A KC Map for Variants of Nondeterministic PDDL

Sergej Scheck, Alexandre Niveau, Bruno Zanuttini

A KC Map for Variants of Nondeterministic PDDL

Sergej Scheck Alexandre Niveau Bruno Zanuttini

Normandie Univ.; UNICAEN, ENSICAEN, CNRS, GREYC, 14 000 Caen, France
{sergej.scheck,alexandre.niveau,bruno.zanuttini}@unicaen.fr

Résumé

Nous étudions différents langages permettant de représenter des actions non-déterministes pour la planification automatique, du point de vue de la compilation de connaissances. Précisément, nous considérons la question de la concision des langages (quelle est la taille de la description d'une action dans chaque langage ?) et des questions de complexité (traitabilité ou dureté de plusieurs requêtes et transformations qui surviennent naturellement dans la planification et le suivi des croyances). Nous étudions une version abstraite et nondéterministe de PDDL, STRIPS conditionnel nondéterministe, et les langages $\text{NPDDL}_{\text{seq}}$ et $\text{NPDDL}_{\text{not}}$ obtenus en ajoutant séquence et négation à PDDL nondéterministe. Nous montrons que ces langages ont une concision et complexité différente pour les requêtes les plus naturelles.

Abstract

We study different languages for representing nondeterministic actions in planning from the point of view of knowledge compilation. Precisely, we consider succintness issues (how succinct is the description of an action in each language?), and complexity issues (tractability or hardness of several queries and transformations which arise naturally in planning and belief tracking). We study an abstract, nondeterministic version of PDDL, nondeterministic conditional STRIPS, and the languages $\text{NPDDL}_{\text{seq}}$ and $\text{NPDDL}_{\text{not}}$ obtained by adding sequence and negation to nondeterministic PDDL. We show that these languages have different succinctness and different complexity for the most natural queries.

1 Introduction

In automated planning, a central aspect of the description of problems is the formal representation of actions. Such representations are indeed needed for specifying the actions available to the agent (PDDL [13] is a standard language for this), and also for planners to operate on them while searching for a plan.

In this paper, we consider different representation languages within the formal framework of the knowledge compilation map [5]. 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 ?).

The knowledge compilation map has been introduced for representations of Boolean functions [5]. As far as we know there has been no systematic study of languages for representing actions per themselves. This is however an important problem, as planners need to query action representations again and again while searching for a plan (e.g., to find out which actions are applicable at the current node of the search tree), and many of them start by transforming the action specifications into some representation suited for this [11, 24, 3, 22]. Hence having a clear picture of the properties of languages is of interest for developing such planners.

This paper is an attempt at a systematic study of various action languages from the point of view of knowledge compilation. Works with related objectives do exist, but the focus has been on other aspects of planning, like the representation of plans [2], axioms [23], or action costs [21]. The only studies about action languages which we are aware of are those pioneered by Bäckström, but they essentially consider the representation of actions up to preservation of plan length [1, 16, 18], while we are interested in a stricter notion whereby the semantics is precisely preserved, hence preserving more aspects of the planning tasks (like the number of plans, relationships between variables, etc.).

We are interested here in (purely) nondeterministic actions, which lie at the core of fully observable nondeterministic planning and of conformant planning [19, 17, 7, 14, 24, 8]. We also focus on propositional domains, in which states are assignments to a given set of propositions.

Our mid-term goal is to give a systematic picture of languages arising from combinations of allowed constructs among the ones introduced in the literature (like nondeterministic choice, iteration, persistency by default, etc.),

and in this work we consider abstract languages which resemble variants and extensions of the well-known STRIPS and PDDL. Orthogonally, we also study two concrete representations of expressions, as syntactic *trees* or as more compact *circuits*, where identical subexpressions are not repeated. The former representation gives a natural measure of the size of action specifications, while the latter is more compact and is the one typically used in knowledge compilation literature [5].

The paper is structured as follows. We first give background about actions and logic (Section 2), then formally define the action languages which we consider (Section 3). Then we give our results : results about the complexity of queries (Section 4); separation results, which allow us to determine the relative succinctness of the languages (Section 5), and results about the complexity of transformations (Section 6). Finally we conclude, discussing some open problems and perspectives of this work (Section 7).

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 nonempty 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.

Actions We consider (purely) nondeterministic actions, which map states to sets of states. Hence a single state may have several successors through the same action, in contrast with deterministic actions (which map states to states), and no relative likelihood is encoded between the successors of a state, in contrast with stochastic actions (which map states to probability distributions over states).

Definition 1 (action)**.** 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 and P is called the *scope* of a.

Note that $a(s)$ is defined for all states s. For our results explicit preconditions are not important; we will consider a to be *applicable* in s if and only if $a(s) \neq \emptyset$. Every action a can be identified with a binary *transition relation* $\|a\|$ on states which is defined via $\|a\| := \{(s, s') \mid s' \in a(s)\}$. Elements of $\|a\|$ are called *state transitions*.

In this article, we are interested in the properties of *representations* of actions in various langugages.

Definition 2. [action language] An *action language* is an ordered pair $\langle L, I\rangle$, where L is a set of expressions and I is a partial function from $L \times 2^{\mathbb{P}}$ to the set of all actions such that, when defined on $\alpha \in L$ and $P \subset \mathbb{P}$, $I(\alpha, P)$ is a P-action. If $\langle L, I\rangle$ is an action language and $L' \subseteq L$ then we call $\langle L', I|_{L' \times 2^{\mathbb{P}}}\rangle$ a *sublanguage* of $\langle L, I\rangle$.

We call the expressions in L *action descriptions*, and call I the *interpretation function* of the language. In this article $I(\alpha, P)$ will be defined if and only if $\mathrm{V}(\alpha) \subseteq P$.

If the language $\langle L, I\rangle$ and the set 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. We might also say "action α", meaning the action described by the action description α, and write $\|\alpha\|$ for $\|I(\alpha, P)\|$.

Translations In this article, we are interested in the existence of translations between languages which preserve the scope of action descriptions, i.e., which do not add new state variables to the scope in the destination language.

Definition 3 (translation)**.** A *translation* from an action language $\langle L_1, I_1\rangle$ to another language $\langle L_2, I_2\rangle$ is a function $f\colon L_1 \times 2^{\mathbb{P}} \to L_2$ such that $I_1(\alpha, P) = I_2(f(\alpha, P), P)$ holds 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.

Here, a translation is *not* allowed to introduce auxiliary variables. This is in contrast with many studies of compilation for planning. The reason why we focus on this setting is that we are interested in translating the *actions*, and not only the solutions to a planning problem. Our strict notion guarantees that translating the actions of a domain will preserve all properties : the existence of solution plans of course, but also their length, their number, and measures of complexity like many notions of width [17, 12, for instance].

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.

Negation Normal Form A Boolean formula φ over a set Q of variables is said to be in *negation normal form* (NNF) if it is built up from literals using conjunctions and disjunctions, i.e., if it is generated by the following grammar (where q ranges over Q) :

$$\varphi ::= q \mid \neg q \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$$

3 Action Languages

We first define the semantics for operators which then will be combined to obtain the languages we study. Applying a nondeterministic action a in state s can be seen as choosing

nondeterministically which groups of variables to assign from a set of possible combinations. This motivates the following definition.

Definition 4 (effect)**.** An *effect* over a set of variables $P \subset \mathbb{P}$ is an ordered pair $\langle Q^+, Q^- \rangle$ with $Q^+, Q^- \subseteq P$ and $Q^+ \cap Q^- = \emptyset$. The set Q^+ (resp. Q^-) is called a *positive* (resp. *negative*) effect.

In the following we are going to define the set $\mathrm{E}(\alpha, P, s)$ of *explicit effects of* α *in* s for action descriptions α, but we already use it to define the interpretation function, which simply formalizes the fact that variables not explicitly set by the action retain their value. For all action descriptions α and sets of variables P with $\mathrm{V}(\alpha) \subseteq P \subset \mathbb{P}$,

$$\forall s \subseteq P\colon\ I(\alpha, P)(s) := \{(s \cup Q^+) \setminus Q^- \mid \langle Q^+, Q^- \rangle \in \mathrm{E}(\alpha, P, s)\}$$

An effect $\langle Q^+, Q^- \rangle$ is said to *witness* a transition (s, s') if $(s \cup Q^+) \setminus Q^- = s'$. We emphasize that for a transition (s, s') there may exist several effects witnessing it. For example, given $P := \{p_1, p_2\}$, both effects $\langle \{p_2\}, \emptyset \rangle$ and $\langle \{p_1, p_2\}, \emptyset \rangle\}$ cause the same transition from $s := \{p_1\}$ to $s' := \{p_1, p_2\}$.

We say that two effects $e_1 := \langle Q_1^+, Q_1^- \rangle$, $e_2 := \langle Q_2^+, Q_2^- \rangle$ *are consistent* if $Q_1^+ \cap Q_2^- = Q_2^+ \cap Q_1^- = \emptyset$; in other words, if one effect assigns $\top$ to a variable, the other does not assign $\bot$ to it. Then the *combination* of e_1 and e_2 is defined to be the effect $\langle Q_1^+ \cup Q_2^+, Q_1^- \cup Q_2^- \rangle$. It can be interpreted as simultaneously executing the assignments of e_1 and e_2.

Definition 5. Our action languages will be defined as various combinations of constructs from the grammar

$$\alpha \ ::= \varepsilon \mid +p \mid -p \mid \varphi \rhd \alpha \mid (\alpha \cup \alpha) \mid \alpha \sqcap \alpha \mid \alpha ; \alpha \mid \neg_{\min} \alpha$$

Intuitively, ε describes the action with no effect ($\forall s\colon\ \varepsilon(s) = \{s\}$), $+p$ (resp. $-p$) is the action which sets p true (resp. false), $\rhd$ denotes conditional execution, $\cup$ denotes (exclusive) nondeterministic choice, $\sqcap$ denotes simultaneous execution, ; denotes sequential execution, and $\neg_{\min}$ describes the negation, and, importantly, variables not explicitly set by the action are assumed to keep their value. Also observe that auxiliary variables are not allowed — only variables in $\mathbb{P}$ can occur.

Definition 6. Now we can define the effects $\mathrm{E}(\alpha, P, s)$ of action descriptions, and thus the semantics of all action languages in this article :

1. $\mathrm{E}(\varepsilon, P, s) := \{\langle \emptyset, \emptyset \rangle\}$,
2. $\mathrm{E}(+p, P, s) := \{\langle \{p\}, \emptyset \rangle\}$, and $\mathrm{E}(-p, P, s) := \{\langle \emptyset, \{p\} \rangle\}$,
3. $\mathrm{E}(\varphi \rhd \alpha, P, s) := \begin{cases} \mathrm{E}(\alpha, P, s) & \text{if } s \models \varphi, \\ \{\langle \emptyset, \emptyset \rangle\} & \text{otherwise,} \end{cases}$
4. $\mathrm{E}(\alpha \cup \beta, P, s) := \mathrm{E}(\alpha, P, s) \cup \mathrm{E}(\beta, P, s)$,
5. $\mathrm{E}(\alpha \sqcap \beta, P, s) := \{\langle Q_\alpha^+ \cup Q_\beta^+, Q_\alpha^- \cup Q_\beta^- \rangle \mid \langle Q_\alpha^+, Q_\alpha^- \rangle \in \mathrm{E}(\alpha, P, s), \langle Q_\beta^+, Q_\beta^- \rangle \in \mathrm{E}(\beta, P, s), Q_\alpha^+ \cap Q_\beta^- = Q_\alpha^- \cap Q_\beta^+ = \emptyset\}$,
6. $\mathrm{E}(\alpha ; \beta, P, s) := \{\langle Q_\beta^+ \cup (Q_\alpha^+ \setminus Q_\beta^-), Q_\beta^- \cup (Q_\alpha^- \setminus Q_\beta^+) \rangle \mid \langle Q_\alpha^+, Q_\alpha^- \rangle \in \mathrm{E}(\alpha, P, s), t := (s \cup Q_\alpha^+) \setminus Q_\alpha^-, \langle Q_\beta^+, Q_\beta^- \rangle \in \mathrm{E}(\beta, P, t)\}$,
7. $\mathrm{E}(\neg_{\min} \alpha, P, s) := \{\langle s' \setminus s, s \setminus s' \rangle \mid s' \notin \alpha(s)\}$

The item 5 says that the effects of $\alpha \sqcap \beta$ in s are all *possible* combinations of effects of α and β in s. As an example, for $\alpha := (+p_1 \cup (-p_2 \sqcap +p_3)) \sqcap (-p_2 \cup +p_2)$, $P := \{p_1, p_2, p_3\}$, and any s, we have $\mathrm{E}(\alpha, P, s) = \{\langle \{p_1\}, \{p_2\} \rangle, \langle \{p_1, p_2\}, \emptyset \rangle, \langle \{p_3\}, \{p_2\} \rangle\}$, since in the last combination, $-p_2 \sqcap +p_3$ and $+p_2$ are not consistent. [1]

Item 6 says that if a variable is assigned by both α and β, then it appears in the effects of $\alpha ; \beta$ with the same polarity as in the effects of β. For example, for $\alpha := +p_1 \sqcap +p_2$ and $\beta := -p_1$, the only effect of $\alpha ; \beta$ in the state $s := \emptyset$ is $\langle \{p_2\}, \{p_1\} \rangle$, because the $-p_1$ in β "beats" the $+p_1$ in α.

And item 7 says that $\neg_{\min} \alpha$ defines the transition from s to exactly all the states which were not accessible via α, with the witnessing effect being the one that assigns only the variables which change their value during this transition. For example, if $P := \{p_1, p_2, p_3\}$, $s := \{p_1\}$ and $\alpha := +p_2 \cup (-p_1 \sqcap +p_3)$, then $(\neg_{\min} \alpha)(s) = \{\emptyset, \{p_1\}, \{p_2\}, \{p_1, p_3\}, \{p_2, p_3\}, \{p_1, p_2, p_3\}\}$, and the effect of $\neg_{\min} \alpha$ witnessing the transition from $\{p_1\}$ to $\{p_1, p_3\}$ is $\langle \{p_3\}, \emptyset \rangle$.

We emphasize that explicit effects (of subactions) matter only for $\sqcap$, in the sense that for other constructs, the transition relation $\|\alpha\|$ of α only depends on the transition relation of its subactions.

Note that the expression $+p \sqcap -p$ (for an arbitrary $p \in \mathbb{P}$) defines an action with no successor (which can be interpreted as an execution failing). We use $\bot$ as a shorthand for it (hence $I(\bot, P)(s) = \emptyset$ for all $P \subset \mathbb{P}, s \subseteq P$). We also highlight that $\rhd$ is not necessarily expressing a precondition. If φ is not satisfied in a state s then $\varphi \rhd \alpha$ simply modifies no variable in s. If we want to express that φ is a precondition for α being applicable we need to write $(\varphi \rhd \alpha) \sqcap (\neg \varphi \rhd \bot)$.

Being given the grammar from Definition 5 and the interpretation from Definition 6 we can now define our action languages. In the following definitions p ranges over $\mathbb{P}$ and φ over formulas in NNF over $\mathbb{P}$.

Definition 7 (**NPDDL**)**.** An **NPDDL** *action description* is an expression α generated by the grammar

$$\alpha \ ::= \varepsilon \mid +p \mid -p \mid \varphi \rhd \alpha \mid (\alpha \cup \alpha) \mid \alpha \sqcap \alpha$$

NPDDL can be seen as an abstract version of grounded PDDL extended to nondeterminism (in the sense of [4]).

1. Note that this is in contrast to the usual semantics of STRIPS, where addition would override deletion, thus "forgetting" $-p_2$.

The next language can be seen as a nondeterministic extension of STRIPS [6] with arbitrary boolean NNF conditions.

Definition 8 (conditional STRIPS)**.** An **NSTRIPS** *action description* is an **NPDDL** expression of the form

$$\prod_{i=0}^{n}\Big(\varphi_i \rhd \big((\ell_i^{1,1}\sqcap\ldots\sqcap\ell_i^{1,j_1})\cup\ldots\cup(\ell_i^{k_i,1}\sqcap\ldots\sqcap\ell_i^{k_i,j_{k_i}})\big)\Big)$$

where each $\ell_i^{k,j}$ is either ε, $+p$ or $-p$ for some $p\in\mathbb{P}$. In words, an **NSTRIPS** action description specifies a set of conditions so that, when the action is applied in a state s, for each condition satisfied by s exactly one of the corresponding effects occurs.

The last two languages can be seen as extensions of **NPDDL** via the sequence or the negation operator.

Definition 9 (**NPDDL**$_{\text{seq}}$, **NPDDL**$_{\text{not}}$)**.** An **NPDDL**$_{\text{seq}}$ action description α is generated by the grammar

$$\alpha ::= \varepsilon \mid +p \mid -p \mid \varphi\rhd\alpha \mid (\alpha\cup\alpha) \mid \alpha\sqcap\alpha \mid \alpha;\alpha$$

An **NPDDL**$_{\text{not}}$ action description α is generated by the grammar

$$\alpha ::= \varepsilon \mid +p \mid -p \mid \varphi\rhd\alpha \mid (\alpha\cup\alpha) \mid \alpha\sqcap\alpha \mid \neg_{\min}\alpha$$

Obviously **NSTRIPS** is a sublanguage of **NPDDL** in the sense of Definition 2, and **NPDDL** is a sublanguage of **NPDDL**$_{\text{not}}$ and **NPDDL**$_{\text{seq}}$. **NSTRIPS** is complete, since any action a can at least be represented by one condition for each state s (satisfied only by s), associated to either (1) a choice ($\cup$) between some "conjunctions" of atoms, one conjunction per a-successor s' of s (setting all variables as in s'), or (2) to the degenerate choice of conjunctions $\bot$, when $a(s)$ is empty. Therefore all languages in this article are complete.

While the choice $\cup$ is a natural construct to allow for expressing nondeterminism, and $\pm p$, $\sqcap$ and $\rhd$ are abstractions of the natural properties of PDDL and STRIPS, the sequence and the negation operators require an illustration of their possible use.

Example 10. *The sequence operator is particularly useful for describing actions featuring an intermediate nondeterministic process whose outcome influences the final result. For example, the* **NPDDL**$_{\text{seq}}$ *expression*

$$\begin{array}{l}+p_{\text{even}}\,;\\ \big(+p_1\sqcap(p_{\text{even}}\rhd-p_{\text{even}})\sqcap(\neg p_{\text{even}}\rhd+p_{\text{even}})\big)\ \cup\ -p_1\,;\\ \ldots\\ \big(+p_n\sqcap(p_{\text{even}}\rhd-p_{\text{even}})\sqcap(\neg p_{\text{even}}\rhd+p_{\text{even}})\big)\ \cup\ -p_n\,;\\ \neg p_{\text{even}}\rhd\bot\end{array}$$

describes an action which maps any state to the set of all states with an even number of p_i's, and p_{even}, set to true. An action description without the sequence connective would need to enumerate all possible combinations directly, resulting in an exponential tree.

Example 11. *Imagine a self-driving car on a road where there might be an obstacle. Then a possible scenario could be a collision, with the following* **NPDDL** *description over the state variables* $P=\{\text{high\_speed},\text{obstacle},\text{tire\_flat},\text{tank\_leaking},\text{bumper\_dented},\text{door\_blocked},\text{traffic\_barrier},\text{upside\_down}\}$ *:*

$$\beta \coloneqq \text{obstacle}\rhd\left(\begin{array}{l}\big((+\text{tire\_flat}\cup\varepsilon)\sqcap(-\text{high\_speed}\cup\varepsilon)\big)\\ \sqcap\big(\text{high\_speed}\rhd\left(\begin{array}{l}(+\text{tank\_leaking}\cup\varepsilon)\\ \sqcap\text{bumper\_dented}\\ \sqcap\text{door\_blocked}\end{array}\right)\big)\\ \sqcap\big(\neg\text{traffic\_barrier}\rhd+\text{upside\_down}\cup\varepsilon\big)\end{array}\right)$$

This "action" describes the effect of the collision, hence $\neg_{\min}\beta$ describes that the effects of the collision do not *occur. As a consequence, if α otherwise describes the effects of one of the actions of the car, action $\alpha\wedge\neg_{\min}\beta$ describes the same action but taking into account the fact that the car has a built-in collision avoidance system.*

Representations Since we are interested in the succinctness of languages, it is crucial to define the *size* of action descriptions. For this we consider two variants of each language. The first variant corresponds to a representation of the expressions α in the language by their syntactic *tree* (including the Boolean formulas occuring in the expression, if any). The second variant corresponds to the representation of these expressions α 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. 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 tree or from a nonreduced circuit.

We write **T-NPDDL**, **T-NPDDL**$_{\text{seq}}$ and **T-NPDDL**$_{\text{not}}$ for those languages with expressions represented as trees, and **C-NNFAT**, **C-NPDDL**, **C-NPDDL**$_{\text{seq}}$ and **C-NPDDL**$_{\text{not}}$ for those languages with expressions represented as reduced circuits. Since **NSTRIPS** is flat (the depth of the underlying graph is bounded), there is no difference between the circuit and the tree versions up to polynomial-time transformations, except for the representation of conditions φ; since, as it turns out, the representation of conditions does not affect the complexity results in this paper, we only write **NSTRIPS** without specifying the representation.

4 Complexity of Queries

We now turn to studying the complexity of *queries* on action descriptions. We concentrate on three natural queries

for planning, corresponding to checking the existence of a transition, deciding applicability of an action, and deciding whether a (sequential) plan reaches a goal.

Definition 12 (queries)**.** Let $\langle L, I\rangle$ be a fixed action language, $\alpha, \alpha_1, \ldots, \alpha_k$ be action descriptions in L, $P \subset \mathbb{P}$ be a set of variables such that $I(\alpha, P)$ and $I(\alpha_1, P), \ldots, I(\alpha_k, P)$ are defined, and s, s' be P-states. We consider the following decision problems.

— Is-Succ : given α, P, s, s', decide whether s' is an α-successor of s.

— Is-Applic : given α, P, s, decide whether $\alpha(s)$ is non-empty.

— Entails : given $\alpha_1, \ldots, \alpha_k$, P, s, and an **NNF** formula φ over P, decide whether $s' \models \varphi$ for all $s' \in (\alpha_1 ; \ldots ; \alpha_k)(s)$. [2]

Note that if $\alpha_1 ; \ldots ; \alpha_k$ has no successors for s, then it automatically entails any formula φ, especially any unsatisfiable formula, say $\varphi := p \wedge \neg p$.

Is-Succ is the most basic query which corresponds to model checking for Boolean formulas. We will see later that it is usually enough to know the complexity of Is-Succ to conclude about the other queries.

The first result is a technical one and will be used later in succinctness proofs. It can be easily deduced from the fact that for a bounded number of variables the number of possible effects is bounded, too.

Lemma 13. *Let $k \in \mathbb{N}$ be fixed, and assume $|P| \leq k$ (i.e. the size of the scope of the described actions is fixed). Then Is-Succ can be solved in polynomial time for all languages which we consider.*

The next two statements were proven in [20].

Proposition 14. *Is-Succ is* NP*-complete for* **NSTRIPS**, **T-NPDDL**, **C-NPDDL** *and* **T-NPDDL**$_{\text{seq}}$.

Proposition 15. *Is-Succ is* PSPACE*-complete for* **C-NPDDL**$_{\text{seq}}$.

In order to prove the theorem about Is-Succ in **NPDDL**$_{\text{not}}$, we require a notation and a lemma.

Notation 16. Let $n \in \mathbb{N}$, and let $X_n := \{x_1, \ldots, x_n\}$ be a set of variables. Observe that there are a cubic number N_n of clauses of length 3 over X_n (any choice of 3 variables with a polarity for each). 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 $\{p_1, p_2, \ldots, p_{N_n}\}$. 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.

2. By $s' \in (\alpha_1 ; \ldots ; \alpha_k)(s)$ we mean that there are $s_0, s_1, \ldots, s_k$ with $s = s_0$, $s_i \in \alpha_i(s_{i-1})$ for $i = 1, \ldots, k$, and $s_k = s'$.

Example 17. *Let $n := 2$, and consider an enumeration of all clauses over $X_2 := \{x_1, x_2\}$ which starts with $\gamma_1 := (x_1 \vee x_1 \vee x_2), \gamma_2 := (x_1 \vee x_1 \vee \neg x_2), \gamma_3 := (x_1 \vee \neg x_1 \vee x_2), \gamma_4 := (x_1 \vee \neg x_1 \vee \neg x_2), \gamma_5 := (\neg x_1 \vee \neg x_1 \vee x_2), \ldots$.*

Then the 3-CNF *$\varphi := (x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_1 \vee x_2)$ is encoded by the state $s(\varphi) = \{p_1, p_5\}$.*

The statement of the lemma can be shown by an easy induction.

Lemma 18. *Let φ, ψ be* 3-CNF *formulas, let $s(\varphi)$ and $s(\psi)$ be as in Notation 16, and define the QBF $\Psi^n_\varphi := \exists x_1 \colon \neg(\exists x_2 \colon \neg(\ldots \neg(\exists x_n \colon \varphi)\ldots))$. Then the following hold :*

1. *If n is odd : if Ψ^n_φ is true and $s(\psi) \subseteq s(\varphi)$ then Ψ^n_ψ is true*
   dually : if Ψ^n_φ is false and $s(\psi) \supseteq s(\varphi)$ then Ψ^n_ψ is false
2. *If n is even : if Ψ^n_φ is true and $s(\psi) \supseteq s(\varphi)$ then Ψ^n_ψ is true*
   dually : if Ψ^n_φ is false and $s(\psi) \subseteq s(\varphi)$ then Ψ^n_ψ is false

Notation 19. Using Notation 16, for all $n \in \mathbb{N}$ we define the **T-NPDDL** action description α^{sat}_n :

$$\alpha^{\text{sat}}_n := \prod_{x \in X_n} \left(\Big(\prod_{\gamma_i \colon x \in \gamma_i} (+p_i \cup \varepsilon) \Big) \cup \Big(\prod_{\gamma_i \colon \neg x \in \gamma_i} (+p_i \cup \varepsilon) \Big) \right)$$

Intuitively, α^{sat}_n chooses an assignment (true or false) to each variable in X_n (outermost $\cup$). Whenever it chooses an assignment for x, for each possible clause which is satisfied by this assignment (innermost $\sqcap$), it chooses whether to include this clause into the result, or not (innermost $\cup$). Hence it builds a formula which is satisfied at least by its choices, and clearly, any satisfiable 3-CNF formula can be built in this manner.

Lemma 20. *Let $n \in \mathbb{N}$, and let φ be a* 3-CNF *formula over X_n. Then φ is satisfiable if and only if $s(\varphi)$ is an α^{sat}_n-successor of the state $\emptyset$.*

Proposition 21. *Is-Succ is* PSPACE*-complete for* **T-NPDDL**$_{\text{not}}$ *and* **C-NPDDL**$_{\text{not}}$.

Proof. For membership observe that every execution of the circuit can be simulated in polynomial space and we can do it until we find a witness for the transition (s, s'), if any.

Not let the sets P_n, X_n and clauses γ_i be as in Notation 16. For hardness, we show that a quantified Boolean formula $\Phi := \exists x_1 \colon \neg(\exists x_2 \colon \neg(\ldots(\exists x_n \colon \varphi)))$, with φ a 3-CNF, is true if and only if $s(\varphi) \in \alpha^n_1(\emptyset)$ with

1. $\alpha^n_n := \chi^n_n$
2. $\alpha^n_k := \chi^n_k \sqcap \Big(\neg_{\min} \neg_{\min} (\rho^n_k \sqcap (\neg_{\min} \alpha^n_{k+1})) \Big)$ with
   — $\chi^n_k := \Big(\prod_{\gamma_i \colon x_k \in \gamma_i} (+p_i \cup \varepsilon) \Big) \cup \Big(\prod_{\gamma_i \colon \neg x_k \in \gamma_i} (+p_i \cup \varepsilon) \Big)$

— $\rho_k^n := (\bigsqcap_{\gamma_i:\, x_k \in \gamma_i} -p_i) \cup (\bigsqcap_{\gamma_i:\, \neg x_k \in \gamma_i} -p_i)$

In the following we give an intuition for what these (sub)actions do in $s := \emptyset$. ρ_k^n chooses an assignment to x_k and explicitly prohibits to add any of the clauses that contain a literal which is true under this assignment (i.e. is satisfied by this assignment). χ_k^n chooses an assignment to x_k and then produces nondeterministically any possible set of clauses which are satisfied by this assignment. Recall that for all actions δ we have $\|\delta\| = \|\neg_{\min}\neg_{\min}\delta\|$; the double negation here serves to make sure that transitions are witnessed by minimal effects.

Throughout the proof we use the following notation : $\nu_{=k}$ refers to an assignment to x_k and $\nu_{\leq \ell}$ to an assignment to $x_1, \ldots, x_\ell$. If $\nu_{\leq k-1}$ and $\nu_{=k}$ are clear from the context, then $\nu_{\leq k}$ refers to the induced joint assignment to $x_1, \ldots, x_k$. Finally, for an assignment ν we denote by $t(\nu)$ the encodings (via Notation 16) of 3-clauses which are satisfied by ν.

We denote by $\Phi_{\nu_{\leq k}}$ the formula $\exists x_{k+1} \colon \neg(\exists x_{k+2} \colon \neg(\ldots \neg(\exists x_n \colon \varphi_{|\nu_{\leq k}})))$. Now we prove the claim that for every $0 \leq k \leq n-1$ we have :

$$\text{for all } \nu_{\leq k} \colon [\Phi_{\nu_{\leq k}} \text{ true } \iff s(\varphi) \setminus t(\nu_{\leq k}) \in \alpha_{k+1}^n(\emptyset)]$$

After having shown the claim for $k = 0$ we are done with showing the main claim because $\Phi_{\nu \leq 0} = \Phi$and $t(\nu_{\leq 0}) = \emptyset$ (since $\nu_{\leq 0}$ is the empty assignement). .

We start with $k := n-1$. In this case, $\Phi_{\nu_{\leq k}}$ is just an existentially quantified 3-CNF which is true (being already conditioned by an assignment $\nu_{\leq k}$ to $x_1, \ldots, x_{n-1}$) if and only if the clauses which have not yet been satisfied by $\nu_{\leq k}$ (i.e. $s(\varphi) \setminus t(\nu_{\leq k})$) can be satisfied by an assignment to x_n, which is equivalent to $s(\varphi) \setminus t(\nu_{\leq k})$ being an α_n^n-successor of $\emptyset$ (the statement is analogous to the statement of Lemma 20). Now we assume that the claim has already been shown for k, and we want to show it for $k-1$: we consider an assignment $\nu_{\leq k-1}$ to $x_1, \ldots, x_{k-1}$, and we prove that $\Phi_{\nu_{\leq k-1}}$ is true if and only if $s(\varphi) \setminus t(\nu_{\leq k-1}) \in \alpha_k^n(\emptyset) = \left(\chi_k^n \sqcap \left(\neg_{\min}\neg_{\min}(\rho_k^n \sqcap (\neg_{\min}\alpha_{k+1}^n))\right)\right)(\emptyset)$.

" $\Longrightarrow$:" Suppose that $\Phi_{\nu_{\leq k-1}}$ is true. Then there exists an assignment $\nu_{=k}$ to x_k such that $\Phi_{\nu_{\leq k}}$ is false. By the induction hypothesis this means that $T := s(\varphi) \setminus t(\nu_{\leq k}) \notin \alpha_{k+1}^n(\emptyset)$ and thus $T \in \neg_{\min}\alpha_{k+1}^n(\emptyset)$. Since $t(\nu_{=k}) \subseteq t(\nu_{\leq k})$, we have $t(\nu_{=k}) \cap T = \emptyset$, therefore the execution of ρ_k^n (re)assigning variables in $t(\nu_{=k})$ to $\bot$ is consistent with the effect of $\neg_{\min}\alpha_{k+1}^n$ leading to T. Hence $T \in (\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$, and adding the double negation we have $T \in \neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$ with the guarantee that the effect e_T witnessing this transition is minimal – and thus, since the transition starts from $\emptyset$, that e_T has no negative effects.

Now observe that, by construction of χ_k^n, any subset U of $t(\nu_{=k})$ is a successor of $\emptyset$ via χ_k^n and the transition is witnessed by a purely positive effect e_U. This applies to the subset $U := (s(\varphi) \setminus t(\nu_{\leq k-1})) \cap t(\nu_{=k})$. All in all, combining e_T and e_U (which are necessarily consistent since both contain only positive effects), we get that $U \cup T \in \chi_k^n \sqcap \left(\neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)\right)(\emptyset) = \alpha_k^n(\emptyset)$. Given that $t(\nu_{\leq k})) = t(\nu_{\leq k-1}) \cup t(\nu_{=k})$, it is not hard to see that $U \cup T = s(\varphi) \setminus t(\nu_{\leq k-1})$, which proves the claim.

" $\Longleftarrow$:" Now assume that $s(\varphi) \setminus t(\nu_{\leq k-1}) \in \alpha_k(\emptyset) = \chi_k^n \sqcap \left(\neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)\right)(\emptyset)$.

We first consider the case when $n-k$ is odd. From the assumption we get that there exist U, T such that $s(\varphi) \setminus t(\nu_{\leq k-1}) = U \cup T$ with $U \in \chi_k^n(\emptyset)$ and $T \in \neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$. So $T \subseteq s(\varphi) \setminus t(\nu_{\leq k-1})$ so there exists a V such that $(s(\varphi) \setminus t(\nu_{\leq k-1})) \setminus V \in \neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$, and therefore $(s(\varphi) \setminus t(\nu_{\leq k-1})) \setminus V \in (\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$. Let $\langle Q^+, Q^- \rangle$ be the effect of $\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n$ witnessing this transition. By definition of ρ_k^n (which is responsible for Q^- since $\neg_{\min}\alpha_{k+1}^n$ has only positive effects in $\emptyset$) there exists an assignment $\nu_{=k}$ to x_k such that $t(\nu_{=k}) = Q^-$. Since $Q^+ \cap Q^- = \emptyset$, it holds that $Q^+ \subseteq (s(\varphi) \setminus t(\nu_{\leq k-1})) \setminus t(\nu_{=k}) = s(\varphi) \setminus t(\nu_{\leq k})$. It also holds that $Q^+ \in \neg_{\min}\alpha_{k+1}^n(\emptyset)$ because ρ_k^n has only negative effects. Thus Q^+ is a set of clauses which encodes a 3-CNF formula ψ with $s(\psi) \subseteq s(\varphi) \setminus t(\nu_{\leq k}) \subseteq s(\varphi)$ and by the inductive assumption $\exists x_{k+1} \colon \neg(\exists x_{k+2} \colon \neg \ldots \neg(\exists x_n \colon \psi_{|\nu_{\leq k}}) \ldots)$ is false. With the first statement of Lemma 18 (recall that $n-k$ and hence the number of quantifiers was odd) it follows that $\exists x_{k+1} \colon \neg(\exists x_{k+2} \colon \neg \ldots \neg(\exists x_n \colon \varphi_{|\nu_{\leq k}}) \ldots)$ is false, too, so $\Phi_{\nu_{\leq k-1}}$ is true.

We now consider the case that $n-k$ is even. For the effect $\langle Q^+, Q^- \rangle$ witnessing the transition from $\emptyset$ to $s(\varphi) \setminus t(\nu_{\leq k-1})$ there must be an effect $\langle U^+, \emptyset \rangle$ of χ_k^n with $U^+ \subseteq Q^+$ and by construction of χ_k^n there exists an assignment $\nu_{=k}$ with $U^+ \subseteq t(\nu_{=k})$. Therefore there must exist an effect $\langle T, \emptyset \rangle$ of $\neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)$ in $\emptyset$ with $T \supseteq Q^+ \setminus U^+ \supseteq (s(\varphi) \setminus t(\nu_{\leq k-1})) \setminus t(\nu_{=k}) = s(\varphi) \setminus t(\nu_{\leq k})$. It follows that $T \in \neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$ and therefore $T \in (\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$ and thus $T \in \neg_{\min}\alpha_{k+1}^n(\emptyset)$ since ρ_k^n does not have positive effects. T encodes a 3-CNF formula ψ such that $s(\psi) \setminus t(\nu_{\leq k}) \supseteq s(\varphi) \setminus t(\nu_{\leq k})$ and by the inductive assumption $\exists x_{k+1} \colon \neg(\exists x_{k+2} \colon \neg \ldots \neg(\exists x_n \colon \psi_{|\nu_{\leq k}}) \ldots)$ is false. Since $n-k$ is even we apply the second statement of Lemma 18 and conclude that $\exists x_{k+1} \colon \neg(\exists x_{k+2} \colon \neg \ldots \neg(\exists x_n \colon \varphi_{|\nu_{\leq k}}) \ldots)$ is false, too. Therefore $\Phi_{\nu_{\leq k-1}}$ is true. □

Lemma 22. *Is-Succ is polynomial-time reducible to Is-Applic for* **T-NPDDL**$_{\text{seq}}$, **C-NPDDL**$_{\text{seq}}$, **T-NPDDL**$_{\text{not}}$ *and* **C-NPDDL**$_{\text{not}}$.

Proof. For a state s let ψ_s denote the formula $(\bigwedge_{p \in s} p) \wedge (\bigwedge_{p \notin s} \neg p)$. Let α be a (tree- or circuit) **NPDDL**$_{\text{seq}}$ action. For all states s, s' it is easy to see that we have $s' \in \alpha(s)$ if and only if the action $(\alpha\,; (\neg\psi_{s'} \rhd \bot))$ is applicable in s.

Now let β be a (tree- or circuit) **NPDDL**$_{\text{not}}$ action description and s, s' be P-states. We define $\rho_{s'} := \left((\bigsqcap_{p \in s'} +p) \sqcap (\bigsqcap_{p \notin s'} -p)\right)$, the deterministic action leading from all states

to s'. Then we have $s' \in \beta(s)$ if and only if the action $\gamma := \neg_{\min}(\neg_{\min}\beta \cup \neg_{\min}\rho_{s'})$ is applicable in s because it is easy to see that γ satisfies $\gamma(s) = \beta(s) \cap \rho_{s'}(s)$. □

Corollary 23. *Is-Applic is* NP*-complete for* **NSTRIPS**, **T-NPDDL**, **C-NPDDL**, *and* **T-NPDDL**$_{\text{seq}}$, *and it is* PSPACE*-complete for* **C-NPDDL**$_{\text{seq}}$, **T-NPDDL**$_{\text{not}}$ *and* **C-NPDDL**$_{\text{not}}$.

Proof. For **NPDDL**$_{\text{seq}}$ and **NPDDL**$_{\text{not}}$ hardness follows from Lemma 22 together with the previous hardness results about Is-Succ (Propositions 15 and 21). For **NSTRIPS** the result has been proven in [20]. Membership is clear because applicability can be witnessed by giving a successor and verifying successorship. □

Lemma 24. *The complement of Is-Applic is polynomial-time reducible to Entails for all languages.*

Proof. An action described by α is non-applicable in s (i.e. it has no successors in s) if and only if all α-successors s' of s satisfy $p \wedge \neg p$ for an arbitrary p, i.e. if and only if α entails $\varphi := p \wedge \neg p$ in s. □

Corollary 25. *Entails is* coNP*-complete for* **NSTRIPS**, **T-NPDDL**, **C-NPDDL**, *and* **T-NPDDL**$_{\text{seq}}$, *and it is* PSPACE*-complete for* **C-NPDDL**$_{\text{seq}}$, **T-NPDDL**$_{\text{not}}$, *and* **C-NPDDL**$_{\text{not}}$.

Proof. We first recall that PSPACE = coPSPACE. Then hardness follows for all the languages from the reduction of non-applicability to entailment together with the hardness results about Is-Applic (Lemma 22). Membership has been shown in [20]. □

5 Succinctness

We now give negative results about the existence of polynomial-*size* translations, hence about the relative *succinctness* of languages [5]. Recall that all the languages which we study are complete, thus our definition is a bit simpler than the usual one.

Definition 26 (succinctness)**.** A complete action language L_1 is at least as succinct as a complete action language L_2 if there exists a polynomial-size translation from L_2 into L_1.

Clearly, if L_2 is a sublanguage of L_1 then L_1 is at least as succinct as L_2.

Our succinctness *separation* results rely on assumptions about the *nonuniform* complexity classes P/poly and NP/poly. Recall that P/poly is the class of all decision problems such that for all $n \in \mathbb{N}$, there is a polynomial-time algorithm which decides the problem for all inputs of size n, NP/poly is defined analogously. The assumptions NP $\not\subseteq$ P/poly, coNP $\not\subseteq$ NP/poly and PSPACE $\not\subseteq$ NP/poly which we use are standard ones.

Given two disjoint sets of variables P, Q a $(P \cup Q)$-action a, and an assignment $t \subseteq Q$ to the variables in Q, we define the *t-conditioning* of a to be the P-action $a_{|t}$ satisfying $\forall s \subseteq P$: $a_{|t}(s) = \{s' \mid (s' \cup t) \in \alpha(s \cup t)\}$. The next lemma will be used to partially separate **NPDDL**$_{\text{seq}}$ from **NPDDL**$_{\text{not}}$.

Lemma 27. *Let P and Q be disjoint sets of variables, α be a* **T-NPDDL**$_{\text{not}}$ *(resp.* **C-NPDDL**$_{\text{not}}$*) expression for a $(P \cup Q)$-action, and $t \subseteq Q$ be an assignment to the variables in Q. Then we can compute a* **T-NPDDL**$_{\text{not}}$ *(resp.* **C-NPDDL**$_{\text{not}}$*) expression $f(\alpha)$ for the t-conditioning of α in time polynomial in $|\alpha|$.*

Proof. Simply replace all leaves of the form $+q$ with $q \in Q \cap t$ by ε, all leaves of the form $+q$ with $q \in Q \setminus t$ by $\bot$ (failure), and dually for $-q$, and for all subexpressions $\varphi \rhd \beta$, simplify φ by the assignment t to Q. □

Proposition 28. *If* NP $\not\subseteq$ P/poly *then there is no polynomial-size translation from* **T-NPDDL**$_{\text{seq}}$ *to* **T-NPDDL**$_{\text{not}}$*, nor from* **C-NPDDL**$_{\text{seq}}$ *to* **C-NPDDL**$_{\text{not}}$.

Proof. Let $n \in \mathbb{N}$, and let φ be a 3-CNF formula over a set of variables $X_n := \{x_1, \ldots, x_n\}$. Recall from Notation 16 that we can encode φ over a set $P_n \subseteq \mathbb{P}$. Finally, let $p_{\text{sat}} \in \mathbb{P}$ be a fresh variable. We define γ_n^{sat} to be the following action :

$$\prod_{i=1}^{n}(+x_i \cup -x_i); (\psi_n \rhd +p_{\text{sat}}) \sqcap (\neg\psi_n \rhd -p_{\text{sat}}); \prod_{i=1}^{n} -x_i$$

where ψ_n is the NNF $\bigwedge_{i=1}^{N_n}(\neg p_i \vee \bigvee_{\ell \in \gamma_i} \ell)$, which is satisfied if and only if each clause γ_i which is in φ (as witnessed by p_i being true) is also satisfied. In words, γ_n^{sat} guesses an assignment to $\mathrm{V}(\varphi)$, then sets p_{sat} according to whether φ is satisfied by this assignment, and finally resets all guessed variables to false. Note that γ_n^{sat} depends on n but not on φ, and that γ_n^{sat} is polynomial in n.

Clearly, $s(\varphi) \cup \{p_{\text{sat}}\}$ is a γ_n^{sat}-successor of $s(\varphi)$ if and only if φ is satisfiable. Hence the following decision problem is NP-hard :

— *Input :* a 3-CNF formula φ

— *Question :* is $s(\varphi) \cup \{p_{\text{sat}}\}$ a γ_n^{sat}-successor of $s(\varphi)$?

Now assume that there is a polynomial-size translation f from **T-NPDDL**$_{\text{seq}}$ to **T-NPDDL**$_{\text{not}}$, and for all $n \in \mathbb{N}$ let $\delta_n^{\text{sat}} := f(\gamma_n^{\text{sat}})$. Let φ be a 3-CNF formula over n variables. Since δ_n^{sat} is in **NPDDL**$_{\text{not}}$, we can apply Lemma 27 with $Q := P_n \cup \{x_1 \ldots, x_n\}$ and $t := s(\varphi)$, to get an expression in which the only occuring variable is p_{sat}, and $\{p_{\text{sat}}\}$ is a successor of $\emptyset$ if and only if $s(\varphi) \cup \{p_{\text{sat}}\}$ is a δ_n^{sat}-sucessor of $s(\varphi)$, that is, if and only if φ is satisfiable. Since this expression has only one variable, with Lemma 13 we see that successorship can be decided in polynomial time, implying NP $\subseteq$ P/poly. The proof is exactly the same for circuits. □

Proposition 29. *If* PSPACE $\not\subseteq$ NP/poly *then there is no polynomial-size translation neither from* **C-NPDDL**$_{\text{not}}$ *into* **C-NPDDL** *nor from* **T-NPDDL**$_{\text{not}}$ *into* **T-NPDDL**.

PROOF. If **NPDDL**$_{\text{not}}$ was translatable into **NPDDL** with only a polynomial increase in size, we could translate α_1^n from Proposition 21 into a polynomial-sized equivalent action $f(\alpha_1^n)$ in **NPDDL**. Since α_1^n does not depend on a formula but only on the number of variables, it would follow that deciding the validity of a quantified Boolean formula of the form $\exists x_1 : \neg(\exists x_2 : \neg(\ldots \neg(\exists x_n : \varphi)))$ with a 3-CNF φ amounts to checking $s(\varphi) \in \alpha_1^n(\emptyset)$, and we would obtain a nonuniform NP-algorithm for a PSPACE-complete problem,implying PSPACE $\subseteq$ NP/poly. □

Proposition 30. *There exists no polynomial-size translation of* **C-NPDDL** *into* **NSTRIPS**.

PROOF. The majority function (which returns true if and only if at least half of its arguments are true) can be computed by a boolean circuit ψ of linear size and logarithmic depth [15]. Consider an action a over $P_n = \{p_1, \ldots, p_n\}$ which is applicable only in $s = \emptyset$ and in this state it produces nondeterministically all s' with $|s'| \geq \frac{n}{2}$. Thus it is obviously representable by a polynomial-size circuit NNF action theory ψ_n', and this can be translated into **C-NPDDL** in polynomial time [20]. Now every **NSTRIPS** representation of a can without loss of generality be assumed to be of the form $(\varphi_n \rhd \alpha_n) \sqcap (\neg\varphi_n \rhd \bot)$ with $\varphi_n = \neg p_1 \wedge \ldots \wedge \neg p_n$ and α_n being an unconditioned **NSTRIPS** expression. By replacing $\sqcap$ by $\wedge$, $\cup$ by $\vee$, $+p$ by p and $-p$ by $\neg p$ we would then obtain a formula over P_n whose models are α_n-successors of $\emptyset$, thus obtaining a boolean circuit of bounded depth of size polynomial in n for the majority function, which contradicts a result from [9]. □

6 Transformations

One of the key aspects of knowledge compilation is the study of transformations that a language supports [5]. In general, we are interested in finding out whether a given transformation is possible in polynomial time, and otherwise, whether it is at least possible without a superpolynomial explosion in size.

Let $\langle L, I\rangle$ be a fixed action language, let $\alpha, \alpha_1, \alpha_2$ be action descriptions in L, $P \subset \mathbb{P}$ be a set of variables such that $I(\alpha, P)$, $I(\alpha_1, P)$, and $I(\alpha_2, P)$ are defined, and let s, s' be P-states. We study the following computational problems.

— SEQUENCE : given $\alpha_1, \alpha_2 \in L$, compute an action description $\beta \in L$ such that for all $s : \beta(s) = \{s' \mid \exists t$ such that $t \in \alpha_1(s)$ and $s' \in \alpha_2(t)\}$.
— CHOICE : given $\alpha_1, \alpha_2 \in L$, compute an action description $\beta \in L$ such that for all $s : \beta(s) = \alpha_1(s) \cup \alpha_2(s)$.
— NEGATION : given $\alpha \in L$, compute an action description $\beta \in L$ such that for all $s : \beta(s) = \{s' \mid s' \notin \alpha(s)\}$.

We insist that for each one it is required that the output be in the same language as the input description(s). We also consider the problem of extracting the precondition.

— EX-PREC : given $\alpha \in L$ and $P \subset \mathbb{P}$, compute an NNF formula φ such that for all $s : \alpha(s) \neq \emptyset \iff s \models \varphi$.

We see it as a transformation because it amounts to computing an action description of the form $\neg\varphi \rhd \bot$ expressing when the action does not satisfy its precondition.

Of course, if a language allows for arbitrary nesting of an operator then it trivially supports a transformation. For example, **NPDDL** allows to express CHOICE of α_1 and α_2 via $\alpha_1 \cup \alpha_2$. In other cases, determining the complexity of a transformation seems to be not much easier than determining the succinctness of the language enriched with the corresponding operator.

Proposition 31. *The following transformations are polynomial-time, in all cases for both the tree and circuit representations :*

— *CHOICE for* **NPDDL**, **NPDDL**$_{\text{not}}$ *and* **NPDDL**$_{\text{seq}}$ *;*
— *SEQUENCE for* **NPDDL**$_{\text{seq}}$ *;*
— *NEGATION for and* **NPDDL**$_{\text{not}}$.

Proposition 32. *If* NP $\not\subseteq$ P/poly *then* **NSTRIPS**, **T-NPDDL**, **C-NPDDL**, **T-NPDDL**$_{\text{not}}$ *and* **C-NPDDL**$_{\text{not}}$ *do not support polynomial-size SEQUENCE.*

PROOF. Consider the action γ_n^{sat} from the proof of Proposition 28. Each of its subactions which are connected via the sequence operator (these are $\bigsqcap_{i=1}^n (+x_i \cup -x_i)$, $(\psi_n \rhd +p_{\text{sat}}) \sqcap (\neg\psi_n \rhd -p_{\text{sat}})$ and $\bigsqcap_{i=1}^n -x_i$) is a **NSTRIPS** (and therefore **NPDDL**, **NPDDL**$_{\text{not}}$) action. Thus the proof of Proposition 28 can be reused to show that if NP $\not\subseteq$ P/poly then SEQUENCE cannot be polynomial-size for **NSTRIPS**, **NPDDL**, nor **NPDDL**$_{\text{not}}$. □

Proposition 33. *If* coNP $\not\subseteq$ NP/poly *then NEGATION is not polynomial-size for* **T-NPDDL**, **C-NPDDL**, *nor* **T-NPDDL**$_{\text{seq}}$.

PROOF. Recall for α_n^{sat} from Lemma 20 that $s(\varphi) \in \alpha_n^{\text{sat}}(\emptyset)$ holds if and only if φ is satisfiable. Now suppose that NEGATION is polynomial-size in **T-NPDDL**, **C-NPDDL**, or **T-NPDDL**$_{\text{seq}}$. Then there exists a polynomial-sized equivalent $f(\alpha_n^{\text{sat}})$ of the negation of α_n^{sat}. Thus $s(\varphi) \in f(\alpha_n^{\text{sat}})$ holds if and only if φ is unsatisfiable, and so there is a nonuniform NP-algorithm for a coNP-complete problem.□

The following results show that it is in general more efficient to represent the precondition of actions implicitly in the description rather than as a separate formula.

Proposition 34. *If* NP $\not\subseteq$ P/poly *then EX-PREC is not polynomial-size for* **NSTRIPS**, *nor for* **NPDDL**, **NPDDL**$_{\text{not}}$, **NPDDL**$_{\text{seq}}$ *under neither the tree nor the circuit representation.*

PROOF. Consider the **NSTRIPS** action over $X_n \cup P_n$ (as in Notation 16) : $\xi_n := \bigsqcap_{p_i} \left(p_i \rhd \left((\bigcup_{x \in \gamma_i} +x) \cup (\bigcup_{\neg x \in \gamma_i} -x)\right)\right)$

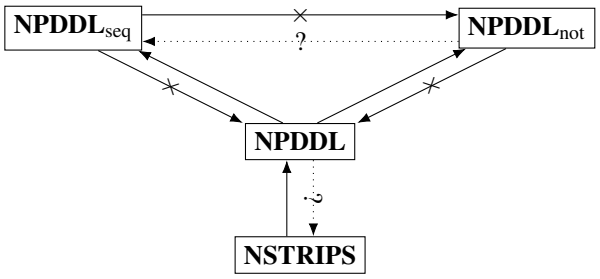

FIGURE 1 – Succinctness results. An arc from L_1 to L_2 denotes that L_1 is a sublanguage of L_2. A crossed out arc from L_1 to L_2 means that there is no polynomial-size translation from L_1 into L_2. These relations hold for both tree and circuit representations. Dotted arcs denote open questions (see text).

and observe that that ξ_n is applicable in $s(\varphi)$ if and only if φ is satisfiable. If there was a polynomial-size NNF representation of the precondition ψ_n of ξ_n, we could check φ for satisfiability by checking whether $s(\varphi) \models \psi_n$, and thus we would obtain a non-uniform polynomial-time algorithm for 3-SAT. To see the claim for the other languages, observe that if EX-PREC was polynomial-size for **NPDDL**, **NPDDL**$_{\text{not}}$ or **NPDDL**$_{\text{seq}}$, it would be polynomial-size for their sublanguage **NSTRIPS** (since the representation of the output does not depend on the language). □

7 Conclusion

We have studied several languages for describing nondeterministic actions along two criteria : succinctness and complexity of different decision problems and transformations natural to automated planning. We have also considered two representations, by trees and by circuits. Our results are summarized in Table 1 and on Figure 1.

An interesting result is the different complexity of queries for **NPDDL**$_{\text{seq}}$ with trees or circuits. While it is intuitively clear that there must be some languages which are strictly less succinct with trees than with circuits, and languages with less tractable queries with circuits than with trees, this gives a concrete example of this phenomenon. Finally, it is interesting that the complexity of the three queries is the same for tree-represented **NPDDL** with or without sequence. Since the language is strictly more succinct with sequence, **T-NPDDL**$_{\text{seq}}$ seems to be a strictly more interesting language than **T-NPDDL**.

We leave some problems open for transformations (see Table 1) and succinctness. For succinctness, a dotted arc from L_1 to L_2 on Figure 1 means that the existence of a polynomial-size translation from L_1 to L_2 is open ; for all these arcs we conjecture that there is actually none.

It would also be interesting to determine the complexity of other queries and transformations. Queries obviously useful to planning are to count and enumerate successors, generate a successor of a given state uniformly at random (as needed in Monte-Carlo approaches), to determine whether an action is deterministic, and whether all executions of a given action sequence are free of dead-ends. As concerns transformations, the complexity of combining two actions via logical conjunction $\wedge$ does not seem so easy to determine. Another interesting transformation, in particular for regression approaches to planning, is the computation of the "reverse" action $\bar{\alpha}$ of α with $s' \in \bar{\alpha}(s) \Leftrightarrow s \in \alpha(s')$.

Our main perspective is a more systematic study, for languages constructed using combinations of features like the sequence operator, modalities, Kleene star, etc. A very expressive language to consider is **DL-PPA** [10]. Another perspective is to consider languages for stochastic actions, and for actions with observations (and hence queries on *belief states* rather than states). The ultimate goal is to draw clear pictures of what language to choose depending on the queries which are used by, e.g., a planning algorithm or a simulator.

Acknowledgements

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

Références

[1] Bäckström, Christer: *Expressive Equivalence of Planning Formalisms*. Artificial Intelligence, 76(1-2) :17–34, 1995.

[2] Bäckström, Christer et Peter Jonsson: *Algorithms and Limits for Compact Plan Representations*. Journal of Artificial Intelligence Research, 44 :141–177, 2012.

[3] Benthem, Johan van, Jan van Eijck, Malvin Gattinger et Kaile Su: *Symbolic Model Checking for Dynamic Epistemic Logic — S5 and Beyond*. Journal of Logic and Computation, 28(2) :367—402, 2018.

[4] Bertoli, Piergiorgio, Alessandro Cimatti, Ugo Dal Lago et Marco Pistore: *Extending PDDL to nondeterminism, limited sensing and iterative conditional plans*. Dans *Proceedings of ICAPS 2003 Workshop on PDDL*, 2003.

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

[6] Fikes, Richard E et Nils J Nilsson: *STRIPS : A new approach to the application of theorem proving to problem solving*. Artificial intelligence, 2(3-4) :189–208, 1971.

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

Language	Is-Succ	Is-Applic	Entails	Choice	Sequence	Negation	Ex-Prec
NSTRIPS	NP-c	NP-c	coNP-c	?	○	?	○
T-NPDDL, **C-NPDDL**	NP-c	NP-c	coNP-c	✓	○	○	○
T-NPDDL$_{\text{seq}}$	NP-c	NP-c	coNP-c	✓	✓	○	○
C-NPDDL$_{\text{seq}}$	PSPACE-c	PSPACE-c	PSPACE-c	✓	✓	?	○
T-NPDDL$_{\text{not}}$, **C-NPDDL**$_{\text{not}}$	PSPACE-c	PSPACE-c	PSPACE-c	✓	○	✓	○

Table 1 – Complexity results for queries and transformations. "✓" means that the transformation can be done in time polynomial in the size of the input. "?" means that the question is open. ○ means that under some complexity-theoretic assumption (see formal statements) the size of the result of the transformation is in general not polynomial in the size of the input.

[8] Geffner, Tomas et Hector Geffner: *Compact Policies for Fully Observable Non-Deterministic Planning as SAT*. Dans *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)*, pages 88–96, 2018.

[9] Hastad, John: *Almost optimal lower bounds for small depth circuits*. Dans *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20, 1986.

[10] Herzig, Andreas, Frédéric Maris et Julien Vianey: *Dynamic logic of parallel propositional assignments and its applications to planning*. Dans *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pages 5576–5582, 2019.

[11] Hoey, Jesse, Robert St Aubin et Craig Boutilier: *SPUDD : stochastic planning using decision diagrams*. Dans *Proceedings of Uncertainty in Artificial Intelligence (UAI). Stockholm, Sweden. Page (s)*, 1999.

[12] Lipovetzky, Nir et Hector Geffner: *Width and Serialization of Classical Planning Problems*. Dans *ECAI*, tome 242 de *Frontiers in Artificial Intelligence and Applications*, pages 540–545. IOS Press, 2012.

[13] McDermott, Drew: *PDDL–the planning domain definition language*. Rapport technique 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).

[14] Muise, Christian J., Sheila A. McIlraith et Vaishak Belle: *Non-Deterministic Planning With Conditional Effects*. Dans *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*, pages 370—374. AAAI Press, 2014.

[15] Muller, David E. et Franco P. Preparata: *Bounds to complexities of networks for sorting and for switching*. Journal of the ACM (JACM), 22(2) :195–201, 1975.

[16] Nebel, Bernhard: *On the compilability and expressive power of propositional planning formalisms*. Journal of Artificial Intelligence Research, 12 :271–315, 2000.

[17] Palacios, Héctor et Hector Geffner: *Compiling Uncertainty Away in Conformant Planning Problems with Bounded Width*. Journal of Artificiall Intelligence Research, 35 :623–675, 2009.

[18] Rintanen, Jussi: *Expressive Equivalence of Formalisms for Planning with Sensing*. Dans *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003)*, pages 185–194. AAAI Press, 2003.

[19] Rintanen, Jussi: *Complexity of Planning with Partial Observability*. Dans *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 345–354. AAAI Press, 2004.

[20] Scheck, Sergej, Alexandre Niveau et Bruno Zanuttini: *Knowledge Compilation for Nondeterministic Action Languages*. Dans *Proceedings of the International Conference on Automated Planning and Scheduling*, tome 31, pages 308–316, 2021.

[21] Speck, David, David Borukhson, Robert Mattmüller et Bernhard Nebel: *On the Compilability and Expressive Power of State-Dependent Action Costs*. Dans *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)*, pages 358–366. AAAI Press, 2021.

[22] Speck, David, Florian Geißer et Robert Mattmüller: *Symbolic Planning with Edge-Valued Multi-Valued Decision Diagrams*. Dans *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)*, pages 250–258. AAAI Press, 2018.

[23] Thiébaux, Sylvie, Joerg Hoffmann et Bernhard Nebel: *In defense of PDDL axioms*. Artificial Intelligence, 168 :38–69, 2005.

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