KNOWLEDGE COMPILATION FOR NONDETERMINISTIC ACTION LANGUAGES

Sergej Scheck, Alexandre Niveau, Bruno Zanuttini
Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, Caen, FRANCE
sergej.scheck@unicaen.fr, alexandre.niveau@unicaen.fr, bruno.zanuttini@unicaen.fr

GREYC
Electronics and Computer Science Laboratory

1. Motivation

Even very basic queries in automated planning - such as deciding applicability of an action in a state, for example - can be intractable. Action descriptions can also become very long with an increasing number of state variables. The complexities of queries and the size of action descriptions depend on the used representation language. Therefore it is necessary to study action languages from the perspective of knowledge compilation (KC). A lot of research in knowledge compilation was done about sets of boolean formulas in negated normal form (short $\mathbf{NNF}$). Boolean formulas can represent nondeterministic actions in a natural way, but the standard language for specifying planning tasks is the planning domain definition language ($\mathbf{PDDL}$). Therefore we study the complexity of $\mathbf{PDDL}$ on an abstract level and compare it to $\mathbf{NNF}$ action theories.

2. The Research

Setting

We consider grounded planning tasks over a set of propositional state variables $P = \{p_1, \ldots, p_n\}$. A state s is a boolean assignment to P (we identify s with the set $\{p \in P \mid s(p) = \top\}$). A nondeterministic action is a function $a : 2^P \to 2^{2^P}$ which maps a state s to the set of its possible successors $a(s)$. In words, if we execute a in a state s we arrive in a state $t \in a(s)$ which is "chosen" nondeterministically.

Action languages

We compare these four nondeterministic representation languages :

- $\mathbf{NNF}$ action theories ($\mathbf{NNFAT}$) :

$$\alpha ::= p \mid \neg p \mid p' \mid \neg p' \mid \alpha \wedge \alpha \mid \alpha \vee \alpha$$

  $P' = \{p' \mid p \in P\}$ is the set of state variables encoding the state after the execution of α. A state t naturally defines an assignment to P' via $t(p') := t(p)$. Then t is an α-successor of s iff $\alpha(s(p_1), \ldots, s(p_n), t(p'_1), \ldots, t(p'_n)) \models \top$.
- Nondeterministic $\mathbf{PDDL}$ ($\mathbf{NPDDL}$) :

$$\alpha ::= \epsilon \mid +p \mid -p \mid \alpha \sqcap \alpha \mid \varphi \rhd \alpha \mid \alpha \cup \alpha$$

  ϵ is the action with no effect. $+p$ and $-p$ sets the variable p to $\top$ or $\bot$ respectively. $\alpha \sqcap \beta$ simultaneously executes α and β combining only effects which are consistent together. $\varphi \rhd \alpha$ executes α only from those states where φ is true, acting like ϵ otherwise. $\alpha \cup \beta$ executes nondeterministically either α or β.
- In nondeterministic conditional STRIPS ($\mathbf{NSTRIPS}$) actions have the following form (with $l_i^{k,j}$ being $+p$ or $-p$) :

$$\textstyle\bigsqcap_{i \in I} \varphi_i \rhd \left((\ell_i^{1,1} \sqcap \ldots \sqcap \ell_i^{1,j_1}) \cup \ldots \cup (\ell_i^{k_i,1} \sqcap \ldots \sqcap \ell_i^{k_i,j_{k_i}})\right)$$

  Is is therefore a set of $\mathbf{NPDDL}$ expressions of bounded depth.
- $\mathbf{NPDDL}_{\mathrm{seq}}$: $\mathbf{NPDDL}$ extended by the sequence operator $;$.

$$\alpha ::= \epsilon \mid +p \mid -p \mid \alpha \sqcap \alpha \mid \varphi \rhd \alpha \mid \alpha \cup \alpha \mid \alpha \,;\, \alpha$$

  with $t \in (\alpha \,;\, \beta)(s) \;:\Leftrightarrow\; \exists s' \in \alpha(s) : \; t \in \beta(s')$

Representations

An expression (e.g. a boolean formula) can be stored as a rooted directed acyclic graph. If the representing graph is a tree, i.e. each node can have only one parent node, we speak of the tree representation. If a node is allowed to have more than one parent, we call it the circuit representation (standard in the KC literature). For each language we compare both representations, and we denote them by T and C (e.g. $\mathbf{NNFAT}$ (T)).

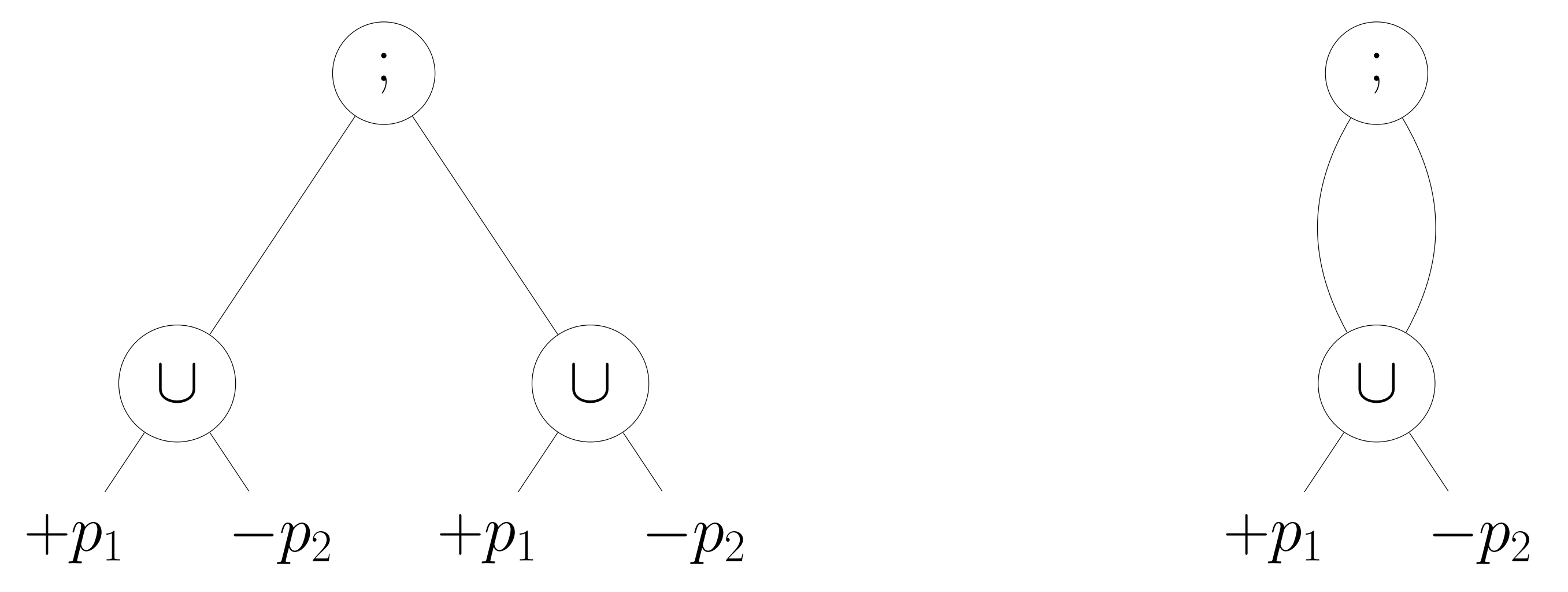

FIGURE 1: *A tree (on the left) and circuit (on the right) representation of the same $\mathbf{NPDDL}_{\mathrm{seq}}$-action $(+p_1 \cup -p_2) \,;\, (+p_1 \cup -p_2)$.*

Complexity of queries

We compare the action languages according to their succinctness and the complexity of the following queries :

- IS-SUCC : given action α and states s, t decide whether $t \in \alpha(s)$
- IS-APPLIC : given action α and state s decide whether $\alpha(s) \neq \emptyset$
- ENTAILS : given state s, formula φ and sequence of actions $\alpha_1, \ldots, \alpha_k$ decide whether $t \models \varphi$ for all $t \in (\alpha_1 \,;\, \ldots \,;\, \alpha_k)(s)$

Succinctness

A language L_1 is strictly more succinct than L_2 (written $L_1 \prec L_2$) iff L_2 expressions can be translated into L_1 expressions with a polynomial bound on the size of the translation, but not vice versa. In other words, there must exist a sequence of L_1 expressions where we would inevitably experience a super-polynomial increase in size when translating them into L_2.

Results

Succinctness proofs often require yet unproven assumptions on the relations between complexity classes. We use a widely accepted assumption that $\mathbf{NP} \not\subseteq \mathbf{P/poly}$. Therefore it can be proven that $\mathbf{NPDDL}_{\mathrm{seq}} \prec \mathbf{NPDDL} \prec \mathbf{NNFAT}$ for both tree and circuit representations.

TABLE 1: *Complexity of* IS-SUCC. *Succinctness strictly increases from top to bottom (assuming $\mathbf{NP} \not\subseteq \mathbf{P/poly}$).*

Language	IS-SUCC (T)	IS-SUCC (C)
$\mathbf{NNFAT}$	linear time	linear time
$\mathbf{NPDDL}$	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete
$\mathbf{NPDDL}_{\mathrm{seq}}$	$\mathbf{NP}$-complete	$\mathbf{PSPACE}$-complete
$\mathbf{NSTRIPS}$	$\mathbf{NP}$-complete	

TABLE 2: *Complexity of* IS-APPLIC *and* ENTAILS. *The representation affects the complexity only for $\mathbf{NPDDL}_{\mathrm{seq}}$.*

Query	$\mathbf{NNFAT}$	$\mathbf{NPDDL}$	$\mathbf{NPDDL}_{\mathrm{seq}}$ (T)	$\mathbf{NPDDL}_{\mathrm{seq}}$ (C)
IS-APPLIC	$\mathbf{NP}$-cpl	$\mathbf{NP}$-cpl	$\mathbf{NP}$-cpl	$\mathbf{PSPACE}$-cpl
ENTAILS	$\mathbf{coNP}$-cpl	$\mathbf{coNP}$-cpl	$\mathbf{coNP}$-cpl	$\mathbf{PSPACE}$-cpl

3. Conclusions

There are nondeterministic actions whose description in $\mathbf{NPDDL}$ cannot be translated into boolean $\mathbf{NNF}$ formulas without an explosion, if $\mathbf{NP} \not\subseteq \mathbf{P/poly}$. Thus it is sometimes useful to specify actions in the more succinct language and work directly with this representation.
It is remarkable that deciding successorship is $\mathbf{NP}$-hard already for $\mathbf{NPDDL}$.
It is also interesting that the representation crucially affects the complexity of queries for $\mathbf{NPDDL}_{\mathrm{seq}}$.