Domain descriptions should be modular (preliminary report)

Andreas Herzig   Ivan Varzinczak*
Institut de Recherche en Informatique de Toulouse
F-31062 Toulouse Cedex 04, France
mailto:{herzig,ivan}@irit.fr

Abstract *In this work we address the problem of what a good domain description for reasoning about actions should look like. We establish some postulates concerning this sore spot and point out the problems that arise when they are violated. Such problems can be overcome with the algorithms we propose.*
Keywords: *Reasoning about actions, domain description, modularity.*

1 Introduction

A lot of logical frameworks for reasoning about actions exist. In all of them some domain is described by means of logical formulas, of which several types can be distinguished, such as domain constraints and effect laws.

Usually, consistency of a domain description (alias action theory) is identified with the existence of a model for it. We here have a closer look at that issue, and formulate some postulates expressing that the different entities of domain descriptions should be arranged in a modular way, such that interactions between them are limited and controlled. We first give some intuitive justifications, pointing out that none of the main existing accounts of actions satisfies all these postulates. In order to get around this, we give algorithms that allow to check whether a given domain description satisfies the postulates, and aid in its generation or correction.

Throughout this work we use a weak version of dynamic logic [6], but our notions and results can be applied to other frameworks as well.

This paper is organized as follows: in Section 2 we define the different types of laws commonly used in the field, establishing the ontology the rest of this work is about. After stating our set of postulates (Section 3), we study each of them in Sections 4–6. In Section 7 we discuss possible strengthenings of them. We assess related work in Section 8 and finally give some conclusions.

*Supported by a fellowship from the government of the Federative Republic of Brazil. Grant: CAPES BEX 1389/01-7.

2 Domain descriptions and inference

Every domain description contains a representation of action effects. We call *effect laws* formulas relating an action to its effects. Statements of conditions under which an action cannot be executed are called *inexecutability laws*. *Executability laws* in turn stipulate the context where an action is guaranteed to be executable. Finally, *static laws* are formulas that do not mention actions and express constraints that must hold in every possible state. These are our four ingredients that we introduce more formally in the sequel.

2.1 Static laws

Frameworks which allow for indirect effects make use of logical formulas that link invariant propositions about the world. Such formulas do not refer to actions and are supposed to characterize the set of possible states.

Definition 2.1 A *static law* is a logical formula of classical propositional logic.[1]

An example of a static law is $Walking \rightarrow Alive$, expressing the fact that if some turkey is walking then it must be alive [14].

We here use the syntax of propositional logic, but all we shall say applies as well to first-order frameworks, in particular to the Situation Calculus [12]. Propositional formulas are noted A, B, C and so forth, and the set of all propositional formulas $PFOR$. Hence we do not consider here causality statements linking propositions as used e.g. in the approaches of Lin [9], McCain and Turner [10], or Giordano *et al.* [5].

$\mathcal{S} \subseteq PFOR$ denotes the set of all static laws of a given domain.

2.2 Effect laws

By nature, logical frameworks for reasoning about actions contain expressions linking actions and their effects. We suppose that such effects might be conditional, and thus get a third component.

Let us denote actions by α, β, and so forth, and the set of all possible actions of a given domain by ACT. From now on, we will use the syntax of propositional dynamic logic (PDL) [6]. $[\alpha]A$ is read as "after the execution of action α, the formula A is true".

Definition 2.2 An *effect law* is of the form $A \rightarrow [\alpha]C$, where $\alpha \in ACT$, $A, C \in PFOR$, and A and C are both consistent.

[1] Static laws are often called domain constraints, but we prefer not to use that term here because it would also apply to the different laws for actions that we shall introduce in the sequence.

The antecedent A is a condition and the consequent C is the effect which obtains when α is executed in a state where the condition holds.[2]

An example of effect law is $[tease]Walking$, expressing that in every possible situation, after teasing the turkey it starts walking. Another example is $Loaded \rightarrow [shoot]\neg Alive$: whenever the gun is loaded then the result of the shoot action is that the turkey is dead.

The consistency requirements for both the condition and effect 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.

$\mathcal{E}(\alpha) = \{A \rightarrow [\alpha]C : A, C \in PFOR\}$ is the set of all effect laws concerning a given action α, and $\mathcal{E} = \bigcup_{\alpha \in ACT} \mathcal{E}(\alpha)$ denotes the set of all effect laws.

2.3 Inexecutability laws

We can also state laws expressing that an action cannot be executed at all in the presence of some condition A. This can be done by stating that the effect of executing α is $\bot$ in every situation where A holds.

Definition 2.3 An *inexecutability law* 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* is inexecutable if one does not have a gun.

This kind of law is not distinguished from effect laws in the literature. Nevertheless, opting for a separate entity allows us to avoid mixing things that are conceptually different: an inexecutability law links an antecedent A and an action α, while an effect law mainly links an action and a consequent C.

We denote the set of all inexecutability laws for α by $\mathcal{I}(\alpha)$. $\mathcal{I} = \bigcup_{\alpha \in ACT} \mathcal{I}(\alpha)$ is the set of all inexecutability laws.

2.4 Executability laws

With only static and effect laws one cannot express the conditions under which executability of an action is guaranteed. In dynamic logic we use the dual $\langle\alpha\rangle A$ defined as $\neg[\alpha]\neg A$ as a means to express executability. $\langle\alpha\rangle\top$ thus reads "the execution of action α is possible".

Definition 2.4 An *executability law* is of the form $A \rightarrow \langle\alpha\rangle\top$ where $\alpha \in ACT$ and $A \in PFOR$ is classically consistent.

An executability law describes a condition under which α is executable. For instance $\langle tease\rangle\top$ expresses that the turkey can always be teased. $HasGun \rightarrow \langle shoot\rangle\top$ formalizes that the action of shooting the turkey can be executed whenever one has a gun.

[2]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.

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: in some approaches [13, 3, 10, 11, 15] it is more or less tacitly considered that they should not be made explicit, but rather inferred by the reasoning mechanism. On the other hand, in several approaches [9, 17] executability laws are first class objects that one can reason about. It seems to us a matter of debate whether one can always do without executability laws. We think that there are many domains where one wants to explicitly state under which conditions a given action is guaranteed to be executable. This holds in particular when we want to express constraints such as that a robot should never get stuck, i.e., it should always be able to execute a move action, or that it should always be able to get home. In any case, allowing for executability laws gives us more flexibility and expressive power.

Most of the approaches that allow for executability laws (most prominently Reiter's) do it in form of biconditionals like $\langle\alpha\rangle\top \leftrightarrow A$. As this is equivalent to $\neg A \leftrightarrow [\alpha]\bot$, it means that information about inexecutability is merged with that about executability. Appealing to modularity, we propose here to decompose such laws into two sets.

We will denote the set of executability laws by $\mathcal{X}$.

2.5 Dynamic logic and the frame problem

Henceforth we will consider that domain descriptions are tuples of the form $\langle\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}\rangle$, where $\mathcal{S}, \mathcal{E}, \mathcal{X}$ and $\mathcal{I}$ are as defined above. What follows from such a domain description? To this end we now formally define dynamic logic, and show how the frame problem can be solved by extending PDL with dependence relations.

We use $P_1, P_2, \ldots$ for propositional constants. $L_1, L_2, \ldots$ denote literals, and $\Phi, \Psi, \ldots$ denote formulas. If $L = \neg P$ then we shall identify $\neg L$ with P. We recall that $A, B, \ldots$ denote classical propositional formulas (without occurrences of action symbols).

A model for an action description in such a language is a triple $M = \langle W, R, I\rangle$ where W is a set of possible worlds, R is a function mapping action constants to binary relations on W, and I is an interpretation function mapping propositional constants to subsets of W.

For a given PDL-model $M = \langle W, R, I\rangle$, $w \models_M [\alpha]\Phi$ if for every w' such that $wR(\alpha)w'$, $w' \models_M \Phi$. We say that a formula Φ is a consequence of the set of global axioms Γ in the class of all PDL-models (noted $\Gamma \models_{\mathsf{PDL}} \Phi$) if and only if for every PDL-model M, if $\models_M \Phi_i$ for every $\Phi_i \in \Gamma$, then $\models_M \Phi$.

As expected, 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 $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\mathsf{PDL}} \mathit{HasGun} \rightarrow [\mathit{load}]\mathit{HasGun}$. Therefore its deductive power 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 [9, 10], several approaches in the literature 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. $\alpha \rightsquigarrow L$ denotes that the execution of action α *may change* the truth value of the literal L. In our running example we have

$$\rightsquigarrow = \{\langle shoot, \neg Loaded\rangle, \langle shoot, \neg Alive\rangle, \langle shoot, \neg Walking\rangle, \langle tease, Walking\rangle\}$$

Hence $shoot \not\rightsquigarrow HasGun$, i.e., $HasGun$ is never caused by $shoot$.

A given dependence relation $\rightsquigarrow$ defines a class of possible worlds models $\mathcal{M}_{\rightsquigarrow}$: every $M \in \mathcal{M}_{\rightsquigarrow}$ must satisfy that whenever $wR(\alpha)w'$ then

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

The associated consequence relation is noted $\models_{\rightsquigarrow}$. Then we indeed obtain for our example $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_{\rightsquigarrow} HasGun \rightarrow [load]HasGun$. We have shown in [2] how Reiter's solution to the frame problem translates to this PDL variant in a straightforward way.

3 Postulates

When does a given domain description have a model? We claim that the approaches that are put forward in the literature are too liberal to such an extent that there are satisfiable domain descriptions that are intuitively incorrect.

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

We put forward the following postulates:

P0. **Logical consistency**:

$$\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\rightsquigarrow} \bot$$

A domain description in dynamic logic should be logically consistent. Postulate P0 is obvious.

P1. **No implicit executability laws**:

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

If an executability law can be inferred from a given 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**:

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

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

P3. **No implicit static laws**:

$$\text{if } \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_{\leadsto} A, \text{ then } \mathcal{S} \models_{\mathsf{PDL}} A$$

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

In most of the approaches of the literature Postulates P2 and P3 are not satisfied. Therefore, in the subsequent sections we discuss each of them by means of examples, and give algorithms to decide whether they are satisfied by a given domain description.

Before that, we just give another postulate about executability laws:

P4. **Maximal executability laws**:

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

This expresses that if no inexecutability for a given action under condition A can be inferred, then its executability under the same condition follows from the executability and static laws. Postulate P4 holds in nonmonotonic frameworks, and we will show that it can easily be ensured in monotonic approaches such as ours by maximizing $\mathcal{X}$.

4 No implicit inexecutability laws

Consider the following domain description:

$$\mathcal{S}_1 = \{\mathit{Walking} \rightarrow \mathit{Alive}\}$$

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

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

For the time being executability laws are irrelevant and we focus on inexecutability ones: from $[\mathit{tease}]\mathit{Walking}$ it somewhat surprisingly follows with $\mathcal{S}_1$ that $[\mathit{tease}]\mathit{Alive}$, i.e., in every situation, after teasing the turkey is alive: $\mathcal{S}_1, \mathcal{E}_1 \models_{\mathsf{PDL}} [\mathit{tease}]\mathit{Alive}$.

Now if we have a correct solution to the frame problem, then we should have that $\mathcal{S}_1, \mathcal{E}_1 \models_{\leadsto} \neg\mathit{Alive} \rightarrow [\mathit{tease}]\neg\mathit{Alive}$ i.e., the status of Alive is not modified by the *tease* action. This is ensured here by $\leadsto$ not containing $\langle \mathit{tease}, \mathit{Alive}\rangle$. From the above, it follows $\mathcal{S}_1, \mathcal{E}_1, \mathcal{X}_1\mathcal{I}_1 \models_{\leadsto} \neg\mathit{Alive} \rightarrow [\mathit{tease}]\bot$, i.e., the turkey cannot be teased if it is dead. But $\mathcal{S}_1, \mathcal{I}_1 \not\models_{\mathsf{PDL}} \neg\mathit{Alive} \rightarrow [\mathit{tease}]\bot$, hence Postulate P2 is violated. The formula $\neg\mathit{Alive} \rightarrow [\mathit{tease}]\bot$ 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 let them implicit and provide mechanisms for inferring them [9, 15]. The other way round, one might argue as well that such implicit qualifications indicate that the domain has not been described in an adequate manner: inexecutability laws have a form that is simpler than that of effect laws, and it might be reasonably expected that it is easier than for the latter to exhaustively describe them. (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.) Thus, in order to be abide by the Postulate P2, a given domain description should not allow the derivability of implicit inexecutability laws, and all the inexecutabilities for a given action in the domain description should be explicitly stated.

Postulate P2 can be checked with the aid of the below algorithm. It uses a function $\mathit{NewCons}_A(B)$ which given two formulas A and B computes the set of strongest clauses that follow from $A \wedge B$, but do not follow from A alone (cf. e.g. [7]). 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 $\{P\}$, the set of prime implicates of $P \wedge (\neg P \vee Q) \wedge (\neg P \vee R \vee T)$ is $\{P, Q, R \vee T\}$, hence $\mathit{NewCons}_P((\neg P \vee Q) \wedge (\neg P \vee R \vee T)) = \{Q, R \vee T\}$.

Algorithm 4.1 (Finding implicit inexecutability laws)

`input:` $\mathcal{S}, \mathcal{E}, \mathcal{I}$ and $\leadsto$.

`output:` a set of implicit inexecutability laws $\mathcal{I}^I$.

`begin`

  $\mathcal{I}^I$ `:=` $\emptyset$

  `forall` $\alpha \in ACT$

    `let` $\mathcal{E}(\alpha)$`:=` $\{A_i \rightarrow [\alpha]C_i : 1 \leq i \leq n\}$

    `forall` $J \subseteq \{1, \ldots, n\}$

      `let` A_J`:=` $\bigwedge_{j \in J} A_j$ and C_J`:=` $\bigwedge_{j \in J} C_j$

      `if` $\mathcal{S} \cup \{A_J\}$ is classically consistent `then`

        `let` Δ`:=` $\mathit{NewCons}_{\mathcal{S}}(C_J)$

        `forall` $L_1 \vee \ldots \vee L_m \in \Delta$

          `if` $\alpha \not\leadsto L_i$ for every i `and`

          $\mathcal{S}, \mathcal{I} \not\models_{\mathsf{PDL}} (A_J \wedge \neg L_1 \wedge \ldots \wedge \neg L_m) \rightarrow [\alpha]\bot$ `then`

$$\mathcal{I}^I := \mathcal{I}^I \cup \{\ (A_J \wedge \neg L_1 \wedge \ldots \wedge \neg L_m) \to [\alpha]\bot\}$$

`end` ■

Example 4.1 Consider $\mathcal{S}_1, \mathcal{E}_1, \mathcal{I}_1$ and $\mathcal{D}$ as given above. Then Algorithm 4.1 returns $\mathcal{I}^I = \{\neg Alive \to [tease]\bot\}$. ■

Theorem 4.1 $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \rangle$ satisfies Postulate P2 if and only if $\mathcal{I}^I = \emptyset$.

Given a way of finding what is going wrong with an action theory, we are also concerned about how to "repair" it. Still considering Example 4.1, given the output $\mathcal{I}^I = \{\neg Alive \to [tease]\bot\}$ produced by Algorithm 4.1, we have three possibilities: 1) let $\mathcal{I}_1 = \mathcal{I}_1 \cup \mathcal{I}^I$, i.e., explicitly state all implicit inexecutabilities; 2) add the dependence $\langle tease, Alive \rangle$ to $\rightsquigarrow$ (which, in this case, is not intuitive); or 3) weaken the law $[tease]\,Walking$ by $Alive \to [tease]\,Walking$. It is easy to see that whatever of these alternatives we opt for, the resulting domain description will satisfy Postulate P2.

Now we turn to another type of implicit laws.

5 No implicit static laws

Whereas executability laws increase expressive power, however, complexity of the framework increases, too: executability laws might conflict with inexecutability laws or even with static and effect laws, while this is not the case in frameworks based on static and effect axioms only (where executability laws are obtained by maximization), there are no implicit static laws. For instance, let $\mathcal{S}_2 = \mathcal{S}_1$, $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{X}_2 = \{\langle tease \rangle \top\}$, and $\mathcal{I}_2 = \{\neg Alive \to [tease]\bot\}$. First note that $\langle \mathcal{S}_2, \mathcal{E}_2, \mathcal{X}_2, \mathcal{I}_2 \rangle$ satisfies Postulate P2. In PDL (and a fortiori in our dependence based variant, and more generally in all logics of action where executability laws can be expressed) this entails the static law *Alive*, i.e., the turkey is immortal. Our example illustrates that sometimes from a given domain description new static laws can be inferred that were not provable from the static laws alone. We call these *implicit static laws*.

The existence of implicit static laws in a domain description may indicate an incorrect specification of executability laws: in the previous example, we derived an implicit static law because we wrongly assumed that action *tease* is always executable. It may also indicate an incomplete formalization of the static laws (see below for an example).

How can we find out whether there are implicit static laws? We here assume that Postulate P2 is satisfied. Then the algorithm below does the job.

Algorithm 5.1 (Finding implicit static laws)

`input:` $\mathcal{S}, \mathcal{X}, \mathcal{I}$.

`output:` a set of implicit static laws $\mathcal{S}^I$.

```
begin
    S^I := ∅
    forall α ∈ ACT
        forall A → [α]⊥ ∈ I and A′ → ⟨α⟩⊤ ∈ X
            if S ⊭_PDL ¬(A ∧ A′) then
                S^I := S^I ∪ {¬(A ∧ A′)}
end
```
■

Example 5.1 Algorithm 5.1 applied to the domain description $\langle \mathcal{S}_2, \mathcal{E}_2, \mathcal{X}_2, \mathcal{I}_2 \rangle$ will output, as expected, the unintended but implicit static law *Alive*, i.e., the agent never dies. ■

Example 5.2 Suppose a domain with a lamp that can be turned on and off by toggling a switch. Suppose also that the following domain description of such a scenario has been given:

$$\mathcal{S}_3 = \emptyset$$

$$\mathcal{E}_3 = \left\{ \begin{array}{c} \mathit{On} \rightarrow [\mathit{toggle}]\neg \mathit{On}, \\ \mathit{Off} \rightarrow [\mathit{toggle}]\mathit{On} \end{array} \right\}$$

$$\mathcal{X}_3 = \{\langle \mathit{toggle} \rangle \top\}$$

$$\mathcal{I}_3 = \{(\mathit{On} \wedge \mathit{Off}) \rightarrow [\mathit{toggle}]\bot\}$$

together with the dependence relation $\leadsto = \{\langle \mathit{toggle}, \mathit{On} \rangle, \langle \mathit{toggle}, \mathit{Off} \rangle\}$. This satisfies Postulate P2. Applying Algorithm 5.1 to this action theory gives us the implicit static law $\neg(\mathit{On} \wedge \mathit{Off})$, i.e., the light cannot be on and off at the same time. ■

Theorem 5.1 Suppose $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \rangle$ satisfies P2. Then Postulate P3 is satisfied if and only if $\mathcal{S}^I = \emptyset$.

Once having derived all implicit static laws, what can we do with them? Examples 5.1 and 5.2 illustrate that we may obtain both "good" and "bad" implicit static laws. Whereas in the latter 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 is thus unintuitive.

6 Maximal executability laws

As we have seen, implicit static laws might be due to executability laws that are too strong. It is perhaps for that reason that in many approaches there are no such laws, and they are inferred a posteriori by some nonmonotonic mechanism. In this section we show how such a maximization can be done in a straightforward way to find all the executability laws that can be consistently added to a given domain description.

Algorithm 6.1 (Finding implicit executability laws)

`input:` $\mathcal{S}, \mathcal{X}, \mathcal{I}$.

`output:` a set of implicit executability laws $\mathcal{X}^I$.

`begin`

$\mathcal{X}^I$ `:=` $\emptyset$

`forall` $\alpha \in ACT$

$A(\alpha)$ `:=` $\bigvee_{\{j : A_j \rightarrow [\alpha]\bot \in \mathcal{I}(\alpha)\}} A_j$

`if` $\mathcal{S}, \mathcal{X} \not\models_{\mathsf{PDL}} \neg A(\alpha) \rightarrow \langle\alpha\rangle\top$ `then`

$\mathcal{X}^I$ `:=` $\mathcal{X}^I \cup \{\neg A(\alpha) \rightarrow \langle\alpha\rangle\top\}$.

`end` ■

Example 6.1 Suppose that $\mathcal{S}_4 = \{\mathit{Walking} \rightarrow \mathit{Alive}\}$, $\mathcal{X}_4 = \emptyset$ and $\mathcal{I}_4 = \{\neg\mathit{Alive} \rightarrow [\mathit{tease}]\bot\}$. Then Algorithm 6.1 yields $\mathcal{X}^I = \{\mathit{Alive} \rightarrow \langle\mathit{tease}\rangle\top\}$. ■

Theorem 6.1 Suppose $\langle\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}\rangle$ satisfies P2 and P3. $\mathcal{X}^I = \emptyset$ if and only if Postulate P4 is satisfied.

What Theorem 6.1 says is that it suffices to take the "complement" of $\mathcal{I}$, $\overline{\mathcal{I}} = \{\neg(A_1 \wedge \ldots \wedge A_{|\mathcal{I}(\alpha)|}) \rightarrow \langle\alpha\rangle\top : A_i \rightarrow [\alpha]\bot \in \mathcal{I}(\alpha), 1 \leq i \leq |\mathcal{I}(\alpha)|\}$, to obtain all the executability laws of the domain. Note that this counts as a solution to the qualification problem, in the sense that all preconditions for guaranteeing executability of actions are known.

For our running example, letting $\mathcal{X}_4$ `:=` $\mathcal{X}_4 \cup \mathcal{X}^I$ establishes maximal executability for such an action theory.

7 Discussion: can we ask for more?

One question that could be raised is: can our postulates be improved, i.e., can we be more restrictive?

Let's suppose we want our domain descriptions not to allow for the deduction of new effect laws. In this case, we should state the following additional postulate:

P5. **No implicit effect laws**:

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

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

In other words, if an effect law can be inferred from a domain description and no inexecutability law for the same action in the same context can be derived, then it should be inferable from the set of static and effect laws alone.

At first glance, this sounds reasonable. However, consider the following intuitively correct domain description, which satisfies Postulates P1, P2, P3, and P4, but does not satisfy P5:

$$\mathcal{S}_5 = \emptyset$$

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

$$\mathcal{X}_5 = \{\mathit{HasGun} \to \langle \mathit{shoot} \rangle \top\}$$

$$\mathcal{I}_5 = \{\neg \mathit{HasGun} \to [\mathit{shoot}]\bot\}$$

together with the dependence relation $\leadsto$ of Example 4.1. Indeed, we have

$$\mathcal{S}_5, \mathcal{E}_5, \mathcal{X}_5, \mathcal{I}_5 \models_{\leadsto} \neg \mathit{HasGun} \vee \mathit{Loaded} \to [\mathit{shoot}]\neg \mathit{Alive}$$

and

$$\mathcal{S}_5, \mathcal{E}_5, \mathcal{X}_5, \mathcal{I}_5 \not\models_{\leadsto} \neg \mathit{HasGun} \vee \mathit{Loaded} \to [\mathit{shoot}]\bot,$$

but

$$\mathcal{S}_5, \mathcal{E}_5 \not\models_{\leadsto} \neg \mathit{HasGun} \vee \mathit{Loaded} \to [\mathit{shoot}]\neg \mathit{Alive}.$$

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**:

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

This expresses that if we have explicitly stated an effect law for a given action in some context, then there should be no inexecutability law in $\mathcal{E}$ for the same action in the same context. It is straightforward to design an algorithm which checks whether this postulate is satisfied. 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)**:

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

For this postulate, and considering the above action theory, we have that

$$\mathcal{E}_5 \models_{\leadsto} (\neg \mathit{HasGun} \land \mathit{Loaded}) \to [\mathit{shoot}]\neg \mathit{Alive},$$

but

$$\mathcal{S}_5, \mathcal{E}_5, \mathcal{X}_5, \mathcal{I}_5 \models_{\leadsto} (\neg \mathit{HasGun} \land \mathit{Loaded}) \to [\mathit{shoot}]\bot.$$

So this version is certainly too strong.

This also illustrates that it is sometimes natural to have some "redundancies" or "overlaps" between $\mathcal{I}$ and $\mathcal{E}$. (Indeed, as we have pointed out, inexecutability laws are a particular type of effect laws, and the distinction here made is conventional.)

8 Related work

Zhang and Chopra [16] have also proposed an assessment of what a good domain description should look like. They develop the ideas in the framework of *EPDL* [17], an extended version of PDL which allows for propositions as modalities in order to represent causal connection between literals. Without getting into the details of their logical framework that is used there, we concentrate here on the main metatheoretical results. They propose a quite expressive normal form for describing action theories, and three different levels of consistency are given.

Roughly speaking, an action theory Σ is *uniformly consistent* if it is globally consistent (i.e., $\Sigma \not\models_{EPDL} \bot$). A formula Φ is Σ-*consistent* if $\Sigma \not\models_{EPDL} \neg\Phi$, for Σ a uniformly consistent theory. *Universal consistency* of Σ w.r.t. a set of fluents means that every logically possible world is accessible, which in our terms corresponds to $\mathcal{S} = \emptyset$.

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

Nevertheless, such satisfaction is not a sufficient 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 in this approach. For instance, consider the following representation in *EPDL* of the scenario of Example 5.2:

$$\Sigma = \left\{ \begin{array}{c} On \rightarrow [toggle]\neg On, \\ Off \rightarrow [toggle]On, \\ [On]\neg Off, \\ [\neg On]Off \end{array} \right\}$$

The causality statement $[On]\neg Off$ means that On causes $\neg Off$. Such an action theory satisfies each of Zhang and Chopra's requirements (in particular it is uniformly consistent, because $\Sigma \not\models_{EPDL} \bot$). Nevertheless, Σ is not safe because the implicit static law $\neg(On \wedge Off)$ cannot be proved.[3]

Focusing on the computational issues of determining what a good domain description is, Lang *et al.* [8] address the problem in a base formalism very similar to the causal laws approach [10]. Instead of using a particular form of minimization, however, they suppose an abstract notion of *completion* of a domain description as a way of overcoming the frame problem.

Given an action theory T_α containing logical information about α's direct effects as well as the indirect effects they can produce, the completion of T_α (noted $cl(T_\alpha)$) roughly speaking is the original theory T_α amended of logical axioms stating the persistence of all non-affected (directly nor indirectly) literals. (Note that such a notion of completion is very close to the underlying semantics of the dependence relation $\rightsquigarrow$ originally appeared in [1] and used throughout the present paper.)

The consistency of a domain description (and by extension the evaluation of how good it is) is achieved by means of testing whether α is executable in all possible initial states (the EXECUTABILITY problem). Such a test amounts to checking whether every possible state s has a successor s' reachable by α such that s and s' both satisfy the completion of T_α. An implicit inexecutability for α being found means that T_α has a problem and should, thus, be revised.

For instance, still considering the lamp scenario, the representation of the action theory for *toggle* is:

$$T_{toggle} = \left\{ \begin{array}{c} On \xrightarrow{toggle} Off, \\ Off \xrightarrow{toggle} On, \\ Off \longrightarrow \neg On, \\ On \longrightarrow \neg Off \end{array} \right\}$$

where the first two formulas are effect laws relating *toggle* and its possible effects under the respective conditions, and the latter two are causal laws in McCain and Turner's sense. We will not dive in the technical details of this

[3] A possible solution could be considering the set of static constraints explicitly in the action theory (viz. in the deductive system). For the running example, taking into account the constraint $On \leftrightarrow \neg Off$ (derived from the causal statements and the *EPDL* global axioms), we can conclude that Σ is safe. On the other hand, all the side effects such a modification could have on the whole theory is to be analyzed.

formalism and just note that checking executability in the completion of this action theory will give "no" as answer, as *toggle* cannot be executed in a state satisfying $On \wedge Off$.

In the referred work, the authors are more concerned with the complexity analysis of the problem of doing such a consistency test and no algorithm for performing it is given, however. In spite of the fact they have the same motivation as us, again what is presented is just a kind of "yes-no tool" which can help in doing a metatheoretical analysis of a given domain description, and many of the comments concerning Zhang and Chopra's approach could be repeated here.

9 Conclusion

We have pointed out throughout this paper the problems that arise when one does not care about encoding a domain description in a modular way. We have argued that domain descriptions should be modular in the sense that the non-dynamic part should not be influenced by the dynamic one.[4]

Such a concept of modularity has motivated us to put forward several postulates. We 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 we have seen, a possible solution comes into its own with Algorithms 4.1, 5.1 and 6.1. Together, they can provide a way of checking how modular a given domain description is, and correct it if needed.

An important point we have not yet talked about, however, is the order of execution of each algorithm. It seems to be natural to run them following their order of presentation.

Acknowledgments

We are grateful to the anonymous referees for their thorough comments on the earlier version of this paper.

References

[1] Marcos A. Castilho, Olivier Gasquet, and Andreas Herzig. Formalizing action and change in modal logic I: the frame problem. *Journal of Logic and Computation*, 9(5):701–735, 1999.

[2] Robert Demolombe, Andreas Herzig, and Ivan Varzinczak. Regression in modal logic. *Journal of Applied Non-Classical Logic*, 13(2):165–185, 2003.

[4] 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.

[3] Patrick Doherty, Witold Łukaszewicz, and Andrzej Szałas. Explaining explanation closure. In *Proc. Int. Symp. on Methodologies for Intelligent Systems*, 1996.

[4] Matthew L. Ginsberg and David E. Smith. Reasoning about actions II: The qualification problem. *Artificial Intelligence*, 35(3):311–342, 1988.

[5] Laura Giordano, Alberto Martelli, and Camilla Schwind. Ramification and causality in a modal action logic. *Journal of Logic and Computation*, 10(5):625–662, 2000.

[6] David Harel. Dynamic logic. In D. M. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic*, volume 2: Extensions of Classical Logic, pages 497–604. D. Reidel Publishing Co., Dordrecht, 1984.

[7] Katsumi Inoue. Linear resolution for consequence finding. *Artificial Intelligence*, 56(2–3):301–353, 1992.

[8] Jérôme Lang, Fangzhen Lin, and Pierre Marquis. Causal theories of action – a computational core. In *Proc. IJCAI'2003*, pages 1073–1078, 2003.

[9] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In C. Mellish, editor, *Proc. IJCAI'95*, pages 1985–1991, 1995.

[10] Norman McCain and Hudson Turner. A causal theory of ramifications and qualifications. In C. Mellish, editor, *Proc. IJCAI'95*, pages 1978–1984, 1995.

[11] Norman McCain and Hudson Turner. Causal theories of action and change. In *Proc. AAAI'97*, pages 460–465, 1997.

[12] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[13] Lenhart K. Schubert. Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In H. E. Kyberg, R. P. Loui, and G. N. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67, 1990.

[14] Michael Thielscher. Computing ramifications by postprocessing. In C. Mellish, editor, *Proc. IJCAI'95*, pages 1994–2000, 1995.

[15] Michael Thielscher. Ramification and causality. *Artificial Intelligence*, 89(1–2):317–364, 1997.

[16] Dongmo Zhang and Samir Chopra. Consistency of action descriptions. In *PRICAI'02, Topics in Artificial Intelligence, Springer*, 2002.

[17] Dongmo Zhang and Norman Y. Foo. EPDL: A logic for causal reasoning. In *Proc. IJCAI'2001*, pages 131–138, 2001.