilation perspective on these processes, by defining the queries and transformations which pertain to belief tracking. We study them for propositional domains, considering a number of representations for belief states, actions, observations, and goals. In particular, for belief states, we consider explicit propositional representations with and without auxiliary variables, as well as implicit representations by the history itself; and for actions, we consider propositional action theories as well as ground PDDL and conditional STRIPS. For all combinations, we investigate the complexity of relevant queries (for instance, whether an action is applicable at a belief state) and transformations (for instance, revising a belief state by an observation); we also discuss the relative succinctness of representations. Though many results show an expected tradeoff between succinctness and tractability, we identify some interesting combinations. We also discuss the choice of representations by existing planners in light of our study.

✉ Bruno Zanuttini
bruno.zanuttini@unicaen.fr

Alexandre Niveau
alexandre.niveau@unicaen.fr

Hector Palacios
hectorpal@gmail.com

Sergej Scheck
sergej.scheck_dd@gmx.de

¹ Normandie University UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

² Work done while at ServiceNow Research, Montréal, QC, Canada

Keywords Knowledge compilation · Nondeterministic planning · Action languages · Partial observability

Mathematics Subject Classification (2010) 68T30 · 68T37

1 Introduction

Nondeterministic planning is the general problem of finding a plan or a policy of actions, under uncertainty about the initial state and the outcome of actions, and under a variety of observation capabilities [1]. These include full observability (the problem is then referred to as “FOND planning”), partial observability (“contingent planning”), and no observability at all (“conformant planning”). In the most general setting, when a history of actions and observations is experienced or simulated, a number of possible states are consistent with the history, constituting what is called the current *belief state*. Obtaining information about the current belief state (e.g., whether an action is guaranteed to be applicable) and maintaining it — typically, updating it when an observation is received or an action is executed — can be framed as the general problem of (offline) *belief tracking* [2]. In this paper we consider the purely logical setting, in particular without probabilities, costs, nor durations of actions.

Planners accept problems in a language for expressing the possible initial states, the goal to be achieved, the actions available for modifying the state, and the sensing actions that trigger observations. While searching for a plan, they typically transform these elements into internal representations of belief states. In this paper we study the properties of languages used by nondeterministic planners, as well as queries and transformations they perform, from a knowledge compilation perspective [3].

We focus on languages based on propositional representations, like ground PDDL for describing actions [4], or CNF for describing the set of initial states. While most planners accept parametric PDDL for describing actions, indeed many of them first ground this representation into a propositional one. Multi-valued variables are also popular as input or state representations, but we leave their study for future work. Given that propositional setting, we study a large number of languages:

- for belief states, explicit representations by propositional formulas (CNF, binary decision diagrams, etc.), with or without auxiliary variables, and implicit representations by the history itself, as used by the FF family of planners [5];
- for actions, logical representations by propositional theories (again, in a number of languages) and standard representations by (ground) PDDL and conditional STRIPS;
- for observations, representations by propositional formulas;
- for goals, logical representations of both ontic goals (about the state of the environment) and epistemic goals (about the belief state, that is, about the information gathered in addition to the state itself).

Overall, we believe that this selection of languages covers most languages considered in the literature, and additionally covers natural variants thereof.

For these, we study the computational complexity of a number of queries and transformations, which we motivate by considering a generic algorithm that covers contingent, conformant, and FOND planning. We focus on *progression*, that is, forward search from the initial belief state to the goal, since regression typically requires to manipulate *sets* of belief states. Our study however applies to a more general problem than offline planning: in particular, we consider observations which can be received independently of actions, as

in generalized belief tracking [2]. The queries which we consider are whether an action is applicable, whether an observation is possible, whether a goal is reached, and whether two belief states are equivalent; the transformations are the progression of a belief state by an action or by an observation.

Not surprisingly, a number of combinations of languages have already been studied in the literature, and part of the complexity results of interest were already derived in isolation, or are consequences of known results, especially available in the knowledge compilation map by [3]. However, our study reviews these results and completes the picture. In particular, we give conditions as weak as possible for certain problems to be (in)tractable. We also consider combinations beyond those studied in the literature; in particular, implicit representations of belief states with action theories, explicit representations with STRIPS actions, and epistemic goals.

The paper is organized as follows. We first give background about planning and knowledge compilation (Section 2) and review related work (Section 3). Then we define and motivate the transformations and queries (Section 4) and languages (Section 5) that are studied in the paper. The complexity results about queries and transformations are given in Section 6 for explicit representations of belief states and in Section 7 for implicit representations; Section 8 covers succinctness results. In Section 9 we review existing planners in light of our results, and we conclude in Section 10. The results are summarized in tables throughout the paper.

2 Background

In this section we give the background about belief tracking, planning, and knowledge compilation.

2.1 Formal setting

Given a finite set F of propositional variables called *fluents*, the finite *state space* generated by F is $\mathcal{S} := 2^F$. Elements of \mathcal{S} are called *states*, hence a state s is an assignment to the fluents in F . We view such a state as a subset of F ; for instance, for $F = \{x_1, x_2, x_3\}$, $s = \{x_1, x_3\}$ denotes the state in which x_1 and x_3 are assigned \top and x_2 is assigned \perp .

We also use formulas over propositional variables which do not directly encode a property of the state. For these we use the same notation, and in particular, we write $s_1 \cup s_2$ for combining assignments to disjoint sets of variables, whether fluents or not.

An (*ontic*) *action* a on \mathcal{S} is a mapping from \mathcal{S} to $2^{\mathcal{S}}$; this means that when a is taken in a state s , this state changes nondeterministically to some $s' \in a(s)$. The image of a at a set of states $S \subseteq \mathcal{S}$, denoted by $a(S)$, is defined to be $\bigcup_{s \in S} a(s)$. An *observation* o on \mathcal{S} is a nonempty subset of \mathcal{S} : the intuition is that when an agent observes $o \subseteq \mathcal{S}$, it learns that the current state is one of the states in o . After actions and observations have taken place, the knowledge of an agent about the current state thus boils down to a set of possible states. Accordingly, we call *belief state* (over \mathcal{S}) a set of states $B \subseteq \mathcal{S}$. It is important to note that the belief state of an agent normally cannot be empty — this would mean that no state is considered to be possibly the actual current state.

An action a is said to be *applicable* at a state s if $a(s) \neq \emptyset$ holds, and *applicable* at a belief state B if it is applicable at all states $s \in B$; this intuitively means that the agent only takes actions which it *knows* to be applicable. An observation o is said to be *fair at a state* s if

$s \in o$ holds; it is said to be *fair at a belief state* B if $B \cap o \neq \emptyset$ holds (in other words, there is at least one state $s \in B$ at which it is fair). Intuitively, this formalizes *reliable* observations.¹

Note that we adopt a general definition of an observation: they might come from sensing actions or from external, uncontrolled events. This allows us to consider belief tracking in a general setting.

Example 1 Consider a robot on a grid. Its location can be specified by two integer values, x (for the horizontal axis) and y (for the vertical axis). Four actions can move the robot one step in all directions: *right* (resp. *left*) increases (resp. decreases) the value of x , but can also change the value y , thus making it move vertically in a nondeterministic fashion. By contrast, *up* (resp. *down*) is deterministic: it increases (resp. decreases) y without affecting x , so the outcome is always the expected one. However, while *right* and *left* are applicable at all the states, it is not always possible to move vertically: there is a switch that can be on or off, and applying action up or down requires the switch to be activated — this is called a *precondition* of these actions. If x or y is at the maximum/minimum value, executing the corresponding action does not move the robot. Finally, the robot can only observe the state of the switch if it is in the leftmost column, i.e., $x = 1$.

Formally, we model this setting for a 2×2 grid using the set of fluents $F := \{x_1, x_2, y_1, y_2, \text{sw_on}\}$. The first four fluents describe the position of the robot in the grid, e.g., x_2 is true if and only if $x = 2$; the fifth one, sw_on , is true if and only if the switch is activated. The state space \mathcal{S} then includes the following states:

$$\begin{array}{ll} s_1 := \{x_1, y_1, \text{sw_on}\} & s_2 := \{x_1, y_1\} \\ s_3 := \{x_1, y_2, \text{sw_on}\} & s_4 := \{x_1, y_2\} \\ s_5 := \{x_2, y_1, \text{sw_on}\} & s_6 := \{x_2, y_1\} \\ s_7 := \{x_2, y_2, \text{sw_on}\} & s_8 := \{x_2, y_2\} \end{array}$$

It also contains states such as $\{x_1, x_2\}$, which have no interpretation as valid configurations of the world. These are harmless as long as they are not reachable by any action. Note that for the 2×2 grid, two fluents would actually suffice to describe the position of the robot, but the scheme used here is more directly generalizable to any grid, which makes for a more illustrative example.

Applying *left* in the state s_1 results in the state s_1 or s_3 . The action *up* is not applicable at the belief state $\{s_1, s_2\}$ since the precondition that the switch be on is not (guaranteed to be) satisfied. Applying *up* in the belief state $\{s_1, s_3\}$ results in the belief state $\{s_3\}$.

We define the observations that the switch is activated (resp. disabled) as the set of states $\text{on} := \{s_1, s_3, s_5, s_7\}$ (resp. $\text{off} := \{s_2, s_4, s_6, s_8\}$), considering only valid states — but adding invalid states to on or off would not change their semantics in practice, so another possibility would be to define them as $\{s \in \mathcal{S} \mid \text{sw_on} \in s\}$ (resp. $\text{sw_on} \notin s$), which can yield simpler representations. Observing on is not fair at belief state $\{s_2, s_4\}$ (it is not fair at any of its possible states, so this observation would not be consistent with the robot's beliefs), but it is fair at $\{s_1, s_2\}$ (and would allow the robot to infer that the real state is s_1).

A *history* over a set of actions A and a set of observations O is a sequence $h = \langle ao_1, \dots, ao_n \rangle$, where for $i = 1, \dots, n$, ao_i is either an action in A or an observation in O .

¹ By definition, at $B = \emptyset$, all actions are applicable and no observation is fair, but this will be of little importance in the rest of the paper.

Given a belief state $B \subseteq S$ and a history h over A and O , the *belief state induced by h at B* , written $h(B)$, is defined inductively as follows:

$$h(B) = \begin{cases} B & \text{for } h = \langle \rangle \\ a(h'(B)) & \text{for } h = h' \cdot a \text{ with } a \in A \\ h'(B) \cap o & \text{for } h = h' \cdot o \text{ with } o \in O \end{cases}$$

A history $h = \langle ao_1, \dots, ao_n \rangle$ is said to be applicable at a belief state B if for all $i = 1, \dots, n$ such that ao_i is an action, this action is applicable at the belief state induced by $\langle ao_1, \dots, ao_{i-1} \rangle$ at B ; and h is said to be fair at B if for all $i = 1, \dots, n$ such that ao_i is an observation, this observation is fair at the belief state induced by $\langle ao_1, \dots, ao_{i-1} \rangle$ at B .

Example 2 (continued from Example 1). Let us consider the belief state $B = \{s_1, \dots, s_8\}$. The history $\langle \text{right}, \text{up} \rangle$ is not applicable at B : indeed, while right is always applicable, we have $\text{right}(B) = \{s_5, s_6, s_7, s_8\}$, and the precondition of up does not hold in s_6 and s_8 , so up is not applicable at $\text{right}(B)$ (even though $\text{up}(\text{right}(B)) \neq \emptyset$). However, the history $h = \langle \text{right}, \text{on}, \text{up} \rangle$ is applicable and fair at B , since (i) on is fair at $\text{right}(B)$, and (ii) the belief state induced by $\langle \text{right}, \text{on} \rangle$ at B is $\{s_5, s_7\}$, at which up is (guaranteed to be) applicable. The belief state induced by h at B is the singleton belief state $\{s_7\}$. Finally, the history $\langle \text{right}, \text{on}, \text{up}, \text{off}, \text{down} \rangle$ is not fair at B — nor at any other belief state, since $\text{on} \cap \text{off} = \emptyset$ and up cannot change the value of sw_on — but it is applicable at B , since down is vacuously applicable at $\langle \text{right}, \text{on}, \text{up}, \text{off} \rangle$.

2.2 Planning instances

We now introduce planning domains and instances, in which observations are triggered by sensing actions.

A (*nondeterministic planning*) domain \mathcal{D} is a tuple $\langle F, A, SA \rangle$, where F is a set of fluents, A is a set of (ontic) actions, and SA is a set of *sensing actions* $\{sa_1, \dots, sa_m\}$. In planning, a sensing action is under the control of the agent: when used, it gives one among a set of observations. Hence a sensing action can be seen as a set of observations $\{o_1, \dots, o_k\}$ satisfying $o_1 \cup o_2 \cup \dots \cup o_k = S$ (recall that we write $S := 2^F$, and that an observation is a subset of S); this covering requirement means that whatever the current state, a sensing action yields a fair observation. We write $o \in sa$ to mean that o is a possible observation when sa is taken. Special cases are *deterministic* sensing actions for which $o_i \cap o_j = \emptyset$ holds for $i \neq j$, and the *void* sensing action $\{S\}$, which always yields the same observation.

We do not introduce preconditions for sensing actions, but this is without loss of generality, since if we want a sensing action $\{o_1, \dots, o_k\}$ to be applicable only if some precondition P is true, we can simply replace it with $\{S, o_1 \cap P, \dots, o_k \cap P\}$, so that it boils down to not observing anything meaningful in case the precondition is not met.

A (*nondeterministic*) *planning instance* \mathcal{P} is a tuple $\langle \mathcal{D}, I, \vec{G} \rangle$, where \mathcal{D} is a domain, $I \subseteq S$ is a nonempty set of initial states, and $\vec{G} \subseteq 2^S$ is a predicate on belief states. Given a planning instance \mathcal{P} , a history h *achieves* the goal if h is applicable and fair at the initial state I , and $h(I) \in \vec{G}$ holds. A specific case called “statewise goals” is when the goal is specified by a nonempty subset G of S of *goal states*, as usual in the planning literature; this implicitly defines the goal $\vec{G} := \{B \subseteq S \mid B \subseteq G\}$, meaning, in words, that all states in the belief state are indeed goal states. For details we refer the reader to [1]. Different kinds of planning problems can be defined on a planning instance, depending on the form of the

solution (plan, policy) and on specific requirements it must satisfy (such as ensuring that the goal be reached, or only reaching it for some executions).

2.3 Knowledge compilation and complexity

We are interested in the complexity of various queries and transformations on belief states, and in the relative succinctness of representations. For these results we will consider concrete representations of belief states and other input components of the problems: we call *language* for a class \mathcal{C} of objects, a set Λ of strings called *expressions*, together with a function $\sigma \mapsto \|\sigma\|$ mapping each string σ to an object $\|\sigma\| \in \mathcal{C}$. A language is said to be *complete* if $\|\cdot\|$ is surjective, i.e., if any object in \mathcal{C} can be represented as an expression in Λ . We denote the number of symbols in the expression σ by $|\sigma|$; concrete languages will be defined in Section 5.

We adopt the definitions standard in the knowledge compilation literature [3]. We call *query* a decision problem with input given by (the representation of) a belief state together with additional components. Contrastingly, a *transformation* is a functional problem, with input given by a belief state together with additional components, and whose output is a belief state, which we require to be in the same representation as the input belief state.

Accordingly, for the complexity of queries we will consider complexity classes of decision problems. We assume familiarity with P, NP, coNP. The class $\Sigma_2\text{P}$ is the class NP^{NP} of decision problems for which there exists a nondeterministic polynomial-time algorithm using an NP oracle (or, equivalently, using a coNP-oracle), and $\Pi_2\text{P} = \text{coNP}^{\text{NP}}$ is the class of decision problems whose complement is in $\Sigma_2\text{P}$. We will also consider a class that is a little less standard: $\Theta_2\text{P} = \text{P}^{\|\text{NP}\|}$ is the class of decision problems for which there exists a deterministic polynomial-time algorithm using *parallel* queries to an NP oracle [6]; that is, the input to one query is not allowed to depend on the output of another one. Clearly, we have $\text{P} \subseteq \{\text{NP}, \text{coNP}\} \subseteq \Theta_2\text{P} \subseteq \{\Sigma_2\text{P}, \Pi_2\text{P}\}$, and all inclusions are widely believed to be proper.

For the complexity of transformations, there are in general several possible output expressions for the same input (intuitively, equivalent expressions — interpreted as the same belief state); we will call them *valid* outputs. We will consider three situations:

- a valid output can be computed in time polynomial in the size of the input (belief state and other components), and hence, in particular, the output has *size* polynomial in the size of the input; we then say that the transformation is *polynomial-time*, written FP;
- there is a valid output of polynomial size, but no valid output can be computed in polynomial time (possibly under some complexity-theoretic assumption); we then say that the transformation is *polynomial-size* but *not polynomial-time*;
- there is no valid output of polynomial size (possibly under some assumption); we then say that the transformation is *not polynomial-size*.

As concerns the relative *succinctness* of belief state representations, we will say that a complete representation language Λ_1^{B} is *at least as succinct* as a complete language Λ_2^{B} , written $\Lambda_1^{\text{B}} \preceq^s \Lambda_2^{\text{B}}$, if there is a polynomial p such that the following holds: for all families of belief states $(B^n)_{n \in \mathbb{N}}$ and all families of representations $(\beta_2^n)_{n \in \mathbb{N}}$ of them in Λ_2^{B} , there exists a family $(\beta_1^n)_{n \in \mathbb{N}}$ of representations of them in Λ_1^{B} such that for all n , $|\beta_1^n| \leq p(|\beta_2^n|)$ holds. Accordingly, Λ_1^{B} is *not at least as succinct* as Λ_2^{B} , written $\Lambda_1^{\text{B}} \not\preceq^s \Lambda_2^{\text{B}}$, if there exists a family in Λ_2^{B} which has no polynomial-size equivalent in Λ_1^{B} .

Finally, the class P/poly of *nonuniform polynomial-time* is the class of problems which, for all $n \in \mathbb{N}$, admit a polynomial-time algorithm for the restriction of the problem to instances of size n . We will use the widely believed assumption $\text{NP} \not\subseteq \text{P/poly}$.

3 Related work

Viewing nondeterministic planning as search in the space of belief states dates back to early work by [7]. It is also a natural approach to follow in the stochastic setting of planning for partially observable Markov Decision Processes [8]. Abstracting away from the mechanics of search, [2, 9, 10] put forward the importance of *belief tracking* in the process of planning, and consider the complexity of the main related problems; in particular, they relate it to different notions of *width* of a planning instance. Their notions of width, albeit various, all depend on the initial belief state and the set of actions available; this makes our setting different, as we consider complexity issues for *languages*, disregarding the features of precise instances. Brafman and Shani [11] later studied that process of belief tracking in an *online* setting (as opposed to planning), in which the given history has been *observed* rather than explored in a planning process. This somehow makes the problems easier, as by construction the history is known to be a valid one; we follow a similar approach when studying implicit representations of belief states by *traces* in Section 7.

A number of planners have been proposed for nondeterministic planning, which use specific representations of the current belief state during the search. We review such planners in Section 9, linking them to our study. Noticeable examples are HSCP [12], which introduced early the use of ordered binary decision diagrams to manipulate the belief states efficiently; the FF family of planners [5], which use an *implicit* representation of the current belief state; and the factored representations introduced by Palacios and Geffner [9], in which the representation uses a set of *factors* built in terms of the initial belief state, the goal and the actions. Beyond the nondeterministic setting, efficient representations of belief states have also been considered in the stochastic setting, especially with ordered *algebraic* decision diagrams [13, 14], and in the relational setting [15].

Besides belief states, various representations of actions have been considered. Nebel [16] studies to what extent a set of actions formulated in conditional STRIPS can be translated into another language, differing on the form of its effects, conditions, and preconditions, while preserving the existence and size of plans for the problem at hand; his results can be used to perform a preprocessing of a classical planning instance for obtaining a pure STRIPS instance. However, such a transformation assumes the state to be fully observable, while conditional effects allow one to progress a belief state without assuming that the conditions are satisfied. Thus, the semantics of a language might be different when progression is defined over single states or over belief states.

Departing from STRIPS and PDDL, which are standard representations of actions that specify how the state is transformed, *action theories* are logical specifications of actions, more declarative in nature. Herzig et al. [17] study such theories and formulate operations on belief states in logical terms, in the perspective of planning for epistemic goals. To et al. [18] also consider such theories, for different representations of belief states; their work is conceptually very close to ours, but we perform a much more systematic study. Recently, [19] also studied both PDDL-like actions and action theories, in the spirit of the knowledge compilation map [3], but their focus is on queries and transformations related to *states* instead of belief states.

Work related to ours also includes studies on the complexity of the planning problem itself. Bonet [20] settles the complexity of conformant planning for a number of restrictions on the languages of actions and observations. Although he uses a fixed syntax for actions, the variations come from their expressivity; specifically, he considers conditions on the belief state rather than on states, and (like us) general epistemic goals. Beyond conformant planning, [21] also studies the complexity of planning under various observation capabilities.

4 Queries and transformations

In this section we define the queries and transformations on belief states which we study in the rest of the paper. We first give a standard generic scheme for planning by forward search in the space of belief states (Section 4.1), which allows us to formally define relevant queries and transformations in Section 4.2.

4.1 A generic scheme for nondeterministic planning

We present a *generic scheme* for planning algorithms covering (non cyclic) classical, FOND, conformant, and contingent planning, where the last is the most general case. Our algorithm essentially formalizes a forward search in belief space, and similar, generic formulations can be found, for instance, in the textbooks by Geffner and Bonet [1] or Ghallab et al. [22]. We do not claim that this algorithm is interesting *per se*, as it is agnostic to search strategies and heuristics; it is however sufficient to derive the set of interesting queries and transformations to the belief state.

In the general case when there are sensing actions available, an agent can act depending on the history of observations received. Hence a *policy* for a planning instance $\langle \mathcal{D}, I, \vec{G} \rangle$, over a domain $\mathcal{D} = \langle F, A, SA \rangle$, is a tree with (i) *action* nodes, labelled by an ontic action $a \in A$ and having only one child, and (ii) *observation* nodes, labelled by a sensing action $sa \in SA$, and having one child per (fair) observation $o \in sa$. The execution semantics is the obvious one. Given a belief state I and a goal \vec{G} , a policy is said to solve the problem associated to the planning instance if all the induced histories achieve \vec{G} .

Given that it searches for a policy, the standard scheme for contingent planning consists of a search in an and-or graph, where or-nodes correspond to the choice of an action with which to expand the current (implicit) policy, and and-nodes correspond to the need of successfully expanding the current policy for all observations which may be received after taking a sensing action.

We depict the decision version of this scheme in Algorithm 1: it returns “true” if there is a (non-cyclic) policy which solves the problem, and “false” otherwise — the policy can be recovered by bookkeeping the tree or rooted DAG of actions and observations chosen by the algorithm. The set \tilde{B}_{OK} keeps track of belief states from which search has already proven that there is a policy achieving the goal, and \tilde{B}_{branch} keeps track of the belief states in the current stack of recursive calls, allowing to detect loops in the search. The fact that the algorithm is correct is straightforward; note that the belief state considered at any given recursive call is guaranteed to be nonempty by construction (it is initially I , and is only progressed by applicable actions and fair observations).

4.2 Definition of queries and transformations

The scheme given in Section 4.1 suggests that the queries and transformations which we are about to introduce in this section are essential to planning. However, they support some other tasks as well, such as updating a belief state with observational events not triggered by the agent’s actions, and more generally, *belief tracking*.

Algorithm 1 Standard scheme for Contingent Planning.**Input:**

planning domain $\langle F, A, SA \rangle$, nonempty belief state B (initially I), goal \vec{G} ;
 sets of belief states \vec{B}_{OK} (global), \vec{B}_{branch} (local), both initially empty

Output: “true” if there is a policy achieving \vec{G} from B , “false” otherwise

```

if  $B \in \vec{B}_{OK}$  then //  $B$  is already known to have a successful policy
  return “true”
else if  $B \in \vec{G}$  then //  $B$  satisfies the goal
  return “true”
else if  $B \in \vec{B}_{branch}$  then // loop detected on the current branch
  return “false”
else
  for all actions  $a \in A$  do // trying ontic actions
    if  $a$  is applicable at  $B$  then
       $B' \leftarrow a(B)$  // progressing  $B$  by  $a$ 
       $result \leftarrow \text{recursive}(B \leftarrow B', \vec{B}_{branch} \leftarrow \vec{B}_{branch} \cup \{B\})$ 
      if  $result = \text{“true”}$  then
         $\vec{B}_{OK} \leftarrow \vec{B}_{OK} \cup \{B\}$ 
        return “true”
      end if
    end if
  end for
  for all sensing actions  $sa \in SA$  do // trying sensing actions
     $result \leftarrow \text{true}$ 
    for all observations  $o \in sa$  do
      if  $o$  is fair at  $B$  then
         $B' \leftarrow B \cap o$  // progressing  $B$  by  $o$ 
         $result \leftarrow result \wedge \text{recursive}(B \leftarrow B', \vec{B}_{branch} \leftarrow \vec{B}_{branch} \cup \{B\})$ 
      end if
    end for
    if  $result = \text{“true”}$  then
       $\vec{B}_{OK} \leftarrow \vec{B}_{OK} \cup \{B\}$ 
      return “true”
    end if
  end for
  return “false”
end if

```

We consider languages for specific classes \mathcal{C} denoted in the following way: Λ^B for belief states, Λ^A for actions, Λ^O for observations, Λ^G for statewise goals, and $\Lambda^{\vec{G}}$ for (generic) goals.

Queries We first define relevant queries on belief states. For complexity matters, all queries depend on the representation languages for their input (belief state, action, observation, goal). We consider these languages to be parameters of the queries, not part of the input. We write, for instance, $\text{APPLIC}(\text{BS}(\text{NNF}), \text{PDDL}_{\text{Terms}, \text{CNF}})$ for the decision problem corresponding to the query denoted by APPLIC , with parameters $\Lambda^B = \text{BS}(\text{NNF})$ and $\Lambda^A = \text{PDDL}_{\text{Terms}, \text{CNF}}$ (these languages will be defined in Section 5).

The queries which we consider are defined in Table 1. They correspond to deciding whether, at a given belief state, a given action is applicable (APPLIC), a given observation is fair (FAIR), a given statewise goal is satisfied (SAT_S), and a given generic goal is satisfied (SAT); and whether two given belief states are equivalent (EQUIV). Observe that SAT is a more general query than SAT_S ; we define them independently because it is natural to use a language for a set of states, rather than for a set of belief states, for statewise goals. Also

Table 1 Definition of queries

Name	Param.	Input	Question
APPLIC	$\Lambda^B; \Lambda^A$	$\beta \in \Lambda^B; \alpha \in \Lambda^A$	is $\ \alpha\ $ applicable at $\ \beta\ $?
FAIR	$\Lambda^B; \Lambda^O$	$\beta \in \Lambda^B; \omega \in \Lambda^O$	is $\ \omega\ $ fair at $\ \beta\ $?
SAT _S	$\Lambda^B; \Lambda^G$	$\beta \in \Lambda^B; \gamma \in \Lambda^G$	does $\ \beta\ \subseteq \ \gamma\ $ hold?
SAT	$\Lambda^B; \Lambda^{\bar{G}}$	$\beta \in \Lambda^B; \bar{\gamma} \in \Lambda^{\bar{G}}$	does $\ \beta\ \in \ \bar{\gamma}\ $ hold?
EQUIV	Λ^B	$\beta_1, \beta_2 \in \Lambda^B$	does $\ \beta_1\ = \ \beta_2\ $ hold?

observe that, though equivalence does not appear as such in Algorithm 1, it is indeed needed for maintaining the sets of belief states \tilde{B}_{OK} and \tilde{B}_{branch} .

Transformations We now turn to two transformations parameterized by languages. They receive as input a belief state and either an action or an observation, and produce a belief state in the same language; we require this because the transformations are used to *maintain* a belief state, as illustrated by Algorithm 1.

Indeed, the transformations which we study, formally defined in Table 2, implement the following two *progression* operations, by an action ($PROG_a$) or by an observation ($PROG_o$), computing the belief state after an event took place. Note that other schemes for planning proceed by *regression*, but we leave the study of the corresponding transformations for future work, as in general they produce a *set* of belief states rather than a single one.

Definition 3 (progression by an action). The *progression* of a nonempty belief state B by an action a is the (nonempty) belief state $B \oplus a := a(B)$ if a is applicable at B ; it is undefined otherwise.

Definition 4 (progression by an observation). The *progression* of a nonempty belief state B by an observation o is the (nonempty) belief state $B \oplus o := B \cap o$ if o is fair at B ; it is undefined otherwise.

We emphasize that consistently with Algorithm 1, we assume that the transformations “make sense”, that is, that the input belief state is nonempty and that the action (resp. observation) is applicable (resp. fair).

Table 2 Definition of transformations

Name	Param.	Input	Output
$PROG_a$	$\Lambda^B; \Lambda^A$	$\beta \in \Lambda^B$ s.t. $\ \beta\ \neq \emptyset$ and $\alpha \in \Lambda^A$ applicable at $\ \beta\ $	$\beta' \in \Lambda^B$ with $\ \beta'\ = \ \beta\ \oplus \ \alpha\ $
$PROG_o$	$\Lambda^B; \Lambda^O$	$\beta \in \Lambda^B$ s.t. $\ \beta\ \neq \emptyset$ and $\omega \in \Lambda^O$ fair at $\ \beta\ $	$\beta' \in \Lambda^B$ with $\ \beta'\ = \ \beta\ \oplus \ \omega\ $

5 Languages

We now define the languages which we consider for all components of a planning problem, except belief states, for which they will be defined in Sections 6 and 7. Since the motivation of our study is belief tracking and planning, the representations we consider are the main ones used in the literature by various algorithms and concrete planners — Section 9 will summarize some combinations of languages used by known planners.

5.1 Propositional languages

A number of components in the definition of a planning problem can be associated to a set of propositional assignments to the set of fluents F : initial belief state, observations, and statewise goals, as well as conditions and preconditions of actions. Hence we study the same, standard, representations for all of them.

The representations defined in this section are propositional formulas (concretely represented by circuits, see Section 5.4). For such a formula φ , we denote by $V(\varphi)$ the set of variables occurring in φ . Given a propositional formula φ and a set of variables $X \supseteq V(\varphi)$, we write $\|\varphi\|_X$ for the set of all assignments to X which satisfy φ under the standard semantics of propositional logic. Depending on the context we will either say that a state s is in $\|\varphi\|_X$ or that s satisfies φ , and accordingly write either $s \in \|\varphi\|_X$ or $s \models \varphi$. Moreover, whenever the set X is clear from the context, we write $\|\varphi\|$ instead of $\|\varphi\|_X$.

Finally, we write \top (resp. \perp) for the propositional formula which is true (resp. false) under all assignments, and we recall that a *literal* λ (over X) is an expression of the form x or $\neg x$, for some variable $x \in X$.

Definition 5 (NNF). The language NNF over a set of variables X is the set of all expressions φ generated by the grammar $\varphi ::= \lambda \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$, with λ ranging over the literals over X .

Definition 6 (CNF). The language CNF over a set of variables X is the set of all expressions φ of the form $\varphi := \bigwedge_{i \in I} \gamma_i$, where I is a finite set of indices and each γ_i is a disjunction of literals over X (called a *clause* over X).

Definition 7 (DNF). The language DNF over a set of variables X is the set of all expressions φ of the form $\varphi := \bigvee_{i \in I} \tau_i$, where I is a finite set of indices and each τ_i is a consistent conjunction of literals over X (called a *term* over X).

For the next definition, we use $\text{ite}(x, \varphi_1, \varphi_2)$ as a shorthand for the formula $(x \wedge \varphi_1) \vee (\neg x \wedge \varphi_2)$.

Definition 8 (OBDD). Let $<$ be a total ordering over a set of variables X . The language $\text{OBDD}_{<}$ over X is the set of all expressions φ generated by the grammar $\varphi ::= \top \mid \perp \mid \text{ite}(x, \varphi, \varphi)$, with x ranging over X and the restriction that for all expressions of the form $\text{ite}(x, \varphi_1, \varphi_2)$, $x < y$ holds for all $y \in V(\varphi_1) \cup V(\varphi_2)$.

Definition 9 (terms). The language Terms over a set of variables X is the set of all consistent conjunctions of literals over X .

The language Terms is incomplete, because not all sets of assignments can be represented with terms. However, it is a standard language in the planning literature, especially for initial belief states, for preconditions of actions, and for statewise goals.

Example 10 (continued from Example 2). We still consider a robot on a 2×2 grid, with fluents $F := \{x_1, x_2, y_1, y_2, \text{sw_on}\}$. The initial belief state with all 8 valid states can be expressed in NNF as $\phi := (x_1 \leftrightarrow \neg x_2) \wedge (y_1 \leftrightarrow \neg y_2)$, using symbols $\rightarrow, \leftrightarrow, \perp, \top$ to enhance readability (we will do so in later examples too).

The observation on as we defined it in Example 1 can be expressed in NNF as $\text{sw_on} \wedge \phi$, but it is not representable in Terms. However, since adding invalid states to on is harmless, we can change our definition of on and use the simpler formula sw_on , which is equivalent in practice — and is a Terms expression. The NNF belief state $x_2 \leftrightarrow \neg \text{sw_on}$ can be expressed in OBDD $_{<}$, with $x_1 < x_2 < y_1 < y_2 < \text{sw_on}$, as $\text{ite}(x_2, \text{ite}(\text{sw_on}, \perp, \top), \text{ite}(\text{sw_on}, \top, \perp))$. The goal of being at $x = 2, y = 2$ can be represented as $x_2 \wedge y_2$. Note that, like on , the sets of states expressed by the last two formulas contain invalid states, but this is harmless. We simply ignore them from now on.

It can be seen that any term is a CNF (with only unit clauses), a DNF (with only one term), and — exact syntax left apart — an OBDD: for instance, the term $x_1 \wedge \neg x_2 \wedge x_3$ can be seen as the OBDD $\text{ite}(x_1, \text{ite}(x_2, \perp, \text{ite}(x_3, \top, \perp)), \perp)$. Moreover, by definition, any CNF, DNF, or OBDD is an NNF formula as well. Hence, slightly abusing the definitions, we will repeatedly use the inclusions $\text{Terms} \subseteq \{\text{CNF}, \text{DNF}, \text{OBDD}_{<}\} \subseteq \text{NNF}$.

5.2 Goal languages

By definition, statewise goals are specified by a subset G of the state space \mathcal{S} . Consequently, for these we will consider the propositional languages defined in Section 5.1.

Such a statewise goal defined by $G \subseteq \mathcal{S}$ can also be seen as the goal of *knowing* that the current state is one in G , which is usually written $\text{K } G$, or $\Box G$ [20]. This generalizes naturally with goals of the form $\neg \text{K } G$, and Boolean combinations of such atoms. We capture this generalization by the logic of knowledge S5.²

Definition 11 (S5). Let X be a set of variables, and let Λ^+, Λ^- be two propositional languages over X . The *S5 language induced by Λ^+, Λ^-* , written $\text{S5}_{\Lambda^+, \Lambda^-}$, is the set of all expressions $\vec{\gamma}$ generated by the grammar $\vec{\gamma} ::= \kappa^+ \mid \kappa^- \mid \vec{\gamma} \vee \vec{\gamma} \mid \vec{\gamma} \wedge \vec{\gamma}$, with κ^+ ranging over all expressions of the form $\text{K } \varphi$ with $\varphi \in \Lambda^+$, and κ^- ranging over all expressions of the form $\neg \text{K } \varphi$ with $\varphi \in \Lambda^-$.

The semantics of an S5 expression is a *set* of belief states, so that a belief state B satisfies a goal expressed by such an expression $\vec{\gamma}$ if $B \in \|\vec{\gamma}\|$ holds.

Definition 12 (S5 semantics). Let X be a set of variables, let Λ^+, Λ^- be two propositional languages, and let $\vec{\gamma}$ be a formula in $\text{S5}_{\Lambda^+, \Lambda^-}$. The *semantics* $\|\vec{\gamma}\|$ of $\vec{\gamma}$ is inductively defined by

$$\begin{aligned} \|\text{K } \varphi\| &:= \{B \subseteq 2^X \mid \forall s \in B : s \models \varphi\} \\ \|\neg \text{K } \varphi\| &:= \{B \subseteq 2^X \mid \exists s \in B : s \not\models \varphi\} \\ \|\vec{\gamma}_1 \vee \vec{\gamma}_2\| &:= \|\vec{\gamma}_1\| \cup \|\vec{\gamma}_2\| \\ \|\vec{\gamma}_1 \wedge \vec{\gamma}_2\| &:= \|\vec{\gamma}_1\| \cap \|\vec{\gamma}_2\| \end{aligned}$$

² The use of two different propositional languages for positive and negative knowledge atoms is motivated by algorithmic efficiency, following [23].

Example 13 (continued from Example 10). The goal of being at $x = x_2$ but not knowing the position along the y axis can be expressed in $S5_{Terms, Terms}$ as $K x_2 \wedge \neg K y_1 \wedge \neg K \neg y_1$, since in this domain, not knowing the value of the variable y_1 implies not knowing the value of the variable y_2 . This formula is satisfied by the belief state B represented by the NNF formula x_2 , since B contains states where y_1 is false (so B is not in $\|K y_1\|$), but also states where it is true (so B is not in $\|K \neg y_1\|$). In contrast, the goal $K x_2 \wedge \neg K y_1$, while also satisfied by B , is satisfied by belief states in which the agent knows its position, such as the one expressed as $x_2 \wedge \neg y_1$.

While negative epistemic goals might seem useless at first sight, they can be used to model goals like “I want to check our common bank account without learning how much my partner spent on my Christmas present”. They can also be used to model weak planning objectives, as reaching $\neg K \neg \varphi$ amounts to reaching a situation in which the goal specified by φ is *possibly* achieved.

5.3 Action languages

An action language is a language for representing a class of ontic actions. Hence their semantics is a mapping from states to sets of states.

PDDL and STRIPS The first action language which we consider is a nondeterministic version of (ground) PDDL. Observe that we take the STRIPS-like semantics, in which if $+p$ and $-p$ are both effects of an action in a state, then the semantics is like if $+p$ alone was the effect.

Definition 14 (PDDL). Let Λ^P, Λ^C be two propositional languages over a set of fluents F . The action language $PDDL_{\Lambda^P, \Lambda^C}$ is the set of all expressions of the form $\alpha = \langle \pi, \eta \rangle$, with $\pi \in \Lambda^P$ and η generated from the grammar

$$\eta ::= +x \mid -x \mid \eta \cup \eta \mid \eta \& \eta \mid \chi \triangleright \eta$$

with $x \in F$ and $\chi \in \Lambda^C$. The expression π is called the *precondition* of α , the expression η is called its *effect*, and each expression χ in the effect is called a *condition*.

The semantics is given by the following definition.

Definition 15 (PDDL semantics). Let Λ^P, Λ^C be two propositional languages over a set of fluents F , and let $\alpha := \langle \pi, \eta \rangle$ be an expression in $PDDL_{\Lambda^P, \Lambda^C}$. The semantics of α is defined by $\|\alpha\|(s) := \emptyset$ if s does not satisfy π , and otherwise by $\|\alpha\|(s) := \{(s \setminus E^-) \cup E^+ \mid (E^+, E^-) \in E(\eta, s)\}$, with

$$\begin{aligned} E(+x, s) &:= \{(\{x\}, \emptyset)\} \\ E(-x, s) &:= \{(\emptyset, \{x\})\} \\ E(\eta_1 \cup \eta_2, s) &:= E(\eta_1, s) \cup E(\eta_2, s) \\ E(\eta_1 \& \eta_2, s) &:= \{(E_1^+ \cup E_2^+, E_1^- \cup E_2^-) \mid (E_i^+, E_i^-) \in E(\eta_i, s) \text{ for } i = 1, 2\} \\ E(\chi \triangleright \eta, s) &:= E(\eta, s) \text{ if } s \text{ satisfies } \chi, \text{ otherwise } \{(\emptyset, \emptyset)\} \end{aligned}$$

Observe that $\eta_1 \cup \eta_2$, $\eta_1 \& \eta_2$, $\chi \triangleright \eta$ are just synonyms for the constructions (one of $\eta_1 \eta_2$), (and $\eta_1 \eta_2$), (when $\chi \eta$) of usual PDDL [4].

Since PDDL allows arbitrary nesting of the constructs, we also consider a widely used syntactic restriction.

Definition 16 (conditional STRIPS). Let Λ^P, Λ^C be two propositional languages over a set of fluents F . The action language $\text{CSTRIPS}_{\Lambda^P, \Lambda^C}$ is the restriction of $\text{PDDL}_{\Lambda^P, \Lambda^C}$ to expressions $\langle \pi, \eta \rangle$ with η of the form

$$\&_{i \in I} (\chi_i \triangleright \bigcup_{j \in J_i} (\pm x_{i,j}^1 \& \cdots \& \pm x_{i,j}^{k_{i,j}}))$$

where I and all the J_i are finite sets of indices and for all i, j, k , $\pm x_{i,j}^k$ is either $+x$ or $-x$ for some $x \in F$.

Example 17 (continued from Example 13). The action down can be written in $\text{PDDL}_{\text{Terms}, \text{Terms}}$ as $\langle \pi, \eta \rangle$, with $\pi = \text{sw_on}$ and $\eta = -y_2 \& +y_1$. The action right can be written in $\text{CSTRIPS}_{\text{Terms}, \text{Terms}}$ as $\langle \pi, \eta \rangle$, with $\pi = \top$ and

$$\eta = (\top \triangleright ((+y_1 \& -y_2) \cup (-y_1 \& +y_2))) \& (x_1 \triangleright (+x_2 \& -x_1))$$

For both $\text{PDDL}_{\Lambda^P, \Lambda^C}$ and $\text{CSTRIPS}_{\Lambda^P, \Lambda^C}$, we will sometimes consider the restriction that there is no precondition (equivalently, that the precondition is tautological), and/or that there is no construct $\chi \triangleright \eta$ (equivalently, that all such constructs are of the form $\top \triangleright \eta$). We will then write \top in place of Λ^P and/or Λ^C ; for instance, $\text{CSTRIPS}_{\top, \Lambda^C}$ denotes the CSTRIPS language restricted to tautological preconditions and to conditions in Λ^C .

Deterministic Restrictions When considering actions, a very natural and common restriction is *determinism*. Deterministic actions map each state which satisfies their precondition to a single successor.

Definition 18 (deterministic PDDL, STRIPS). Let Λ^P, Λ^C be two propositional languages over a set of fluents F . The action language $\text{PDDL-Det}_{\Lambda^P, \Lambda^C}$ (resp. $\text{CSTRIPS-Det}_{\Lambda^P, \Lambda^C}$) is the restriction of $\text{PDDL}_{\Lambda^P, \Lambda^C}$ (resp. $\text{CSTRIPS}_{\Lambda^P, \Lambda^C}$) to expressions in which the choice operator \cup does not occur.

Observe that we define determinism *syntactically*, so that an expression in which the choice operator does occur, but which in effect is deterministic (like $\alpha := +x \cup +x$) does *not* fall in the scope of Definition 18. The reason is that we want languages to be defined syntactically, and determining whether a given PDDL or STRIPS action is “semantically” deterministic is hard in general (this involves checking the satisfiability of conditions, among others).

Also observe that $\text{CSTRIPS-Det}_{\text{Terms}, \top}$ is essentially the historical (deterministic, unconditional) STRIPS language. The representation of action down in Example 17 is an example of an expression in this language.

Action Theories We will finally consider a number of action languages built on top of propositional languages as follows. Given a set of fluents F , we write F' for the set of variables (assumed to be fresh) $\{x' \mid x \in F\}$. Given an assignment s to F , we write s' — or $(s)'$ in case of ambiguity — for the assignment to F' defined by $\forall x' \in F' : s'(x') := s(x)$. The variables in F' are used to express the values of the fluents *after* the action took place.

Definition 19 (action theories). Let F be a set of fluents, and let Λ be a propositional language. The action language of *action theories* over Λ and F , written $\text{AT}(\Lambda)$, is the set of all expressions α of Λ satisfying $\forall(\alpha) \subseteq F \cup F'$. The semantics $\|\alpha\|$ of $\alpha \in \text{AT}(\Lambda)$ is defined by $\forall s_1 \in 2^F : \|\alpha\|(s_1) = \{s_2 \in 2^{F'} \mid s_1 \cup s_2' \models \alpha\}$.

For conciseness, we write NNFAT (resp. CNFAT, DNFAT, OBDDAT_<) instead of AT(NNF) (resp. AT(CNF), AT(DNF), AT(OBDD_<)).

Example 20 (continued from Example 17). Still assuming that the initial belief state only contains valid states, the action down in Example 17 can be expressed in NNFAT as

$$\text{sw_on} \wedge (y'_1 \wedge \neg y'_2) \wedge (x'_1 \leftrightarrow x_1) \wedge (x'_2 \leftrightarrow x_2) \wedge (\text{sw_on}' \leftrightarrow \text{sw_on})$$

Notice that the first conjunct implements the precondition, the second conjunct implements fluent changes, and the last three make sure that the other fluents keep their value.

Observe that in action theories, we need to specify explicitly the persistency of fluents. For instance, if we remove the subexpression $(\text{sw_on}' \leftrightarrow \text{sw_on})$ from the expression in Example 20, we get an action after which the value of sw_on is arbitrary.

For the action language OBDDAT_<, since the binary decision diagrams involve variables both in F and in F' , an important question is how variables in F are ordered by $<$ with respect to variables in F' . As we will see, this indeed affects the complexity of some queries and transformations. First, throughout the paper we assume that $<$ is such that variables are ordered the same in F and in F' , in the sense that for all $x_1, x_2 \in F$, $x_1 < x_2$ holds if and only if $x'_1 < x'_2$ holds. For the interleaving of variables in F and in F' , we consider three situations:

- all variables in F are ordered before all variables in F' , written $F < F'$;
- all variables in F are ordered after all variables in F' , written $F' < F$;
- for each variable $x \in F$, x is ordered just before x' ; precisely, if $F := \{x_1, \dots, x_n\}$ is ordered by $x_{i_1} < x_{i_2} < \dots < x_{i_n}$, then $F \cup F'$ is ordered by $x_{i_1} < x'_{i_1} < x_{i_2} < x'_{i_2} < \dots < x_{i_n} < x'_{i_n}$; this situation we write $F \parallel_{<} F'$.³

Admittedly, these situations do not cover all possible orderings of $F \cup F'$; they however cover the natural cases where F and F' have the same ordering, so that the order of the fluents does not depend on the timestep (before or after the action took place).

Finally, note that we do *not* consider the deterministic versions of action theories, as there is no obvious *syntactic* criterion ensuring the determinism of an action theory and, like for PDDL, determining whether an action theory is “semantically” deterministic is typically hard.

5.4 Size of expressions

For complexity analyses, it is crucial to define the *size* of expressions. As is standard in the knowledge compilation literature, we will assume that for all propositional languages, any subformula of a propositional formula φ which occurs several times in φ is represented only once (and pointed to as much as necessary). While this does not change (up to a polynomial) the size of CNFs, DNFs, and terms, this is crucial for the complexity of queries and transformations on OBDD_<, and this *a priori* matters for NNF.

Hence we define the *size* $|\varphi|$ of an expression φ in any propositional language to be the number of symbols in this representation, and we extend this definition to action theories in $\text{AT}(\Lambda)$ and to general goals in $\text{S5}_{\Lambda^+, \Lambda^-}$ for all propositional languages Λ , Λ^+ , Λ^- .

For $\text{PDDL}_{\Lambda^P, \Lambda^C}$ and $\text{CSTRIPS}_{\Lambda^P, \Lambda^C}$, we also assume this representation for preconditions $\pi \in \Lambda^P$ and conditions $\chi \in \Lambda^C$. On the other hand, we assume that the rest of the structure

³ The complexity results would be exactly the same for $x'_{i_1} < x_{i_1} < x'_{i_2} < x_{i_2} < \dots < x'_{i_n} < x_{i_n}$ instead.

in a PDDL expression is represented by a tree, because we leave the complexity of some problems open with the circuit representation. We define the size $|\alpha|$ of an action expression α in $\text{PDDL}_{\Lambda^P, \Lambda^C}$ accordingly, counting each occurrence of each expression; we will however point out where the results depend on this choice. Observe that this assumption does not matter for $\text{CSTRIPS}_{\Lambda^P, \Lambda^C}$ (up to a polynomial), since the expressions there have bounded nesting.

6 Representations of belief states by propositional formulas

In this section we give the complexity results about queries and transformations, for belief states represented in a propositional language. We study two variants of such representations, depending on whether we allow auxiliary variables in the expressions. A number of these results follow from previous studies on knowledge compilation, especially from the results by Darwiche and Marquis [3], but others are new since they concern the interplay between different propositional languages, or because they concern *non-logical* action languages (PDDL and CSTRIPS).

6.1 Belief state propositional languages

Since, by definition, a belief state on a set of fluents F is a subset of 2^F , it is natural to represent a belief state B by a propositional expression β , for instance in NNF, with $V(\beta) \subseteq F$. This is what we call a *plain propositional representation* of B .

However, in the context of planning, it is also natural to consider representations in which auxiliary variables are allowed, so that a belief state $B \subseteq 2^F$ is represented by a propositional expression β with $V(\beta) \subseteq F \cup Y$, for some set of *auxiliary variables* Y . Then the semantics is that β represents the belief state $\exists Y \cdot \|\beta\|_{F \cup Y}$, where for a set of assignments $S \subseteq 2^{F \cup Y}$, $\exists Y \cdot S$ denotes the set of assignments obtained by projecting out the variables in Y , an operation also known as eliminating or *\exists -forgetting* Y . Formally, $\exists Y \cdot B$ is defined to be $\{s \in 2^F \mid \exists s_Y \in 2^Y : s \cup s_Y \in B\}$.

A typical use of such auxiliary variables in planning is for progressing a belief state by an action. Suppose for instance that we are progressing a belief state B represented by $\beta := (x_1 \vee x_2)$ by an action a represented by $\alpha := (x'_1 \leftrightarrow \neg x_1) \wedge (x'_2 \leftrightarrow x_2)$. The action α switches the value of x_1 but leaves x_2 unchanged. Then computing a plain representation of $B \oplus a$ amounts to computing $\beta \wedge \alpha$, then projecting out the variables x_1, x_2 but not x'_1, x'_2 , and finally renaming x'_1, x'_2 to x_1, x_2 . This indeed yields $(\neg x_1 \vee x_2)$, as desired. However, since existential forgetting is typically a costly operation, incurring a possible blowup in size, it can be better to represent $B \oplus a$ by $\beta' := (x_1^- \vee x_2^-) \wedge (x_1 \leftrightarrow \neg x_1^-) \wedge (x_2 \leftrightarrow x_2^-)$. Such a representation uses only conjunction and renaming, and it is easy to see that the semantics $\|\beta'\| = \exists\{x_1^-, x_2^-\} \cdot \|\beta'\|_{\{x_1^-, x_2^-, x_1, x_2\}}$ is the same as $\|\neg x_1 \vee x_2\|$.

This precise use of auxiliary variables amounts to keeping track of the history which yielded the represented belief state; it can be seen as projecting out in a lazy fashion. Beyond this, we consider general representations, using an arbitrary set of auxiliary variables, which we call *existential propositional representations*.

All in all, for each propositional language Λ , we define two belief state languages.

Definition 21 (plain belief state language). Let F be a set of fluents, and let Λ be a propositional language. The *plain belief state (propositional) language induced by Λ* , written $\text{BS}(\Lambda)$,

Table 3 Complexity of $\text{APPLIC}(\Lambda^B, \Lambda^A)$ for representations of belief states by propositional formulas

$\Lambda^B =$	$\Lambda^A =$	NNFAT or CNFAT	DNFAT	OBDDAT _{<}	$\text{P/C}(-\text{Det})_{\Lambda^P, \Lambda^C}$
$\text{BS}_{\exists}(\Lambda)$ or $\text{BS}(\Lambda)$, $\Lambda \in \{\text{NNF}, \text{CNF}\}$		$\Pi_2\text{P-c}$ (m: 26)	coNP-c (m: 26)	coNP-c (m: 26)	P for $\Lambda^P = \top$ (obvious); coNP-c for $\Lambda^P \supseteq \text{Terms}$ (m: 26, h: 28)
$\text{BS}_{\exists}(\Lambda)$ or $\text{BS}(\Lambda)$, $\Lambda \in \{\text{DNF}, \text{OBDD}_{<}, \text{Terms}\}$		$\Pi_2\text{P-c}$ (h: 27)	coNP-c (h: 28 + 25)	coNP-c for $F' < F$ or $F \parallel < F'$ (h: 29); P for $F < F'$ (m: 30)	coNP-c for $\Lambda^P \supseteq \text{DNF}$ (h: 28); P for $\Lambda^P \subseteq \text{CNF}$ or $\Lambda^P \subseteq \text{OBDD}_{<}$ (m: 31)

$\text{P/C}(-\text{Det})_{\Lambda^P, \Lambda^C}$ stands for $\text{PDDL}_{\Lambda^P, \Lambda^C}$, $\text{CSTRIPS}_{\Lambda^P, \Lambda^C}$, $\text{PDDL-Det}_{\Lambda^P, \Lambda^C}$, or $\text{CSTRIPS-Det}_{\Lambda^P, \Lambda^C}$. “h” stands for “(proof of) hardness” and “m” for “(proof of) membership”

is defined to be the set of propositional expressions $\beta \in \Lambda$ with $V(\beta) \subseteq F$, with the semantics defined by $\|\beta\| := \|\beta\|_F$.

Definition 22 (existential belief state language). Let F be a set of fluents, and let Λ be a propositional language. The *existential belief state (propositional) language induced by Λ* , written $\text{BS}_{\exists}(\Lambda)$, is defined to be the set of propositional expressions $\beta \in \Lambda$ with $V(\beta) \subseteq F \cup Y$ for some set of variables Y disjoint from F . The semantics of such an expression is defined by $\|\beta\| := \exists Y \cdot \|\beta\|_{F \cup Y}$.

For the language $\text{BS}_{\exists}(\text{OBDD}_{<})$, we will sometimes need to distinguish two situations, depending on whether the ordering $<$ orders the variables in Y after or before the variables in F , denoted by $F < Y$ and $Y < F$, respectively. This does not cover all possible orderings of $F \cup Y$, but it covers the most natural situations, and additionally abstracts away from the identity of the variables in F or in Y .

6.2 Complexity of queries

We start by studying the complexity of queries. The results are summarized in Tables 3, 4, 5, 6 and 7; the numbers there point to the propositions where the result is proved, with “h” standing for “(proof of) hardness” and “m” for “(proof of) membership”.

Table 4 Complexity of $\text{FAIR}(\Lambda^B, \Lambda^O)$ for representations of belief states by propositional formulas

$\Lambda^B =$	$\Lambda^O =$	NNF/CNF	DNF	OBDD _{<}	Terms
$\text{BS}_{\exists}(\Lambda)$ or $\text{BS}(\Lambda)$, $\Lambda \in \{\text{NNF}, \text{CNF}\}$		NP-c (m: 32)	NP-c	NP-c	NP-c (h: 33)
$\text{BS}_{\exists}(\Lambda)$ or $\text{BS}(\Lambda)$, $\Lambda \in \{\text{DNF}, \text{OBDD}_{<}, \text{Terms}\}$		NP-c (h: 33)	P (m: 34)	P (m: 34)	P

Table 5 Complexity of $\text{SAT}_S(\Lambda^B, \Lambda^G)$ for representations of belief states by propositional formulas. The results follow from Table 3 and Lemma 35

Λ^B =	Λ^G =	NNF/DNF	CNF	OBDD _{<}	Terms
$\text{BS}_{\exists}(\Lambda)$ or $\text{BS}(\Lambda)$, $\Lambda \in \{\text{NNF}, \text{CNF}\}$		coNP-c	coNP-c	coNP-c	coNP-c
$\text{BS}_{\exists}(\Lambda)$ or $\text{BS}(\Lambda)$, $\Lambda \in \{\text{DNF}, \text{OBDD}_{<}, \text{Terms}\}$		coNP-c	P	P	P

Table 6 Complexity of $\text{SAT}(\Lambda^B, S5_{\Lambda^+, \Lambda^-})$ for representations of belief states by propositional formulas

Λ^B =	$\Lambda^+ =$ $\Lambda^- =$	NNF/DNF	C/O/T	NNF/DNF	C/O/T
		NNF/DNF	NNF/DNF	C/O/T	C/O/T
$\text{BS}_{\exists}(\Lambda)$ or $\text{BS}(\Lambda)$, $\Lambda \in \{\text{NNF}, \text{CNF}\}$		$\Theta_2\text{P-c}$ (m: 36)	$\Theta_2\text{P-c}$	$\Theta_2\text{P-c}$	$\Theta_2\text{P-c}$ (h: 37)
$\text{BS}_{\exists}(\Lambda)$ or $\text{BS}(\Lambda)$, $\Lambda \in \{\text{DNF}, \text{OBDD}_{<}, \text{Terms}\}$		$\Theta_2\text{P-c}$ (h: 37)	NP-c (m, h: 38)	coNP-c (m, h: 38)	P (m: 39)

A column header means that the results in the column hold for any combination of one of the positive languages with one of the negative languages. C/O/T stands for CNF/OBDD_</Terms. The (epistemic) goal language $S5_{\Lambda^+, \Lambda^-}$ can also be used as a precondition language Λ^P for APPLIC of $\text{PDDL}_{\Lambda^P, \Lambda^C}$, complementing the results in Table 3 for epistemic preconditions

Table 7 Complexity of $\text{EQUIV}(\Lambda^B)$ for representations of belief states by propositional expressions

Λ^B	complexity
$\text{BS}_{\exists}(\text{NNF})$	$\Pi_2\text{P-c}$ (m: 40)
$\text{BS}(\text{NNF})$	coNP-c (m: 43)
$\text{BS}_{\exists}(\text{CNF})$	$\Pi_2\text{P-c}$ (h: 41)
$\text{BS}(\text{CNF})$	coNP-c (h: 43)
$\text{BS}_{\exists}(\text{DNF})$ or $\text{BS}(\text{DNF})$	coNP-c (m: 40, h: 43)
$\text{BS}_{\exists}(\text{OBDD}_{<})$	coNP-c for $Y_1 \cup Y_2 < F$ (m: 40, h: 42); P for $F < Y_1 \cup Y_2$ (m: 43 + 23)
$\text{BS}(\text{OBDD}_{<})$	P (m: 43)
$\text{BS}_{\exists}(\text{Terms})$ or $\text{BS}(\text{Terms})$	P (m: 44)

In the tables, we tacitly use a number of reductions between problems, which can be easily derived from the inclusion relations between languages. Common to all queries is a belief state language Λ^B as a parameter. Then if Λ_1^B, Λ_2^B are two belief state languages with $\Lambda_1^B \subseteq \Lambda_2^B$, then for all action languages Λ^A , the problem $\text{APPLIC}(\Lambda_1^B, \Lambda^A)$ is trivially polynomial-time reducible to the problem $\text{APPLIC}(\Lambda_2^B, \Lambda^A)$, and similarly for queries FAIR , SAT_S , SAT , and Equiv . We will also tacitly use similar reductions derived from inclusion relations between the other languages (Λ^A, Λ^O , etc.). In tables, results derived from such relations are written without any reference to a proposition.

Recall from Section 5.1 the inclusions $\text{Terms} \subseteq \{\text{CNF}, \text{DNF}, \text{OBDD}_{<}\} \subseteq \text{NNF}$. We will also use the obvious inclusion $\text{BS}(\Lambda) \subseteq \text{BS}_{\exists}(\Lambda)$ (for all propositional languages Λ). Moreover, we will use the polynomial-time reductions induced by the following translations.

Lemma 23 *Let Λ be either DNF, or $\text{OBDD}_{<}$ with $F < Y$. Then given an expression β in $\text{BS}_{\exists}(\Lambda)$, one can compute in polynomial-time an expression β' in $\text{BS}(\Lambda)$ with $\|\beta'\| = \|\beta\|$.*

Proof This follows from the fact that \exists -forgetting can be done in polynomial time for both languages. This is shown by [3, Table 7] for DNF. For $\text{OBDD}_{<}$ with $F < Y$, we can compute β' as follows: given $\beta \in \text{OBDD}_{<}$, replace bottom-up each subexpression $\text{ite}(y, \varphi_1, \varphi_2)$, for $y \in Y$, by \top (resp. \perp) if $\varphi_1 = \top$ or $\varphi_2 = \top$ holds (resp. if both $\varphi_1 = \perp$ and $\varphi_2 = \perp$ hold). It is easy to see that β' can be built in polynomial time, satisfies $V(\beta') = F$, and satisfies $\|\beta'\| = \|\beta\|$. \square

Finally, recall that we define the queries to take as input a nonempty belief state. The following lemma will allow us to ignore this when giving reductions to queries (except for SAT).

Lemma 24 *For all belief state languages Λ^B of the form $\text{BS}(\Lambda)$ or $\text{BS}_{\exists}(\Lambda)$ with $\Lambda \in \{\text{NNF}, \text{CNF}, \text{DNF}, \text{OBDD}_{<}, \text{Terms}\}$, there is a polynomial-time reduction from the problem with possibly empty belief states as input, to the problem with nonempty belief states, for $\text{APPLIC}(\Lambda^B, \Lambda^A)$, $\text{FAIR}(\Lambda^B, \Lambda^O)$, $\text{SAT}_S(\Lambda^B, \Lambda^G)$, and $\text{Equiv}(\Lambda^B)$, for all languages $\Lambda^A, \Lambda^O, \Lambda^G$ considered in this paper.*

Proof First observe that the result holds vacuously for $\Lambda = \text{Terms}$, since we define terms to be consistent.

For the other languages, let $x \notin F$ be a fresh variable. First, it is easy to see that given an expression $\beta \in \Lambda^B$, an expression β' in Λ^B with $\|\beta'\| = \|\beta \vee x\|$ can be computed in polynomial time, and $\|\beta \vee x\|$ is nonempty. Now it can be seen that:

- for any ω in any observation language considered, $\|\omega\|$ is fair at $\|\beta\|$ if and only if $\|\omega \wedge \neg x\|$ is fair at $\|\beta'\|$; moreover, all observation languages allow the computation of an expression for $\omega \wedge \neg x$ in polynomial time;
- for any γ in any $\Lambda^G \in \{\text{NNF}, \text{CNF}, \text{DNF}, \text{OBDD}_{<}\}$, $\|\beta\| \subseteq \|\gamma\|$ if and only if $\|\beta'\| \subseteq \|\gamma \vee x\|$, and all these languages allow the computation of an expression for $\gamma \vee x$ in polynomial time;
- for any γ in $\Lambda^G = \text{Terms}$, $\|\beta\| \subseteq \|\gamma\|$ if and only if $\|\beta \vee \gamma\| \subseteq \|\gamma\|$, $\beta \vee \gamma$ is consistent because γ is in Terms , and all languages $\Lambda \in \{\text{NNF}, \text{CNF}, \text{DNF}, \text{OBDD}_{<}\}$ allow one to compute an expression for $\beta \vee \gamma$ in polynomial time.
- a similar reasoning as for SAT_S shows the result for applicability.

Finally, for equivalence, it is again easy to see that $\|\beta_1\| = \|\beta_2\|$ holds if and only if $\|\beta_1 \vee x\| = \|\beta_2 \vee x\|$ holds. \square

Applicability For APPLIC, the results are summarized in Table 3. Let us emphasize that, as the notation suggests, when considering the belief state language $\text{OBDD}_{<}$ and the action language $\text{OBDDAT}_{<}$, we assume that the ordering on $F \cup F'$ (for actions) extends that on F (for belief states). This is not a real restriction since, in a typical implementation, one uses the same ordering for all decision diagrams so as to maximize the benefits of memory sharing and caching.

As a complement to the results given in Table 3, note that deciding applicability of an action expression in $\text{PDDL}_{\Lambda^P, \Lambda^C}$ amounts to checking the satisfaction of the precondition expression, in Λ^P . It follows that these results can be complemented by the results about goal satisfaction. For instance, using Table 6 one can derive what would be the complexity of checking applicability of a PDDL expression with an epistemic precondition in $\text{SS}_{\Lambda^+, \Lambda^-}$.

We now turn to the results; we will use the following lemma.

Lemma 25 *For all belief state languages Λ^B and all propositional languages Λ^P, Λ^C , the problem $\text{APPLIC}(\Lambda^B, \text{PDDL}_{\Lambda^P, \Lambda^C})$ is polynomial-time reducible to the problem $\text{APPLIC}(\Lambda^B, \text{AT}(\Lambda^P))$.*

Proof An action $\alpha = \langle \pi, \eta \rangle \in \text{PDDL}_{\Lambda^P, \Lambda^C}$ is applicable at a belief state B if and only if all states $s \in B$ satisfy π . Now π is in Λ^P by definition, so it is a special case of an expression in $\text{AT}(\Lambda^P)$ (one not mentioning any primed variable), and as such it is applicable at B if and only if all states $s \in B$ satisfy π . \square

Proposition 26 *The problem $\text{APPLIC}(\text{BS}_{\exists}(\text{NNF}), \text{NNFAT})$ is in $\Pi_2\text{P}$, and the problems $\text{APPLIC}(\text{BS}_{\exists}(\text{NNF}), \text{DNFAT})$, $\text{APPLIC}(\text{BS}_{\exists}(\text{NNF}), \text{OBDDAT}_{<})$, and $\text{APPLIC}(\text{BS}_{\exists}(\text{NNF}), \text{PDDL}_{\text{NNF}, \text{NNF}})$ are in coNP .*

Proof Given $\beta \in \text{BS}_{\exists}(\text{NNF})$ with $V(\beta) \subseteq F \cup Y$ and α an expression in any action language, the problem amounts to deciding the validity of

$$\forall s \in 2^F : \forall s_Y \in 2^Y : s \cup s_Y \models \beta \implies (\exists s' \in 2^F : s' \in \|\alpha\|(s))$$

Deciding $s \cup s_Y \models \beta$ is in P . For $\alpha \in \text{NNFAT}$, deciding $\exists s' \in 2^F : s' \in \|\alpha\|(s)$ is in NP , hence the problem is in $\Pi_2\text{P}$. Now for DNFAT and $\text{OBDDAT}_{<}$, deciding $\exists s' \in 2^F : s' \in \|\alpha\|(s)$ amounts to deciding whether α is satisfiable after conditioning by s , which is in P ; for $\text{PDDL}_{\text{NNF}, \text{NNF}}$, it amounts to deciding whether s satisfies the precondition of the action, again a question in P . Hence in all these cases, the problem is in coNP . \square

Proposition 27 *The problem $\text{APPLIC}(\text{BS}(\text{Terms}), \text{CNFAT})$ is $\Pi_2\text{P-hard}$.*

Proof Let $\forall V_1 : \exists V_2 : \varphi$ be a QBF formula with φ in CNF. Let $F := V_1 \cup V_2$, and define α_φ to be the action expression in CNFAT obtained from φ by renaming each $x \in V_2$ to x' . Then it is easy to see that the QBF is valid if and only if α_φ is applicable at the complete belief state 2^F , which has a polynomial-size expression as the empty term. This gives a polynomial-time reduction from the problem of deciding whether a $\forall\exists$ -QBF is valid, which is $\Pi_2\text{P-hard}$ [as a corollary of Th. 17.10 of 24, p. 428]. \square

Proposition 28 *The problems $\text{APPLIC}(\text{BS}(\text{CNF}), \text{CSTRIPS-Det}_{\text{Terms}, \top})$ and $\text{APPLIC}(\text{BS}(\text{Terms}), \text{CSTRIPS-Det}_{\text{DNF}, \top})$ are coNP-hard .*

Proof Deciding applicability of $\alpha = \langle \pi, \eta \rangle$ at a belief state β amounts to deciding whether β entails π , which is coNP-hard in both cases. Indeed, deciding whether a CNF $\gamma_1 \wedge \dots \wedge \gamma_n$ is inconsistent is equivalent to introducing a fresh variable x and deciding whether the consistent CNF $(\gamma_1 \vee x) \wedge \dots \wedge (\gamma_n \vee x)$ entails x , which is a consistent term; and deciding whether a DNF is valid is equivalent to deciding whether it is entailed by \top (the empty term). Note that η does not matter in the proof, so assuming it is deterministic yields a stronger statement at no cost. \square

Proposition 29 *Assume $F' < F$ or $F \parallel < F'$. Then the problem $\text{APPLIC}(\text{BS}(\text{Terms}), \text{OBDDAT}_{<})$ is coNP-hard.*

Proof We adapt a construction by [3, proof of Prop. 5.1, p. 258] to give a reduction from the problem of deciding whether a given DNF formula is tautological. Let $\varphi := \bigwedge_{i=1}^m \tau_i$ be a DNF formula over a set of variables $V(\varphi) := \{x_1, \dots, x_n\}$. We define the set of fluents F to be $\{x_1, \dots, x_n, x_{n+1}, \dots, x_{n+m}\}$, and the ordering $<$ on $F \cup F'$ to extend $x_{n+1} < x_{n+2} < \dots < x_{n+m} < x_1 < x_2 < \dots < x_n$; clearly, this is possible while respecting either $F' < F$ or $F \parallel < F'$.

Now we define the propositional expression α by $\alpha := \alpha_1$, with $\alpha_{m+1} := \perp$ and for $i = 1, \dots, m$, $\alpha_i := \text{ite}(x'_{n+i}, \tau_i, \alpha_{i+1})$ (where, abusing notation, we identify τ_i with an expression in $\text{OBDDAT}_{<}$). Clearly, this expression is in $\text{OBDDAT}_{<}$ (whatever the assumed ordering $<$), and it can be constructed in polynomial time.

We claim that $\|\alpha\|$ is applicable at the belief state represented by \top (empty term) if and only if φ is tautological. Assume first that $\|\alpha\|$ is applicable at $\|\top\| = 2^F$; then by definition of applicability, for all assignments $s \in 2^F$, there is an assignment $s' \in 2^{F'}$ such that $s \cup s' \models \alpha$; then it follows from the construction of α that for all assignments $s \in 2^F$, there is an i such that s satisfies τ_i , and hence s satisfies φ ; so φ is tautological. Conversely, if $\|\alpha\|$ is not applicable at some $s \in 2^F$, then for all assignments $s' \in 2^{F'}$ we have $s \cup s' \not\models \alpha$; hence for all $i = 1, \dots, m$, this holds in particular for any assignment s'_i which makes x'_{n+i} true, and x'_{n+j} false for all $j \neq i$; it is easily seen that this implies $s \not\models \tau_i$ for all i , and thus in the end $s \not\models \varphi$, so that φ is not tautological. \square

Proposition 30 *Assume $F < F'$. Then the problem $\text{APPLIC}(\text{BS}_{\exists}(\Lambda), \text{OBDDAT}_{<})$ is in P for $\Lambda \in \{\text{DNF}, \text{OBDD}_{<}\}$.*

Proof Using the same construction as in Lemma 23, given $\alpha \in \text{OBDDAT}_{<}$ with $F < F'$, we can compute in polynomial time an expression $\pi \in \text{OBDDAT}_{<}$ with $V(\pi) \subseteq F$ and $\|\pi\| = \exists F' \cdot \|\alpha\|$, that is, π represents the (implicit) precondition of α . It remains to decide whether β entails π , a polynomial-time problem for $\beta \in \text{BS}_{\exists}(\Lambda)$ with $\Lambda \in \{\text{OBDD}_{<}, \text{DNF}\}$: indeed, deciding whether an OBDD entails another OBDD with the same ordering is polynomial-time [3, “sentential entailment” query], and for DNF we can decide whether each term in β (which we see as an OBDD) entails π ; and for coping with auxiliary variables, we simply observe that $\exists Y \cdot \beta$ entails π if and only if β entails π , since we have $V(\pi) \cap Y = \emptyset$. \square

Proposition 31 *The problem $\text{APPLIC}(\text{BS}_{\exists}(\Lambda), \text{PDDL}_{\Lambda^P, \text{NNF}})$ is in P for $\Lambda \in \{\text{DNF}, \text{OBDD}_{<}\}$ and $\Lambda^P \in \{\text{CNF}, \text{OBDD}_{<}\}$.*

Proof This follows directly from the fact that deciding whether a formula in Λ entails a formula in Λ^P is in P for all four combinations. Indeed, we just saw the two cases when $\Lambda^P = \text{OBDD}_{<}$ (in the proof of Prop. 30); moreover, deciding whether a formula ϕ entails a formula ψ in CNF is equivalent to deciding whether ϕ entails each clause of ψ , which is polynomial-time if ϕ is in DNF or $\text{OBDD}_{<}$ [3, “clausal entailment” query]. \square

Fairness For fairness, the results are summarized in Table 4.

Proposition 32 *The problem $\text{FAIR}(\text{BS}_{\exists}(\text{NNF}), \text{NNF})$ is in NP.*

Proof Define a witness to be an assignment to all variables in the belief state expression (including the auxiliary variables) and the observation expression, which satisfies both. Such a witness is clearly of polynomial size, and can be verified in polynomial time. \square

Proposition 33 *The problems $\text{FAIR}(\text{BS}(\text{CNF}), \text{Terms})$ and $\text{FAIR}(\text{BS}(\text{Terms}), \text{CNF})$ are NP-hard.*

Proof Deciding whether a CNF formula φ is satisfiable can be reduced to deciding whether the observation $\omega := \top$ is fair to the belief state $\beta := \varphi$, or, symmetrically, to deciding whether $\omega := \varphi$ is fair to $\beta := \top$. \square

Proposition 34 *The problem $\text{FAIR}(\text{BS}_{\exists}(\Lambda), \Lambda^{\text{O}})$ is in P for $\Lambda, \Lambda^{\text{O}} \in \{\text{DNF}, \text{OBDD}_{<}\}$.*

Proof Deciding whether ω is fair to β amounts to deciding whether $\beta \wedge \omega$ is satisfiable; note that this holds even with auxiliary variables in β , since for all formulas φ , $\exists Y : \varphi$ is satisfiable if and only if $\varphi \wedge x$ is satisfiable. For $\Lambda^{\text{B}} = \Lambda^{\text{O}} = \text{DNF}$, this can be decided by distributing \wedge to obtain a DNF (of quadratic size), the satisfiability of which can be decided in linear time. For $\Lambda^{\text{B}} = \Lambda^{\text{O}} = \text{OBDD}_{<}$, again an OBDD for $\beta \wedge \omega$ can be computed in quadratic time, and its satisfiability decided in linear time. Finally, assume by symmetry $\Lambda^{\text{B}} = \text{DNF}$ and $\Lambda^{\text{O}} = \text{OBDD}_{<}$. Then $\beta \wedge \omega$ is satisfiable if and only if there is a term τ in β such that $\tau \wedge \omega$ is satisfiable, which again can be decided in polynomial time. \square

Statewise Goal Satisfaction For clarity, we give the results in Table 5, but all of them can be read from Table 3 in virtue of the following lemma.

Lemma 35 *For all belief state languages Λ^{B} considered, and for all propositional languages Λ^{G} , the problems $\text{SAT}_{\mathcal{S}}(\Lambda^{\text{B}}, \Lambda^{\text{G}})$ and $\text{APPLIC}(\Lambda^{\text{B}}, \text{PDDL}_{\Lambda^{\text{G}}, \top})$ are polynomial-time reducible to each other.*

Proof By definition of $\text{SAT}_{\mathcal{S}}$, a belief state expression $\beta \in \Lambda^{\text{B}}$ satisfies a statewise goal $\gamma \in \Lambda^{\text{G}}$ over F if and only if for all assignments $s \in \|\beta\|$, s satisfies γ . By definition of APPLIC , this is equivalent to the action expression $\langle \gamma, \eta \rangle \in \text{PDDL}_{\Lambda^{\text{G}}, \top}$ being applicable at β for an arbitrary effect η . \square

General Goal Satisfaction For general goal satisfaction, the results are summarized in Table 6.

Proposition 36 *The problem $\text{SAT}(\text{BS}_{\exists}(\text{NNF}), \text{S5}_{\text{NNF}, \text{NNF}})$ is in $\Theta_2\text{P}$.*

Proof Deciding whether β satisfies \vec{G} can be done by deciding independently, for each positive atom $K\varphi$ whether β entails φ , and for each negative atom $\neg K\varphi$ whether $\beta \wedge \neg\varphi$ is satisfiable, then combining the results following the structure of \vec{G} . Since the evaluation of each atom can be done with one call to an NP-oracle, we get the result. \square

Proposition 37 *The problems $\text{SAT}(\text{BS}(\text{CNF}), \text{S5}_{\text{Terms}, \text{Terms}})$ and $\text{SAT}(\text{BS}(\text{Terms}), \text{S5}_{\text{DNF}, \text{DNF}})$ are $\Theta_2\text{P}$ -hard.*

Proof We start with $\text{SAT}(\text{BS}(\text{CNF}), S5_{\text{Terms}, \text{Terms}})$. We give a reduction from the problem of deciding, given k propositional formulas ψ_1, \dots, ψ_k satisfying $\psi_i \models \psi_{i+1}$ for $i = 1, \dots, k-1$, whether the smallest i such that ψ_i is satisfiable is even. This problem is $\Theta_2\text{P}$ -complete [6]. Without loss of generality we assume that k is even (otherwise we add $\varphi_{k+1} := \top$).

So let $\psi_1, \dots, \psi_{2\ell}$ be propositional formulas as above.

First observe that it is NP-complete to decide whether $\varphi \wedge x$ is satisfiable for a given satisfiable CNF formula φ and a given variable x ; indeed, given an arbitrary CNF formula $\varphi_0 := \bigwedge C$, φ_0 is satisfiable if and only if $\varphi \wedge x$ is satisfiable, where x is a fresh variable and φ is the satisfiable CNF $\bigwedge_{C \in \varphi_0} (C \vee \neg x)$. For $i = 1, \dots, 2\ell$, we write $\langle \varphi_i, x_i \rangle$ for the image of ψ_i under this reduction, that is, ψ_i is satisfiable if and only if $\varphi_i \wedge x_i$ is satisfiable; moreover, by renaming the variables as necessary, we assume $(\bigvee(\varphi_i) \cup \{x_i\}) \cap (\bigvee(\varphi_{i'}) \cup \{x_{i'}\}) = \emptyset$ for $i \neq i'$.

Now we claim that the smallest i such that ψ_i is satisfiable is even, if and only if the belief state expression $\beta := \bigwedge_{i=1}^{2\ell} \varphi_i$ satisfies the goal $\vec{\gamma} := \bigvee_{j=1}^{\ell} (\text{K} \neg x_{2j-1} \wedge \neg \text{K} \neg x_{2j})$. Observe that $\|\beta\|$ is nonempty, since each φ_i is satisfiable and all are on disjoint sets of variables.

Indeed, by definition of satisfaction we have that β satisfies $\vec{\gamma}$ if and only if there is a j such that β satisfies $(\text{K} \neg x_{2j-1} \wedge \neg \text{K} \neg x_{2j})$. Now β satisfies $(\text{K} \neg x_{2j-1} \wedge \neg \text{K} \neg x_{2j})$ if and only if β entails $\neg x_{2j-1}$ and β does not entail $\neg x_{2j}$. This holds if and only if $\beta \wedge x_{2j-1}$ is unsatisfiable, and $\beta \wedge x_{2j}$ is satisfiable. Now because β is $\bigwedge_{i=1}^{2\ell} \varphi_i$, the sets of variables occurring in x_i and in $x_{i'}$ are disjoint for $i \neq i'$, and each φ_i is satisfiable, we have that for $i = 1, \dots, 2\ell$, $\beta \wedge x_i$ is satisfiable if and only if $\varphi_i \wedge x_i$ is satisfiable. This is by construction if and only if ψ_i is satisfiable.

In the end, we get that β satisfies $\vec{\gamma}$ if and only if there is a j such that ψ_{2j-1} is unsatisfiable and ψ_{2j} is satisfiable. Since by assumption $\psi_i \models \psi_{i+1}$ for all i , hence no $\psi_{j'}, j' < 2j-1$, can be satisfiable, this is if and only if there is j satisfying that $2j$ is the smallest i such that ψ_i is satisfiable, as desired.

The proof for $\text{SAT}(\text{BS}(\text{Terms}), S5_{\text{DNF}, \text{DNF}})$ is similar: since it is coNP-complete to decide whether a variable x entails a DNF formula φ , we use a polynomial-time reduction from the problem of deciding the unsatisfiability of a CNF to associate a pair $\langle x_i, \varphi_i \rangle$ to each ψ_i , such that x_i entails φ_i if and only if ψ_i is unsatisfiable (with disjoint sets of variables for $i \neq i'$). Then we define β to be the term $\bigwedge_{i=1}^{2\ell} x_i$ and $\vec{\gamma}$ to be $\bigvee_{j=1}^{\ell} (\text{K} \varphi_{2j-1} \wedge \neg \text{K} \varphi_{2j})$. We conclude by observing that β satisfies $\vec{\gamma}$ if and only if there is a j such that x_{2j-1} entails φ_{2j-1} and x_{2j} does not entail φ_{2j} , which by construction holds if and only if there is a j such that ψ_{2j-1} is unsatisfiable and ψ_{2j} is satisfiable, as desired. \square

Proposition 38 *The problem $\text{SAT}(\text{BS}_{\exists}(\Lambda), S5_{\Lambda^+, \Lambda^-})$ is NP-complete for $\Lambda \in \{\text{DNF}, \text{OBDD}_{<}, \text{Terms}\}$, $\Lambda^+ \in \{\text{CNF}, \text{OBDD}_{<}, \text{Terms}\}$, and $\Lambda^- \in \{\text{NNF}, \text{DNF}\}$. It is coNP-complete for $\Lambda \in \{\text{DNF}, \text{OBDD}_{<}, \text{Terms}\}$, $\Lambda^+ \in \{\text{NNF}, \text{DNF}\}$, and $\Lambda^- \in \{\text{CNF}, \text{OBDD}_{<}, \text{Terms}\}$. The same results hold for $\text{BS}(\Lambda)$ instead of $\text{BS}_{\exists}(\Lambda)$.*

Proof We start by showing membership for the NP-complete cases. For all combinations of languages, with or without auxiliary variables, the problem of deciding whether $\beta \in \Lambda$ entails $\varphi \in \Lambda^+$ is in P (see proof of Prop. 31), hence deciding whether β satisfies $\text{K} \varphi$ is in P; moreover, deciding whether $\beta \in \Lambda$ satisfies $\varphi \in \Lambda^-$ is coNP-complete, hence deciding whether β satisfies $\neg \text{K} \varphi$ is NP-complete. It follows that deciding whether $\beta \in \text{BS}_{\exists}(\Lambda)$ satisfies $\vec{\gamma}$ can be done by guessing a subset of the negative atoms $\neg \text{K} \varphi$ occurring in $\vec{\gamma}$, together with a witness of $\beta \models \neg \text{K} \varphi$ for each of them, which amounts to guessing a polynomial-size witness. Verifying such a witness can be done by deciding which positive

atoms $K\varphi$ are satisfied by β (in polynomial time), verifying the witnesses for negative atoms $\neg K\varphi$ (in polynomial time again), and finally verifying that the satisfied (positive and negative) atoms make the Boolean combination $\vec{\gamma}$ evaluate to true, again in polynomial time. Hence the problem is in NP.

Now hardness for the NP-complete cases follows from the fact that it is already NP-hard to decide whether $\|\beta\|$ satisfies a general goal $\vec{\gamma}$ reduced to a single negative atom $\neg K\varphi$, as observed above; this holds also without auxiliary variables, and under the restriction that $\|\beta\|$ is a nonempty belief state, because β does not entail φ if and only if $(\beta \vee x)$ does not entail $(\varphi \vee x)$, where x is a fresh variable, and the languages Λ^B and Λ^- under consideration support polynomial-time disjunction with x .

For the coNP-complete cases, we simply observe that $\|\beta\|$ does *not* satisfy $\vec{\gamma} \in S5_{\Lambda^+, \Lambda^-}$ if and only if it satisfies the negation of $\vec{\gamma}$, which by De Morgan's laws can be read as a formula in $S5_{\Lambda^-, \Lambda^+}$. Hence the problem $\text{SAT}(\text{BS}(\Lambda), S5_{\Lambda^+, \Lambda^-})$ and the complement of $\text{SAT}(\text{BS}(\Lambda), S5_{\Lambda^-, \Lambda^+})$ are polynomial-time reducible to each other, and the result follows from the first part of the statement.

The same results holds for $\text{BS}_{\exists}(\Lambda)$ and $\text{BS}(\Lambda)$ because we have shown membership for the former and hardness for the latter. \square

Proposition 39 *The problem $\text{SAT}(\text{BS}_{\exists}(\Lambda), S5_{\Lambda^+, \Lambda^-})$ is in P for $\Lambda \in \{\text{DNF}, \text{OBDD}_{<}\}$ and $\Lambda^+, \Lambda^- \in \{\text{CNF}, \text{OBDD}_{<}\}$.*

Proof It suffices to observe that for all the combinations of Λ, Λ^+ in the statement, it is polynomial-time to decide whether β entails φ for $\beta \in \text{BS}_{\exists}(\Lambda)$ and $\varphi \in \Lambda^+$, and for all the combinations of Λ, Λ^- , it is polynomial-time to decide whether $\beta \wedge \neg\varphi$ is satisfiable for $\beta \in \text{BS}_{\exists}(\Lambda)$ and $\varphi \in \Lambda^-$. Hence whether β satisfies $\vec{\gamma}$ can be decided by evaluating satisfaction of each atom in $\vec{\gamma}$ in polynomial time, then combining the results following the structure of $\vec{\gamma}$. \square

Equivalence For equivalence, the results are summarized in Table 7. For a set of fluents F and a belief state expression β_i in $\text{BS}_{\exists}(\Lambda)$ (for some propositional language Λ), we write Y_i for the set of auxiliary variables used in β_i .

Proposition 40 *The problem $\text{Equi}v(\text{BS}_{\exists}(\text{NNF}))$ is in $\Pi_2\text{P}$, and the problems $\text{Equi}v(\text{BS}_{\exists}(\text{DNF}))$ and $\text{Equi}v(\text{BS}_{\exists}(\text{OBDD}_{<}))$ are in coNP.*

Proof For NNF, we have $\|\beta_0\| \neq \|\beta_1\|$ if and only if there is an assignment $s \in 2^F$, an index $i \in \{0, 1\}$, and an assignment $s_{Y_i} \in 2^{Y_i}$ such that (i) $s \cup s_{Y_i}$ satisfies β_i and (ii) for all assignments $s_{Y_{1-i}} \in 2^{Y_{1-i}}$, $s \cup s_{Y_{1-i}}$ does not satisfy β_{1-i} . The statements $s \cup s_{Y_i} \models \beta_i$ and $s \cup s_{Y_{1-i}} \not\models \beta_{1-i}$ can be checked in polynomial time, so given s and s_{Y_i} , the first condition is in P and the second in coNP. This proves that checking non-equivalence is in $\Sigma_2\text{P}$, hence the result.

For DNF and $\text{OBDD}_{<}$, the reasoning is the same but given $s \in 2^F, i \in \{0, 1\}$, and $s_{Y_i} \in 2^{Y_i}$, the second condition amounts to deciding whether β_{1-i} conditioned by s is unsatisfiable, which can be decided in polynomial time. \square

Proposition 41 *The problem $\text{Equi}v(\text{BS}_{\exists}(\text{CNF}))$ is $\Pi_2\text{P-hard}$.*

Proof Consider a QBF of the form $\forall F \exists Y_1 : \varphi$, where φ is a CNF formula, and define $\beta_1 := \varphi \in \text{BS}_{\exists}(\text{CNF})$ and $\beta_2 := \top \in \text{BS}_{\exists}(\text{CNF})$ (empty CNF formula) over the set of fluents F . Then since all states in 2^F satisfy β_2 , $\|\beta_1\| = \|\beta_2\|$ holds if and only if all states in 2^F satisfy β_1 , that is, by definition of $\text{BS}_{\exists}(\text{CNF})$, if and only if $\forall F \exists Y_1 : \varphi$ is valid. This gives a reduction from the problem of deciding whether a $\forall \exists$ -QBF is valid. \square

For binary decision diagrams, since in general there is no bijection between F and the auxiliary variables involved in an existential representation of a belief state, we only consider the situations $F < Y_1 \cup Y_2$ and $Y_1 \cup Y_2 < F$.

Proposition 42 *Let $<$ be an ordering over $F \cup Y_1 \cup Y_2$ satisfying $Y_1 \cup Y_2 < F$. Then the problem $\text{Equiv}(\text{BS}_{\exists}(\text{OBDD}_{<}))$ is coNP-hard.*

Proof As shown in the proof of Proposition 29, it is coNP-hard to decide whether an expression $\beta_1 \in \text{BS}_{\exists}(\text{OBDD}_{<})$ with $V(\beta_1) \subseteq F \cup Y_1$ and $Y_1 < F$, satisfies $\exists Y_1 \cdot \|\beta_1\| = 2^F$. Since this is the same as deciding whether β_1 is equivalent to the belief state expression $\beta_2 := \top \in \text{BS}_{\exists}(\text{OBDD}_{<})$, we have the result. \square

The next result follows directly from well-known facts about propositional logic [3].

Proposition 43 *The problem $\text{Equiv}(\text{BS}(\Lambda))$ is coNP-complete for $\Lambda \in \{\text{NNF}, \text{DNF}, \text{CNF}\}$. The problem $\text{Equiv}(\text{BS}(\text{OBDD}_{<}))$ is in P.*

Proposition 44 *The problem $\text{Equiv}(\text{BS}_{\exists}(\text{Terms}))$ is in P.*

Proof Recall first that we define terms in the language Terms to be consistent. Given a term τ with $V(\tau) \subseteq F \cup Y$, one can compute in polynomial time a term τ' over F satisfying $\|\tau'\| = \exists Y \cdot \|\tau\|$ by simply removing all literals over a variable in Y . Then deciding equivalence of $\tau_1, \tau_2 \in \text{BS}_{\exists}(\text{Terms})$ can be done by computing τ'_1 and τ'_2 and comparing them for equality. \square

6.3 Complexity of transformations

We now study the complexity of transformations.

Importantly, contrary to queries, in general we cannot derive a result for a belief state language Λ_1^B from a result for a belief state language $\Lambda_2^B \supseteq \Lambda_1^B$ (nor vice-versa). This is because the belief state language defines a class of inputs, but also puts a restriction on the output, since we always require the output belief state to be in the same language as the input belief state. For that reason, we also ignore $\text{BS}_{\exists}(\text{Terms})$ and $\text{BS}(\text{Terms})$ as representations of belief states, since they are not complete languages and hence, in general they cannot express the result of an operation (even when the input is also a term).

On the other hand, we will tacitly use the reductions induced by the inclusion between the other languages (Λ^A and Λ^O), since they do not constrain the output of the transformations.

Also note that we restrict our study to “one-shot” transformations, in the sense that we define a transformation to be polynomial-time if applying it *once* is polynomial-time, without considering whether it is polynomial-time to perform a *sequence* of instances of that transformation; in the terms of the knowledge compilation map [3], we only consider *bounded* transformations. However, whether each transformation can be applied efficiently in sequence can be read easily from the proofs.

Progression by an Action For progression by an action, the results are summarized in Table 8.

If φ is a propositional formula, we write $\varphi[F'/F/F^-]$ for the expression obtained from φ by renaming simultaneously each variable $x \in F$ to a fresh variable x^- , and each variable $x' \in F'$, to x ; variables from $F^- := \{x^- \mid x \in F\}$ will typically become auxiliary variables in the set Y for existential representations. Moreover, if φ does not contain any primed

Table 8 Complexity of $\text{PROG}_a(\Lambda^B, \Lambda^A)$ for representations of belief states by propositional formulas

$\Lambda^B =$	$\Lambda^A =$	N/C-AT	DNFAT	OBDDAT _{<}	P/C $_{\Lambda^P, \Lambda^C}$	P/C-Det $_{\Lambda^P, \Lambda^C}$
BS \exists (NNF)		FP (46)	FP	FP	FP (51)	FP
BS(NNF)		•* (56)	polysize but $\notin \text{FP}^*$ (53)	•* for $F \parallel_{<} F'$ (59); otherwise open	•*	•* (62)
BS \exists (CNF)		FP (48)	FP	FP	FP (51)	FP
BS(CNF)		•* (56)	• (55)	• (55)	•*	•* (62)
BS \exists (DNF) or BS(DNF)		•	FP (46)	• (57)	•* (63)	FP for $\Lambda^C = \top$ (52); • for $\Lambda^C \supseteq \text{Terms}$ (61)
BS \exists (OBDD _{<})		• (57)	FP for $Y < F$ (50); • for $F < Y$ (55)	FP (46)	FP for $\Lambda^C = \top$ (52); • for $\Lambda^C \supseteq \text{Terms}$	FP for $\Lambda^C = \top$; • for $\Lambda^C \supseteq \text{Terms}$ (61)
BS(OBDD _{<})		• (57)	• (55)	• for $F < F'$, $F \parallel_{<} F'$ (60); FP for $F' < F$ (46 + 23)	•	• (62)

By • we denote the fact that the problem is not polynomial-size; •*, $\notin \text{FP}^*$ indicate that the result is conditional on some complexity-theoretic assumption. N/C-AT stands for NNFAT or CNFAT; P/C $_{\Lambda^P, \Lambda^C}$ stands for PDDL $_{\Lambda^P, \Lambda^C}$ or CSTRIPS $_{\Lambda^P, \Lambda^C}$; and P/C-Det $_{\Lambda^P, \Lambda^C}$ stands for PDDL-Det $_{\Lambda^P, \Lambda^C}$ or CSTRIPS-Det $_{\Lambda^P, \Lambda^C}$.

variable, we write $\varphi[F/F']$ for the expression obtained from φ by renaming simultaneously each variable $x \in F$ to x' .

We will use the following lemma, which follows directly from the definitions.

Lemma 45 *Let Λ, Λ_A be two propositional languages, and $\alpha \in \text{AT}(\Lambda_A)$ be an action theory. Then*

- for β in BS $\exists(\Lambda)$, $\beta' := (\beta \wedge \alpha)[F'/F/F^-]$ satisfies $\|\beta'\| = \|\beta\| \oplus \|\alpha\|$;⁴
- for β in BS (Λ) , an expression β' satisfies $\|\beta'\| = \|\beta\| \oplus \|\alpha\|$ if and only if $\|\beta'\| = \|\beta''[F'/F/F^-]\|$ for some expression β'' satisfying $\|\beta''\| = \exists F \cdot \|\beta\| \wedge \alpha$.

As a consequence, for existential propositional representations of belief states, using positive results from Darwiche and Marquis [3] about conjunction, we get the following result.

Proposition 46 *The problems $\text{PROG}_a(\text{BS}\exists(\text{NNF}), \text{NNFAT})$, $\text{PROG}_a(\text{BS}\exists(\text{DNF}), \text{DNFAT})$ and $\text{PROG}_a(\text{BS}\exists(\text{OBDD}_{<}), \text{OBDDAT}_{<})$ are in FP.*

⁴ Since β' here is in BS $\exists(\Lambda)$, $\|\beta'\|$ is by definition $\exists Y \cup F^- \cdot \|\beta'\|_{F \cup Y \cup F^-}$, where Y is the set of auxiliary variables in β .

For CNF, we will use the following lemma, which can be shown using the Tseitin transform of an NNF formula [25].

Lemma 47 *Given a formula $\varphi \in \text{NNF}$ over a set of variables F , one can compute in polynomial time a CNF formula φ' over $F \cup Y$, for some set of fresh variables Y , such that $\exists Y \cdot \|\varphi'\|_{F \cup Y} = \|\varphi\|_F$ holds.*

Proposition 48 *The problem $\text{PROG}_a(\text{BS}_\exists(\text{CNF}), \text{NNFAT})$ is in FP.*

Proof Given $\beta \in \text{BS}_\exists(\text{CNF})$ with $V(\beta) \subseteq F \cup Y$ and $\alpha \in \text{NNFAT}$, we can use Lemma 47 to compute in polynomial time an action expression $\alpha' \in \text{CNFAT}$ satisfying $V(\alpha') \subseteq F \cup F' \cup Y_\alpha$ (for some set Y_α of fresh variables) and $\exists Y_\alpha \cdot \|\alpha'\|_{F \cup F' \cup Y_\alpha} = \|\alpha\|$. Then by construction, for the CNF $\beta' := (\beta \wedge \alpha')[F'/F/F^-]$ we have the desired property $\exists Y \cup Y_\alpha \cup F^- \cdot \|\beta'\| = \|\beta\| \oplus \|\alpha\|$, and β' can be computed in polynomial time. \square

For DNF action theories, we will use the following lemma.

Lemma 49 *Let F be a set of fluents, $\beta \in \text{BS}_\exists(\text{NNF})$ be the representation of a belief state, and let $\alpha := \bigvee_{i \in I} \tau_i$ be an action theory in DNFAT. Then the expression $\beta' \in \text{DNF}$ defined as follows satisfies $\|\beta'\| = \|\beta\| \oplus \|\alpha\|$:*

$$\beta' := \left(\bigvee \{ \exists F \cdot \tau_i \mid i \in I, \beta \wedge \tau_i \text{ is satisfiable} \} \right) [F'/F/F^-]$$

where $\exists F \cdot \tau_i := \bigwedge \{ \lambda \in \tau_i \mid V(\lambda) \subseteq F' \}$ denotes the restriction of τ_i to the variables in F' .

Proof Let s be a state in $\|\beta\| \oplus \|\alpha\|$, and Y be the set of auxiliary variables used in β . Then by definition of progression, there are $s^- \in 2^F$ and $s_Y \in 2^Y$ such that $s^- \cup s_Y$ satisfies β and $s^- \cup (s')'$ satisfies α , hence $s^- \cup (s')'$ satisfies τ_i for some $i \in I$. It follows that $s^- \cup s_Y \cup (s')'$ satisfies $\beta \wedge \tau_i$, which is thus satisfiable, and that $(s')'$ satisfies $\exists F \cdot \tau_i$.

Conversely, let $s \in 2^F$ be a state such that $(s')'$ satisfies $\exists F \cdot \tau_i$ for some $i \in I$ such that $\beta \wedge \tau_i$ is satisfied by some state $s^- \cup s_Y \cup s'_0$. Then since $\beta \wedge \tau_i$ is logically equivalent to $\beta \wedge (\exists F' \cdot \tau_i) \wedge (\exists F \cdot \tau_i)$, and the two first conjuncts of this expression are over $F \cup Y$, we get that $s^- \cup s_Y$ satisfies β and $s^- \cup s_Y \cup (s')'$ satisfies $\beta \wedge \tau_i$ and hence, $\beta \wedge \alpha$; hence s is in $\|\beta\| \oplus \|\alpha\|$. \square

Proposition 50 *The problem $\text{PROG}_a(\text{BS}_\exists(\text{OBDD}_{<}), \text{DNFAT})$ is in FP for $Y < F$.*

Proof Using Lemma 49, one can get a DNF expression β' for $\|\beta\| \oplus \|\alpha\|$ by considering each term of α in turn, since deciding whether a term is consistent with an OBDD is in P. Then we can use the construction by [3, Proof of Prop. 5.1, p. 258] already described in the proof of Prop. 29 to compute in polynomial time an expression in $\text{BS}_\exists(\text{OBDD}_{<})$ which represents β' . \square

For PDDL languages, despite the fact that expressions are not action theories, we can introduce a set of auxiliary variables Y , and translate them into propositional formulas over $F \cup F' \cup Y$. This is enough for obtaining the following positive result about existential representations.

Proposition 51 *The problems $\text{PROG}_a(\text{BS}_\exists(\text{NNF}), \text{PDDL}_{\text{NNF}, \text{NNF}})$ and $\text{PROG}_a(\text{BS}_\exists(\text{CNF}), \text{PDDL}_{\text{NNF}, \text{NNF}})$ are in FP.*⁵

⁵ The proof of this statement relies on the fact that for PDDL expressions, we measure the size of a tree representation (except for propositional preconditions and conditions).

Proof Recall first from the definition of PROG_a that in an input $\langle \beta, \alpha \rangle$, the action $\|\alpha\|$ is assumed to be applicable at the belief state $\|\beta\|$, so that we can ignore the precondition of the action.

Given an action expression $\alpha := \langle \pi, \eta \rangle$ in $\text{PDDL}_{\text{NNF}, \text{NNF}}$, we define an NNF formula $\varphi(\eta)$ which uses a set Y_α of polynomially many auxiliary variables, can be computed in polynomial time, and is such that two states $s \in 2^F$, $s' \in 2^{F'}$ verify $s' \in \alpha(s)$ if and only if there is an assignment $s_{Y_\alpha} \in 2^{Y_\alpha}$ with $s \cup s' \cup s_{Y_\alpha} \models \varphi(\eta)$.

It will follow from Lemma 45 that $(\beta \wedge \varphi(\eta))[F'/F/F^-]$ is a representation in $\text{BS}_\exists(\text{NNF})$ of $\|\beta\| \oplus \|\alpha\|$; for $\text{BS}_\exists(\text{CNF})$, Lemma 47 will allow us to conclude.

Intuitively, the construction of $\varphi(\eta)$ emulates the step-by-step transformation of s into s' by η , by keeping track of the changes in the values of the variables at each timestep, using additional variables. We first compute the set of necessary timesteps as follows. First, going bottom-up in the expression η , we define the *minimum number* of timesteps $T^\#(\eta_0)$ required by each subexpression η_0 of η by defining $T^\#(+x) := T^\#(-x) := 1$, $T^\#(\eta_1 \cup \eta_2) := \max(T^\#(\eta_1), T^\#(\eta_2))$, $T^\#(\eta_1 \& \eta_2) := T^\#(\eta_1) + T^\#(\eta_2)$, and $T^\#(\chi \triangleright \eta_1) := T^\#(\eta_1)$. Then, going top-down in the expression η , we define the *set* of timesteps $T(\eta)$ to be $[1..T^\#(\eta)]$, and for subexpressions η_0 of η we compute the set of timesteps for the children of η_0 as follows, where we write $T(\eta_0) := [b(\eta_0)..e(\eta_0)]$:

$$\begin{aligned} \text{for } \eta_0 = \eta_1 \cup \eta_2 : T(\eta_1) &:= T(\eta_2) := T(\eta_0) \\ \text{for } \eta_0 = \eta_1 \& \eta_2 : &\begin{cases} T(\eta_1) := [b(\eta_0)..(b(\eta_0) + T^\#(\eta_1) - 1)] \\ T(\eta_2) := [(b(\eta_0) + T^\#(\eta_1))..e(\eta_0)] \end{cases} \\ \text{for } \eta_0 = \chi \triangleright \eta_1 : T(\eta_1) &:= T(\eta_0) \end{aligned}$$

By construction, for all subexpressions η_0 of η , $T(\eta_0)$ is a set of at least $T^\#(\eta_0)$ contiguous integers of the form $[b(\eta_0)..e(\eta_0)]$, and there is a polynomial number of timesteps introduced.

We now define an intermediate NNF formula $\psi(\eta_0)$ by induction as follows, where for all $x \in F$ and all relevant timesteps t , 1_x^t , 0_x^t denote fresh variables introduced in Y_α (to be read “ x is set to \top —resp. to \perp —at timestep t ”).⁶

$$\psi(\eta_0) := \begin{cases} \text{for } \eta_0 = +x : & \left(1_x^{b(\eta_0)} \wedge \neg 0_x^{b(\eta_0)} \wedge \bigwedge_{t=b(\eta_0)+1}^{e(\eta_0)} (\neg 1_x^t \wedge \neg 0_x^t) \right. \\ & \left. \wedge \bigwedge_{t=b(\eta_0)}^{e(\eta_0)} \bigwedge_{x_0 \in F, x_0 \neq x} (\neg 1_{x_0}^t \wedge \neg 0_{x_0}^t) \right) \\ \text{for } \eta_0 = -x : & \left(\neg 1_x^{b(\eta_0)} \wedge 0_x^{b(\eta_0)} \wedge \bigwedge_{t=b(\eta_0)+1}^{e(\eta_0)} (\neg 1_x^t \wedge \neg 0_x^t) \right. \\ & \left. \wedge \bigwedge_{t=b(\eta_0)}^{e(\eta_0)} \bigwedge_{x_0 \in F, x_0 \neq x} (\neg 1_{x_0}^t \wedge \neg 0_{x_0}^t) \right) \\ \text{for } \eta_0 = \eta_1 \cup \eta_2 : & (\psi(\eta_1) \vee \psi(\eta_2)) \\ \text{for } \eta_0 = \eta_1 \& \eta_2 : & (\psi(\eta_1) \wedge \psi(\eta_2)) \\ \text{for } \eta_0 = \chi \triangleright \eta_1 : & \left(\begin{aligned} &(\chi \rightarrow \psi(\eta_1)) \\ &\wedge (\neg \chi \rightarrow \bigwedge_{t=b(\eta_0)}^{e(\eta_0)} \bigwedge_{x \in F} (\neg 1_x^t \wedge \neg 0_x^t)) \end{aligned} \right) \end{cases}$$

Clearly, $\psi(\eta_0)$ can be computed in polynomial time. Now it is easy to show by induction that the following hold. First, for all assignments s_ψ to $V(\psi(\eta_0))$ which satisfy $\psi(\eta_0)$, for

⁶ We use the connective \rightarrow for readability, with $\varphi \rightarrow \psi$ being a shorthand for the NNF formula $\neg \varphi \vee \psi$; the shorthand $x \leftrightarrow y$, for variables x, y , for the NNF formula $(\neg x \vee y) \wedge (x \vee \neg y)$; and for an NNF formula φ , we use $\neg \varphi$ for the NNF obtained from negating φ using De Morgan’s laws.

all $x \in F$ and $t \in [b(\eta_0) .. e(\eta_0)]$, it is *not* the case that both 1_x^t and 0_x^t are in s_ψ . Second, for all states $s \in 2^F$, the effect $\langle E^+, E^- \rangle$ is in $E(\eta_0, s)$ if and only if there is an assignment s_ψ to $V(\psi(\eta_0))$ which satisfies $\psi(\eta_0)$ and verifies for all $x \in F$:

$$x \in E^+ \iff \exists t : b(\eta_0) \leq t \leq e(\eta_0) \text{ and } 1_x^t \in s_\psi \quad (1)$$

$$x \in E^- \iff \exists t : b(\eta_0) \leq t \leq e(\eta_0) \text{ and } 0_x^t \in s_\psi \quad (2)$$

Now to complete the construction, we define $\varphi(\eta)$ as follows:

$$\varphi(\eta) := \psi(\eta) \wedge \bigwedge_{x \in F} \left(\begin{array}{l} \left(\left(\bigwedge_{t=1, \dots, T^\#(\eta)} (\neg 1_x^t \wedge \neg 0_x^t) \rightarrow (x' \leftrightarrow x) \right) \right) \\ \wedge \left(\left(\bigvee_{t=1, \dots, T^\#(\eta)} 1_x^t \rightarrow x' \right) \right) \\ \wedge \left(\left(\bigvee_{t=1, \dots, T^\#(\eta)} 0_x^t \right) \wedge \left(\bigwedge_{t=1, \dots, T^\#(\eta)} \neg 1_x^t \right) \rightarrow \neg x' \right) \end{array} \right)$$

As can be seen, this construction sets the values of the variables in F' depending on the values of the variables in F and the values of the auxiliary variables in $\psi(\eta_0)$, and it follows from the properties of $\psi(\eta)$ that for all states $s \in 2^F$, $s' \in 2^{F'}$, $s' \in \|\alpha\|(s)$ holds if and only if there is an assignment $s_{Y_\alpha} \in 2^{Y_\alpha}$ such that $s \cup s' \cup s_{Y_\alpha}$ satisfies $\varphi(\eta)$, as desired. \square

Proposition 52 *The problems $\text{PROG}_a(\text{BS}(\text{DNF}), \text{PDDL-Det}_{\text{NNF}, \top})$ and $\text{PROG}_a(\text{BS}\exists(\text{OBDD}_{<}), \text{PDDL-Det}_{\text{NNF}, \top})$ are in FP.*

Proof Recall that we assume the actions to be applicable, so we can ignore the precondition and work with the effect expression η . Since there are no conditions and no nondeterministic choices, η can be transformed to an expression of the form $\&_{x \in V} \pm x$. Hence applying it to a belief state expression amounts to forgetting the variables in V , then conjoining the literals corresponding to the $\pm x$'s. This concludes the proof since forgetting and conjunction with terms are in FP for DNF and for $\text{OBDD}_{<}$ with auxiliary variables (for the latter, first rename the variables to be forgotten, then conjoin the term). \square

The following can be seen as a mixed positive/negative result.

Proposition 53 *The problem $\text{PROG}_a(\text{BS}(\text{NNF}), \text{DNFAT})$ is polynomial-size but, assuming $P \neq NP$, not polynomial-time.*

Proof The fact that the problem is polysize follows directly from the construction of Lemma 49. Now let ψ be an NNF formula over a set of variables F , and let x be a fresh variable. Define $\beta := \psi \vee x \in \text{BS}(\text{NNF})$ and $\alpha := (\neg x \wedge x') \vee (\neg x') \in \text{DNFAT}$; observe that $\|\beta\|$ is nonempty and that α is applicable at all states. Then by Lemma 49 again, the progression $\beta \oplus \alpha$ is equivalent to \top if ψ is satisfiable, and to $\neg x'$ if it is not. As a consequence, if a representation of $\beta \oplus \alpha$ in $\text{BS}(\text{NNF})$ can be computed in polynomial time, then one can check whether an arbitrary assignment s to 2^F (say the assignment of all variables to \top) is such that $(s)' \cup \{x'\}$ satisfies it, and conclude whether ψ is satisfiable, contradicting $P \neq NP$. \square

We now turn to negative results, starting with progression by action theories. First, using the following result together with the results by [3, Table 3] sets a number of cases.

Lemma 54 *Let Λ, Λ_A be two propositional languages such that Λ has a polynomial-size representation of \top (in the number of variables in F). If Λ is not at least as succinct as Λ_A , then the problem $\text{PROG}_a(\text{BS}(\Lambda), \text{AT}(\Lambda_A))$ is not polynomial size.*

Proof Given a formula $\psi \in \Lambda_A$, write F for $V(\psi)$. Then the formula $\alpha := \psi[F/F']$ is an action expression in $AT(\Lambda_A)$ (which does not mention any unprimed variable). Then by Lemma 45, $\beta = \top$ is such that the progression of β by α is an expression in Λ which is logically equivalent to ψ ; moreover, $\|\beta\|$ is nonempty and α is applicable at β , hence the result. \square

Corollary 55 *The problems $\text{PROG}_a(\text{BS}(\text{CNF}), \text{DNFAT})$, $\text{PROG}_a(\text{BS}(\text{CNF}), \text{OBDDAT}_{<})$, $\text{PROG}_a(\text{BS}(\text{OBDD}_{<}), \text{DNFAT})$, and $\text{PROG}_a(\text{BS}_{\exists}(\text{OBDD}_{<}), \text{DNFAT})$ for $F < Y$, are not polynomial size.*

Proof The results follow directly from Lemma 54 together with the results by [3, Table 3], together with Lemma 23 for $\text{BS}_{\exists}(\text{OBDD}_{<})$. \square

Proposition 56 *Assume $\text{NP} \not\subseteq \text{P/poly}$, and let Λ be NNF or CNF. Then the problem $\text{PROG}_a(\text{BS}(\Lambda), \text{CNFAT})$ is not polynomial-size.*

Proof Consider $\beta = \top$. Then by Lemma 45, computing the progression of β by an expression α in a plain representation, amounts to compute the existential forgetting of F in α (and renaming the variables).

For all $n \in \mathbb{N}$, let V_n be the set of fluents $\{x_1, \dots, x_n\}$, and C_n be the set of fluents $\{c_1, \dots, c_{8\binom{n}{3}}\}$, with the intended meaning that c_i encodes whether the i th clause of length 3 over V_n (which we denote by $\gamma_{n,i}$) is present in a given CNF formula (for some arbitrary, fixed enumeration of these clauses). Finally, write $F_n := V_n \cup C_n$. We consider the following action theory in CNFAT:

$$\alpha_n := \bigwedge_{i=1}^{8\binom{n}{3}} (\neg c'_i \vee \bigvee_{\lambda \in \gamma_{n,i}} \lambda) \wedge \bigwedge_{i=1}^n x'_i$$

It is easy to see that given a CNF formula ψ over V_n , ψ is satisfiable if and only if for the assignment $s' := \{c'_i \mid \gamma_{n,i} \in \psi\} \cup V'_n$, there is an assignment $s \in 2^{F_n}$ such that $s \cup s'$ satisfies α_n , that is, if and only if s' satisfies $\top \oplus \alpha_n$. Since $s' \models \top \oplus \alpha_n$ can be verified in polynomial time for $\text{BS}(\Lambda)$, if there was a polynomial-size expression for $\top \oplus \alpha_n$, this expression would define a polynomial-time algorithm for verifying the satisfiability of a 3CNF formula over n variables, implying $\text{NP} \subseteq \text{P/poly}$. Since $\beta := \top$ is a nonempty belief state, with a polynomial size representation in NNF and CNF, and α_n is applicable at all states (for all s , $s \cup \{x'_i \mid x_i \in V_n\}$ satisfies α_n), we get the result. \square

Proposition 57 *Let Λ^A be one of CNFAT , $\text{OBDDAT}_{<}$. Then the problem $\text{PROG}_a(\text{BS}(\text{DNF}), \Lambda^A)$ is not polynomial-size. Additionally, the problems $\text{PROG}_a(\text{BS}(\text{OBDD}_{<}), \text{CNFAT})$ and $\text{PROG}_a(\text{BS}_{\exists}(\text{OBDD}_{<}), \text{CNFAT})$ are not polynomial-size.*

Proof We start with $\text{BS}(\text{DNF})$ and CNFAT . Without loss of generality, let $F = \{x_1, x_2, \dots, x_{2\ell}\}$ be a set of fluents (if there are an odd number of fluents, we can simply leave one out of the construction). Let $\beta \in \text{BS}(\text{DNF})$ be the DNF formula \top (reduced to an empty term). Finally, let α be the expression defined by

$$\alpha := \bigwedge_{j=1}^{\ell} ((x'_j \vee x'_{j+\ell}) \wedge (\neg x'_j \vee \neg x'_{j+\ell}))$$

It is easy to see that $\|\beta\| \oplus \|\alpha\|$ is $\{s \in 2^F \mid \text{for } j = 1, \dots, \ell : x_j \in s \Leftrightarrow x_{j+\ell} \notin s\}$, which we claim has no polynomial-size representation in $\text{BS}(\text{DNF})$.

Indeed, let $\beta' \in \text{BS}(\text{DNF})$ be a representation, and let τ be a consistent term in it. Assume towards contradiction that τ does not have a literal over x_i for some $i \in \{1, \dots, 2\ell\}$; then for any model of τ , the same model with the value of x_i changed must also be a model of τ , hence also of β' ; this is a contradiction because no two assignments in $\|\beta\| \oplus \|\alpha\|$ differ over the value of exactly one x_i . Hence in the end, each term in β' must have one literal over each x_i , and hence covers only one element in $\|\beta\| \oplus \|\alpha\|$. Since there are an exponential number of these elements, β' must have exponentially many terms.

For $\text{OBDDAT}_{<}$, assume without loss of generality that $<$ orders F' by $x'_1 < x'_{1+\ell} < x'_2 < x'_{2+\ell} < \dots < x'_\ell < x'_{2\ell}$ (otherwise we simply reindex them). Then it is easy to see that α above has the same semantics as $\alpha := \alpha_1$, with $\alpha_{\ell+1} := \top$ and, for $j = 1, \dots, \ell$:

$$\eta := \left(\bigwedge_{j=1}^{\ell} (x_j \triangleright (-x_{j+\ell})) \right) \& \left(\bigwedge_{j=1}^{\ell} (\neg x_j \triangleright (+x_{j+\ell})) \right)$$

Finally, for $\text{BS}(\text{OBDD}_{<})/\text{BS}_{\exists}(\text{OBDD}_{<})$ and CNFAT , assume without loss of generality that $<$ orders F by $x_1 < x_2 < \dots < x_{2\ell}$. Let $\beta := \top$, and consider again the expression $\alpha \in \text{CNFAT}$ defined above. Then the progression of β by α is again equivalent to $\bigwedge_{j=1}^{\ell} (x_j \neq x_{j+\ell})$, which is well-known to have only exponential representations in $\text{OBDD}_{<}$ (there cannot be fewer than 2^ℓ nodes at the $x_{\ell+1}$ level, since each assignment to the x_1, \dots, x_ℓ induces a distinct model set over $x_{\ell+1}, \dots, x_{2\ell}$ — see e.g. [26]), and clearly in $\text{BS}_{\exists}(\text{OBDD}_{<})$ as well (using the same argument, and noticing that if two model sets over $x_{\ell+1}, \dots, x_{2\ell}$ are distinct, adding auxiliary variables cannot make them equal).

To conclude, observe that $\beta := \top$ is nonempty, and that α is applicable at all states. \square

For the next results, we first show the following lemma, which is a stronger version of a result from [3] (they show that the problem is not polynomial-time assuming $\text{P} \neq \text{NP}$).

Lemma 58 *Assuming $\text{NP} \not\subseteq \text{P/poly}$, existential forgetting is not polynomial-size in NNF nor in CNF.*

Proof We use essentially the same construction as in Proposition 56: if the variables in V_n could be forgotten in polynomial-size from the expression

$$\varphi_n := \bigwedge_{i=1}^{8\binom{n}{3}} (\neg c_i \vee \bigvee_{\lambda \in \gamma_{n,i}} \lambda)$$

then the resulting family of NNF formulas would give a non-uniform polynomial-time algorithm for deciding the satisfiability of a 3CNF, implying $\text{NP} \subseteq \text{P/poly}$. Since φ_n is in CNF, we get the result for both NNF and CNF. \square

Proposition 59 *Assume $\text{NP} \not\subseteq \text{P/poly}$. Then for $F \parallel_{<} F'$, the problem $\text{PROG}_a(\text{BS}(\text{NNF}), \text{OBDDAT}_{<})$ is not polynomial-size.*

Proof Let ψ be an NNF formula, and let V be a subset of $\text{V}(\psi)$. Let x be a fresh variable. Define $V := \text{V}(\psi)$, $\beta := \psi \vee \neg x$, and α to be an expression in $\text{OBDDAT}_{<}$ for the action which sets all variables in V to \top and leaves all other variables unchanged; clearly, such an action has a polynomial size expression in $\text{OBDDAT}_{<}$ for $F \parallel_{<} F'$. Moreover, observe that $\|\beta\|$ is nonempty and that α is applicable at all states. Now clearly, for $\beta' := \beta \oplus \alpha$, the expression $\beta'_{|V \cup \{x\} = \top}$ (β' conditioned by the assignment of all fluents in $V \cup \{x\}$ to \top) is equivalent to the forgetting of V in ψ . Since conditioning is polynomial-time in NNF, but forgetting is not polysize (Lemma 58), we get the result. \square

Proposition 60 Assume $F < F'$ or $F \parallel_{<} F'$. Then the problem $\text{PROG}_a(\text{BS}(\text{OBDD}_{<}), \text{OBDDAT}_{<})$ is not polynomial-size.

Proof We use the fact that forgetting is not polynomial-size for OBDDs even under the restriction that the forgotten variables occur first in the ordering [3, Table 7 and proof of Prop. 5.1, p. 258]. Given an expression $\psi \in \text{OBDD}_{<}$ and a set of variables V (with the aim of forgetting V in ψ), we define F to be $V(\psi)$, and the action theory α to be obtained from ψ by renaming each variable $x \in V(\psi) \setminus V$ to $x' \in F'$. Since by assumption the variables V occur first in the ordering, α is in $\text{OBDDAT}_{<}$ for some ordering $<$ with $F < F'$; it is also in $\text{OBDDAT}_{<}$ for $F \parallel_{<} F'$ since for no variable x do both x and x' occur. We finally define $\beta := \top \in \text{BS}(\text{OBDD}_{<})$, and with Lemma 45 we conclude that $\beta \oplus \alpha$ should be an expression for the result of forgetting V in ψ (modulo a renaming of the variables), and hence it is not polynomial-size.

Observe that in the construction above, β is nonempty but α is not necessarily applicable at all states, as is required by the definition of the progression transformation. However, one can compute $\beta \oplus \alpha$ by computing first $\beta \oplus ((\alpha \wedge \neg x') \vee x')$ for some fresh variable x , then conditioning the result on $x = \perp$. Since an OBDD for $(\alpha \wedge \neg x') \vee x'$, and the conditioning of an OBDD, can be computed in polynomial time, and the action $(\alpha \wedge \neg x') \vee x'$ is applicable at all states, we get the result. \square

We finally turn to negative results for PDDL and STRIPS.

Proposition 61 The problems $\text{PROG}_a(\text{BS}_{\exists}(\text{OBDD}_{<}), \text{CSTRIPS-Det}_{\top, \text{Terms}})$ and $\text{PROG}_a(\text{BS}(\text{DNF}), \text{CSTRIPS-Det}_{\top, \text{Terms}})$ are not polynomial-size.

Proof We start with $\text{OBDD}_{<}$. We use a similar construction as in the proof of Proposition 57. Assume without loss of generality that $<$ orders F by $x_1 < x_2 < \dots < x_{2\ell}$. Let $\beta := \top$, and define α to be $\langle \top, \eta \rangle$, with

$$\eta := \left(\bigwedge_{j=1}^{\ell} (x_j \triangleright (-x_{j+\ell})) \right) \& \left(\bigwedge_{j=1}^{\ell} (\neg x_j \triangleright (+x_{j+\ell})) \right)$$

Observe that $\|\beta\|$ is nonempty and that α is applicable at all states. Then we have $B' := \|\beta\| \oplus \|\alpha\| = \{s \in 2^F \mid \text{for } j = 1, \dots, \ell : x_j \in s \Leftrightarrow x_{j+\ell} \notin s\}$, and we use the standard argument for binary decision diagrams: the projection of $\|B'\|$ onto $\{x_{\ell+1}, \dots, x_{2\ell}\}$ is a different set of assignments for each assignment to $\{x_1, \dots, x_{\ell}\}$, hence from $x_1, \dots, x_{\ell} < x_{\ell+1}, \dots, x_{2\ell}$ it follows that there must be an exponential number of nodes in β' .

For DNF, the proof is exactly the same, with the fact that a DNF for B' has exponential size being proved in Proposition 57. \square

Proposition 62 Let Λ be one of NNF, CNF, $\text{OBDD}_{<}$. Then the problem $\text{PROG}_a(\text{BS}(\Lambda), \text{CSTRIPS-Det}_{\top, \top})$ is not polynomial-size, assuming $\text{NP} \not\subseteq \text{P/poly}$ for NNF, CNF.

Proof The construction is similar to that in the proof of Proposition 59. Let ψ be an expression in Λ , and let V be a subset of $V(\psi)$. Let x be a fresh variable. Define $F := V(\psi)$, $\beta := \psi \vee \neg x$, and $\alpha := \langle \top, \eta \rangle \in \text{CSTRIPS-Det}_{\top, \top}$, with $\eta := \top \triangleright \bigwedge_{x_i \in V} +x_i$; observe that $\|\beta\|$ is nonempty and that α is applicable at all states. Now clearly, for $\beta' := \beta \oplus \alpha$, the expression $\beta'_{|V \cup \{x\} = \top}$ (β' conditioned by the assignment of all fluents in $V \cup \{x\}$ to \top) is equivalent to the forgetting of V in ψ . Since conditioning is polynomial-time in NNF, CNF, $\text{OBDD}_{<}$, but forgetting is not polysize ([3, Table 7] for $\text{OBDD}_{<}$, and Lemma 58 for NNF, CNF), we get the result. \square

Proposition 63 Assuming $\text{NP} \not\subseteq \text{P/poly}$, the problem $\text{PROG}_a(\text{BS}(\text{DNF}), \text{CSTRIPS}_{\top, \top})$ is not polynomial-size.

Proof The proof is similar to that of Proposition 56, and we use the same notation. For $n \in \mathbb{N}$, consider the set of fluents C_n , and the action description $\alpha_n := \langle \top, \eta_n \rangle \in \text{CSTRIPS}_{\top, \top}$ defined by:

$$\eta_n := \bigotimes_{i=1}^n (\top \triangleright \bigcup_{\langle \gamma_{i1,n}, \gamma_{i2,n}, \gamma_{i3,n} \rangle \in \text{choose}(x_i) \cup \text{choose}(\neg x_i)} (+c_{i1} \& +c_{i2} \& +c_{i3}))$$

where for a literal λ over F_n , $\text{choose}(\lambda)$ denotes the set of all triples (possibly with repetition) of clauses of length 3 over V_n which contain λ . Let β be such that $\|\beta_n\| = \{\emptyset\}$ holds, that is, β_n represents the belief state containing only the assignment of all fluents to \perp . Observe that $\|\beta_n\|$ is nonempty and that α_n is applicable at all states.

It is easily seen that $\|\beta_n\| \oplus \|\alpha_n\|$ contains all states of the form $s_I := \{c_i \mid i \in I\}$ such that $\bigwedge_{i \in I} \gamma_{i,n}$ is a satisfiable 3CNF formula over n variables with at most 3 occurrences of each variable, and contains no encoding of an unsatisfiable formula at all; the fact that the formula is satisfiable is ensured by the fact that for $i = 1, \dots, n$, the action must choose $\lambda_i := x_i$ or $\lambda_i := \neg x_i$, and add only clauses which are satisfied by λ_i .

Hence a polynomial representation of $\|\beta_n\| \oplus \|\alpha_n\|$ in $\text{BS}(\text{DNF})$ would give a polynomial time algorithm for deciding whether a 3CNF formula over n variables, with at most 3 occurrences of each literal, is satisfiable, implying $\text{NP} \subseteq \text{P/poly}$. \square

Progression by an Observation

For progression by an observation, the results are summarized in Table 9.

Since $B \oplus o$ is by definition $B \cap o$, a number of positive results follow directly from known results about the complexity of (bounded) conjunction.

Proposition 64 Let Λ be one of NNF, CNF, DNF, $\text{OBDD}_{<}$. Then the problems $\text{Prog}_o(\text{BS}_{\exists}(\Lambda), \Lambda)$ and $\text{Prog}_o(\text{BS}(\Lambda), \Lambda)$ are in FP.

For existential representations in CNF, we can use Lemma 47 to get an expression for $\beta \wedge \omega$:

Table 9 Complexity of $\text{PROG}_o(\Lambda^B, \Lambda^O)$ for representations of belief states by propositional formulas. By \bullet we denote the fact that the problem is not polynomial-size

$\Lambda^B =$	$\Lambda^O =$	NNF	CNF	DNF	$\text{OBDD}_{<}$	Terms
$\text{BS}_{\exists}(\text{NNF})$ or $\text{BS}(\text{NNF})$		FP (64)	FP	FP	FP	FP
$\text{BS}_{\exists}(\text{CNF})$		FP (65)	FP (64)	FP	FP	FP
$\text{BS}(\text{CNF})$		\bullet	FP (64)	\bullet (66)	\bullet (66)	FP
$\text{BS}_{\exists}(\text{DNF})$ or $\text{BS}(\text{DNF})$		\bullet	\bullet (66)	FP (64)	\bullet (66)	FP
$\text{BS}_{\exists}(\text{OBDD}_{<})$		\bullet	\bullet (67)	\bullet for $F < Y$ (66 + 23); FP for $Y < F$ (68)	FP (64)	FP
$\text{BS}(\text{OBDD}_{<})$		\bullet	\bullet (66)	\bullet (66)	FP (64)	FP

Proposition 65 *The problem $\text{Prog}_o(\text{BS}_\exists(\text{CNF}), \text{NNF})$ is in FP.*

As for negative results, we can infer a number of results from the relative succinctness of languages.

Proposition 66 *Let Λ, Λ^O be two propositional languages. If there is a polynomial-size representation of \top and a constant-size representation of \perp in Λ (in the number of variables in F), and Λ is not at least as succinct as Λ^O , then the problem $\text{Prog}_o(\text{BS}(\Lambda), \Lambda^O)$ is not polynomial-size.*

Proof First observe that Λ is not at least as succinct as Λ^O even when restricting to satisfiable formulas, since \perp has a constant-size representation in Λ and hence, Λ is at least as succinct for unsatisfiable formulas.

Now consider $\beta := \top$ and an expression $\omega \in \Lambda^O$ which is satisfiable (hence fair to β). Then the progression of $\|\beta\|$ by $\|\omega\|$ is an expression $\beta' \in \Lambda$ with $\|\beta'\| = \|\omega\|$, hence it is a representation of ω in Λ , from what the result follows. \square

Finally, for binary decision diagrams, we have the following.

Proposition 67 *The problem $\text{Prog}_o(\text{BS}_\exists(\text{OBDD}_{<}), \text{CNF})$ is not polynomial-size.*

Proof The idea is similar to that in the proof of Proposition 61. Assume without loss of generality $F = \{x_1, x_2, \dots, x_{2\ell}\}$ (otherwise, if it has an odd number of elements, we simply leave one out of the construction), and that $<$ orders F as $x_1 < x_2 < \dots < x_{2\ell}$ (otherwise we simply reindex the fluents).

We define the expression $\omega \in \text{CNF}$ by $\omega := \bigwedge_{j=1}^{\ell} ((x_j \vee x_{j+\ell}) \wedge (\neg x_j \vee \neg x_{j+\ell}))$. Then for $\beta := \top$, we have $\|\beta\| \cap \|\omega\| = \{s \in 2^F \mid \text{for } j = 1, \dots, \ell : x_j \in s \Leftrightarrow x_{j+\ell} \notin s\}$ (in particular, ω is fair to β), and we conclude as in the proof of Proposition 61. \square

Proposition 68 *The problem $\text{Prog}_o(\text{BS}_\exists(\text{OBDD}_{<}), \text{DNF})$ is in FP for $Y < F$.*

Proof Given $\beta \in \text{BS}_\exists(\text{OBDD}_{<})$ with $V(\beta) \subseteq F \cup Y$ and $\omega := \bigvee_{i=1}^m \tau_i \in \text{DNF}$, where each τ_i is a term, we first compute, for $i = 1, \dots, m$, an expression $\beta'_i \in \text{OBDD}_{<}$ which is logically equivalent to $\beta \wedge \tau_i$; since this is a binary conjunction of two binary decision diagrams (viewing τ_i as one), this can be done in polynomial time. By definition of progression by an observation and the fact that existential quantification distributes over disjunction, it is easy to see that $\bigcup_{i=1}^m \|\beta'_i\| = \|\beta\| \cap \|\omega\|$ holds. Then we can use the same construction as in the proof of Proposition 29 to get an expression for $\bigcup_{i=1}^m \|\beta'_i\|$ in $\text{BS}_\exists(\text{OBDD}_{<})$. \square

7 Implicit representation by traces

We now investigate implicit representations of belief states. These differ from the representations of Section 6 because they do not use a representation by a propositional formula.

Implicit representations by traces were introduced by the Conformant-FF planner [27]. A trace simply consists of an initial belief state and a history of actions and observations; such a trace is taken to represent the belief state resulting from the progression of the initial belief state by the history.

Definition 69 (trace language). Let Λ^I, Λ^O be propositional languages and Λ^A be an action language. The *trace language induced by $\Lambda^I, \Lambda^O, \Lambda^A$* , written $\text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$, is the set of

all expressions of the form $\beta := \langle \iota, \alpha\omega_1, \alpha\omega_2, \dots, \alpha\omega_k \rangle$, where $\iota \in \Lambda^I$ is an expression for a nonempty initial belief state; for $i = 1, \dots, k$, $\alpha\omega_i$ is either an action expression in Λ^A or an observation expression in Λ^O ; and the history $h(\beta) := \langle \|\alpha\omega_1\|, \|\alpha\omega_2\|, \dots, \|\alpha\omega_k\| \rangle$ is applicable and fair at $\|\iota\|$. The semantics of such an expression is given by $\|\beta\| := h(\beta)(\|\iota\|)$.

We emphasize that the history of a trace is assumed to be applicable and fair at $\|\iota\|$. Though this cannot be decided efficiently in general, it is indeed how planners typically use them [5]. This is moreover motivated by *online belief tracking* as defined by [11].

Example 70 (continued from Examples 2, 10, 17). The belief state induced by history $\langle \text{right}, \text{on} \rangle$ at B , in the context of Example 2, can be represented in the language $\text{Traces}_{\text{NNF}, \text{CSTRIPS}_{\text{Terms}, \text{Terms}, \text{Terms}}}$ as the expression $\beta := \langle \iota, \alpha\omega_1, \alpha\omega_2 \rangle$, where

- ι is the initial belief state expression $(x_1 \leftrightarrow \neg x_2) \wedge (y_1 \leftrightarrow \neg y_2)$
- $\alpha\omega_1$ is the action expression $\langle \pi, \eta \rangle$, with $\pi = \top$ and

$$\eta = (\top \triangleright ((+y_1 \ \& \ -y_2) \cup (-y_1 \ \& \ +y_2))) \ \& \ (x_1 \triangleright (+x_2 \ \& \ -x_1))$$

- $\alpha\omega_2$ is the observation expression sw_on .

The action $\alpha\omega_1$ is applicable at (the belief state represented by) trace $\langle \iota \rangle$ since its precondition is always satisfied. The observation $\alpha\omega_2$ is fair at (the belief state represented by) trace $\langle \iota, \alpha\omega_1 \rangle$, since sw_on is consistent with the resulting belief state.

Throughout this section, for a sequence of states $t := \langle s_0, s_1, \dots, s_k \rangle$ (also called a *trajectory*) and a trace $\beta := \langle \iota, \alpha\omega_1, \alpha\omega_2, \dots, \alpha\omega_k \rangle$, we say that t *satisfies* β , and write $t \models \beta$, if (i) s_0 satisfies ι , (ii) for $i = 1, \dots, k$, if $\alpha\omega_i$ is an action expression, then $s_i \in \|\alpha\omega_i\|(s_{i-1})$ holds, and (iii) for $i = 1, \dots, k$, if $\alpha\omega_i$ is an observation expression, then $s_i = s_{i-1}$ and $s_i \in \|\alpha\omega_i\|$ holds.

Complexity Results

As we shall see, most complexity results can be easily inferred from the complexity for propositional languages. The results are summarized in Table 10.

Similarly to Section 6, we will tacitly use the fact that for all languages $\Lambda_1^I, \Lambda_2^I, \Lambda^O, \Lambda^A$, if $\Lambda_1^I \subseteq \Lambda_2^I$ holds then $\text{Traces}_{\Lambda_1^I, \Lambda^A, \Lambda^O} \subseteq \text{Traces}_{\Lambda_2^I, \Lambda^A, \Lambda^O}$ holds, and similarly for $\Lambda_1^O \subseteq \Lambda_2^O$ or $\Lambda_1^A \subseteq \Lambda_2^A$.

We start with upper bounds; for these we will use the following lemma.

Lemma 71 *For all propositional languages Λ^I, Λ^O , and for all action languages Λ^A with $\Lambda^A \subseteq \text{NNFAT}$ or $\Lambda^A \subseteq \text{PDDL}_{\text{NNF}, \text{NNF}}$, the problem of deciding $s \in \|\beta\|$ for given $s \in 2^F$ and $\beta \in \text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$, is in NP.*

Proof First observe that for all action languages Λ^A as in the statement, deciding whether for a given expression $\alpha \in \Lambda^A$ and given states $s_1, s_2 \in 2^F$, $s_2 \in \|\alpha\|(s_1)$ holds, is a problem in NP. Indeed, it is clearly in P for action theories, since deciding $s_2 \in \|\alpha\|(s_1)$ amounts to deciding $s_1 \cup (s_2)' \models \alpha$. Now for $\text{PDDL}_{\text{NNF}, \text{NNF}}$, this amounts to guessing a child for each U-node, then simulating the resulting *deterministic* action on s_1 (in polynomial time), and finally deciding whether the resulting state is s_2 . Now for all propositional initial state languages Λ^I (resp. for all propositional observation languages Λ^O), deciding whether a given state s is in $\|\iota\|$ for a given expression $\iota \in \Lambda^I$ (resp. in $\|\omega\|$ for a given $\omega \in \Lambda^O$) is clearly in P. Hence we can decide $s \in \|\beta\|$ by guessing a trajectory t and the witnesses of $s_2 \in \|\alpha\|(s_1)$ all along t , and verifying that t satisfies β with these witnesses. \square

Table 10 Complexity results for queries and transformations with $\Lambda^B = \text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$

	powerful (Def. 73)	$\Lambda^I = \text{DNF/OBDD}_{<}$ $\Lambda^A = \text{DNFAT}$ $\Lambda^O = \text{Terms}$	$\Lambda^I = \text{DNF}$ or $\text{OBDD}_{<}$ or Terms; arbitrary Λ^A, Λ^O
APPLIC (NNFAT/CNFAT)	$\Pi_2\text{P-c}$	$\Pi_2\text{P-c}$	$\Pi_2\text{P-c}$
APPLIC (DNFAT)	coNP-c	coNP-c	coNP-c
APPLIC(Λ^A) for $\Lambda^A = \text{OBDDAT}_{<}$ or $\text{P/C}_{\Lambda^P, \Lambda^C}$ or $\text{P/C-Det}_{\Lambda^P, \Lambda^C}$	coNP-c, except $\text{PDDL}_{\top, \text{NNF}}$: P (obvious)	see Table 3	open
FAIR (NNF/CNF)	NP-c	NP-c	NP-c
FAIR (D/O/T)	NP-c	P	open
$\text{SAT}_{\mathcal{S}}$ (NNF/DNF)	coNP-c	coNP-c	coNP-c
$\text{SAT}_{\mathcal{S}}$ (C/O/T)	coNP-c	P	open
SAT	$\Theta_2\text{P-c}$	see Table 6	open
EQUIV	$\Pi_2\text{P-c}$	coNP-c	open
$\text{PROG}_a(\Lambda^A), \text{PROG}_o(\Lambda^O)$	trivial	trivial	trivial

C/O/T stands for CNF/OBDD_</Terms; D/O/T stands for DNF/OBDD_</Terms; $\text{P/C}_{\Lambda^P, \Lambda^C}$ stands for $\text{PDDL}_{\Lambda^P, \Lambda^C}$ or $\text{CSTRIPS}_{\Lambda^P, \Lambda^C}$; and $\text{P/C-Det}_{\Lambda^P, \Lambda^C}$ stands for $\text{PDDL-Det}_{\Lambda^P, \Lambda^C}$ or $\text{CSTRIPS-Det}_{\Lambda^P, \Lambda^C}$. The line about progression assumes that the action or observation progressed by is in the language used by the trace

The results for the first column follow from Proposition 72 for membership, and Corollary 75 for hardness; for the second column, from Propositions 76/77 and the results in Section 6; and for the third column, from Proposition 78 and the results in Section 6

In the cells marked “open”, the complexity of some, but not all, subcases, can be derived from the results in this section and in Section 6

In the rest of this section, to avoid ambiguity we use notation Λ^A, Λ^O for languages of the *queries*, as opposed to those used in the traces.

Proposition 72 *For all propositional languages $\Lambda^I, \Lambda^O, \Lambda^G, \Lambda^+, \Lambda^-$ and all action languages Λ^A, Λ^A with $\Lambda^A \subseteq \text{NNFAT}$ or $\Lambda^A \subseteq \text{PDDL}_{\text{NNF}, \text{NNF}}$ and $\Lambda^A \subseteq \text{NNFAT}$ or $\Lambda^A \subseteq \text{PDDL}_{\text{NNF}, \text{NNF}}$, the following holds:*

1. the problem $\text{APPLIC}(\text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}, \Lambda^A)$ is in $\Pi_2\text{P}$;
2. if in addition $\Lambda^A \in \{\text{DNFAT}, \text{OBDDAT}_{<}, \text{PDDL}_{\text{NNF}, \text{NNF}}\}$ holds, then the problem $\text{APPLIC}(\text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}, \Lambda^A)$ is in coNP;
3. the problem $\text{FAIR}(\text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}, \Lambda^O)$ is in NP;
4. the problem $\text{SAT}_{\mathcal{S}}(\text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}, \Lambda^G)$ is in coNP;
5. the problem $\text{SAT}(\text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}, S5_{\Lambda^+, \Lambda^-})$ is in $\Theta_2\text{P}$;
6. the problem $\text{Equiv}(\text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O})$ is in $\Pi_2\text{P}$.

Proof We show each item in turn.

1. Given $\beta \in \text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$ of length k and $\alpha \in \Lambda^A$, $\|\alpha\|$ is not applicable at $\|\beta\|$ if and only if there is a trajectory $t = \langle s_0, s_1, \dots, s_k \rangle$ which satisfies β , and is such that $\|\alpha\|$ is not applicable at s_k . Since deciding $t \models \beta$ is in NP by Lemma 71, the latter assertion holds if and only if there is a trajectory $t = \langle s_0, s_1, \dots, s_k \rangle$ and a polynomial-size witness, such

- that (i) this is indeed a witness of $t \models \beta$, and (ii) $\|\alpha\|$ is not applicable at s_k . Condition (i) can be verified in polynomial time, and Condition (ii) can be verified using a coNP-oracle for verifying that for no $s' \in 2^{F'}$ does $s' \in \|\alpha\|(s)$ hold (for PDDL, a polynomial-time check that s_k does not satisfy the precondition suffices). Hence the complement of the problem is in $\text{NP}^{\text{coNP}} = \text{NP}^{\text{NP}} = \Sigma_2\text{P}$, so that the problem is in $\Pi_2\text{P}$.
2. For all the given action languages Λ^A , the problem of deciding whether $\alpha \in \Lambda^A$ is applicable at $s \in 2^F$, is in P. Hence using the same reasoning as for Item 1, we conclude that the problem is in coNP.
 3. Given $\beta \in \text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$ of length k and $\omega \in \Lambda^O$, $\|\omega\|$ is fair at $\|\beta\|$ if and only if there exists a trajectory $t = \langle s_0, s_1, \dots, s_k \rangle$ which satisfies β , and is such that s_k satisfies ω ; since deciding $s_k \models \omega$ is in P for all considered observation languages, reasoning as for Item 1 we get that the problem is in NP.
 4. Given $\beta \in \text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$ of length k and $\gamma \in \Lambda^G$, $\|\beta\|$ does *not* satisfy γ if and only if there exists a trajectory $t = \langle s_0, s_1, \dots, s_k \rangle$ which satisfies β , and is such that s_k does *not* satisfy γ ; since deciding $s_k \models \gamma$ is in P for all considered goal languages, reasoning as for Item 1 we get that the complement of the problem is in NP, hence the problem is in coNP.
 5. Given $\beta \in \text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$ and $\vec{\gamma} \in S5_{\Lambda^+, \Lambda^-}$, we can evaluate whether β satisfies κ for each atom in $\vec{\gamma}$ using an NP-oracle: indeed, for $\kappa = K\varphi$ we decide whether β entails φ , and for $\kappa = \neg K\varphi$ we decide whether $\beta \wedge \neg\varphi$ is satisfiable. This constitutes a polynomial number of independent calls to an NP-oracle. Then deciding whether β satisfies $\vec{\gamma}$ amounts to combining the answers to these calls following the structure of $\vec{\gamma}$, in polynomial-time, hence the problem is in $\text{P}^{\text{NP}} = \Theta_2\text{P}$.
 6. Given $\beta_1, \beta_2 \in \text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$, deciding whether $\|\beta_1\| = \|\beta_2\|$ does *not* hold can be done by deciding whether there exists a trajectory t_1 which satisfies β_1 , and is such that there does not exist a trajectory t_2 which satisfies β_2 and in which the last state is the same as in t_1 (or vice-versa, swapping the roles of β_1 and β_2). Hence we can show that $\|\beta_1\| = \|\beta_2\|$ does *not* hold by guessing t_1 and a witness that it satisfies β_1 , and verifying that for all trajectories t_2 and all candidate witnesses that t_2 would satisfy β_2 , (i) the witness for t_1 is indeed a witness, and (ii) the witness for t_2 is not a witness, or the last states of t_1 and t_2 are not the same. Since checking Conditions (i) and (ii) can be done in polynomial time (Lemma 71), we get that the complement of the problem is in $\Sigma_2\text{P}$, and hence the problem is in $\Pi_2\text{P}$.

□

For the next results, we will use the following definition, which is motivated by the lemma that follows.

Definition 73 (powerful trace language) Let F be a set of fluents, Λ^I, Λ^O be propositional languages, and Λ^A be an action language. The trace language $\text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$ is said to be *powerful* if one of the following holds:

- $\Lambda^I \supseteq \text{BS}(\text{CNF})$ and $\Lambda^A \supseteq \text{CSTRIPS-Det}_{\top, \top}$ (and Λ^O is arbitrary, possibly empty); or
- there is a polynomial-size representation (in the number of fluents) of \top in Λ^I ; $\Lambda^O \supseteq \text{Terms}$ holds; and Λ^A is a superset of CNFAT , $\text{OBDDAT}_{<}$ with $F \parallel_{<} F'$, or $\text{CSTRIPS-Det}_{\top, \text{Terms}}$.

We now show a technical lemma, which essentially says that traces in a powerful language can efficiently represent existential CNF belief states. For a fluent x , we write a_x^\top for the action which sets x to \top and leaves the value of all other fluents unchanged.

Lemma 74 *Let $\text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$ be a powerful trace language. Then for all sets of variables V, Y , and all satisfiable propositional CNF formulas ψ over $V \cup Y$, there are a set of variables Y' and an expression β in $\text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$ over $F := V \cup Y \cup Y'$, such that $\|\beta\| = \|(\exists Y \cdot \psi) \wedge \bigwedge_{x \in Y \cup Y'} x\|$ holds, and Y', β can be computed in polynomial time.*

Proof For the whole proof, write $Y := \{x_1, \dots, x_k\}$. Observe that for all action languages under consideration, and for all variables x , there is a polynomial-size expression α_x^\top for the action α_x^\top ; for instance, $\alpha_x^\top := x' \wedge \bigwedge_{y \neq x} ((\neg y \vee y') \wedge (y \vee \neg y'))$ is suitable for CNFAT, and $\alpha_x^\top := +x$ is suitable for CSTRIPS-Det $_{\top, \top}$. Also observe that these actions are applicable at all states.

For $\Lambda^I \supseteq \text{BS}(\text{CNF})$ and $\Lambda^A \supseteq \text{CSTRIPS-Det}_{\top, \top}$, we define $Y' := \emptyset$ and $\beta := (\psi, +x_1, \dots, +x_k)$. Clearly, the result holds.

Now for $\Lambda^A \supseteq \text{CNFAT}$, write $\psi := \bigwedge_{i=1}^m \gamma_i$, and define Y' to be the set of fluents $Y' := \{c_i \mid i = 1, \dots, m\}$, where the c_i 's are fresh variables (to be read “the i th clause of ψ is satisfied”). For $i = 1, \dots, m$, let $\alpha_i \in \text{CNFAT}$ be defined by (using \leftrightarrow for readability)

$$\alpha_i := (\gamma_i \vee \neg c'_i) \wedge \left(\bigwedge_{\lambda \in \gamma_i} (\neg \lambda \vee c'_i) \right) \wedge \left(\bigwedge_{x \in V \cup Y} (x \leftrightarrow x') \right) \wedge \left(\bigwedge_{j \in \{1, \dots, m\} \setminus \{i\}} (c_j \leftrightarrow c'_j) \right)$$

By definition, α_i does not modify the value of the fluents $x \in V \cup Y$, and sets c_i according to whether γ_i is satisfied; moreover, it is applicable at all states. Hence after progressing $\iota := \top$ by $\alpha_1, \dots, \alpha_m$, we get the belief state $\{s \in 2^{V \cup Y \cup Y'} \mid s \models \psi, s \models \bigwedge_{i=1}^m c_i\} \cup \{s \in 2^{V \cup Y \cup Y'} \mid s \not\models \psi, s \not\models \bigwedge_{i=1}^m c_i\}$. Now define the observation $\omega := (\bigwedge_{i=1}^m c_i) \in \Lambda^O$, and the trace $\beta := (\top, \alpha_1, \alpha_2, \dots, \alpha_m, \omega, \alpha_{x_1}^\top, \dots, \alpha_{x_k}^\top)$. Since ψ is satisfiable, $\|\omega\|$ is indeed fair to the beginning of the trace, and we have $\|\beta\| = \{s \in 2^{V \cup Y \cup Y'} \mid s \models (\exists Y \cdot \psi) \wedge \bigwedge_{i=1}^m c_i \wedge \bigwedge_{i=1}^m c_i\}$, as desired.

Finally, for $\Lambda^A \supseteq \text{OBDDAT}_{<}$ with $F \parallel_{<} F'$ and for $\Lambda^A \supseteq \text{CSTRIPS-Det}_{\top, \text{Terms}}$, we simply observe that the action $\|\alpha_i\|$ also has a polynomial-size representation, and reason as for $\Lambda^A \supseteq \text{CNFAT}$. \square

Corollary 75 *Let $\Lambda^B = \text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$ be a powerful trace language. Then the following hold:*

1. for $\Lambda^A \in \{\text{NNFAT}, \text{CNFAT}\}$, the problem $\text{APPLIC}(\Lambda^B, \Lambda^A)$ is $\Pi_2\text{P-hard}$;
2. for $\Lambda^A \in \{\text{DNFAT}, \text{OBDDAT}_{<}, \text{CSTRIPS-Det}_{\text{Terms}, \top}\}$, the problem $\text{APPLIC}(\Lambda^B, \Lambda^A)$ is coNP-hard ;
3. the problem $\text{FAIR}(\Lambda^B, \text{Terms})$ is NP-hard ;
4. the problem $\text{SAT}_{\mathcal{S}}(\Lambda^B, \text{Terms})$ is coNP-hard ;
5. the problem $\text{SAT}(\Lambda^B, \text{S5}_{\text{Terms}, \text{Terms}})$ is $\Theta_2\text{P-hard}$;
6. the problem $\text{Equiv}(\Lambda^B)$ is $\Pi_2\text{P-hard}$.

Proof Observe first that for all cases, by the results in Section 6 the corresponding problem is hard for the belief state language is $\text{BS}_{\exists}(\text{CNF})$ instead of Λ^B . Hence for each problem, it is enough to give a polynomial-time reduction from the problem with a belief state in $\text{BS}_{\exists}(\text{CNF})$.

For this, let F be a set of fluents, Y be a set of auxiliary variables, and let β^0 be an expression in $\text{BS}_{\exists}(\text{CNF})$ (hence with $\|\beta^0\| = \exists Y \cdot \|\beta^0\|_{F \cup Y}$). Using Lemma 74, we compute in polynomial time a set of variables Y' and an expression $\beta \in \Lambda^B$, such that $\|\beta\| = \|(\exists Y \cdot \beta^0) \wedge \bigwedge_{x \in Y \cup Y'} x\|$ holds.

Let $\mathbf{1}_{Y \cup Y'} \in 2^{Y \cup Y'}$ be the assignment of \top to all variables in $Y \cup Y'$.

1. For applicability, we have that $\exists s' \in 2^F : s' \in \|\alpha\|(s)$ is equivalent to $\exists s' \in 2^{F \cup Y \cup Y'} : s' \in \|\alpha\|_{F \cup Y \cup Y'}(s \cup \mathbf{1}_{Y \cup Y'})$, since in NNFAT and in CNFAT the fluents not mentioned in

- α can take any value. It follows that α is applicable at $\|\beta^0\|$ if and only if it is applicable at $\{s \cup \mathbf{1}_{Y \cup Y'} \mid s \in \|\beta^0\|\} = \|\beta\|$. This gives the desired reduction.
2. Idem.
 3. For fairness, we have that $s \in \|\omega\|$ is equivalent to $s \cup \mathbf{1}_{Y \cup Y'} \in \|\omega\|_{F \cup Y \cup Y'}$, since ω does not mention any variable in $Y \cup Y'$; hence again this gives the desired reduction.
 4. For statewise goals, we reason as for applicability.
 5. For S5 goals, we reason as for applicability for positive atoms $K\varphi$, and as for fairness for negative atoms $\neg K\varphi$.
 6. For equivalence, given β_1^0 over $F \cup Y_1$ and β_2^0 over $F \cup Y_2$, with $\|\beta_1^0\| = \exists Y_1 \cdot \|\beta_1^0\|_{F \cup Y_1}$ and $\|\beta_2^0\| = \exists Y_2 \cdot \|\beta_2^0\|_{F \cup Y_2}$, using Lemma 74 we compute in polynomial time $Y'_1, Y'_2, \beta_1, \beta_2$ such that $\|\beta_1\| = \|(\exists Y_1 \cdot \beta_1^0) \wedge \bigwedge_{x \in Y_1 \cup Y'_1} x\|$ and $\|\beta_2\| = \|(\exists Y_2 \cdot \beta_2^0) \wedge \bigwedge_{x \in Y_2 \cup Y'_2} x\|$ hold. We moreover ensure that Y_1, Y'_1, Y_2, Y'_2 are pairwise disjoint, by renaming variables as necessary. Now let β'_1 (resp. β'_2) be the trace obtained from β_1 (resp. β_2) by appending action expressions for a_x^\top for all $x \in Y'_2$ (resp. for all $x \in Y'_1$). Then we get $\|\beta'_1\| = \|(\exists Y_1 \cdot \beta_1^0) \wedge \bigwedge_{x \in Y \cup Y'_1 \cup Y'_2} x\|$ and $\|\beta'_2\| = \|(\exists Y_2 \cdot \beta_2^0) \wedge \bigwedge_{x \in Y \cup Y'_1 \cup Y'_2} x\|$. Hence we have $\exists Y_1 \cdot \|\beta_1^0\| = \exists Y_2 \cdot \|\beta_2^0\|$ if and only if β'_1 and β'_2 are equivalent, which gives the desired reduction.

□

Proposition 76 *For all decision problems, the restriction to input belief states in DNF and the restriction to input belief states in $\text{Traces}_{\text{DNF}, \text{DNFAT}, \text{Terms}}$ are polynomial-time reducible to each other.*

Proof An input β in DNF can be emulated by the trace $\langle \beta \rangle$, that is, the trace reduced to the initial belief state β . Conversely, given a trace $\beta := \langle \iota, ao_1, ao_2, \dots, ao_k \rangle$ in $\text{Traces}_{\text{DNF}, \text{DNFAT}, \text{Terms}}$, we can progress ι iteratively using Lemma 49 for actions and conditioning and conjunction for observations. By construction, this yields a belief state $\beta' \in \text{DNF}$ with $\|\beta'\| = \|\beta\|$, and it is easy to see that the iterated progression can be performed in polynomial time (in particular, the size of the belief state does not blow up with k , as it remains less than the size of the last action theory progressed with). □

Proposition 77 *For all decision problems, the restriction to input belief states in DNF and the restriction to input belief states represented by traces in $\text{Traces}_{\text{OBDD}_<, \text{DNFAT}, \text{Terms}}$ containing at least one action, are polynomial-time reducible to each other.*

Proof An input β in DNF can be emulated by the trace $\langle \top, \beta[F/F'] \rangle$, that is, the trace reduced to the initial belief state \top followed by an action producing exactly β . Conversely, given a trace $\beta := \langle \iota, ao_1, ao_2, \dots, ao_k \rangle$ in $\text{Traces}_{\text{OBDD}_<, \text{DNFAT}, \text{Terms}}$ containing at least one action, we can progress ι iteratively using Lemma 49 for actions and conditioning and conjunction for observations. We conclude as in the proof of Proposition 76 (the important point is that we progress an OBDD by a DNF action theory into a DNF belief state). □

Using Propositions 76 and 77 allows the transfer to traces of a number of results from Section 6, which we report in Table 10; for $\Lambda^1 = \text{OBDD}_<$, to cope with traces containing no action (and hence, in which iterated progression gives a binary decision diagram again), we use the fact that, as it turns out, the complexity of queries is the same for DNF and $\text{OBDD}_<$ (except for *Equiv*, but it suffices to observe that the problem is already coNP-complete for DNF).

Finally, considering the empty trace, we immediately have the following, which settles further cases, precisely those for which the complexity is already the hardest for Λ^1 . We leave the remaining cases open for future work.

Proposition 78 Let $\Lambda^B := \text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$ be a trace language. Then for all decision problems, there is a polynomial-time reduction from the restriction of the problem to input belief states in Λ^I to the restriction of the problem to input belief states in Λ^B .

8 Succinctness

In this section we discuss the relative succinctness of languages for representing belief states. We focus on the relative succinctness of trace languages vs propositional representations, since the relationships between propositional representations are mostly directly derivable from the results by [3]. Note that existential representations are not studied there, however, we know that

- $\text{BS}_{\exists}(\Lambda)$ is at least as succinct as $\text{BS}(\Lambda)$, for all complete propositional languages Λ ;
- $\text{BS}_{\exists}(\text{CNF})$ is at least as succinct as all other complete — plain or existential — propositional representations (Lemma 47), and hence so is $\text{BS}_{\exists}(\text{NNF})$;
- $\text{BS}_{\exists}(\text{DNF})$ is equally succinct as $\text{BS}(\text{DNF})$ (essentially because forgetting one variable can be done in linear time), and $\text{BS}_{\exists}(\text{OBDD}_{<})$ is equally succinct as $\text{BS}(\text{OBDD}_{<})$ for $F < Y$ (Lemma 23);
- if existential forgetting is not polynomial-size for a propositional language Λ , then $\text{BS}(\Lambda)$ is not at least as succinct as $\text{BS}_{\exists}(\Lambda)$.

Of course, this does not cover all pairs of languages (especially when $\text{BS}_{\exists}(\text{OBDD}_{<})$ is involved with $Y < F$), but a complete picture is out of the scope of this paper.

We now give a number of results relating traces with propositional representations in terms of succinctness. The first of these is obvious.

Proposition 79 Let $\Lambda^I, \Lambda_A, \Lambda^O$ be complete propositional languages; we have $\text{Traces}_{\Lambda^I, \text{AT}(\Lambda_A), \Lambda^O} \preceq^s \text{BS}(\Lambda^I)$. Moreover, if \top has a polynomial-size representation (in the number of variables in F) in Λ^I , then the following hold: $\text{Traces}_{\Lambda^I, \text{AT}(\Lambda_A), \Lambda^O} \preceq^s \text{BS}(\Lambda_A)$, $\text{Traces}_{\Lambda^I, \text{AT}(\Lambda_A), \Lambda^O} \preceq^s \text{BS}(\Lambda^O)$.

The next two results allow the derivation of results about relative succinctness from results about the complexity of transformations. Recall that for a set of fluents F and $x \in F$, we write a_x for the action which sets x to \top and leaves the value of all other fluents unchanged.

Proposition 80 Let Λ^I, Λ^O be propositional languages, and let Λ^A be an action language in which there is a polynomial-size representation (in the number of fluents) of a_x , for all x . If existential forgetting is not polynomial size in Λ^I , then $\text{BS}(\Lambda^I) \not\preceq^s \text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$ holds.

Proof To every expression $\beta \in \text{BS}(\Lambda^I)$ and set of variables $V := \{x_1, \dots, x_k\} \subseteq V(\beta)$, we associate the trace $\beta' := \langle \beta, \alpha_{x_1}, \alpha_{x_2}, \dots, \alpha_{x_k} \rangle \in \text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$, where α_{x_i} is a representation of a_{x_i} in Λ^A . Let ϕ be a Λ^I expression representing β' . By construction of β' , the semantics of ϕ is $\|(\exists V \cdot \beta) \wedge \bigwedge_{x \in V} x\|$, and hence the expression $\phi' := \phi|_{V=\top}$ satisfies $\|\phi'\| = \exists V \cdot \|\beta\|$. Since conditioning is polynomial in propositional languages, we get that ϕ cannot be of polynomial size. \square

A consequence of Proposition 80 is that $\text{BS}(\text{OBDD}_{<})$ is not at least as succinct as $\text{Traces}_{\text{OBDD}_{<}, \text{CSTRIPS-DetTerms}, \top, \Lambda^O}$, or, said otherwise, that it is not always worth explicitly progressing a belief state as an OBDD (by classical STRIPS actions) as compared to storing the history, as far as succinctness is concerned.

Proposition 81 *Let Λ^I, Λ^O be propositional languages, and let Λ^A be an action language. Then $\text{BS}(\Lambda^I) \preceq^s \text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$ holds if and only if iterated progression of expressions in $\text{BS}(\Lambda^I)$ by action expressions in Λ^A and observation expressions in Λ^O is a polynomial-size transformation.*

Proof Assume first $\text{BS}(\Lambda^I) \preceq^s \text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$. Then for all families of expressions $\beta \in \Lambda^I$ and histories $\langle \alpha\omega_1, \dots, \alpha\omega_k \rangle$, with $\alpha\omega_i \in \Lambda^A \cup \Lambda^O$ for $i = 1, \dots, k$, the expression $\beta' := \langle \beta, \alpha\omega_1, \alpha\omega_2, \dots, \alpha\omega_k \rangle$ is a trace in $\text{Traces}_{\Lambda^I, \Lambda^A, \Lambda^O}$ and hence, by the assumption, for all families of such expressions, there is an equivalent family of polynomial size expressions in $\text{BS}(\Lambda^I)$. The result follows from the fact that $\|\beta'\|$ is exactly the iterated progression of β by $\alpha\omega_1, \dots, \alpha\omega_k$.

The converse is shown similarly. \square

From Proposition 81, and from the results in Section 6, we can conclude that $\text{BS}(\text{DNF})$ is at least as succinct as $\text{Traces}_{\text{DNF}, \text{DNFAT}, \text{Terms}}$, that is, in this case it is worth maintaining the explicit belief state.

To conclude this section, we observe that planners which use traces typically have a fixed initial belief state (the one of the planning problem at hand) and a fixed set of actions and observations (the available ones). Hence it seems interesting to study the relative succinctness of traces in that precise setting. However, it is easy to see that in general, traces cannot be succinct representations of belief states in that setting, due to the fact that there are planning problems which require exponentially long plans (in the classical setting) or doubly exponentially long plans (in the conformant setting). As a simple example, consider an initial (singleton) belief state where all variables are assigned to \perp , and a set of deterministic actions allowing to transform the current state into the subsequent one in a lexicographic order. Then the smallest representation of the (singleton) belief state β , where all variables are assigned to \top , by a trace, is exponentially long, while β itself has linear size.

Still, if plan length is bounded (like in SatPlan), it may be worth studying the succinctness of traces relative to explicit belief states. We leave this as an interesting direction for future work.

9 Nondeterministic planners

In this section, we review existing planners from the literature, in terms of the languages, queries, and transformations that they use. Taking advantage of our results to improve some of them with more tractable languages for their internal representations, or to generalize them to more general input languages while retaining the complexity of operations, is indeed a promising matter of future work.

Most planners under incomplete information use variations of the PDDL syntax as input [4]. However, in practice, they do not efficiently support general PDDL as we define it, as they translate PDDL into Conditional STRIPS. For many planners, the initial belief state is expected to be in CNF. In many benchmarks, that initial belief state contains exclusive disjunctions (`oneof`), that some planners exploit for further efficiency. Finally, most contingent planners that accept Conditional STRIPS actions, accept sensing actions of the form $\langle \pi, \omega \rangle$, where π is a precondition and ω or $\neg\omega$ are the possible observations.

9.1 Search in belief state space

We first consider planners that rely on forward search from the initial belief state as illustrated in Algorithm 1.

GPT introduced heuristic search in belief state space [28]. It accepts planning problems in a language close to Functional STRIPS, featuring numerical variables and probabilities [29], supporting nondeterminism and sensing actions. GPT represents beliefs states as a list of possible states, equivalent to the propositional language MODS [3].

Planners using OBDDs to represent belief states include HSCP [12], KACMBP [30], and POND [31]. All of them support nondeterministic actions. HSCP and KACMBP accept problems in a language of action theories used in model checking (NuSMV, [32]). In contrast, POND receives PDDL with nondeterministic effects.

Some planners also use DNF or CNF to represent belief states. To et al. [33, 34] propose conformant planners, while [35] propose a contingent planner. They all receive deterministic Conditional STRIPS. The algorithms aim to reduce the size of the belief state by continuous simplification. They rely on the initial belief state being in CNF. Most deterministic conformant and contingent benchmarks are in prime implicate form, enabling more efficient updates of CNF belief states. It is worth noting that in practice, benchmarks with initial belief states using exclusive disjunctions are already in prime implicate form [9].

Finally, the implicit representation of belief states by *traces* was introduced with Conformant-FF [27], which accepts deterministic Conditional PDDL. Albore et al. [36] also propose a planner that uses traces and accepts deterministic Conditional PDDL.

9.2 Action theories and logical approaches

We now consider planners which do not rely directly on and-or search.

Some planners build formulas describing the transition function using variations of the action language NNFAT, while others use formulas related to the formulation used by SatPlan [37]. Rintanen [38] describes QBFPlan, which builds QBF formulas for solving conformant and contingent planning with nondeterministic actions. The QBF problems are solved using a standalone theorem prover.

Cimatti et al. [39] describe CMBP, which simultaneously expands all the actions using a representation in $OBDDAT_{<}$. They propose an algorithm operating by progression ($PROG_a$) and another operating by regression. Additional heuristics are implemented as logical operations on OBDDs.

The approach by [40] creates a formula similar to that of SatPlan for solving conformant problems with a given makespan, for problems expressed in Conditional STRIPS. The formula is compiled into d-DNNF, a language related to $OBDD_{<}$ [3]. Since the formula encodes all plans for all the possible initial states, they rely on search in the space of possible plans using model counting operations to prune nodes. Palacios and Geffner [41] use the same d-DNNF formulation, but create a new NNF built as the logical conjunction of the subcircuits expressing the plans for each initial state. The NNF is translated into CNF using auxiliary variables, and a single call to a SAT solver yields a conformant plan, if any, for the given makespan. The formulas proposed by [41] can also be used to create a QBF formula whose models are conformant plans [42].

9.3 Other approaches

We finally review some approaches which do not directly fit our framework, but are still related to it.

Palacios and Geffner [9] present a representation for conformant planning, later extended to contingent planning [10]. Both receive problems in deterministic Conditional STRIPS, accepting an initial belief state in CNF. The idea is to track the relationship between literals in the current belief state and terms in the initial belief state using atoms of the form $k\lambda/\chi$ in a language which we here denote by K_{Λ^I, Λ^C} . Informally, if the atom $k\lambda/\chi$ is in the representation of a belief state B , this means that if χ was satisfied by the actual (unknown) initial state, then λ must be true in B . In particular, if λ is true for a set of χ_i such that $\bigvee_i \chi_i$ is entailed by the initial belief state, then λ must be true in the current belief state.

Palacios and Geffner [9] hence introduce the K_k translation of an instance. This translation is an instance of K_{Λ^I, Λ^C} , with Λ^C corresponding to consistent terms spanning k clauses in the initial belief state expression ι , so that the disjunction of those χ_i 's is entailed by ι . Verifying the consistency of χ_i amounts to checking whether $\iota \wedge \chi_i$ is satisfiable, a problem in NP for ι in CNF. Palacios and Geffner [9] assume that ι is in prime implicate form, so the consistency check becomes polynomial [3].

More recently, [43, 44] tackle conformant planning by attempting to get a plan for a subset of the initial belief state, using the same language as [9]. A SAT solver is used to verify if a candidate plan is conformant; if the answer is negative, the model encodes an initial state where the conformant planner fails. If a conformant plan was obtained for a subset of initial states, those initial states are called a *basis*. In turn, a basis can be mapped into a set conditions $\{\chi_i\}$ so that the initial belief state ι entails $\bigvee_{i=1, \dots, k} \chi_i$, enabling efficient planners that incrementally build a complete and possibly compact translation.

Finally, [2] present algorithms for belief tracking for both conformant and contingent planning with nondeterministic actions and multi-valued variables. [45] present a planner which combines ideas of previous effective belief tracking techniques, focusing on efficiency. [46, 47] rely on state sampling, leading to an unsound but effective algorithm based on replanning [48], in contrast with [36], who use an unsound translation but represent belief states as traces. All these works exploit properties at the level of domains and problems, which is out of the scope of the present paper but opens interesting perspectives.

10 Conclusion

We have studied the complexity of queries and transformations pertaining to belief tracking and to planning by forward search in the space of belief states, for a number of representations of belief states, actions, observations, and goals, as well as the relative succinctness of these languages. Our study covers both explicit and implicit representation of belief states, and both logical and imperative (PDDL-like) representations of actions. Beyond the planning problem, our queries and transformations cover belief tracking with observations not triggered by the agent's actions, but by external events. This can be useful for applications where we need to monitor a situation, not necessarily for planning.

Overall, we give an almost complete picture of complexity and succinctness. This picture allows one to consider the most adequate combination of languages for designing an algorithm or an implementation which uses our queries and transformations. For instance, planners representing belief states in DNF or OBDD_< might be extended to epistemic goals expressed

in S5, with polynomial precondition and goal checking in some non-trivial cases (see Table 6). On the other hand, such extensions seem less promising for planners using NNF or CNF representations. Further investigation on prime implicate forms and on d-DNNF might reveal some more tractable cases. The complexity results in fairness, and progression of actions and observations, are also interesting for reusing the representations that planners use for monitoring, execution, and dealing with external events and actions beyond the intent of the agent [49]. This might lead to highest forms of robustness beyond the initial uncertainty and the actions the agent can perform. Finally, it can be seen from our results that $BS_{\exists}(OBDD_{<})$, with $Y < F$, has interesting computational properties; it might be worth investigating its practical use in planners.

We have considered planning by progression; it would naturally be useful to complete the picture by queries and transformations pertaining to planning by regression, that is, by backwards search in the space of belief states. The main difference with our study is that regression in general produces several belief states instead of a single one. Backwards search for planning typically filters the regression to include only the actions that modified the states, so studying regression might need to consider multiple variants of the transformation. More generally, an important perspective of this work is to study representations of *sets* of belief states rather than belief states in isolation. Apart from regression, this would be useful to planning by progression, for considering approaches which progress a *frontier* of reachable belief states, like SatPlan [37] and generalizations to conformant problems [30, 41]. For such representations, natural languages, beyond mere lists of belief states, are representations tailored to the logic of knowledge S5, in particular S5-DNF [23] and epistemic splitting diagrams [50].

Generalizing our study to sets of belief states would also be of particular interest for encompassing planning with PDDL at the relational level and lifted planning, in which a plan is searched for a class of instances of a planning problem, rather than for a single ground instance [51, 52]. Similarly, sets of belief states are also useful when considering generalized planning, which aims to obtain a single policy for a set of problems from the same lifted problem [53–55].

Finally, an interesting perspective is to consider approaches where the choice of a representation language can be “problem aware”, that is, dependent on the initial state, on the available actions, and on the goal of the problem. Such a setting indeed allows one to consider representations which take advantage of a small *width* of the problem at hand, like the approaches by [2, 9, 10] for conformant and contingent planning, and for belief tracking as discussed in Section 9.3. While the worst-case complexity of such approaches matches the known complexities of the problems, situations are characterized where planning and belief tracking are provably more tractable. Hence considering these approaches from a knowledge compilation perspective would open interesting research avenues.

Funding This work has been supported by the French National Research Agency (ANR) through project PING/ACK (ANR-18-CE40-0011).

Data availability Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Geffner, H., Bonet, B.: A concise introduction to models and methods for automated planning. *Synth. Lect. Artif. Intell. Mach. Learn.* **8**(1), 1–141 (2013)
2. Bonet, B., Geffner, H.: Belief tracking for planning with sensing: Width, complexity and approximations. *J. Artif. Intell. Res.* **50**, 923–970 (2014)
3. Darwiche, A., Marquis, P.: A knowledge compilation map. *J. Artif. Intell. Res.* **17**, 229–264 (2002)
4. McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D.: PDDL – The Planning Domain Definition Language – Version 1.2. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University (1998)
5. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res.* **14**, 253–302 (2001)
6. Wagner, K.W.: Bounded query classes. *SIAM J. Comput.* **19**(5), 833–846 (1990)
7. Bonet, B., Geffner, H.: Planning with incomplete information as heuristic search in belief space. In: Chien, S., Kambhampati, S., Knoblock, C.A. (eds.) *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000)*, pp. 52–61. AAAI Press (2000)
8. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. *Artif. Intell.* **101**(1), 99–134 (1998)
9. Palacios, H., Geffner, H.: Compiling uncertainty away in conformant planning problems with bounded width. *J. Artif. Intell. Res.* **35**, 623–675 (2009)
10. Albore, A., Palacios, H., Geffner, H.: Compiling uncertainty away in non-deterministic conformant planning. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, pp. 465–470. IOS Press (2010)
11. Brafman, R.I., Shani, G.: Online belief tracking using regression for contingent planning. *Artif. Intell.* **241**, 131–152 (2016)
12. Bertoli, P., Cimatti, A., Roveri, M.: Heuristic search + symbolic model checking = efficient conformant planning. In: Nebel, B. (ed.) *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pp. 467–472. Morgan Kaufmann (2001)
13. Hoey, J., St-Aubin, R., Hu, A., Boutilier, C.: SPUDD: Stochastic planning using decision diagrams. In: *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI 1999)*, pp. 279–288 (1999)
14. Lesner, B., Zanuttini, B.: Efficient policy construction for MDPs represented in probabilistic PDDL. In [56] (2011)
15. Wang, C., Joshi, S., Khordon, R.: First order decision diagrams for relational MDPs. *J. Artif. Intell. Res.* **31**, 432–472 (2008)
16. Nebel, B.: On the compilability and expressive power of propositional planning formalisms. *J. Artif. Intell. Res.* **12**, 271–315 (2000)
17. Herzig, A., Lang, J., Marquis, P.: Action representation and partially observable planning using epistemic logic. In: Gottlob, G., Walsh, T. (eds.) *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pp. 1067–1072. Morgan Kaufmann (2003)
18. To, S.T., Son, T.C., Pontelli, E.: A generic approach to planning in the presence of incomplete information: Theory and implementation. *Artif. Intell.* **227**, 1–51 (2015)
19. Scheck, S., Niveau, A., Zanuttini, B.: Knowledge compilation for nondeterministic action languages. In [57], pp. 308–316 (2021)
20. Bonet, B.: Conformant plans and beyond: Principles and complexity. *Artif. Intell.* **174**(3–4), 245–269 (2010)
21. Rintanen, J.: Complexity of planning with partial observability. In: Zilberstein, S., Koehler, J., Koenig, S. (eds.) *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pp. 345–354. AAAI Press (2004)
22. Ghallab, M., Nau, D.S., Traverso, P.: *Automated planning and acting*. Cambridge University Press (2016)
23. Bienvenu, M., Fargier, H., Marquis, P.: Knowledge compilation in the modal logic S5. In: *Proceedings of the 25th Conference on Artificial Intelligence (AAAI 2010)*, pp. 261–266 (2010)
24. Papadimitriou, C.H.: *Computational Complexity*. Addison-Wesley (1994)
25. Tseitin, G.S.: On the complexity of derivation in propositional calculus. *Automation of reasoning: 2: Classical papers on computational logic 1967–1970*, pp. 466–483 (1983)
26. Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. *IEEE Trans. Comput.* **35**(8), 677–691 (1986)
27. Hoffmann, J., Brafman, R.I.: Conformant planning via heuristic forward search: A new approach. *Artif. Intell.* **170**(6–7), 507–541 (2006)

28. Bonet, B., Geffner, H.: mGPT: A probabilistic planner based on heuristic search. *J. Artif. Intell. Res.* **24**, 933–944 (2005)
29. Geffner, H.: Functional Strips: A more flexible language for planning and problem solving. In: Minker, J. (ed.) *Logic-Based Artificial Intelligence*, volume 597 of *Kluwer International Series In Engineering And Computer Science*, chap. 9, pp. 187–209. Kluwer, Dordrecht (2000)
30. Bertoli, P., Cimatti, A.: Improving heuristics for planning as search in belief space. In: Ghallab, M., Hertzberg, J., Traverso, P. (eds.) *Proceedings of the 6th International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*. AAAI Press (2002)
31. Bryce, D., Kambhampati, S., Smith, D.E.: Planning graph heuristics for belief space search. *J. Artif. Intell. Res.* **26**, 35–99 (2006)
32. Cimatti, A., Clarke, E., Giunchiglia, F., Roveri, M.: NuSMV: A new symbolic model checker. *Int. J. Softw. Tools Technol. Transfer* **2**, 410–425 (2000)
33. To, S., Pontelli, E., Son, T.: A conformant planner with explicit disjunctive representation of belief states. In: Gerevini, A., Howe, A., Cesta, A., Refanidis, I. (eds.) *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*. AAAI Press (2009)
34. To, S.T., Son, T.C., Pontelli, E.: A new approach to conformant planning using CNF. In: Brafman, R., Geffner, H., Hoffmann, J., Kautz, H. (eds.) *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS 2010)*. AAAI Press (2010)
35. To, S.T., Son, T.C., Pontelli, E.: Contingent planning as and/or forward search with disjunctive representation. In [56] (2011)
36. Albore, A., Ramirez, M., Geffner, H.: Effective heuristics and belief tracking for planning with incomplete information. In [56] (2011)
37. Kautz, H., Selman, B., Hoffmann, J.: SatPlan: Planning as satisfiability. In: *5th International Planning Competition (IPC-5): Planner Abstracts* (2006)
38. Rintanen, J.: Constructing conditional plans by a theorem-prover. *J. Artif. Intell. Res.* **10**, 323–352 (1999)
39. Cimatti, A., Roveri, M., Bertoli, P.: Conformant planning via symbolic model checking and heuristic search. *Artif. Intell.* **159**(1–2), 127–206 (2004)
40. Palacios, H., Bonet, B., Darwiche, A., Geffner, H.: Pruning conformant plans by counting models on compiled d-DNNF representations. In: Biundo, S., Myers, K., Rajan, K. (eds.) *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005)*. AAAI Press (2005)
41. Palacios, H., Geffner, H.: Mapping conformant planning into SAT through compilation and projection. In: *Proceedings of the 11th Conference of the Spanish Association for Artificial Intelligence (CAEPIA 2005)*, pp. 311–320. Springer (2006)
42. Palacios, H.: The quantified boolean formulas satisfiability library. Conformant family problems. http://www.qbflib.org/family_detail.php?idFamily=707 (2007). Accessed 2 Feb 2023
43. Grastien, A., Scala, E.: CPCEs: A planning framework to solve conformant planning problems through a counterexample guided refinement. *Artif. Intell.* **284**, 103271 (2020)
44. Scala, E., Grastien, A.: Non-deterministic conformant planning using a counterexample-guided incremental compilation to classical planning. In [57] (2021)
45. Bonet, B., Geffner, H.: Flexible and scalable partially observable planning with linear translations. In: Chien, S., Fern, A., Ruml, W., Do, M. (eds.) *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS 2014)*. AAAI Press (2014b)
46. Brafman, R. I., Shani, G.: A multi-path compilation approach to contingent planning. In: Hoffmann, J., Selman, B. (eds.) *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI 2012)*, pp. 1868–1874. AAAI Press (2012a)
47. Brafman, R.I., Shani, G.: Replanning in domains with partial information and sensing actions. *J. Artif. Intell. Res.* **45**, 565–600 (2012)
48. Yoon, S., Fern, A., Givan, R.: FF-Replan: A baseline for probabilistic planning. In: Boddy, M., Fox, M., Thiébaux, S. (eds.) *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS 2007)*, pp. 352–360. AAAI Press (2007)
49. Muise, C., McIlraith, S.A., Beck, J.C. (2011). Monitoring the execution of partial-order plans via regression. In: Walsh, T. (ed.) *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*. AAAI Press (1998)
50. Niveau, A., Zanuttini, B.: Efficient representations for the modal logic S5. In: *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)* (2016)
51. Corrêa, A.B., Pommerening, F., Helmert, M., Francès, G.: Lifted successor generation using query optimization techniques. In: Beck, J.C., Karpas, E., Sohrabi, S. (eds.) *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS 2020)*, pp. 80–89. AAAI Press (2020)

52. Lauer, P., Torralba, Á., Fišer, D., Höller, D., Wichlacz, J., Hoffmann, J.: Polynomial-time in PDDL input size: Making the delete relaxation feasible for lifted planning. In: Zhou, Z.H. (ed.), *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI 2021)*. IJCAI (2021)
53. Martín, M., Geffner, H.: Learning generalized policies from planning examples using concept languages. *Artif. Intell.* **20**(1), 9–19 (2004)
54. Segovia, J., Jiménez, S., Jonsson, A.: Generalized planning with procedural domain control knowledge. In: Coles, A., Coles, A., Edelkamp, S., Magazzeni, D., Sanner, S. (eds.) *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS 2016)*, pp. 285–293. AAAI Press (2016)
55. Segovia-Aguas, J., Jiménez, S., Jonsson, A.: Generating context-free grammars using classical planning. In: Sierra, C. (ed.) *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, pp. 4391–4397. IJCAI (2017)
56. Bacchus, F., Domshlak, C., Edelkamp, S., Helmert, M. (eds.): *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS 2011)*. AAAI Press (2011)
57. Goldman, R.P., Biundo, S., Katz, M. (eds.): *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS 2021)*. AAAI Press (2021)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.