Explicit Representations of Persistency for Propositional Action Theories

Sergej Scheck, Alexandre Niveau, Bruno Zanuttini

Explicit Representations of Persistency for Propositional Action Theories

Sergej Scheck[1] Alexandre Niveau[1] Bruno Zanuttini[1]

[1] Normandie Univ.; UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

sergej.scheck,alexandre.niveau,bruno.zanuttini@unicaen.fr

Résumé

Nous envisageons d'enrichir la représentation des actions en logique propositionnelle par des opérateurs syntaxiques pour représenter la persistance des variables. Ceci est motivé par le fait que le problème du cadre n'est pas résolu de manière satisfaisante par les langages propositionnels, tels que les diagrammes de décision binaires ou DNF. Nous introduisons deux de ces opérateurs, permettant de représenter différents types de persistance, et considérons les langages obtenus à partir de la logique propositionnelle en les ajoutant à n'importe quel niveau d'imbrication. Nous étudions les langages résultantes du point de vue de leur concision relative et de la complexité de la décision de successeur. Nous montrons une image intéressante de divers résultats de complexité.

Mots Clef

Planification, compilation de connaissances, théories d'actions, persistance, problème du cadre, circonscription

Abstract

We consider enriching the representation of actions in propositional logic by syntactic operators for representing the persistency of variables. This is motivated by the fact that the frame problem is not satisfactorily solved by propositional languages, such as binary decision diagrams or DNF. We introduce two such operators, allowing to represent different kinds of persistency, and consider the languages obtained from propositional logic by adding them at any level of nesting. We study the resulting languages from the point of view of their relative succinctness and the complexity of deciding successorship. We show an interesting picture of diverse complexity results.

Keywords

Planning, knowledge compilation, action theories, persistency, frame problem, circumscription

1 Introduction

In automated planning, a central aspect of the description of problems is the formal representation of actions. Such representations are needed for specifying the available actions, and for the planners to operate on them while computing a plan. **PDDL** [15] is a standard language for this. More generally, the formal representation of actions is central to reasoning about actions, programs, and change using logic. Frameworks for this include proposals as diverse as the Situation Calculus [18], the Event Calculus [13], **PDL** [9], **DL-PA** [3], and many others.

We view such languages as either *imperative* or *declarative*. Declarative languages allow one to specify the properties of situations, actions, events, while imperative languages concentrate on how the effects are brought about. In this view, the Situation and Event Calculi are declarative, as well as **PDL**, while **PDDL** and **DL-PA** are imperative.

On the other hand, a well-known question when representing action and change is how to succinctly specify the *non-effects* of actions, that is, to ensure that the specification precludes fluents to change value while this is not intended. This is known as the *frame problem*. While imperative languages naturally come with a solution to the frame problem, because operational semantics literally transform a situation into another one, the frame problem is crucial in declarative languages, and it has been thoroughly studied, in particular for the Situation Calculus [18].

We are interested here in the frame problem for actions specified in the simple language of *propositional action theories*, that is, as Boolean formulas describing the possible combinations of values for fluents *before* and *after* the action is taken. Though this language is very simple, it is indeed used in automated planning, because it makes operations on sets of states (aka *belief states*) conceptually simple [6, 5, 20].

We consider action theories represented in Negation Normal Form (NNF), which encompasses representations usually used like ordered binary decision diagrams or formulas in disjunctive normal form. Such theories are adequate for representing (purely) nondeterministic actions, which lie at the core of fully observable nondeterministic planning and conformant planning [19, 1, 10, 16, 20, 11]. Our contribution is to propose two different operators for representing the persistency of fluents, and to consider the language of NNF actions theories as enriched by one or the other. The originality of this contribution lies in the facts that (i) the language is *enriched* with an operator, which, we argue, allows one to specify the persistency of variables more naturally and succinctly than expressions in the plain underlying logic (like successor-state axioms for the Situation Calculus), and (ii) we allow the operators to occur anywhere in the formula, including nested occurrences, which again facilitates the description of actions.

Our operators are one whose interpretation is dependent on the *syntax* of the action in its scope, and one whose interpretation depends only on its semantics (as a relation between the states before and after the action). The "semantic" one corresponds to interpreting the action in its scope under *circumscription* [14], here used as a semantics of minimal change through the action.

We consider the resulting extensions of the language of NNF action theories in the formal framework of the knowledge compilation map [7]. This framework deals with the study of formal languages under the point of view of queries (how efficient is it to answer various queries depending on the language?), transformations (how efficient is it to transform or combine different representations in a given language?), and succinctness (how concise is it to represent knowledge in each language?). We focus on queries related to automated planning (deciding whether the action can lead from a state to another one, whether it is applicable at some state) and on succinctness issues.

Naturally, there is a tradeoff between tractability of queries and succinctness of the languages. We give a complete picture for the languages which we consider. Precisely, we show that the syntactic operator can be added to NNF action theories without changing complexity of queries nor succinctness, since it can be compiled away in polynomial time; this shows that actions can be specified in the richer language without harming further calculations. On the other hand, we show that the semantic operator yields a more succinct language when allowed only at the root of expressions, and even more succinct when allowed everywhere, but that the complexity of answering queries increases accordingly.

2 Preliminaries

We consider a countable set of propositional *state* variables $\mathbb{P} = \{p_i \mid i \in \mathbb{N}\}$. Let $P \subset \mathbb{P}$ be a finite set of state variables; a subset of P is called a *P-state*, or simply a *state*. The intended interpretation of a state $s \in 2^P$ is the assignment to P in which all variables in s are true, and all variables in $P \setminus s$ are false. For instance, for $P = \{p_1, p_2, p_3\}$, $s = \{p_1, p_3\}$ denotes the state in which p_1, p_3 are true and p_2 is false. We write $\mathrm{V}(\varphi)$ for the set of variables occuring in an expression φ; note that expressions may involve both variables in $\mathbb{P}$ and variables not in $\mathbb{P}$, so in general we do *not* have $\mathrm{V}(\varphi) \subseteq \mathbb{P}$.

Actions We consider (purely) nondeterministic actions, *i.e.*, actions with which a single state may have several successors.

Definition 1. Let $P \subset \mathbb{P}$ be a finite set of variables. A *P-action* is a mapping a from 2^P to $2^{(2^P)}$. The states in $a(s)$ are called *a-successors* of s.

Note that $a(s)$ is defined for all states s. We will consider a to be *applicable* in s if and only if $a(s) \neq \emptyset$ holds.

Definition 2. An *action language* is an ordered pair $\langle L, I \rangle$, where L is a set of expressions and I is a partial function on $L \times 2^{\mathbb{P}}$ such that, when defined on $\alpha \in L$ and $P \subset \mathbb{P}$, $I(\alpha, P)$ is a P-action.

We call the expressions in L *action descriptions*, and call I the *interpretation function* of the language. Observe that those sets P's such that $I(\alpha, P)$ is defined are *a priori* not related to $\mathrm{V}(\alpha)$; α may involve auxiliary variables (not in $\mathbb{P}$) which are not part of the state descriptions, and dually, a state may assign variables of $\mathbb{P}$ which do not occur in α. If L, I, P are fixed or clear from the context, then we write $\alpha(s)$ instead of $I(\alpha, P)(s)$ for the set of all α-successors of s. In this case we call P the *scope* of α. In this article we will consider action descriptions α which are intendedly constructed to ensure that $I(\alpha, P)$ is defined.

Definition 3. A *translation* from an action language $\langle L_1, I_1 \rangle$ to another language $\langle L_2, I_2 \rangle$ is a function $f : L_1 \times 2^{\mathbb{P}} \to L_2$ satisfying $I_1(\alpha, P) = I_2(f(\alpha, P), P)$ for all $\alpha \in L_1$ and $P \subset \mathbb{P}$ such that $I_1(\alpha, P)$ is defined.

In words, this means that the L_1-expression α and the L_2-expression $f(\alpha, P)$ describe the same P-action. Again, when P is clear from the context, we write $f(\alpha)$ for $f(\alpha, P)$. The translation f is said to be *polynomial-time* if it can be computed in time polynomial in the size of α and P, and *polynomial-size* if the size of $f(\alpha, P)$ is bounded by a fixed polynomial in the size of α and P. Clearly, a polynomial-time translation is necessarily also a polynomial-size one, but the converse is not true in general.

Logic A Boolean formula φ over a set Q of variables is in *negation normal form* (NNF) if it is built up from literals using conjunctions and disjunctions, *i.e.*, if it is generated by the grammar $\varphi ::= q \mid \neg q \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$, where q ranges over Q. Similarly to other expressions, Q may involve state variables (in $\mathbb{P}$) and other variables (not in $\mathbb{P}$).

It is important to note that a formula φ with $\mathrm{V}(\varphi) \subseteq Q$ for some set of variables Q can be viewed as a formula over Q (and the truth value of the corresponding Boolean function does not depend on the variables in $Q \setminus \mathrm{V}(\varphi)$). For a Boolean formula φ over Q and an assignment t to the variables in Q, we write $t \models \varphi$ if φ evaluates to true under the assignment t.

For readability, we sometimes use the symbols $\leftrightarrow$ and $\rightarrow$ in Boolean formulas and still call them **NNF** formulas. Indeed, it will always be the case that there are equivalent **NNF** formula of the same size, up to a polynomial.

We always use notation $s, t, \ldots$ for states, $\alpha, \beta, \ldots$ for action descriptions, and $\varphi, \psi, \ldots$ for logical formulas.

Action theories We define the action language of (NNF) action theories, whose extensions we are going to study. To prepare the definition we associate an auxiliary variable

$p' \notin \mathbb{P}$ to each variable $p \in \mathbb{P}$; p' denotes the value of p *after* the action took place, while p denotes the value *before*.

Definition 4. An **NNFAT** *action description* is a Boolean formula α in NNF over $P_\alpha \cup \{p' \mid p \in P_\alpha\}$ for some set of state propositions $P_\alpha \subset \mathbb{P}$. The interpretation of an **NNFAT** action description α is defined for all $P \subseteq \mathbb{P}$ such that $P \supseteq \mathrm{V}(\alpha) \cap \mathbb{P}$ (that is, when all state propositions have a value) by

$$\forall s \subseteq P\colon\ I(\alpha, P)(s) = \{s' \mid (s, s') \models \alpha\}$$

where $(s, s') := s \cup \{p' \mid p \in s'\}$ is the assignment to $P \cup \{p' \mid p \in P_\alpha\}$ induced by s, s'. For $P \not\supseteq \mathrm{V}(\alpha) \cap \mathbb{P}$, $I(\alpha, P)$ is not defined. In words, an **NNFAT** expression represents the set of all ordered pairs $\langle s, s' \rangle$ such that s' is a successor of s, as a Boolean formula over variables in $\mathbb{P} \cup \{p' \mid p \in \mathbb{P}\}$.
Importantly, **NNFAT** does not assume persistency of values, so that if, for example, a variable does not appear at all in an **NNFAT** expression, then this means that its value after the execution of the action can be arbitrary.

Example 5. Let $P = \{p_1, p_2, p_3\}$, $s_1 = \emptyset$, $s_2 = \{p_1\}$, and $\alpha := p_1 \vee (p'_1 \wedge p'_2)$, which can be read "when p_1 is false before the action is taken, both p_1 and p_2 become true after the action, and when p_1 is true anything can occur". Then $\alpha(s_1) = \{\{p_1, p_2\}, \{p_1, p_2, p_3\}\}$, and $\alpha(s_2) = 2^P$.

Circuit representation Since we study the succinctness of languages, it is crucial to define the *size* of action descriptions. For this we use the same setting as typically used in knowledge compilation studies about propositional logic [7], and assume that all action descriptions α are represented by the directed acyclic graph, or *circuit*, obtained from the syntactic tree of α by iteratively identifying the roots of two isomorphic subexpressions to each other, until no more reduction is possible (like for binary decision diagrams [4]). Clearly, for all expressions α, the circuit associated to α in this manner is unique, and it can be computed in polynomial time from the plain expression or from a nonreduced circuit.

3 Frame operators

Our proposal is to enrich **NNFAT** with operators for expressing persistency of variable values. Operators in action languages are used to construct new action descriptions from existing ones, and they can be roughly divided into two types: the interpretation of a *syntactic* operator depends on its argument action descriptions, while that of a *semantic* operator depends on the actions but not on their description.
It is important to recall that **NNF** is a complete language for propositional logic and hence, that **NNFAT** is able to express any action. As a consequence, by *enriching* **NNFAT** we mean providing languages in which it is more convenient to express actions, as we will illustrate, but there is no action which those enriched languages can encode, that **NNFAT** cannot encode itself.

Semantic frame operator The semantic frame operator which we study builds on *circumscription* [14], which is a nonmonotonic semantics for formulas enforcing a form of closed-world assumption, and especially by *propositional* circumscription [8, 17]. However, by introducing an *operator* for this interpretation, we allow circumscription to be enforced only on some parts of an expression.
Let α be an action description and s be a P-state. For all partitions $\{X, V, F\}$ of P, we introduce the operator $\mathsf{C}_{X,V,F}$ so that $\mathsf{C}_{X,V,F}(\alpha)(s)$ chooses those α-successors of s which change variables from X minimally among all states with the same values over F.
Precisely, we define a state s' to be *preferred* to a state s'' with respect to a state s and to X, V, F, which we write $s' \prec^s_{X,V,F} s''$, if $s' \cap F = s'' \cap F$ and $(s'' \Delta s) \cap X \subset (s' \Delta s) \cap X$ hold, where Δ denotes symmetric difference for sets.[1]

Definition 6. The action language $\mathbf{NNFAT_C}$ is the language $\langle L_\mathbf{C}, I_\mathbf{C} \rangle$, where the expressions with scope P in $L_\mathbf{C}$ are defined by the grammar

$$\alpha ::= p \mid p' \mid \neg p \mid \neg p' \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \mathsf{C}_{X,V,F}(\alpha),$$

where p ranges over P and $\langle X, V, F \rangle$ over partitions of P, and $I_\mathbf{C}$ is defined as for **NNFAT**, extended with

$$\begin{aligned} & I(\mathsf{C}_{X,V,F}(\alpha), P)(s) \\ & = \{s' \in I(\alpha, P)(s) \mid \nexists s'' \in I(\alpha, P)(s) : s'' \prec^s_{X,V,F} s'\} \end{aligned}$$

Example 7. Let $P = \{p_1, \ldots, p_5\}$, $X = \{p_1, p_2\}$, $V = \{p_3\}$, and $F = \{p_4, p_5\}$. Let $\alpha = (p'_1 \vee p'_3) \wedge (p'_2 \vee p'_4) \wedge (p'_5)$ and $s = \emptyset$. Then $\{p_1, p_2, p_5\}$ is an α-successor, but not a $\mathsf{C}_{X,V,F}(\alpha)$-successor, of s, because $s'' = \{p_2, p_3, p_5\}$ is also an α-successor of s with $s'' \cap F = \{p_5\} = s' \cap F$ and $(s'' \Delta s) \cap X = \{p_2\} \subset \{p_1, p_2\} = (s' \Delta s) \cap X$. On the other hand, s'' is a $\mathsf{C}_{X,V,F}(\alpha)$-successor of s even though $s''' = \{p_3, p_4, p_5\}$ changes fewer values over X, since s'' and s''' differ over F and hence are incomparable with each other.

It can be seen that the $\mathsf{C}_{X,V,F}$ operator is convenient in particular for expressing actions which involve external causes (to be put in the set F) of changes of values for variables of interest (set X), in the presence of ramifications (set V).

Example 8. Consider encoding the action of driving from work to home. A particularly succinct description is

$$\mathsf{C}_{\{\mathsf{home}\},\{\mathsf{at\_work}\},\{\mathsf{flat\_tire},\mathsf{engine\_ok}\}} \Big((\mathsf{engine\_ok}' \wedge \neg \mathsf{flat\_tire}') \rightarrow \mathsf{home}') \wedge (\mathsf{home}' \leftrightarrow \neg \mathsf{at\_work}') \Big)$$

Consider $s = \{\mathsf{engine\_ok}, \mathsf{at\_work}\}$. Minimization of change over $\{\mathsf{home}\}$ entails that when the causes are not met (for instance, when $\mathsf{engine\_ok}'$ is false), among the possible successors of s only the ones with $\neg\mathsf{home}'$ are retained, ignoring the ramification $\mathsf{at\_work}'$; successors with

[1] As mnemonics, variables in V may vary; those in F are fixed.

home' are not retained, reflecting the fact that there is no "proof" provided by the action that home should change value. On the other hand, due to their presence in the set F, all combinations of causes will be retained. Precisely, the successors of s are $\{\mathsf{engine\_ok}, \mathsf{at\_work}, \mathsf{flat\_tire}\}$, $\{\mathsf{at\_work}\}$, $\{\mathsf{at\_work}, \mathsf{flat\_tire}\}$, and $\{\mathsf{engine\_ok}, \mathsf{home}\}$.

Syntactic frame operator We now define a *syntactic* operator, for representing the notion of persistency of languages like $\mathbf{PDDL}$, where a variable does not change value if there is no *explicit* reason for this. Concretely, we want an operator F such that $\{p\}$ is an $\mathsf{F}(p' \vee \neg p')$-successor of $s = \emptyset$, but not an $\mathsf{F}(q' \vee \neg q')$-successor for $q \neq p$, although $p' \vee \neg p'$ describes the same action as $q' \vee \neg q'$. Therefore we need to formalize the intuition that some change is "explicitly mentioned" in an action description.
For this, we refine the notion of successor by considering the *effects* which apply in a state, themselves decomposed into *explicit* and *implicit* effects.

Definition 9. An *effect* (over $P \subseteq \mathbb{P}$) is a quadruple $\langle e^+, e^-, i^+, i^- \rangle$ such that e^+, e^-, i^+, i^- are pairwise disjoint subsets of P. The *explicit* (resp. *implicit*) part of such an effect is the pair $\langle e^+, e^- \rangle$ (resp. $\langle i^+, i^- \rangle$).

The positive ($e^+ \cup e^-$) and negative ($e^- \cup i^-$) are similar to add- and del-lists in STRIPS: $\langle e^+, e^-, i^+, i^- \rangle$ provokes a transition from a state s to $s' = (s' \cup e^+ \cup i^+) \setminus (e^- \cup i^-)$.
We now define effects for $\mathbf{NNFAT}$ action descriptions. For combinations of effects via $\wedge$, let $\varepsilon_1 = \langle e_1^+, e_1^-, i_1^+, i_1^- \rangle$ and $\varepsilon_2 = \langle e_2^+, e_2^-, i_2^+, i_2^- \rangle$. If $e_1^+ \cup i_1^+ = e_2^+ \cup i_2^+$ holds then we write $\varepsilon_1 \approx \varepsilon_2$ and set $\varepsilon_1 + \varepsilon_2 := \langle e_1^+ \cup e_2^+, e_1^- \cup e_2^-, i_1^+ \cap i_1^-, i_2^+ \cap i_2^- \rangle$. Intuitively, $\varepsilon_1 \approx \varepsilon_2$ means that they provoke exactly the same transitions, but possibly with different explicit/implicit changes, and $\varepsilon_1 + \varepsilon_2$ is the effect which makes explicit any change which is explicit in one of them.

Definition 10. Let α be an $\mathbf{NNFAT}$ action description and s be a state. Then the set of *effects* of α in s, written $\mathrm{E}(\alpha, s)$, is defined inductively by

$\mathrm{E}(p, s) = \{\langle \emptyset, \emptyset, A, B \rangle \mid A, B \subseteq P\}$ for $s \models p$,
$\mathrm{E}(p, s) = \emptyset$ for $s \not\models p$;
and dually for $\mathrm{E}(\neg p, s)$

$$\mathrm{E}(p', s) = \begin{array}{l} \{\langle \emptyset, \emptyset, A, B \rangle \mid A, B \subseteq P \setminus \{p\}\} \\ \cup \{\langle \{p\}, \emptyset, A, B \rangle \mid A, B \subseteq P \setminus \{p\}\} \end{array} \text{ for } s \models p,$$

$\mathrm{E}(p', s) = \{\langle \{p\}, \emptyset, A, B \rangle \mid A, B \subseteq P \setminus \{p\}\}$ for $s \not\models p$;
and dually for $\mathrm{E}(\neg p', s)$

$$\begin{aligned} &\mathrm{E}(\alpha_1 \wedge \alpha_2, s) \\ &= \{\varepsilon_1 + \varepsilon_2 \mid \varepsilon_1 \in \mathrm{E}(\alpha_1, s), \varepsilon_2 \in \mathrm{E}(\alpha_2, s), \varepsilon_1 \approx \varepsilon_2\}; \end{aligned}$$

$$\mathrm{E}(\alpha_1 \vee \alpha_2, s) = \mathrm{E}(\alpha_1, s) \cup \mathrm{E}(\alpha_2, s) \cup \mathrm{E}(\alpha_1 \wedge \alpha_2, s).$$

where A, B are always disjoint from each other.

Intuitively, the first case says that the action p is not applicable if s does not satisfy p (second line), and otherwise imposes no constraint on the successor state, but moreover does not set any value *explicitly*: a transition may occur from, say, $\{p, q\}$ to $\{r\}$, but then the positive effects $A = \{r\}$ and negative effects $B = \{p, q\}$ are considered to be *implicit*.
For atomic actions of the form p', we distinguish two cases. When s does not already satisfy p, then all effects of p' *explicitly* set p. Constrastingly, when s already satisfies p, then the action p' leaves the value unchanged, and we include both effects with an explicit setting of p (to the same value) and effects with no setting of p at all (neither implicit nor explicit). Of course, for fixed A, B, those two effects provoke a transition from s to the same successor s'. Nevertheless, including the effects which do not set p at all turns out to be necessary for $\wedge$ to behave as expected in the extension of $\mathbf{NNFAT}$ with the F operator (to be defined soon).
Finally, it is worth noting that we include the effects of $\alpha_1 \wedge \alpha_2$ in those of $\alpha_1 \vee \alpha_2$. Of course, the *transitions* of $\alpha_1 \wedge \alpha_2$ are already included in those of α_1 and in those of α_2, but not necessarily with the same explicit parts. For instance, for $\alpha_1 = p'$, $\alpha_2 = q'$, and $s = \emptyset$, $s' = \{p, q\}$ is both an α_1- and an α_2-successor of s, but the "fully" explicit effect $\langle \{p, q\}, \emptyset, \emptyset, \emptyset \rangle$ is only one of $\alpha_1 \wedge \alpha_2$.
With this in hand, we can define the *framing* operator F_X. Intuitively, $\mathsf{F}_X(\alpha)$ retains only those effects of α which include no implicit effect on variables of X. As can be seen, this is equivalent to removing variables of X from the implicit part of all effects. So we define $\mathrm{E}(\mathsf{F}_X(\alpha), s)$ to be

$$\{\langle e^+, e^-, i^+ \setminus X, i^- \setminus X \rangle \mid \langle e^+, e^-, i^+, i^- \rangle \in \mathrm{E}(\alpha, s)\}$$

Definition 11. The action language $\mathbf{NNFAT_F}$ is the language $\langle L_\mathbf{F}, I_\mathbf{F} \rangle$, where the expressions with scope P in $L_\mathbf{F}$ are defined by the grammar

$$\alpha ::= p \mid p' \mid \neg p \mid \neg p' \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \mathsf{F}_X(\alpha),$$

where p ranges over P and X over subsets of P, and $I_\mathbf{F}$ is defined for all α, s, $I_\mathbf{F}(\alpha, P)(s)$ is defined to be

$$\{(s \cup e^+ \cup i+) \setminus (e^- \cup i^-) \mid \langle e^+, e^-, i^+, i^- \rangle \in \mathrm{E}(\alpha, s)\}$$

Example 12. Consider the action of leaving one's bike in a garage for them to repair exactly one wheel, but not knowing which one in advance.[2] The action may have two effects: making the front wheel or the back wheel ok (in both cases not affecting the other). However, in the process of repairing the back wheel, it might occur that the gear is changed. Additionally, in no case would the brakes be affected. Such an action could be encoded by

$$\mathsf{F}_{\mathsf{brakes}}\big(\mathsf{F}_{\mathsf{f\_wheel\_ok}}(\mathsf{b\_wheel\_ok}') \vee \mathsf{F}_{\mathsf{b\_wheel\_ok,gear}}(\mathsf{f\_wheel\_ok}')\big)$$

[2] *E.g.*, knowing only that they are not aligned which each other.

Importantly, in general, pushing all occurrences of F to the root of the expression changes its interpretation. For instance, in Example 12, this would yield the expression

$$\mathsf{F}_{\mathsf{brakes,f\_wheel\_ok,b\_wheel\_ok,gear}}\left(\mathsf{b\_wheel\_ok}' \vee \mathsf{f\_wheel\_ok}'\right)$$

according to which the gear can never change value.

Another important observation is that F_X is *not* a special case of $\mathsf{C}_{X,V,F}$. It might seem that F_X is nothing more than $\mathsf{C}_{X,\emptyset,P\setminus X}$. However, taking $P = \{p\}$, $\alpha = p' \vee \neg p'$, and $s = \emptyset$, it can be seen that both $\emptyset$ and $\{p\}$ are $\mathsf{F}_X(\alpha)$-successors of s, while only $\emptyset$ is a $\mathsf{C}_{X,\emptyset,P\setminus X}(\alpha)$-successor. Indeed, $\mathsf{F}_X(\alpha)$ takes into account the fact that both alternatives are explicitly mentioned in the formula, while $\mathsf{C}_{X,\emptyset,P\setminus X}(\alpha)$ considers only the semantics of α and hence, is equivalent to $\mathsf{C}_{X,\emptyset,P\setminus X}(\top)$.

Restriction on circuits In addition to $\mathbf{NNFAT_C}$ and $\mathbf{NNFAT_F}$, we will also study their natural restrictions where the operator $\mathsf{C}_{X,V,F}$ or F_X, respectively, occurs only at the root of expressions. Namely, $\mathbf{NNFAT_C}$ is $\mathbf{NNFAT}$ augmented only with expressions of the form $\mathsf{C}_{X,V,F}(\alpha)$, where α is an $\mathbf{NNFAT}$ action description. We denote by $\mathbf{NNFAT_{rC}}$ the resulting language. Similarly, we define the restricted language $\mathbf{NNFAT_{rF}}$. Observe that $\mathbf{PDDL}$, for instance, can be seen as a language without framing, together with an (implicit) operator F_P at the root of all expressions.

4 Compiling the Syntactic Operator Away

In this section, we show that the syntactic operator F_X can be eliminated from an $\mathbf{NNFAT_F}$ expression to yield an expression in $\mathbf{NNFAT}$ which has the same interpretation and size (up to a polynomial). Moreover, the elimination can be done in polynomial time. This means that $\mathbf{NNFAT_F}$ can be used as a convenient language for describing actions, still the algorithms which manipulate those descriptions (e.g. planners) need not be extended to cope with F, since all its occurrences can simply be eliminated in polynomial time and space.

Let us mention that the equivalent result is rather obvious for plain (tree-like) representations of formulas, but that we consider here *circuit representations*, in which a single subcircuit may occur in an exponential number of paths.

Our procedure essentially amounts to replacing all occurrences of F with subformulas similar to successor-state axioms. Before we show the result, we need two lemmas (the proofs are by induction on the structure of α).

The first lemma defines an expression $\mathrm{Expl}(\alpha, p)$, and shows that this expression states that if p changes its value via α, then there is at least one effect which does it explicitly. For states s, s' and action description α, write $\mathrm{E}(\alpha, s, s')$ for the set of effects which lead from s to s': $\mathrm{E}(\alpha, s, s') := \{\langle e^+, e^-, i^+, i^-\rangle \in \mathrm{E}(\alpha, s) \mid s' = (s \cup e^+ \cup i^+) \setminus (e^- \cup i^-)\}$.

Lemma 13. *Let α be a $\mathbf{NNFAT_F}$ action description with scope P, $s, s' \subseteq P$ be two states and $p \in s\Delta s'$. Then there is an effect $\langle e^+, e^-, i^+, i^-\rangle \in \mathrm{E}(\alpha, s, s')$ with $p \in e^+_e \cup e^-_e$ if and only if $(s, s') \models \mathrm{Expl}(\alpha, s)$, where the $\mathbf{NNFAT}$ expression $\mathrm{Expl}(\alpha, s)$ is defined to be*

- *α for $\alpha = p'$ or $\alpha = \neg p'$,*
- *$\bot$ for $\alpha = q'$ or $\alpha = \neg q'$ with $q \neq p$,*
- *$\bot$ for $\alpha = q$ or $\alpha = \neg q$, for all variables q,*
- *$(\mathrm{Expl}(\beta, p) \wedge \gamma) \vee (\beta \wedge \mathrm{Expl}(\gamma, p))$ for $\alpha = \beta \wedge \gamma$,*
- *$\mathrm{Expl}(\beta, p) \vee \mathrm{Expl}(\gamma, p)$ for $\alpha = \beta \vee \gamma$,*
- *$\bigwedge_{x \in X \cup \{p\}} \big((x \leftrightarrow x') \vee \mathrm{Expl}(\beta, x)\big)$ for $\alpha = \mathsf{F}_X(\beta)$.*

The second lemma states properties of sets of effects.

Lemma 14. *Let α be an $\mathbf{NNFAT_F}$ action description with scope $P \supseteq s$ and $\varepsilon_1, \varepsilon_2 \in \mathrm{E}(\alpha, s, s')$. Then it holds:*

1. *If $\varepsilon_1 \approx \varepsilon_2$ then $\varepsilon_1 + \varepsilon_2 \in \mathrm{E}(\alpha, s)$*
2. *There is at least one $\langle e^+, e^-, i^+, i^-\rangle \in \mathrm{E}(\alpha, s, s')$ such that $e^+ \cup i^+ = s' \setminus s$ and $e^- \cup i^- = s \setminus s'$*

Proposition 15. $\mathbf{NNFAT_F}$ *is translatable into* $\mathbf{NNFAT}$ *in polynomial time.*

Proof. Let α be a $\mathbf{NNFAT_F}$ action description with scope P. The translation $f(\alpha)$ is obtained by replacing each node $\mathsf{F}_X(\beta)$ (with $X \subseteq P$) of the circuit of α by $\beta \wedge \bigwedge_{p \in X} \big((p \leftrightarrow p') \vee \mathrm{Expl}(\beta, p)\big)$ and keeping the other nodes.

The circuit of $\mathrm{Expl}(\beta, p)$ can be computed in polynomial time (in each step we create a bounded amount of edges and nodes). Thus $f(\alpha)$ can be computed in polynomial time, too.

Now we show that $f(\alpha)$ describes the same action as α. Suppose that $f(\beta) \equiv \beta$ and $(s, s') \models f(\alpha) = \beta \wedge \bigwedge_{p \in X} \big((p \leftrightarrow p') \vee \mathrm{Expl}(\beta, p)\big)$. Then $s' \in \beta(s)$, and $(s, s') \models \mathrm{Expl}(\beta, p)$ for all $p \in X$. Then by lemmas 13 and 14 for every $p \in (s'\Delta s) \cap X$ there exists an effect $\langle e^+_{\beta,p}, e^-_{\beta,p}, i^+_{\beta,p}, i^-_{\beta,p}\rangle$ with $p \in e^+_{\beta,p} \cup e^-_{\beta,p}$ which mentions only variables from $s\Delta s'$ and by lemma 14 the sum $\langle e^+, e^-, i^+, i^-\rangle$ of these effects is as well in $\mathrm{E}(\beta, s, s')$ and $(s\Delta s') \cap X \subseteq e^+ \cup e^-$ and thus $\langle e^+, e^-, i^+, i^-\rangle \in \mathrm{E}(\alpha, s)$. This implies $s' \in (\mathsf{F}_X(\beta))(s)$. Conversely, if $s' \in (\mathsf{F}_X(\beta))(s)$ then there exists an effect $\langle e^+, e^-, i^+, i^-\rangle$ of β in s witnessing this with $(s\Delta s') \cap X \subseteq e^+ \cup e^-$. Therefore, by Lemma 13 all $\mathrm{Expl}(\beta, p)$ with $p \in (s\Delta s') \cap X$ from the definition of $f(\alpha)$ are satisfied by (s, s'), and for the rest the expression $p \leftrightarrow p'$ is satisfied. And β is satisfied by the inductive assumption. For $\alpha = \alpha_1 \wedge \alpha_2$ or $\alpha = \alpha_1 \vee \alpha_2$ we trivially have that $f(\alpha)$ describes the same action as α if the claim was proven for α_1 and α_2. $\square$

5 Complexity of queries

We now turn to studying the complexity of *queries* to expressions. We concentrate on two natural queries for planning: checking the existence of a transition, and deciding applicability of an action in a state. These queries arguably are at the basis of most other reasonable queries.
Let $\langle L, I\rangle$ be a fixed action language, and let α denote an expression in L, $P \subseteq \mathbb{P}$ denote a set of variables such that $I(\alpha, P)$ is defined, and s, s' denote two P-states.

Definition 16. The decision problem SUCC takes as input α, P, s, s', and asks whether $s' \in \alpha(s)$.

Definition 17. The decision problem APPLIC takes as input α, P, s, and asks whether $\alpha(s) \neq \emptyset$.

SUCC for $\mathbf{NNFAT}$ amounts to model-checking of an NNF formula, and for $\mathbf{NNFAT_F}$ the complexity follows from Proposition 15.

Proposition 18. SUCC *is in* $\mathbf{P}$ *for* $\mathbf{NNFAT}$ *and* $\mathbf{NNFAT_F}$.

To prepare further results we introduce a notation.

Notation 19. Let $n \in \mathbb{N}$ and $X_n = \{x_1, \ldots, x_n\}$ be a set of variables. Observe that there are a cubic number N_n of clauses of length 3 over X_n. We fix an arbitrary enumeration $\gamma_1, \gamma_2, \ldots, \gamma_{N_n}$ of all these clauses, and we define $P_n \subset \mathbb{P}$ to be the set of state variables $\{p_1, p_2, \ldots, p_{N_n}\}$. Write $\ell \in \gamma_i$ if the literal ℓ occurs in the clause γ_i. Then to any 3-CNF formula φ we associate the P_n-state $s(\varphi) = \{p_i \mid i \in \{1, \ldots, N_n\}, \gamma_i \in \varphi\}$, and dually, to any P_n-state s, we associate the 3-CNF formula over X_n, written $\varphi(s)$, which contains exactly those clauses γ_i for which $p_i \in s$ holds.We set $\psi_n := \bigwedge_{i=1}^{N_n}(\neg p_i \vee \bigvee_{\ell\in\gamma_i} \ell)$. In words, ψ_n is satisfied by an assignment t to $P_n \cup \{x_1, \ldots, x_n\}$ if and only if the 3-CNF over $\{x_1, \ldots, x_n\}$ encoded by $t \cap P_n$, is satisfied by the assignment to $\{x_1, \ldots, x_n\}$ encoded by $t \cap \{x_1, \ldots, x_n\}$.

Example 20. Consider an enumeration of all clauses over $X_2 = \{x_1, x_2\}$ which starts with $\gamma_1 = (x_1 \vee x_1 \vee x_2), \gamma_2 = (x_1 \vee x_1 \vee \neg x_2), \gamma_3 = (x_1 \vee \neg x_1 \vee x_2), \ldots$ Then $\varphi = (x_1 \vee x_1 \vee x_2) \wedge (x_1 \vee \neg x_1 \vee x_2)$ is encoded by $s(\varphi) = \{p_1, p_3\}$.

Proposition 21. SUCC *is* $\mathbf{coNP}$*-complete for* $\mathbf{NNFAT_{rC}}$.

Proof. For a formula ψ over the variables $\{q_i \mid j \in J\}$, write ψ' for the formula obtained by replacing all variables q_j by q'_j. Consider the (polynomial-sized) $\mathbf{NNFAT_{rC}}$ action description with scope $P_n := X_n \cup \{p_1, \ldots, p_{N_n}\}$ as in Notation 19:

$$\alpha_n := \mathsf{C}_{P_n,\emptyset,\emptyset}\big(\psi'_n \vee \bigwedge\nolimits_{1\leq i\leq N_n} p'_i\big)$$

Let φ be a 3-CNF over X_n which is not satisfied by the assignment "$\forall i : x_i = \top$". We claim that φ is unsatisfiable if and only if $s(\varphi)\cup X_n$ is an α_n-successor of $s(\varphi)$. Indeed, if φ is satisfiable then by assumption a satisfying assignment ($t \subsetneq X_n$) contains at least one variable set to $\bot$. Then by definition of $\mathsf{C}_{P_n,\emptyset,\emptyset}$ the state $s(\varphi) \cup X_n$ can't be a successor of $s(\varphi)$ because $s(\varphi) \cup t$ changes fewer variables from P_n. Conversely, if $s(\varphi) \cup X_n$ is an α_n-successor of $s(\varphi)$ then the only subset $t \subseteq X_n$ with $t \cup s(\varphi) \in \alpha_n(s(\varphi))$ is $t = X_n$, which is by a assumption not a satisfying assignment. Therefore SUCC is $\mathbf{coNP}$-hard. For membership: $s' \notin \mathsf{C}_{X,V,F}(\alpha)(s)$ can be justified in polynomial time by either showing that $s' \notin \alpha(s)$ or that there exists $s'' \in \alpha(s)$ with $(s''\Delta s)\cap X \subsetneq (s'\Delta s)\cap X$ and $s'\cap F = s''\cap F$ (both justifications can be checked in polynomial time since α is in $\mathbf{NNFAT}$). ☐

Proposition 22. SUCC *is* $\mathbf{PSPACE}$*-complete for* $\mathbf{NNFAT_C}$.

Proof. We modify Notation 19 by introducing for every variable x_j two variables q_j, r_j and write $q_j \in \gamma_i$ if $x_j \in \gamma_i$, and $r_j \in \gamma_i$ if $\neg x_j \in \gamma_i$. We set $Q_n := \{q_j, r_j \mid 1 \leq j \leq n\}$. $S_n := Q_n \cup \{p_1, \ldots, p_{N_n}\}$. We obtain β^n_{n+1} by replacing all x_j in ψ_n by q'_j and all $\neg x_j$ by r'_j and then define recursively

$$\beta^n_i := \mathsf{C}_{\{q_i,r_i\},\emptyset,S_n\setminus\{q_i,r_i\}}\big((q_i\wedge r_i)\vee(q_i\wedge\beta^n_{i+1})\vee(r_i\wedge\beta^n_{i+1})\big)$$

Let $\Phi := \forall x_1 : \exists x_2 : \ldots : \forall x_n : \varphi$, which is equivalent to $\forall x_1 : \neg(\forall x_2 : \neg(\ldots\neg(\forall x_n : \varphi)\ldots)$, be a quantified Boolean formula with a 3-CNF φ. Deciding the validity of such formulas is obviously $\mathbf{PSPACE}$-complete. We claim that Φ is true if and only if $s(\varphi) \cup Q_n \in \beta^n_1(s(\varphi))$. Indeed, first observe that for all i and all states $s \subseteq S_n \setminus \{q_i, r_i\}$, $t \subseteq \{p_1, \ldots, p_{N_n}\}$: $s \cup \{q_i, r_i\} \in \beta^n_i(t) \Leftrightarrow s \cup \{q_i\}, s\cup\{r_i\} \notin \beta^n_{i+1}(t)$. We set $V_i := \{q_j, r_j \mid i \leq j \leq n\}$ and $W_i := \{q_i, r_i\}$ and it follows with $t := s(\varphi)$

$$\begin{aligned}
&s(\varphi) \cup Q_n \in \beta^n_1(s(\varphi))\\
\Leftrightarrow &s(\varphi) \cup V_2 \cup \{q_1\}, s(\varphi) \cup V_2 \cup \{r_1\} \notin \beta^n_2(s(\varphi))\\
\Leftrightarrow &\forall z_1 \in W_1 : \neg(s(\varphi) \cup V_2 \cup \{z_1\} \in \beta^n_2(s(\varphi)))\\
&\quad\ldots\\
\Leftrightarrow &\forall z_1 \in W_1 : \neg\big(\forall z_2 \in W_2 : \neg(\forall z_3 \in W_3 : \neg\ldots\\
&(s(\varphi) \cup \{z_1, \ldots, z_n\} \in \beta^n_{n+1}(s(\varphi)))))\big)
\end{aligned}$$

$s(\varphi)\cup\{z_1, \ldots, z_n\} \in \beta^n_{n+1}(s(\varphi))$ in the last line is equivalent to φ being true under the assignment defined by $x_i := (z_i = q_i)$. We have proven the claim and thus $\mathbf{PSPACE}$-hardness of SUCC. For membership: SUCC can be reduced to deciding the truth of a fully quantified boolean formula (because $s' \in \mathsf{C}_{X,V,F}(\alpha)(s) \Leftrightarrow \forall s'' : ((s''\Delta s) \cap X \subsetneq (s'\Delta s) \cap X \wedge s'' \cap F = s' \cap F \Rightarrow s'' \notin \alpha(s)))$. ☐

Proposition 23. APPLIC *is* $\mathbf{NP}$*-complete for* $\mathbf{NNFAT}$, $\mathbf{\Sigma^P_2}$*-complete for* $\mathbf{NNFAT_{rC}}$ *and* $\mathbf{PSPACE}$*-complete for* $\mathbf{NNFAT_C}$.

Proof. Satisfiability of a 3-CNF φ can be reduced to applicability in $\mathbf{NNFAT}$ by replacing each x by x' and checking whether the obtained action description describes an action which is applicable in $s = \emptyset$.

For $\mathbf{\Sigma_2^p}$-hardness in $\mathbf{NNFAT_{rC}}$: let $\bar{x} = (x_1, \ldots, x_n)$, $\bar{y} = (y_1, \ldots, y_m)$, $X_n = \{x_1, \ldots, x_n\}, Y_m = \{y_1, \ldots, y_m\}$. A quantified boolean formula $\exists\bar{y}\forall\bar{x}\varphi(\bar{x}, \bar{y})$ is true if and only if the $S := \{q\} \cup X_n \cup Y_m$ action $\mathsf{C}_{X_n \cup \{q\}, \emptyset, Y_m}((\neg q' \wedge \neg\varphi(\bar{x}', \bar{y}')) \vee q') \wedge q'$ is applicable in $\emptyset$, because it is applicable only if q can be set to true meaning that $\neg\varphi(\bar{x}', \bar{y}')$ is unsatisfiable by any assignment to $\bar{x}'$. For membership: if we have an oracle for SUCC then we can justify applicability in polynomial time by giving a successor and checking successorship with the oracle.

For $\mathbf{PSPACE}$-hardness in $\mathbf{NNFAT_C}$: $s' \in \alpha(s)$ if and only if $\alpha \wedge \bigwedge_{p \in s'} p' \wedge \bigwedge_{p \notin s'} \neg p'$ is applicable in s. For membership: to check for applicability of α in s we need to check for all s' whether $s' \in \alpha(s)$. □

6 Succinctness

Recall that all languages are fully expressive, so we use the following definition [7].

Definition 24. A language $\mathbf{L}_1$ is *at least as succinct as* $\mathbf{L}_2$ if there exists a polynomial-size translation from $\mathbf{L}_2$ into $\mathbf{L}_1$.

Our separation results rely on yet unproven assumptions on nonuniform complexity classes. Recall that $\mathbf{P/poly}$ (resp. $\mathbf{coNP/poly}$) is the class of all decision problems such that for all $n \in \mathbb{N}$, there is a polytime algorithm (resp. a nondeterministic polytime algorithm for the complement) which decides the problem for all inputs of size n [2]. The assumptions $\mathbf{coNP} \not\subseteq \mathbf{P/poly}$ and $\mathbf{PSPACE} \not\subseteq \mathbf{coNP/poly}$ which we use are standard ones; in particular, $\mathbf{coNP} \subseteq \mathbf{P/poly}$ would imply a collapse of the polynomial hierarchy at the second level (Karp-Lipton theorem), and $\mathbf{PSPACE} \subseteq \mathbf{coNP/poly}$ would imply a collapse at the third level[21].

We first observe that since $\mathbf{NNFAT}$ is translatable into $\mathbf{NNFAT_F}$ via the identity function, and $\mathbf{NNFAT_F}$ is a superlanguage of $\mathbf{NNFAT}$, so they are equally succinct.

Proposition 25. *If* $\mathbf{coNP} \not\subseteq \mathbf{P/poly}$ *then* $\mathbf{NNFAT_{rC}}$ *is strictly more succinct than* $\mathbf{NNFAT}$.

Proof. Recall from the proof of proposition 21 that $s(\varphi) \cup X_n$ is an α_n-successor of $s(\varphi)$ if and only if φ (assumed not to be satisfied by assigning $\top$ to all variables) is unsatisfiable, and that α_n depends only on the number n of variables in φ (not on φ itself). Now suppose that there exists a poly-size translation f from $\mathbf{NNFAT_{rC}}$ into $\mathbf{NNFAT}$. Then we can check whether φ is unsatisfiable by checking if it is not satisfied by the all-$\top$ assignment, and whether $s(\varphi) \cup X_n$ is an $f(\alpha_n)$-successor of $s(\varphi)$. This gives a nonuniform polytime algorithm (Proposition 18) for non-satisfiability, which is a $\mathbf{coNP}$-complete problem. □

The next proposition says that nesting of $\mathsf{C}_{X,V,F}$ operators contributes to succinctness. We omit the proof since it is very similar to that of Proposition 25 (using Proposition 22).

Proposition 26. *If* $\mathbf{PSPACE} \not\subseteq \mathbf{coNP/poly}$ *then* $\mathbf{NNFAT_{rC}}$ *is strictly less succinct than* $\mathbf{NNFAT_C}$.

7 Conclusion

We studied extensions of $\mathbf{NNF}$ action theories with operators expressing two different types of persistency of variables, with the goal of enriching the language. We gave a picture of the resulting languages *à la* knowledge compilation map. It turns out that using a frame operator resembling that of PDDL at any level of nesting does not change time nor space complexity; hence this operator can be used when specifying actions, then compiled away efficiently so as to use algorithms designed for (standard) NNF action theories. The languages resulting for our second operator (related to the interpretation of formulas under circumscription) are more succinct but also have a greater complexity for basic queries.

Our results raise new open knowledge compilation-related questions. For example, we are interested in comparing these new languages to already well-known languages like variants of $\mathbf{PDDL}$ or $\mathbf{DL\text{-}PPA}$ [12] in terms of succinctness. We are also interested in the complexity of queries other than studied here, e.g. whether all successors of a state via a given sequential plan satisfy some property. Our long-term goal is to study action description languages as defined by allowed operators or constructs, so as to get a complete picture.

Acknowledgements

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

References

[1] Alexandre Albore, Héctor Palacios, and Hector Geffner. Compiling uncertainty away in non-deterministic conformant planning. In *Proc. 19th European Conference on Artificial Intelligence (ECAI 2010)*, volume 215, pages 465–470, 2010.

[2] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[3] P. Balbiani, A. Herzig, and N. Troquard. Dynamic logic of propositional assignments: A well-behaved variant of pdl. In *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 143–152, 2013.

[4] Randal E Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys (CSUR)*, 24(3):293–318, 1992.

[5] Daniel Bryce, Subbarao Kambhampati, and David E. Smith. Planning graph heuristics for belief space search. *Journal of Artificial Intelligence Research*, 26:35–99, 2006.

[6] Alessandro Cimatti and Marco Roveri. Conformant planning via symbolic model checking. *Journal of Artificial Intelligence Research*, 13:305–338, 2000.

[7] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[8] Thomas Eiter and Georg Gottlob. Propositional circumscription and extended closed-world reasoning are πp2-complete. *Theoretical Computer Science*, 114(2):231–245, 1993.

[9] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194 – 211, 1979.

[10] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

[11] Tomas Geffner and Hector Geffner. Compact policies for fully observable non-deterministic planning as SAT. In *Proc. 28th International Conference on Automated Planning and Scheduling (ICAPS 2018)*, pages 88–96, 2018.

[12] Andreas Herzig, Frédéric Maris, and Julien Vianey. Dynamic logic of parallel propositional assignments and its applications to planning. In *Proc. 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pages 5576–5582, 2019.

[13] Robert A. Kowalski and Marek J. Sergot. A logic-based calculus of events. *New Gener. Comput.*, 4(1):67–95, 1986.

[14] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial intelligence*, 13(1-2):27–39, 1980.

[15] Drew McDermott. PDDL–the planning domain definition language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998. Available at: `www.cs.yale.edu/homes/dvm` (consulted on 2020/03/16).

[16] Christian J. Muise, Sheila A. McIlraith, and Vaishak Belle. Non-deterministic planning with conditional effects. In *Proc. 24th International Conference on Automated Planning and Scheduling (ICAPS 2014)*, pages 370—374, 2014.

[17] Gustav Nordh. A trichotomy in the complexity of propositional circumscription. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 257–269. Springer, 2005.

[18] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In *Artificial and Mathematical Theory of Computation*, 1991.

[19] Jussi Rintanen. Complexity of planning with partial observability. In *Proc. 14th International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 345–354, 2004.

[20] Son Thanh To, Tran Cao Son, and Enrico Pontelli. A generic approach to planning in the presence of incomplete information: Theory and implementation. *Artificial Intelligence*, 227:1–51, 2015.

[21] Chee K Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical computer science*, 26(3):287–300, 1983.