Domain descriptions should be modular

Andreas Herzig and **Ivan Varzinczak**[1]

Abstract. This work is about the metatheory of actions, and here we address the problem of what a good domain description for reasoning about actions should look like. We state some postulates concerning this sore spot, which establishes the notion of a modular domain description. We point out the problems that arise when modularity is violated and propose algorithms to overcome them.

1 INTRODUCTION

In logic-based approaches to reasoning about actions a domain is described by a set of logical formulas Σ. At first glance satisfiability is the only criterion logic provides to check the quality of such a description. In this paper we go beyond that, and argue that we should require more than the mere existence of a model for Σ. Our starting point is that in reasoning about actions one usually distinguishes several kinds of logical formulas. Among these are effect axioms, precondition axioms, and domain constraints.

We prefer here to speak of effect, executability, and static laws, respectively. Moreover we separate inexecutability laws from effect laws. Given these ingredients, suppose the language is powerful enough to state that action α is inexecutable in contexts where A holds, and executable in contexts where B holds. It follows that there can be no context where $A \wedge B$ holds. Now $\neg(A \wedge B)$ is a static law that is independent of α. It is therefore natural to expect that it follows from these laws alone! By means of examples we show that if this is not the case then unexpected conclusions might follow from Σ. A similar case can be made against implicit inexecutability laws.

This motivates postulates requiring that the different ingredients of domain descriptions should be arranged in a modular way, such that interactions between them are limited and controlled. It turns out that in all existing accounts which allow for these four kinds of laws [8, 9, 13, 1, 15], consistent domain descriptions can be written that violate some of these postulates. We here give algorithms that allow one to check whether a domain description satisfies the postulates. With such algorithms, the task of correcting badly written descriptions can be made easier.

This paper is organized as follows: after the preliminary definitions (Sections 2 and 3) we state (Section 4) and study (5–7) three postulates. Finally we discuss strengthenings (Section 8) and assess related work (Section 9).

2 DOMAIN DESCRIPTIONS

In this section we establish the ontology of domain descriptions.

Static laws Frameworks which allow for indirect effects make use of logical formulas that link invariant propositions about the world. Such formulas characterize the set of possible states. They do not refer to actions, and we suppose they are expressed as formulas of classical propositional logic. *PFOR* $= \{A, B, \ldots\}$ is the set of all classical formulas.

A *static law*[2] is a formula $A \in$ *PFOR* that is consistent. An example is *Walking* $\rightarrow$ *Alive*, saying that if a turkey is walking, then it must be alive [13].

Effect laws Let *ACT* $= \{\alpha, \beta, \ldots\}$ be the set of all actions of a given domain. To speak about action effects we use the syntax of propositional dynamic logic (PDL) [5]. The formula $[\alpha]A$ expresses that A is true after every possible execution of α.

An *effect law*[3] *for* α is of the form $A \rightarrow [\alpha]C$, where $A, C \in$ *PFOR*, with A and C both classically consistent. ('Classically consistent' is a shorthand for 'consistent in classical propositional logic'.) The consequent C is the effect which obtains when α is executed in a state where the antecedent A holds. An example is *Loaded* $\rightarrow$ [*shoot*]$\neg$*Alive*, saying that whenever the gun is loaded, after shooting the turkey is dead. Another one is [*tease*]*Walking*: in every situation, the result of teasing is that the turkey starts walking.

Note that the consistency requirements for A and C make sense: if A is inconsistent then the effect law is superfluous; if C is inconsistent then we have an inexecutability law, that we consider to be a separate entity.

Inexecutability laws We suppose that effect laws with inconsistent consequents are a particular kind of law. This allows us to avoid mixing things that are conceptually different: for an action α, an effect law mainly associates it with a consequent C, while an inexecutability law only associates it with an antecedent A.

An *inexecutability law for* α is of the form $A \rightarrow [\alpha]\bot$, where $A \in$ *PFOR* is classically consistent. For example $\neg$*HasGun* $\rightarrow$ [*shoot*]$\bot$ expresses that *shoot* cannot be executed if the agent has no gun.

Executability laws With only static and effect laws one cannot guarantee that *shoot* is executable if the agent has a gun. Whereas all the extant approaches in the literature that allow for indirect effects of actions contain static and effect laws, the status of executability laws is less consensual. Some authors [12, 3, 9, 13] more or less tacitly consider that executability laws should not be made explicit but rather inferred by the reasoning mechanism. Others [8, 15] have executability laws as first class objects one can reason about.

[1] The authors are with the Institut de Recherche en Informatique de Toulouse (IRIT), Toulouse, France. e-mail: {herzig,ivan}@irit.fr

[2] Static laws are often called *domain constraints*, but the different laws for actions that we shall introduce in the sequel could in principle also be called like that.

[3] Effect laws are often called *action laws*, but we prefer not to use that term here because it would also apply to executability laws that are to be introduced in the sequel.

It seems strange to us just stating information about necessary conditions for execution of an action (inexecutabilities) and saying nothing about the sufficient ones. This is the reason why we think we need executability laws. Indeed, in several domains one wants to explicitly state under which conditions a given action is guaranteed to be executable, e.g. that a robot should never get stuck and should always be able to execute a move action. In any case, allowing for executability laws gives us more flexibility and expressive power.

In dynamic logic the dual $\langle\alpha\rangle A$, defined as $\neg[\alpha]\neg A$, can be used to express executability. $\langle\alpha\rangle\top$ thus reads "the execution of action α is possible". An *executability law*[4] *for* α is of the form $A \rightarrow \langle\alpha\rangle\top$, where $A \in \mathit{PFOR}$ is classically consistent. For instance $\mathit{HasGun} \rightarrow \langle \mathit{shoot}\rangle\top$ says that shooting can be executed whenever the agent has a gun, and $\langle \mathit{tease}\rangle\top$ establishes that the turkey can always be teased.

Domain descriptions $\mathcal{S} \subseteq \mathit{PFOR}$ denotes the set of all static laws of a given domain. For a given action $\alpha \in \mathit{ACT}$, $\mathcal{E}_\alpha$ is the set of its effect laws, $\mathcal{X}_\alpha$ is the set of its executability laws, and $\mathcal{I}_\alpha$ is the set of its inexecutability laws. We define $\mathcal{E} = \bigcup_{\alpha\in \mathit{ACT}} \mathcal{E}_\alpha$, $\mathcal{X} = \bigcup_{\alpha\in \mathit{ACT}} \mathcal{X}_\alpha$, and $\mathcal{I} = \bigcup_{\alpha\in \mathit{ACT}} \mathcal{I}_\alpha$. A *domain description* is a tuple of the form $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}\rangle$.

3 DYNAMIC LOGIC AND THE FRAME PROBLEM

Given a domain description $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}\rangle$, we need a consequence relation solving the frame problem. To this end we now give the semantics of PDL, and extend it with dependence relations.

$P_1, P_2, \ldots$ denote propositional constants, $L_1, L_2, \ldots$ literals, and $\Phi, \Psi, \ldots$ formulas. (We recall that $A, B, \ldots$ denote classical formulas.) If $L = \neg P$ then we identify $\neg L$ with P.

A PDL-*model* is a triple $M = \langle W, R, I\rangle$ where W is a set of possible worlds, R maps action constants α to accessibility relations $R_\alpha \subseteq W \times W$, and I maps propositional constants to subsets of W. Given a PDL-model $M = \langle W, R, I\rangle$, $\models_M \Phi$ if for all $w \in W$, $w \models_M \Phi$; $w \models_M [\alpha]\Phi$ if $w' \models_M \Phi$ for every w' such that $wR_\alpha w'$. A formula Φ is a *consequence of the set of global axioms* $\{\Phi_1, \ldots, \Phi_n\}$ in the class of all PDL-models (noted $\Phi_1, \ldots, \Phi_n \models_{\mathsf{PDL}} \Phi$) if and only if for every PDL-model M, if $\models_M \Phi_i$ for every Φ_i, then $\models_M \Phi$.

PDL alone does not solve the frame problem. For instance, if $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}\rangle$ describes our shooting domain then $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\mathsf{PDL}} \mathit{HasGun} \rightarrow [\mathit{load}]\mathit{HasGun}$. The deductive power of PDL has to be augmented in order to ensure that the relevant frame axioms follow. The presence of static constraints makes that this is a delicate task, and starting with [8, 9], several authors have argued that some notion of causality is needed. We here opt for the dependence based approach presented in [1], where dependence information has been added to PDL. In [2] it has been shown how Reiter's solution to the frame problem can be recast in PDL. $\alpha \leadsto L$ denotes that the execution of action α *may change* the truth value of the literal L. In our example we have

$$\leadsto = \left\{ \begin{array}{l} \langle \mathit{shoot}, \neg \mathit{Loaded}\rangle, \langle \mathit{shoot}, \neg \mathit{Alive}\rangle, \\ \langle \mathit{shoot}, \neg \mathit{Walking}\rangle, \langle \mathit{tease}, \mathit{Walking}\rangle \end{array} \right\}$$

Because $\langle \mathit{load}, \neg \mathit{HasGun}\rangle \notin \leadsto$, we have $\mathit{load} \not\leadsto \neg \mathit{HasGun}$, i.e., $\neg \mathit{HasGun}$ is never caused by *load*. We also have $\mathit{tease} \not\leadsto \mathit{Alive}$ and $\mathit{tease} \not\leadsto \neg \mathit{Alive}$.

[4] Some approaches (most prominently Reiter's) use biconditionals $A \leftrightarrow \langle\alpha\rangle\top$, called precondition axioms. This is equivalent to $\neg A \leftrightarrow [\alpha]\bot$, which illustrates that they thus merge information about inexecutability with information about executability.

We here suppose that $\leadsto$ is finite. A given dependence relation $\leadsto$ defines a class of possible worlds models $\mathcal{M}_\leadsto$: every $M \in \mathcal{M}_\leadsto$ must satisfy that whenever $wR_\alpha w'$ then

- $\alpha \not\leadsto P$ and $w \notin I(P)$ implies $w' \notin I(P)$;
- $\alpha \not\leadsto \neg P$ and $w \in I(P)$ implies $w' \in I(P)$.

The associated consequence relation is noted $\models_\leadsto$. In our example we obtain $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_\leadsto \mathit{HasGun} \rightarrow [\mathit{load}]\mathit{HasGun}$.

4 POSTULATES

Our central hypothesis is that the different types of laws should be neatly separated, and should only interfere in one sense: static laws allow one to infer action laws that do not follow from the action laws alone. The other way round, action laws should not allow to infer new static laws, effect laws should not allow to infer inexecutability laws, etc. Here are the postulates for that:

P0. **Logical consistency**: $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_\leadsto \bot$

A domain description should be logically consistent.

P1. **No implicit executability laws**:

$$\text{if } \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_\leadsto A \rightarrow \langle\alpha\rangle\top, \text{ then } \mathcal{S}, \mathcal{X} \models_{\mathsf{PDL}} A \rightarrow \langle\alpha\rangle\top$$

If an executability law can be inferred from the domain description, then it should already "be" in $\mathcal{X}$, in the sense that it should also be inferable in PDL from the set of executability and static laws alone.

P2. **No implicit inexecutability laws**:

$$\text{if } \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_\leadsto A \rightarrow [\alpha]\bot, \text{ then } \mathcal{S}, \mathcal{I} \models_{\mathsf{PDL}} A \rightarrow [\alpha]\bot$$

If an inexecutability law can be inferred from the domain description, then it should be inferable in PDL from the static and inexecutability laws alone.

P3. **No implicit static laws**: if $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_\leadsto A$, then $\mathcal{S} \models_{\mathsf{PDL}} A$.

If a static law can be inferred from the domain description, then it should be inferable in PDL (and even classically) from the set of static laws alone.

Postulate P0 is obvious. P1 can be ensured by maximizing $\mathcal{X}$. This suggests a stronger version of P1:

P4. **Maximal executability laws**:

$$\text{if } \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_\leadsto A \rightarrow [\alpha]\bot, \text{ then } \mathcal{S}, \mathcal{X} \models_{\mathsf{PDL}} A \rightarrow \langle\alpha\rangle\top$$

It expresses that if in context A no inexecutability for α can be inferred, then the respective executability follows in PDL from the executability and static laws. P4 generally holds in nonmonotonic frameworks, and can be enforced in monotonic approaches such as ours by maximizing $\mathcal{X}$.

Things are less obvious for Postulates P2 and P3. They are violated by domain descriptions designed in all approaches in the literature that allow to express the four kinds of laws. We therefore discuss each of them in the subsequent sections by means of examples, and give algorithms to decide whether they are satisfied.

5 NO IMPLICIT INEXECUTABILITY LAWS

Consider the following domain description (and $\leadsto$ as above):

$$\mathcal{S}_1 = \{\textit{Walking} \to \textit{Alive}\},\ \mathcal{E}_1 = \left\{ \begin{array}{c} [\textit{tease}]\textit{Walking}, \\ \textit{Loaded} \to [\textit{shoot}]\neg\textit{Alive} \end{array} \right\},$$

$$\mathcal{X}_1 = \mathcal{I}_1 = \emptyset$$

From $[\textit{tease}]\textit{Walking}$ it follows with $\mathcal{S}_1$ that $[\textit{tease}]\textit{Alive}$, i.e., in every situation, after teasing the turkey is alive: $\mathcal{S}_1, \mathcal{E}_1 \models_{\mathsf{PDL}} [\textit{tease}]\textit{Alive}$. Now as $\textit{tease} \not\leadsto \textit{Alive}$, the status of *Alive* is not modified by the *tease* action, and we have $\mathcal{S}_1, \mathcal{E}_1 \models_{\leadsto} \neg\textit{Alive} \to [\textit{tease}]\neg\textit{Alive}$. From the above, it follows $\mathcal{S}_1, \mathcal{E}_1, \mathcal{X}_1, \mathcal{I}_1 \models_{\leadsto} \neg\textit{Alive} \to [\textit{tease}]\bot$, i.e., the turkey cannot be teased if it is dead. But $\mathcal{S}_1, \mathcal{I}_1 \not\models_{\mathsf{PDL}} \neg\textit{Alive} \to [\textit{tease}]\bot$, hence Postulate P2 is violated. The formula $\neg\textit{Alive} \to [\textit{tease}]\bot$ is an example of what we call an *implicit inexecutability law*.

In the literature, such laws are also known as *implicit qualifications* [4], and it has been argued that it is a positive feature of reasoning about actions frameworks to leave them implicit and provide mechanisms for inferring them [8, 13]. The other way round, one might argue as well that implicit qualifications indicate that the domain has not been described in an adequate manner: inexecutability laws have a form simpler than that of effect laws, and it might be reasonably expected that it is easier to exhaustively describe them.[5] Thus, all the inexecutabilities should be explicitly stated, and this is what Postulate P2 says.

How can we check whether P2 is violated? First we need a definition. Given classical formulas A and B, the function $\mathit{NewCons}_A(B)$ computes the set of strongest clauses that follow from $A \wedge B$, but do not follow from A alone (cf. e.g. [6]). It is known that $\mathit{NewCons}_A(B)$ can be computed by subtracting the prime implicates of A from those of $A \wedge B$. For example, the set of prime implicates of P is just $\{P\}$, that of $P \wedge (\neg P \vee Q) \wedge (\neg P \vee R \vee T)$ is $\{P, Q, R \vee T\}$, hence $\mathit{NewCons}_P((\neg P \vee Q) \wedge (\neg P \vee R \vee T)) = \{Q, R \vee T\}$. And for our example, $\mathit{NewCons}_{\textit{Walking}\to\textit{Alive}}(\textit{Walking}) = \{\textit{Alive}, \textit{Walking}\}$.

Algorithm 1 (Finding implicit inexecutability laws)
```
input: $\mathcal{S}, \mathcal{E}, \mathcal{I}, \leadsto$
output: a set of implicit inexecutability laws $\mathcal{I}^I$
  $\mathcal{I}^I := \emptyset$
  for all $\alpha \in ACT$ do
    for all $J \subseteq \mathcal{E}_\alpha$ do
      $A_J := \bigwedge\{A_i : A_i \to [\alpha]C_i \in J\}$
      $C_J := \bigwedge\{C_i : A_i \to [\alpha]C_i \in J\}$
      if $\mathcal{S} \cup \{A_J\}$ is classically consistent then
        for all $\bigvee L_i \in \mathit{NewCons}_{\mathcal{S}}(C_J)$ do
          if $\forall i,\ \alpha \not\leadsto L_i$ and $\mathcal{S}, \mathcal{I} \not\models_{\mathsf{PDL}} (A_J \wedge \bigwedge \neg L_i) \to [\alpha]\bot$ then
            $\mathcal{I}^I := \mathcal{I}^I \cup \{(A_J \wedge \bigwedge \neg L_i) \to [\alpha]\bot\}$
```

Example 1 Consider $\mathcal{S}_1, \mathcal{E}_1, \mathcal{I}_1$ and $\leadsto$ as given above. Then Algorithm 1 returns $\mathcal{I}^I = \{\neg\textit{Alive} \to [\textit{tease}]\bot\}$.

Theorem 1 $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \rangle$ satisfies Postulate P2 if and only if $\mathcal{I}^I = \emptyset$.

This is the key algorithm of the paper. We are aware that it comes with considerable computational costs: first, the number of formulas A_J and C_J is exponential in the size of $\mathcal{E}_\alpha$, and second, the computation of $\mathit{NewCons}_{\mathcal{S}}(C_J)$ might result in exponential growth. While we might expect $\mathcal{E}_\alpha$ to be reasonably small in practice, the size of $\mathit{NewCons}_{\mathcal{S}}(C_J)$ is more difficult to control. Note that the algorithm terminates because we have assumed $\leadsto$ finite.

The algorithm not only decides whether the postulate is satisfied, its output $\mathcal{I}^I$ also can provide a way to "repair" the domain description. Basically there are three options, that we illustrate with our example: 1) add $\neg\textit{Alive} \to [\textit{tease}]\bot$ to $\mathcal{I}_1$; 2) add the (unintuitive) dependence $\langle \textit{tease}, \textit{Alive} \rangle$ to $\leadsto$; or 3) weaken the law $[\textit{tease}]\textit{Walking}$ to $\textit{Alive} \to [\textit{tease}]\textit{Walking}$. It is easy to see that whatever we opt for, the new domain description will satisfy P2.

6 NO IMPLICIT STATIC LAWS

Executability laws increase expressive power, but might conflict with inexecutability laws. For instance, let $\mathcal{S}_2 = \mathcal{S}_1$, $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{X}_2 = \{\langle \textit{tease} \rangle \top\}$, and $\mathcal{I}_2 = \{\neg\textit{Alive} \to [\textit{tease}]\bot\}$. (Note that Postulate P2 is satisfied.) We have the unintuitive $\mathcal{X}_2, \mathcal{I}_2 \models_{\mathsf{PDL}} \textit{Alive}$: the turkey is immortal! This is an *implicit static law* because *Alive* does not follow from $\mathcal{S}_2$ alone: P3 is violated.

How can we find out whether there are implicit static laws? We assume that Postulate P2 is satisfied, i.e., all inexecutabilities are captured by $\mathcal{I}$.

Algorithm 2 (Finding implicit static laws)
```
input: $\mathcal{S}, \mathcal{X}, \mathcal{I}$
output: a set of implicit static laws $\mathcal{S}^I$
  $\mathcal{S}^I := \emptyset$
  for all $\alpha \in ACT$ do
    for all $A \to [\alpha]\bot \in \mathcal{I}$ and $A' \to \langle\alpha\rangle\top \in \mathcal{X}$ do
      if $\mathcal{S} \not\models_{\mathsf{PDL}} \neg(A \wedge A')$ then
        $\mathcal{S}^I := \mathcal{S}^I \cup \{\neg(A \wedge A')\}$
```

Example 2 For $\langle \mathcal{S}_2, \mathcal{E}_2, \mathcal{X}_2, \mathcal{I}_2 \rangle$, Algorithm 2 returns $\mathcal{S}^I = \{\textit{Alive}\}$.

The existence of implicit static laws may thus indicate too strong executability laws: in our example, we wrongly assumed that *tease* is always executable. It may also indicate that the inexecutability laws are too strong, or that the static laws are too weak:

Example 3 Suppose a computer representation of the line of integers, in which we can be at a strictly positive number, *Positive*, or at a negative one or zero, $\neg$*Positive*. Let *MaxInt* and *MinInt*, respectively, be the largest and the smallest representable integer number. *goleft* is the action of moving to the biggest integer smaller than the one at which we are. Consider the following domain description for this scenario ($\textit{At}_i$ means we are at number i):

$$\mathcal{S}_3 = \{\textit{At}_i \to \textit{Positive} : i > 0\} \cup \{\textit{At}_i \to \neg\textit{Positive} : i \leq 0\}$$

$$\mathcal{E}_3 = \begin{array}{c} \{\textit{At}_{\textit{MinInt}} \to [\textit{goleft}]\textit{Underflow}\} \cup \\ \{\textit{At}_i \to [\textit{goleft}]\textit{At}_{i-1} : i > \textit{MinInt}\} \end{array}$$

$$\mathcal{X}_3 = \{\langle \textit{goleft} \rangle \top\},\ \mathcal{I}_3 = \emptyset$$

with the dependence relation ($\textit{MinInt} \leq i \leq \textit{MaxInt}$):

$$\leadsto = \left\{ \begin{array}{c} \langle \textit{goleft}, \textit{At}_i \rangle, \langle \textit{goleft}, \textit{Positive} \rangle, \\ \langle \textit{goleft}, \neg\textit{Positive} \rangle, \langle \textit{goleft}, \textit{Underflow} \rangle \end{array} \right\}$$

In order to satisfy Postulate P2, we run Algorithm 1 and obtain $\mathcal{I}_3 = \{(\textit{At}_1 \wedge \textit{At}_2) \to [\textit{goleft}]\bot\}$. Now applying Algorithm 2 to this action theory gives us the implicit static law $\neg(\textit{At}_1 \wedge \textit{At}_2)$, i.e., we cannot be at 1 and 2 at the same time.

Theorem 2 Suppose $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \rangle$ satisfies P2. Then Postulate P3 is satisfied if and only if $\mathcal{S}^I = \emptyset$.

[5] Note that nevertheless this is not related to the qualification problem, which basically says that it is difficult to state all the executability laws of a domain.

What shall we do with an implicit static law? Again, several options show up: whereas in the latter example the implicit static law should be added to $\mathcal{S}$, in the former the implicit static law is due to an executability law that is too strong and should be weakened.

So, in order to satisfy Postulate P3, a domain description should contain a complete set of static laws or, alternatively, should not make so strong assumptions about executability. This means that eliminating implicit static laws may require revision of $\mathcal{S}$ or completion of $\mathcal{X}$. In the next section we approach the latter option.

7 MAXIMAL EXECUTABILITY LAWS

Implicit static laws only show up when there are executability laws. Which executability laws can be consistently added to a given domain description?

Algorithm 3 (Finding implicit executability laws)

```
input: $\mathcal{S}, \mathcal{X}, \mathcal{I}$
output: a set of implicit executability laws $\mathcal{X}^I$
  $\mathcal{X}^I := \emptyset$
  for all $\alpha \in ACT$ do
    $A_\alpha := \bigvee\{A_i : A_i \rightarrow [\alpha]\bot \in \mathcal{I}_\alpha\}$
    if $\mathcal{S} \not\models_{\mathsf{PDL}} A_\alpha$ and $\mathcal{S}, \mathcal{X} \not\models_{\mathsf{PDL}} \neg A_\alpha \rightarrow \langle\alpha\rangle\top$ then
      $\mathcal{X}^I := \mathcal{X}^I \cup \{\neg A_\alpha \rightarrow \langle\alpha\rangle\top\}$
```

Example 4 Suppose $\mathcal{S}_4 = \{\mathit{Walking} \rightarrow \mathit{Alive}\}$, $\mathcal{X}_4 = \emptyset$ and $\mathcal{I}_4 = \{\neg\mathit{Alive} \rightarrow [\mathit{tease}]\bot\}$. Then Algorithm 3 yields $\mathcal{X}^I = \{\mathit{Alive} \rightarrow \langle\mathit{tease}\rangle\top\}$.

Theorem 3 Suppose $\langle\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}\rangle$ satisfies P2 and P3. Postulate P4 is satisfied if and only if $\mathcal{X}^I = \emptyset$.

What Theorem 3 says is that it suffices to take the 'complement' of $\mathcal{I}$ to obtain all the executability laws of the domain. Note that this counts as a solution to the qualification problem given that all preconditions for guaranteeing executability of actions are thus known.

8 DISCUSSION

In this section we discuss other properties related to consistency and modularity of domain descriptions. Some will follow from ours, while some others look natural at first glance, but turn out to be too strong.

Theorem 4 If $\langle\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}\rangle$ satisfies P3, then $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_{\leadsto} \bot$ iff $\mathcal{S} \models_{\mathsf{PDL}} \bot$.

This means that if there are no implicit static laws then consistency of a domain description (P0) can be checked by just checking consistency of $\mathcal{S}$.

Theorem 5 If $\langle\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}\rangle$ satisfies P3, then $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_{\leadsto} A \rightarrow [\alpha]C$ iff $\mathcal{S}, \mathcal{E}_\alpha, \mathcal{I}_\alpha \models_{\leadsto} A \rightarrow [\alpha]C$.

This means that under P3 we have modularity inside $\mathcal{E}$, too: when deducing the effects of α we need not consider the action laws for other actions. Versions for executability and inexecutability can be stated as well.

Remark 8.1 Although in the present paper concurrency is not taken into account, we conjecture that Theorem 5 holds when we have concurrent action execution.

Theorem 6 There exist domain descriptions $\langle\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}\rangle$ not satisfying P3 such that $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_{\leadsto} A \rightarrow [\alpha]C$ and $\mathcal{S}, \mathcal{E}_\alpha, \mathcal{I}_\alpha \not\models_{\leadsto} A \rightarrow [\alpha]C$.

For example, we have $\mathcal{S}_2, \mathcal{E}_2, \mathcal{X}_2, \mathcal{I}_2 \models_{\leadsto} \neg\mathit{Alive} \rightarrow [\mathit{shoot}]\mathit{Alive}$, but $\mathcal{S}_2, \mathcal{E}_{2\mathit{shoot}}, \mathcal{I}_{2\mathit{shoot}} \not\models_{\leadsto} \neg\mathit{Alive} \rightarrow [\mathit{shoot}]\mathit{Alive}$.

Now we turn to postulates that are too strong. First, it seems to be in line with the other postulates to require domain descriptions not to allow for the deduction of new effect laws: if an effect law follows from a domain description, and no inexecutability law for the same action in the same context can be derived, then it should follow from the set of static and effect laws alone. This means we should have:

P5. **No implicit effect laws**:

$$\text{if } \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_{\leadsto} A \rightarrow [\alpha]C \text{ and } \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\leadsto} A \rightarrow [\alpha]\bot,$$

$$\text{then } \mathcal{S}, \mathcal{E} \models_{\leadsto} A \rightarrow [\alpha]C$$

But consider the following intuitively correct domain description:

$$\mathcal{S}_5 = \emptyset,\ \mathcal{E}_5 = \left\{ \begin{array}{c} \mathit{Loaded} \rightarrow [\mathit{shoot}]\neg\mathit{Alive}, \\ (\neg\mathit{Loaded} \wedge \mathit{Alive}) \rightarrow [\mathit{shoot}]\mathit{Alive} \end{array} \right\}$$

$$\mathcal{X}_5 = \{\mathit{HasGun} \rightarrow \langle\mathit{shoot}\rangle\top\},\ \mathcal{I}_5 = \{\neg\mathit{HasGun} \rightarrow [\mathit{shoot}]\bot\}$$

together with the dependence relation $\leadsto$ of Example 1. It satisfies Postulates P1, P2, P3, and P4, but does not satisfy P5. Indeed, we have that $\mathcal{S}_5, \mathcal{E}_5, \mathcal{X}_5, \mathcal{I}_5 \models_{\leadsto} \neg\mathit{HasGun} \vee \mathit{Loaded} \rightarrow [\mathit{shoot}]\neg\mathit{Alive}$ and $\mathcal{S}_5, \mathcal{E}_5, \mathcal{X}_5, \mathcal{I}_5 \not\models_{\leadsto} \neg\mathit{HasGun} \vee \mathit{Loaded} \rightarrow [\mathit{shoot}]\bot$, but $\mathcal{S}_5, \mathcal{E}_5 \not\models_{\leadsto} \neg\mathit{HasGun} \vee \mathit{Loaded} \rightarrow [\mathit{shoot}]\neg\mathit{Alive}$. So, Postulate P5 would not help us to deliver the goods.

Another though obvious possibility of amending our modularity criteria could be by stating the following postulate:

P6. **No unattainable effects**:

$$\text{if } A \rightarrow [\alpha]C \in \mathcal{E},\ \text{then } \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\leadsto} A \rightarrow [\alpha]\bot$$

This expresses that if we have explicitly stated an effect law for α in some context, then there should be no inexecutability law for the same action in the same context. We do not investigate this further here, but just observe that the slightly stronger version below leads to unintuitive consequences:

P6′. **No unattainable effects (strong version)**:

$$\text{if } \mathcal{S}, \mathcal{E} \models_{\leadsto} A \rightarrow [\alpha]C,\ \text{then } \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\leadsto} A \rightarrow [\alpha]\bot$$

Indeed, for the above domain description we have that $\mathcal{E}_5 \models_{\leadsto} (\neg\mathit{HasGun} \wedge \mathit{Loaded}) \rightarrow [\mathit{shoot}]\neg\mathit{Alive}$, but $\mathcal{S}_5, \mathcal{E}_5, \mathcal{X}_5, \mathcal{I}_5 \models_{\leadsto} (\neg\mathit{HasGun} \wedge \mathit{Loaded}) \rightarrow [\mathit{shoot}]\bot$. This is certainly too strong. Our example also illustrates that it is sometimes natural to have some 'redundancies' or 'overlaps' between $\mathcal{I}$ and $\mathcal{E}$.

9 RELATED WORK

Pirri and Reiter have investigated the metatheory of the situation calculus [11]. In a spirit similar to ours, they simplify the entailment problem for this calculus, and show for several problems such as consistency or regression that only some of the modules of a domain description are necessary. Note that in their domain descriptions $\mathcal{S} = \emptyset$. This allows them to show that such theories are always consistent.

Zhang *et al.* [14] have also proposed an assessment of what a good domain description should look like. They develop the ideas in the framework of EPDL [15], an extended version of PDL which allows for propositions as modalities to represent causal connection between literals. We do not present the details of that, but concentrate on the main metatheoretical results.

Zhang *et al.* propose a normal form for describing action theories, and investigate three levels of consistency. Roughly speaking, a domain description Σ is *uniformly consistent* if it is globally consistent (i.e., $\Sigma \not\models_{\mathsf{EPDL}} \bot$); a formula Φ is Σ*-consistent* if $\Sigma \not\models_{\mathsf{EPDL}} \neg\Phi$, for Σ a uniformly consistent theory; Σ is *universally consistent* if $\Sigma \models_{\mathsf{EPDL}} A$ implies $\models_{\mathsf{EPDL}} A$.

Given these definitions, they propose algorithms to test the different versions of consistency for a domain description Σ that is in normal form. This test essentially amounts to checking whether Σ is *safe*, i.e., whether $\Sigma \models_{\mathsf{EPDL}} \langle\alpha\rangle\top$, for every α. Success of this check should mean the domain description under analysis satisfies the consistency requirements.

Nevertheless, this is only a necessary condition: it is not hard to imagine domain descriptions that are uniformly consistent but in which we can still have implicit inexecutabilities that are not caught by the algorithm. Consider for instance a scenario with a lamp that can be turned on and off by a toggle action, and its EPDL representation given by $\{On \rightarrow [toggle]\neg On,\ Off \rightarrow [toggle]On,\ [On]\neg Off,\ [\neg On]Off\}$.

The causal statement $[On]\neg Off$ means *On* causes $\neg$*Off*. Such a domain description satisfies each of the consistency requirements (in particular it is uniformly consistent, as $\Sigma \not\models_{\mathsf{EPDL}} \bot$). However, Σ is not safe because the implicit static law $\neg(On \wedge Off)$ cannot be proved.

Lang *et al.* [7] address consistency in the causal laws approach [9], focusing on the computational aspects. They suppose an abstract notion of completion of a domain description solving the frame problem. Given a domain description Σ_α containing logical information about α's direct effects as well as the indirect effects that may follow, the completion of Σ_α roughly speaking is the original theory Σ_α amended of logical axioms stating the persistence of all non-affected (directly nor indirectly) literals.

Their EXECUTABILITY problem is to check whether α is executable in all possible initial states (Zhang *et al.*'s safety property). This amounts to testing whether every possible state w has a successor w' reachable by α such that w and w' both satisfy the completion of Σ_α. For instance, still considering the lamp scenario, the representation of the domain description for *toggle* is $\{On \xrightarrow{toggle} Off,\ Off \xrightarrow{toggle} On,\ Off \longrightarrow \neg On,\ On \longrightarrow \neg Off\}$, where the first two formulas are conditional effect laws for *toggle*, and the latter two causal laws in McCain and Turner's sense. We will not dive in the technical details, and just note that the executability check will return "no" for this example as *toggle* cannot be executed in a state satisfying $On \wedge Off$.

10 CONCLUSION

We have tried to point out some of the problems that can arise when domain descriptions are not modular. In particular we have argued that the non-dynamic part of domain descriptions should not be influenced by the dynamic one.[6]

We have put forward several postulates, and have in particular tried to demonstrate that when there are implicit inexecutability and static laws then one has slipped up in designing the domain description under consideration. As shown, a possible solution comes into its own with Algorithms 1, 2 and 3, which can give us some guidelines in correcting a domain description if needed.

Given the difficulty of exhaustively enumerating all the preconditions under which a given action is executable and also those under which such an action cannot be executed, there is always going to be some executability precondition A or some inexecutability precondition B that together lead to a contradiction, forcing, thus, an implicit static law $\neg(A \wedge B)$. This is the reason we propose to state some information about both executabilities and inexecutabilities, complete the latter and then, after deriving all implicit static laws, complete the former. As a final result we will have complete $\mathcal{S}$, $\mathcal{X}$ and $\mathcal{I}$.

Throughout this work we used a weak version of PDL, but our notions and results can be applied to other frameworks as well. It is worth noting however that for first-order based frameworks the consistency check of Algorithm 1 is undecidable. (We can get rid of this by assuming that $\langle\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}\rangle$ is finite and there is no function symbol in the language. In this way, the result of *NewCons* is finite and the algorithm terminates.)

Our postulates do not take into account causality statements linking propositions. This could be a topic for further investigation.

[6] It might be objected that it is only by doing experiments that one learns the static laws that govern the universe. But note that this involves *learning*, whereas here — as always done in the reasoning about actions field — the static laws are known once forever, and do not evolve.

ACKNOWLEDGMENTS

Ivan Varzinczak has been supported by a fellowship from the government of the Federative Republic of Brazil, grant CAPES.

REFERENCES

[1] M. A. Castilho, O. Gasquet, and A. Herzig, 'Formalizing action and change in modal logic I: the frame problem', *J. of Logic and Computation*, **9**(5), 701–735, (1999).
[2] R. Demolombe, A. Herzig, and I. Varzinczak, 'Regression in modal logic', *J. of Applied Non-classical Logics (JANCL)*, **13**(2), 165–185, (2003).
[3] P. Doherty, W. Łukaszewicz, and A. Szałas, 'Explaining explanation closure', in *Proc. Int. Symposium on Methodologies for Intelligent Systems*, Zakopane, Poland, (1996).
[4] M. L. Ginsberg and D. E. Smith, 'Reasoning about actions II: The qualification problem', *Artificial Intelligence*, **35**(3), 311–342, (1988).
[5] D. Harel, 'Dynamic logic', in *Handbook of Philosophical Logic*, eds., D. M. Gabbay and F. Günthner, volume II, 497–604, D. Reidel, Dordrecht, (1984).
[6] K. Inoue, 'Linear resolution for consequence finding', *Artificial Intelligence*, **56**(2–3), 301–353, (1992).
[7] J. Lang, F. Lin, and P Marquis, 'Causal theories of action – a computational core', in *Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*, eds., V. Sorge, S. Colton, M. Fisher, and J. Gow, pp. 1073–1078, Acapulco, (2003). Morgan Kaufmann Publishers.
[8] F. Lin, 'Embracing causality in specifying the indirect effects of actions', In Mellish [10], pp. 1985–1991.
[9] N. McCain and H. Turner, 'A causal theory of ramifications and qualifications', In Mellish [10], pp. 1978–1984.
[10] C. Mellish, ed. *Proc. 14th Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*, Montreal, 1995. Morgan Kaufmann Publishers.
[11] F. Pirri and R. Reiter, 'Some contributions to the metatheory of the situation calculus', *Journal of the ACM*, **46**(3), 325–361, (1999).
[12] L. K. Schubert, 'Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions', in *Knowledge Representation and Defeasible Reasoning*, eds., H. E. Kyberg, R. P. Loui, and G. N. Carlson, 23–67, Kluwer Academic Publishers, (1990).
[13] M. Thielscher, 'Computing ramifications by postprocessing', In Mellish [10], pp. 1994–2000.
[14] D. Zhang, S. Chopra, and N. Y. Foo, 'Consistency of action descriptions', in *PRICAI'02, Topics in Artificial Intelligence*. Springer-Verlag, (2002).
[15] D. Zhang and N. Y. Foo, 'EPDL: A logic for causal reasoning', in *Proc. 17th Int. Joint Conf. on Artificial Intelligence (IJCAI'01)*, ed., B. Nebel, pp. 131–138, Seattle, (2001). Morgan Kaufmann Publishers.