# On the Modularity of Theories

Andreas Herzig and Ivan Varzinczak

ABSTRACT. In this paper we give the notion of modularity of a theory and analyze some of its properties, especially for the case of action theories in reasoning about actions. We propose algorithms to check whether a given action theory is modular and that also make it modular, if needed. Completeness, correctness and termination results are demonstrated.

## 1 Introduction

In many cases knowledge is represented by logical theories containing multiple modalities $\alpha_1, \alpha_2, \ldots$ Then it is often the case that we have modularity, in the sense that our theory $\mathcal{T}$ can be partitioned into a union of theories

$$\mathcal{T} = \mathcal{T}^{\emptyset} \cup \mathcal{T}^{\alpha_1} \cup \mathcal{T}^{\alpha_2} \cup \ldots$$

such that

- $\mathcal{T}^{\emptyset}$ contains no modal operators, and
- the only modality of $\mathcal{T}^{\alpha_i}$ is $\alpha_i$.

We call these subtheories *modules* (some modules might by empty). Examples of such theories can be found in reasoning about actions, where each $\mathcal{T}^{\alpha_i}$ contains descriptions of the atomic action $\alpha_i$ in terms of preconditions and effects, and $\mathcal{T}^{\emptyset}$ is the set of static laws (alias domain constraints, alias integrity constraints), i.e., those formulas that hold in every possible state of a dynamic system, and are thus global axioms.

For example, consider the following theory:

$$\mathcal{T}^{marry} = \left\{ \begin{array}{c} \neg Married \rightarrow \langle marry \rangle \top, \\ [marry] Married \end{array} \right\}$$

$$\mathcal{T}^{\emptyset} = \{\neg(Married \wedge Bachelor)\}$$

Such a theory is composed of two subtheories, one for expressing the dynamic part of the theory, $\mathcal{T}^{marry}$, and one to formalize the constraints of the domain, $\mathcal{T}^{\emptyset}$. $\mathcal{T}^{marry}$ formalizes the behavior of the action of getting married, in this case the precondition for executing *marry* (viz. $\neg$*Married*) and the effect that obtains after its execution (viz. *Married*). $\mathcal{T}^{\emptyset}$ formalizes the domain constraint according to which it is not possible to be married and bachelor at the same time.

Another example is when mental attitudes such as knowledge, beliefs or goals of several independent agents are represented: then each $\mathcal{T}^{\alpha_i}$ contains the respective mental attitude of agent $\alpha_i$.[1]

Let the underlying multimodal logic be independently axiomatized (i.e., the logic is a fusion and there is no interaction between the modal operators), and suppose we want to know whether $\mathcal{T} \models \varphi$, i.e., whether a formula $\varphi$ follows from the theory $\mathcal{T}$. Then it is natural to expect that we only have to consider those elements of $\mathcal{T}$ which concern the modal operators occurring in $\varphi$. For instance the proof of some consequences of action $\alpha_1$ should not involve laws for other actions $\alpha_2$; querying the belief base of agent $\alpha_1$ should not require bothering with that of agent $\alpha_2$. Moreover, intensional information in any $\mathcal{T}^{\alpha_i}$ should not influence information about the laws of the world (encoded in $\mathcal{T}^{\emptyset}$). Note that this is not the case if the logic is not independently axiomatized, and there are interaction axioms such as $[\alpha_1]\varphi \rightarrow [\alpha_2]\varphi$.

Similar modularity principles can also be found in structural and object-oriented programming: a commonly used guideline in software development is to divide the software into modules, based on their functionality or on the similarity of the information they handle. This means that instead of having a "jack of all trades" program, it is preferable to split it up into specialized subprograms. For instance, a program made of a module for querying a database and a module for checking its integrity is more modular than a single module that does these two tasks at the same time.

The major benefits of modular systems are reusability, scalability and better management of complexity.[2] Among the criteria commonly used for evaluating how modular a piece of software is are the notions of *cohesion* and *coupling* [15, 17]. Roughly, cohesion is about how well defined a module is, while coupling is about how modules are interdependent. A common sense maxim in object-oriented design is maximize cohesion of modules and diminish their coupling, and this paradigm can also be applied to reasoning about actions [1, 4, 5].

[1]Here we should assume more generally that $[\alpha_i]$ is the only outermost modal operator of $\mathcal{T}^{\alpha_i}$; we think that this case could be analyzed in a way that is similar to ours.

[2]Observe that this is closely related to the concept of *elaboration tolerance* [12] in reasoning about actions.

In this work we pursue the following plan: after some logical preliminaries (Section 2) we formalize the concepts of modularity to be used throughout the paper (Sections 3 and 4). In Section 5 we focus then in a particular kind of theories that are commonly used in reasoning about actions and discuss how to decide (Section 6) and guarantee (Section 7) its modular property. We finish by addressing related work in the field and making some concluding remarks.

## 2 Preliminaries

Let $MOD = \{\alpha_1, \alpha_2, \ldots\}$ be the set of modal operators. Formulas are constructed in the standard way from these and the set of atomic formulas $ATM$. They are denoted by $\varphi, \psi, \ldots$ Formulas without modal operators (propositional formulas) are denoted by $PFOR = \{A, B, C, \ldots\}$.

Let $mod(\varphi)$ return the set of modal operators occurring in formula $\varphi$, and let $mod(\mathcal{T}) = \bigcup_{\psi \in \mathcal{T}} mod(\psi)$. For instance $mod([\alpha_1](p \to [\alpha_2]q)) = \{\alpha_1, \alpha_2\}$. If $\mathfrak{M} \subseteq MOD$ is a nonempty set of modalities, then we define

$$\mathcal{T}^{\mathfrak{M}} = \{\varphi \in \mathcal{T} : mod(\varphi) \cap \mathfrak{M} \neq \emptyset\}$$

For $\mathfrak{M} = \emptyset$, we define

$$\mathcal{T}^{\emptyset} = \{\varphi \in \mathcal{T} : mod(\varphi) = \emptyset\}$$

For example, if

$$\mathcal{T} = \left\{ \begin{array}{c} \neg(Married \land Bachelor), \\ \neg Married \to \langle marry \rangle \top, [marry]Married, \\ Married \to \langle divorce \rangle \top, [divorce]\neg Married \end{array} \right\}$$

then

$$\mathcal{T}^{\{divorce\}} = \{Married \to \langle divorce \rangle \top, [divorce]\neg Married\}$$

and

$$\mathcal{T}^{\emptyset} = \{\neg(Married \land Bachelor)\}$$

We write $\mathcal{T}^{\alpha}$ instead of $\mathcal{T}^{\{\alpha\}}$.

We suppose from now on that $\mathcal{T}$ is *partitioned*, in the sense that $\{\mathcal{T}^{\emptyset}\} \cup \{\mathcal{T}^{\alpha_i} : \alpha_i \in MOD\}$ is a partition of $\mathcal{T}$. We thus exclude $\mathcal{T}^{\alpha_i}$ containing more than one modal operator.

Models of the logic under concern are of the form $M = \langle W, R, V \rangle$, where $W$ is a set of possible worlds, $R : MOD \longrightarrow W \times W$ associates an accessibility relation to every modality, and $V : W \longrightarrow 2^{ATM}$ associates a valuation to every possible world.

Satisfaction of a formula $\varphi$ in world $w$ of model $M$ ($M, w \models \varphi$) and truth of a formula $\varphi$ in $M$ (denoted $M \models \varphi$) are defined as usual. Truth of a set of formulas $\mathcal{T}$ in $M$ (denoted $M \models \mathcal{T}$) is defined by: $M \models \mathcal{T}$ if and only if $M \models \psi$ for every $\psi \in \mathcal{T}$. $\mathcal{T}$ has global consequence $\varphi$ (denoted $\mathcal{T} \models \varphi$) if and only if ($M \models \mathcal{T}$ implies $M \models \varphi$). Note that the underlying logic is an extension of classical propositional logic: if $A$ is a logical consequence of $\mathcal{T}$ in classical propositional logic, then $\mathcal{T} \models A$.

We suppose that the logic under consideration is *compact*.

Given these fundamental concepts, we are able to formally define modularity of a theory.

## 3 Modularity

We make the following hypothesis:

$$\{\mathcal{T}^{\emptyset}\} \cup \{\mathcal{T}^{\alpha_i} : \alpha_i \in \mathit{MOD}\} \text{ partitions } \mathcal{T}^{3} \qquad \text{(H)}$$

We are interested in the following principle of modularity:

DEFINITION 1 A theory $\mathcal{T}$ is *modular* if and only if for every formula $\varphi$,

$$\mathcal{T} \models \varphi \text{ implies } \mathcal{T}^{mod(\varphi)} \cup \mathcal{T}^{\emptyset} \models \varphi$$

Modularity means that when investigating whether $\varphi$ is a consequence of $\mathcal{T}$, the only formulas of $\mathcal{T}$ that are relevant are those whose modal operators occur in $\varphi$ and the classical formulas in $\mathcal{T}^{\emptyset}$.

This is reminiscent of interpolation, which more or less[4] says:

DEFINITION 2 A theory $\mathcal{T}$ has the *interpolation property* if and only if for every formula $\varphi$, if $\mathcal{T} \models \varphi$, then there is a theory $\mathcal{T}_{\varphi}$ such that

- $mod(\mathcal{T}_{\varphi}) \subseteq mod(\mathcal{T}) \cap mod(\varphi)$
- $\mathcal{T} \models \psi$ for every $\psi \in \mathcal{T}_{\varphi}$
- $\mathcal{T}_{\varphi} \models \varphi$

Our definition of modularity is a strengthening of interpolation because it requires $\mathcal{T}_{\varphi}$ to be a subset of $\mathcal{T}$.

[3] $\{\mathcal{T}^{\emptyset}\} \cup \{\mathcal{T}^{\alpha_i} : \alpha_i \in \mathit{MOD}\}$ partitions $\mathcal{T}$ if and only if $\mathcal{T} = \mathcal{T}^{\emptyset} \cup \bigcup_{\alpha_i \in \mathit{MOD}} \mathcal{T}^{\alpha_i}$, and $\mathcal{T}^{\emptyset} \cap \mathcal{T}^{\alpha_i} = \emptyset$, and $\mathcal{T}^{\alpha_i} \cap \mathcal{T}^{\alpha_j} = \emptyset$, if $\alpha_i \neq \alpha_j$. Note that $\mathcal{T}^{\emptyset}$ and $\mathcal{T}^{\alpha_i}$ might be empty.

[4] We here present a version in terms of global consequence, as opposed to local consequence or material implication versions that can be found in the literature [6, 7]. We were unable to find such global versions in the literature.

Contrary to interpolation, modularity does not generally hold. Clearly if (H) is not satisfied, then modularity fails. To witness, consider

$$\mathcal{T} = \{p \rightarrow [\alpha][\beta]q, [\alpha][\beta]q \rightarrow r\}$$

Then $\mathcal{T} \models p \rightarrow r$, but $\mathcal{T}^{\emptyset} \not\models p \rightarrow r$.

Nevertheless even under our hypothesis modularity may fail to hold. For example, let

$$\mathcal{T} = \{p \vee [\alpha]\bot, p \vee \neg[\alpha]\bot\}$$

Then $\mathcal{T}^{\emptyset} = \emptyset$, and $\mathcal{T}^{\alpha} = \mathcal{T}$. Now $\mathcal{T} \models p$, but clearly $\mathcal{T}^{\emptyset} \not\models p$.

Being modular is a useful feature of theories: beyond being a reasonable principle of design that helps avoiding mistakes, it clearly restricts the search space, and thus makes reasoning easier. To witness, if $\mathcal{T}$ is modular then consistency of $\mathcal{T}$ amounts to consistency (in classical logic) of its propositional part $\mathcal{T}^{\emptyset}$. This is what we address in the following section.

## 4 Propositional modularity

How can we know whether a given theory $\mathcal{T}$ is modular? The following criterion is simpler:

DEFINITION 3 A theory $\mathcal{T}$ is *propositionally modular* if and only if for every propositional formula $A$,

$$\mathcal{T} \models A \text{ implies } \mathcal{T}^{\emptyset} \models A$$

And it will suffice to guarantee modularity:

THEOREM 4 Let the underlying logic be a fusion, and let $\mathcal{T}$ be a partitioned theory. If $\mathcal{T}$ is propositionally modular, then $\mathcal{T}$ is modular.

**Proof.** Let $\mathcal{T}$ be propositionally modular. Suppose $\mathcal{T}^{mod(\varphi)} \cup \mathcal{T}^{\emptyset} \not\models \varphi$. Hence there is a model $M = \langle W, R, V \rangle$ such that $M \models \mathcal{T}^{mod(\varphi)} \cup \mathcal{T}^{\emptyset}$, and there is some $w$ in $M$ such that $M, w \not\models \varphi$. We prove that $\mathcal{T} \not\models \varphi$ by constructing from $M$ a model $M'$ such that $M' \models \mathcal{T}$ and $M', w \not\models \varphi$.

First, as we have supposed that our logic is an extension of classical propositional logic and that it is compact, propositional modularity implies that for every propositional valuation $val \subseteq 2^{ATM}$ which is a model of $\mathcal{T}^{\emptyset}$ there is a possible worlds model $M_{val} = \langle W_{val}, R_{val}, V_{val} \rangle$ such that $M_{val} \models \mathcal{T}$, and there is some $w$ in $M_{val}$ such that $V_{val}(w) = val$. In other words, for every propositional model of $\mathcal{T}^{\emptyset}$ there is a model of $\mathcal{T}$ containing that propositional model.

Second, taking the disjoint union of all these models we obtain a 'big model' $M_{big}$ such that $M_{big} \models \mathcal{T}$, and for every propositional model $val \subseteq 2^{ATM}$ of $\mathcal{T}^{\emptyset}$ there is a possible world $w$ in $M_{big}$ such that $V(w) = val$.

Now we can use the big model to adjust those accessibility relations $R(\alpha)$ of $M$ whose $\alpha$ does not appear in $\varphi$, in a way such that the resulting model satisfies the rest of the theory $\mathcal{T} \setminus \mathcal{T}^{mod(\varphi)}$: let $M' = \langle W', R', V' \rangle$ be such that

- $W' = \{u_v : u \in W, v \in W_{big}, \text{ and } V(u) = V_{big}(v)\}$
- if $\alpha \in mod(\varphi)$, then $u_v R'(\alpha) u'_{v'}$ if and only if $uR(\alpha)u'$
- if $\alpha \notin mod(\varphi)$, then $u_v R'(\alpha) u'_{v'}$ if and only if $vR(\alpha)v'$
- $V'(u_v) = V(u) = V_{big}(v)$

$W'$ is nonempty because $M \models \mathcal{T}^{\emptyset}$. $M'$ is a model of the underlying logic because the latter is a fusion. Then for the sublanguage constructed from $mod(\varphi)$ it can be proved by structural induction that for every formula $\psi$ of the sublanguage and every $u \in W$ and $v \in W_{big}$, $M, u \models \psi$ if and only if $M', u_v \models \psi$. The same can be proved for the sublanguage constructed from $MOD \setminus mod(\varphi)$. As, by hypothesis, $\mathcal{T}$ is partitioned, $\mathcal{T}^{\emptyset}$ and each of our modules $\mathcal{T}^{\alpha}$ are in at least one of these sublanguages, thus we have proved that $M' \models \mathcal{T}$, and $M', w_v \not\models \varphi$ for every $v$. ■

In the rest of the paper we investigate how it can be automatically checked whether a given theory $\mathcal{T}$ is modular or not, and how to make it modular, if needed. We do this for a particular kind of theories commonly used in reasoning about actions. First of all we say what an action theory is.

## 5 Action theories

We suppose that the underlying logic is multimodal K. (Note that this is a fusion and that it is compact.)

Every formalization of a dynamic domain contains a representation of action effects. We call *effect laws* formulas relating an action to its effects. *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.

**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 formulas of classical propositional logic $A, B, \ldots \in PFOR$.

A *static law*[5] is a formula $A \in \mathit{PFOR}$ that is consistent. An example is $\mathit{Walking} \rightarrow \mathit{Alive}$, saying that if a turkey is walking, then it must be alive [19]. In our action theories $\mathcal{T}$, static laws correspond to $\mathcal{T}^{\emptyset}$.

**Effect laws** To speak about action effects we use the syntax of propositional dynamic logic (PDL) [3]. The formula $[\alpha]A$ expresses that $A$ is true after every possible execution of $\alpha$.

An *effect law*[6] *for* $\alpha$ is of the form $A \rightarrow [\alpha]C$, where $A, C \in \mathit{PFOR}$. The consequent $C$ is the effect which obtains when $\alpha$ is executed in a state where the antecedent $A$ holds. An example is $\mathit{Loaded} \rightarrow [\mathit{shoot}]\neg\mathit{Alive}$, saying that whenever the gun is loaded, after shooting the turkey is dead. Another one is $[\mathit{tease}]\mathit{Walking}$: in every circumstance, the result of teasing is that the turkey starts walking.

A particular case of effect laws are *inexecutability laws* of the form $A \rightarrow [\alpha]\bot$. For example $\neg\mathit{HasGun} \rightarrow [\mathit{shoot}]\bot$ 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.[7] 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 $\alpha$ is possible".

An *executability law*[8] *for* $\alpha$ is of the form $A \rightarrow \langle\alpha\rangle\top$, where $A \in \mathit{PFOR}$. For instance $\mathit{HasGun} \rightarrow \langle\mathit{shoot}\rangle\top$ says that shooting can be executed whenever the agent has a gun, and $\langle\mathit{tease}\rangle\top$ says that the turkey can always be teased.

**Action theories** $\mathcal{S} \subseteq \mathit{PFOR}$ denotes the set of all static laws of a domain. For a given action $\alpha \in \mathit{MOD}$, $\mathcal{E}_{\alpha}$ is the set of its effect laws, and $\mathcal{X}_{\alpha}$ is the set of its executability laws. We define $\mathcal{E} = \bigcup_{\alpha \in \mathit{MOD}} \mathcal{E}_{\alpha}$, and $\mathcal{X} = \bigcup_{\alpha \in \mathit{MOD}} \mathcal{X}_{\alpha}$. An *action theory* is a tuple of the form $\langle \mathcal{S}, \mathcal{E}, \mathcal{X} \rangle$. We suppose that $\mathcal{S}, \mathcal{E}$ and $\mathcal{X}$ are finite.

[5]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.

[6]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.

[7]Some authors [16, 2, 11, 19] more or less tacitly consider that executability laws should not be made explicit, but rather inferred by the reasoning mechanism. Others [10, 21] have executability laws as first class objects one can reason about. It seems a matter of debate whether one can always do without, but we think that 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.

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

EXAMPLE 5 Consider the following formalization of a transaction domain:

$$\mathcal{S} = \{\neg Adult \rightarrow \neg OblgPay\}$$

$$\mathcal{E} = \left\{ \begin{array}{c} [order]\,OblgPay, \\ \neg Adult \rightarrow [order]\neg Adult \end{array} \right\}$$

$$\mathcal{X} = \{\langle order\rangle\top\}$$

Observe that by the fact that $\mathcal{S}, \mathcal{E}, \mathcal{X} \models \neg Adult \rightarrow [order]\bot$ we have $\mathcal{S}, \mathcal{E}, \mathcal{X} \models Adult$. But $\mathcal{S} \not\models Adult$, hence $\langle\mathcal{S}, \mathcal{E}, \mathcal{X}\rangle$ is also an example of an action theory that is not modular.

Our central hypothesis here is that the different types of laws in an action theory should be neatly separated and only interfere in one sense: static laws together with action laws for $\alpha$ may have consequences that do not follow from the action laws for $\alpha$ alone. The other way round, action laws should not allow to infer new static laws. That is what modularity of action theories establishes and we develop it in the sequel.

## 6 Deciding modularity

How can we check whether a given action theory $\mathcal{T} = \langle\mathcal{S}, \mathcal{E}, \mathcal{X}\rangle$ is modular? Following Theorem 4, it is enough to check for propositional modularity.

DEFINITION 6 $A \in PFOR$ is an *implicit static law* of an action theory $\langle\mathcal{S}, \mathcal{E}, \mathcal{X}\rangle$ if and only if $\mathcal{S}, \mathcal{E}, \mathcal{X} \models A$ and $\mathcal{S} \not\models A$.

In Example 5, *Adult* is an example of an implicit static law.

Theorem 4 tells us that an action theory is modular if and only if it has no implicit static law. Hence, checking the existence of such laws provides us a way to decide modularity of a given action theory. Assuming $\mathcal{T}$ is finite, the algorithm below does the job:

ALGORITHM 7 (Finding some implicit static laws)

**input:** $\langle\mathcal{S}, \mathcal{E}, \mathcal{X}\rangle$
**output:** a set of implicit static laws $\mathcal{S}^I$

```
$\mathcal{S}^I := \emptyset$
for all $\alpha \in mod(\mathcal{E}) \cap mod(\mathcal{X})$ do
  for all $B \rightarrow \langle\alpha\rangle\top \in \mathcal{X}$ do
    for all $\{A_1 \rightarrow [\alpha]C_1, \ldots, A_n \rightarrow [\alpha]C_n\} \subseteq \mathcal{E}_\alpha$ do
      if $\mathcal{S} \cup \{C_1, \ldots, C_n\} \vdash \bot$ and $\mathcal{S} \cup \{B, A_1, \ldots, A_n\} \not\vdash \bot$ then
        $\mathcal{S}^I := \mathcal{S}^I \cup \{\neg(B \wedge A_1 \wedge \ldots \wedge A_n)\}$
```

THEOREM 8 Algorithm 7 terminates.

**Proof.** Straightforward from finiteness of $\mathcal{X}$ and $\mathcal{E}$. ■

LEMMA 9 For every $A \in \mathit{PFOR}$ such that $\mathcal{S}, \mathcal{E}, \mathcal{X} \models A$, if $A \in \mathcal{S}^I$, then $A$ is an implicit static law of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X} \rangle$.

**Proof.** Let $A \in \mathit{PFOR}$ be such that $A \in \mathcal{S}^I$ and $\mathcal{S}, \mathcal{E}, \mathcal{X} \models A$. $A$ is of the form $\neg(B \wedge A_1 \wedge \ldots \wedge A_n)$, for some $B, A_1, \ldots, A_n$, and $\mathcal{S} \wedge \neg(B \wedge A_1 \wedge \ldots \wedge A_n) \nvdash \bot$ is the case. Hence, $\mathcal{S} \wedge \neg A \nvdash \bot$, which means that $\mathcal{S} \not\models A$. Therefore $A$ is an implicit static law. ■

REMARK 10 The converse of Lemma 9 does not hold: consider the quite simple action theory

$$\langle \mathcal{S}, \mathcal{E}, \mathcal{X} \rangle = \left\langle \begin{array}{c} \{\neg p_n\}, \\ \{p_{i-1} \rightarrow [\alpha]p_i,\ 1 \leq i \leq n\}, \\ \{\langle \alpha \rangle \top\} \end{array} \right\rangle$$

Thus, $\langle \mathcal{S}, \mathcal{E}, \mathcal{X} \rangle \models \neg p_i$, for $0 \leq i \leq n$, but running Algorithm 7 returns only $\mathcal{S}^I = \{\neg p_{n-1}\}$. This suggests that it is necessary to iterate the algorithm in order to find all implicit static laws. We shall do this in the next section, and now just observe that:

THEOREM 11 An action theory $\langle \mathcal{S}, \mathcal{E}, \mathcal{X} \rangle$ is modular if and only if $\mathcal{S}^I = \emptyset$.

**Proof.** The left-to-right direction is straightforward, by Lemma 9.

For the right-to-left direction, suppose $\mathcal{S}^I = \emptyset$. Therefore for all subsets $\{A_1 \rightarrow [\alpha]C_1, \ldots, A_n \rightarrow [\alpha]C_n\}$ of $\mathcal{E}_\alpha$ and all $B \rightarrow \langle \alpha \rangle \top \in \mathcal{X}$ we have that

(1) if $\mathcal{S} \cup \{B, A_1, \ldots, A_n\} \nvdash \bot$, then $\mathcal{S} \cup \{C_1, \ldots, C_n\} \nvdash \bot$.

By Theorem 4, then, it suffices to prove that then $\langle \mathcal{S}, \mathcal{E}, \mathcal{X} \rangle$ is propositionally modular. Therefore, suppose $\mathcal{S} \not\models A$ for some propositional $A$. Let $W$ be the set of all propositional valuations satisfying $\mathcal{S}$ that falsify $A$. As $\mathcal{S} \not\models A$, $\mathcal{S} \cup \{\neg A\}$ is satisfiable, hence $W$ must be nonempty. For every $w \in W$ let

$$\mathcal{E}_\alpha(w) = \{A : A \rightarrow [\alpha]C \in \mathcal{E}_\alpha \text{ and } w \models A\}.$$

We define $R(\alpha)$ such that $wR(\alpha)w'$ if and only if

- $w \models B$ for some $B \rightarrow \langle \alpha \rangle \top \in \mathcal{X}$, and
- $w' \models C$ for every $A \rightarrow [\alpha]C \in \mathcal{E}_\alpha$ such that $A \in \mathcal{E}_\alpha(w)$.

Taking the obvious definition of $V$ we obtain a model $M = \langle W, R, V \rangle$. We have that $M \models \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X}$, because:

- $M \models \mathcal{S}$: by definition of $W$;
- $M \models \mathcal{E}$: for every world $w$, every $\alpha$ and every $A \rightarrow [\alpha]C \in \mathcal{E}_\alpha$ if $w \models A$, then, by the definition of $R(\alpha)$, $w' \models C$ for all $w' \in W$ such that $wR(\alpha)w'$;
- $M \models \mathcal{X}$: for every world $w$, every $\alpha$ and every $B \rightarrow \langle\alpha\rangle\top \in \mathcal{X}$, if $w \models B$, then from (1) and the definition of $R(\alpha)$, there exists at least one $w'$ such that $wR(\alpha)w'$.

Clearly $M \not\models A$, by the definition of $W$. Hence $\mathcal{S}, \mathcal{E}, \mathcal{X} \not\models A$. ■

## 7 Making action theories modular

Considering the action theory in Remark 10, we can see that running Algorithm 7 on $\langle \mathcal{S} \cup \{\neg p_{n-1}\}, \mathcal{E}, \mathcal{X}\rangle$ will give us $\mathcal{S}^I = \{\neg p_{n-2}\}$. This means that some of the implicit static laws of an action theory may be needed in order to derive others. Hence, Algorithm 7 must be iterated to get $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}\rangle$ modular. This is achieved with the following algorithm, which iteratively feeds the set of static laws considered into the **if**-test of Algorithm 7.

ALGORITHM 12 (Finding all implicit static laws)

```
input: $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}\rangle$
output: $\mathcal{S}^I_{\text{total}}$, the set of all implicit static laws of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}\rangle$
  $\mathcal{S}_{\text{new}} := \mathcal{S}$
  $\mathcal{S}^I_{\text{total}} := \emptyset$
  repeat
    $\mathcal{S}^I := \mathit{find\_imp\_stat}(\langle \mathcal{S}_{\text{new}}, \mathcal{E}, \mathcal{X}\rangle)$ /* a call to Algorithm 7 */
    $\mathcal{S}_{\text{new}} := \mathcal{S}_{\text{new}} \cup \mathcal{S}^I$
    $\mathcal{S}^I_{\text{total}} := \mathcal{S}^I_{\text{total}} \cup \mathcal{S}^I$
  until $\mathcal{S}^I = \emptyset$
```

THEOREM 13 Algorithm 12 terminates.

**Proof.** First, for given $\alpha$ the set of candidates to be an implicit static law is

$$\{\neg(B \wedge \bigwedge_{A_i \rightarrow [\alpha]C_i \in \mathcal{E}'_\alpha} A_i) : B \rightarrow \langle\alpha\rangle\top \in \mathcal{X} \text{ and } \mathcal{E}'_\alpha \subseteq \mathcal{E}_\alpha\}$$

This set is finite.

In each step either the algorithm ends because $\mathcal{S}^I = \emptyset$, or at least one of the candidates is put into $\mathcal{S}^I$ (by a call to Algorithm 7, which terminates). Such a candidate is not going to be put into $\mathcal{S}^I$ in future steps, because once added to $\mathcal{S}_{\text{new}}$, it will be in the set of laws of all subsequent calls to

Algorithm 7, falsifying its respective **if**-test for such a candidate. Hence the **repeat**-loop is bounded by the number of candidates, and therefore Algorithm 12 terminates. ∎

THEOREM 14 Let $\mathcal{S}^I_{\text{total}}$ be the output of Algorithm 12 on input $\langle \mathcal{S}, \mathcal{E}, \mathcal{X} \rangle$. Then

1. $\langle \mathcal{S} \cup \mathcal{S}^I_{\text{total}}, \mathcal{E}, \mathcal{X} \rangle$ is modular.
2. $\mathcal{S}, \mathcal{E}, \mathcal{X} \models \bigwedge \mathcal{S}^I_{\text{total}}$.

**Proof.** Item 1. is straightforward from the termination of Algorithm 12 and Theorem 11. Item 2. follows from the fact that by the **if**-test in Algorithm 7, the only formulas that are put in $\mathcal{S}^I_{\text{total}}$ at each execution of the loop are exactly those that are implicit static laws of the original theory. ∎

COROLLARY 15 For all $A \in \mathit{PFOR}$, $\mathcal{S}, \mathcal{E}, \mathcal{X} \models A$ if and only if $\mathcal{S} \cup \mathcal{S}^I_{\text{total}} \models A$.

**Proof.**
For the left-to-right direction, let $A \in \mathit{PFOR}$ be such that $\mathcal{S}, \mathcal{E}, \mathcal{X} \models A$. Then $\mathcal{S} \cup \mathcal{S}^I_{\text{total}}, \mathcal{E}, \mathcal{X} \models A$, by monotonicity. By Theorem 14-1., $\langle \mathcal{S} \cup \mathcal{S}^I_{\text{total}}, \mathcal{E}, \mathcal{X} \rangle$ is modular, hence $\mathcal{S} \cup \mathcal{S}^I_{\text{total}} \models A$.

The right-to-left direction is straightforward by Theorem 14-2. ∎

This establishes that Algorithm 12 finds all implicit static laws of a given action theory $\langle \mathcal{S}, \mathcal{E}, \mathcal{X} \rangle$. Adding such laws to the original set of static laws $\mathcal{S}$ guarantees, hence, modularity of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X} \rangle$.

In the next section we assess existing work on the field in the literature, emphasizing the points that make our approach a step further on a more fine-grained characterization of modularity.

## 8 Related work

Pirri and Reiter have investigated the metatheory of the Situation Calculus [14]. In a spirit similar to ours, they use executability laws and effect laws. Contrary to us, their executability laws are equivalences and are thus at the same time inexecutability laws. There are no static laws, i.e., $\mathcal{S} = \emptyset$. For this setting they give a syntactical condition on effect laws guaranteeing that they do not interact with the executability laws in the sense that they do not entail implicit static laws. Basically, the condition says that when there are effect laws $A \rightarrow [\alpha]C$ and $A' \rightarrow [\alpha]\neg C$, then $A$ and $A'$ are inconsistent (which essentially amounts to having in their theories a kind of "implicit static law schema" of the form $\neg(A \wedge A')$).

This then allows them to show that such theories are always consistent. Moreover they thus simplify the entailment problem for this calculus, and show for several problems such as consistency or regression that only some of the modules of an action theory are necessary.

Amir [1] focuses on design and maintainability of action descriptions applying many of the concepts of the object-oriented paradigm in the Situation Calculus. In that work, guidelines for a partitioned representation of a given theory are presented, with which the inference task can also be optimized, as it is restricted to the part of the theory that is really relevant to a given query. This is observed specially when different agents are involved: the design of an agent's theory can be done with no regard to others', and after the integration of multiple agents, queries about an agent's beliefs do not take into account the belief state of other agents.

In the above mentioned work, executabilities are as in [14] and the same condition on effect laws is assumed, which syntactically precludes the existence of implicit static laws.

Despite of using many of the object-oriented paradigm tools and techniques, no mention is made to the concepts of cohesion and coupling. In the approach presented in [1], even if modules are highly cohesive, they are not minimally coupled, due to the dependence between objects in the reasoning phase. We do not investigate this further here, but conjecture that this could be done there by, during the reasoning process defined for that approach, avoiding passing to a module a formula of a type different from those it contains.

The present work generalizes and extends Pirri and Reiter's result to the case where $\mathcal{S} \neq \emptyset$ and both Pirri and Reiter's and Amir's where the syntactical restriction on effect laws is not made. This gives us more expressive power, as we can reason about inexecutabilities, and a better modularity in the sense that we do not combine formulas that are conceptually different (viz. executabilities and inexecutabilities).

Zhang *et al.* [20] have also proposed an assessment of what a good action theory should look like. They develop the ideas in the framework of $\mathsf{EPDL}$ [21], an extended version of $\mathsf{PDL}$ 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,[9] and

[9]But not as expressive as one might think: For instance, in modeling the non-

investigate three levels of consistency. Roughly speaking, an action theory $\mathcal{T}$ is *uniformly consistent* if it is globally consistent (i.e., $\mathcal{T} \not\models_{\mathsf{EPDL}} \bot$); a formula $\varphi$ is *$\mathcal{T}$-consistent* if $\mathcal{T} \not\models_{\mathsf{EPDL}} \neg\varphi$, for $\mathcal{T}$ a uniformly consistent theory; $\mathcal{T}$ is *universally consistent* if (in our terms) every logically possible world is accessible.

Furthermore, two assumptions are made to preclude the existence of implicit qualifications. Satisfaction of such assumptions means the action theory under consideration is *safe*, i.e., it is uniformly consistent. Such a normal form justifies the two assumptions made and on whose validity relies their notion of good action theories.

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

Nevertheless, this is only a necessary condition: it is not hard to imagine action theories that are uniformly consistent but in which we can still have implicit laws 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 $\mathsf{EPDL}$ representation given by:

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

The causal statement $[\mathit{On}]\neg\mathit{Off}$ means that $\mathit{On}$ causes $\neg\mathit{Off}$. Such an action theory satisfies each of the consistency requirements (in particular it is uniformly consistent, as $\mathcal{T} \not\models_{\mathsf{EPDL}} \bot$). Nevertheless, $\mathcal{T}$ is not safe because the static law $\neg(\mathit{On} \wedge \mathit{Off})$ cannot be proved.[10]

Although they are concerned with the same kind of problems that have been discussed in this paper, they take an overall view of the subject, in

deterministic action of dropping a coin on a chessboard, we are not able to state $[\mathit{drop}](\mathit{Black} \vee \mathit{White})$. Instead, we should write something like $[\mathit{drop}_{\mathit{Black}}]\mathit{Black}$, $[\mathit{drop}_{\mathit{White}}]\mathit{White}$, $[\mathit{drop}_{\mathit{Black},\mathit{White}}]\mathit{Black}$ and $[\mathit{drop}_{\mathit{Black},\mathit{White}}]\mathit{White}$, where $\mathit{drop}_{\mathit{Black}}$ is the action of dropping the coin on a black square (analogously for the others) and $\mathit{drop} = \mathit{drop}_{\mathit{Black}} \cup \mathit{drop}_{\mathit{White}} \cup \mathit{drop}_{\mathit{Black},\mathit{White}}$, with "$\cup$" the nondeterministic composition of actions.

[10] 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 $\mathit{On} \leftrightarrow \neg\mathit{Off}$ (derived from the causal statements and the $\mathsf{EPDL}$ global axioms), we can conclude that $\mathcal{T}$ is safe. On the other hand, all the side effects such a modification could have on the whole theory has yet to be analyzed.

the sense that all problems are dealt with together. This means that in their approach no special attention (in our sense) is given to the different components of the action theory, and then every time something is wrong with it this is taken as a global problem inherent to the action theory as a whole. Whereas such a "systemic" view of action theories is not necessarily a drawback (we have just seen the strong interaction that exists between the different sets of laws composing an action theory), being modular in our sense allows us to circumscribe the "problematic" laws and take care of them. Moreover, the advantage of allowing to find the set of laws which must be modified in order to achieve the desired consistency is made evident by the algorithms we have proposed (while their results only allow to decide whether a given theory satisfies some consistency requirement).

Lang *et al.* [9] address consistency in the causal laws approach [11], focusing on the computational aspects. They suppose an abstract notion of completion of an action theory solving the frame problem. Given an action theory $\mathcal{T}^{\alpha}$ containing logical information about $\alpha$'s direct effects as well as the indirect effects that may follow, the completion of $\mathcal{T}^{\alpha}$ roughly speaking is the original theory $\mathcal{T}^{\alpha}$ amended of logical axioms stating the persistence of all non-affected (directly nor indirectly) literals.

Their EXECUTABILITY problem is to check whether $\alpha$ 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 $\alpha$ such that $w$ and $w'$ both satisfy the completion of $\mathcal{T}^{\alpha}$. For instance, still considering the lamp scenario, the representation of the action theory for *toggle* is:

$$\mathcal{T}^{toggle} = \left\{ \begin{array}{c} On \overset{toggle}{\longrightarrow} Off, \\ Off \overset{toggle}{\longrightarrow} On, \\ Off \longrightarrow \neg On, \\ On \longrightarrow \neg Off \end{array} \right\}$$

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 $On \wedge Off$.

In the mentioned 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 action theory, and many of the comments concerning Zhang and Chopra's approach could be repeated here.

## 9 Discussion and conclusion

In the perspective of independently axiomatized multimodal logics that are not serial we have investigated several criteria of modularity for simple theories. We have demonstrated the usefulness of modularity in reasoning about actions, where we have given an algorithmic checking for modularity of a given action theory.

We can have our criterion of modularity refined by taking into account polarity. Let $mod^{\pm}(\varphi)$ be the set of modalities of *MOD* occurring in $\varphi$ together with their polarity. For instance $mod^{\pm}([\alpha_1]([\alpha_2]p \rightarrow q)) = \{+\alpha_1, -\alpha_2\}$. $mod^{\pm}(\mathcal{T})$ is defined accordingly. If $\mathcal{M}$ is a set of modalities with polarity then we define: $\mathcal{T}^{\mathcal{M}} = \{\varphi \in \mathcal{T} : mod^{\pm}(\varphi) \cap \mathcal{M} \neq \emptyset\}$.

DEFINITION 16 A theory $\mathcal{T}$ is $\pm$-modular if and only if for every formula $\varphi$,

$$\mathcal{T} \models \varphi \text{ implies } \mathcal{T}^{mod^{\pm}(\varphi)} \cup \mathcal{T}^{\emptyset} \models \varphi$$

There are other theories that are modular but not $\pm$-modular, e.g.,

$$\mathcal{T} = \{\neg[\alpha]p, [\alpha]p \vee [\alpha]\neg p\}$$

Indeed, $\mathcal{T} \models [\alpha]\neg p$, but $\mathcal{T}^{+\alpha} \cup \mathcal{T}^{\emptyset} \not\models [\alpha]\neg p$.

For the restricted case of action theories this has been proved in [4]. Moreover, a monotonic solution to the frame problem has been integrated there in such an algorithm. This makes the algorithm a bit more complex as it involves computing prime implicates. For the sake of simplicity this has not been done here.

With regard to the action theory in Example 5, it can be argued that unintuitive consequences in action theories are mainly due to badly written axioms and not to the lack of modularity. True enough, but what we have presented here is the case that making a domain description modular gives us a tool to detect at least some of such problems and correct it. (But note that we do not claim to correct badly written axioms automatically and once for all). Besides this, having separate entities in the ontology and controlling their interaction help us to localize where the problems are, which can be crucial for real world applications.

A topic for further investigations could be considering the notion of *coherence* defined in [8] as a guideline for "repairing" a given theory. Roughly, given an action theory $\mathcal{T}$ and an *unintuitive* implicit static law $A$, the formulas in $\mathcal{T}$ that are most likely to be revised are exactly those whose utility, in Kwok *et al.*'s sense, for deriving $A$ are the highest.

## Acknowledgments

Ivan Varzinczak has been supported by a fellowship from the government of the FEDERATIVE REPUBLIC OF BRAZIL. Grant: CAPES BEX 1389/01-7.

## BIBLIOGRAPHY

[1] E. Amir. (De)composition of situation calculus theories. In *Proc. 17th Nat. Conf. on Artificial Intelligence (AAAI'2000)*, pages 456–463, Austin, 2000. AAAI Press/MIT Press.

[2] P. Doherty, W. Łukaszewicz, and A. Szałas. Explaining explanation closure. In *Proc. Int. Symposium on Methodologies for Intelligent Systems*, Zakopane, Poland, 1996.

[3] D. Harel. Dynamic logic. In D. M. Gabbay and F. Günthner, editors, *Handbook of Philosophical Logic*, volume II, pages 497–604. D. Reidel, Dordrecht, 1984.

[4] A. Herzig and I. Varzinczak. Domain descriptions should be modular. In R. López de Mántaras and L. Saitta, editors, *Proc. 16th Eur. Conf. on Artificial Intelligence (ECAI'04)*, pages 348–352, Valencia, 2004. IOS Press.

[5] A. Herzig and I. Varzinczak. On the cohesion and coupling of action theories. Technical report, Institut de recherche en informatique de Toulouse (IRIT), Université Paul Sabatier, March 2005. (Internal Report).

[6] M. Kracht and F. Wolter. Properties of independently axiomatizable bimodal logics. *J. of Symbolic Logic*, 56(4):1469–1485, 1991.

[7] M. Kracht and F. Wolter. Simulation and transfer results in modal logic: A survey. *Studia Logica*, 59:149–177, 1997.

[8] R. Kwok, N. Foo, and A. Nayak. Coherence of laws. In Sorge et al. [18], pages 1400–1401.

[9] J. Lang, F. Lin, and P Marquis. Causal theories of action – a computational core. In Sorge et al. [18], pages 1073–1078.

[10] F. Lin. Embracing causality in specifying the indirect effects of actions. In Mellish [13], pages 1985–1991.

[11] N. McCain and H. Turner. A causal theory of ramifications and qualifications. In Mellish [13], pages 1978–1984.

[12] J. McCarthy. *Mathematical logic in artificial intelligence*. Daedalus, 1988.

[13] C. Mellish, editor. *Proc. 14th Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*, Montreal, 1995. Morgan Kaufmann Publishers.

[14] F. Pirri and R. Reiter. Some contributions to the metatheory of the situation calculus. *Journal of the ACM*, 46(3):325–361, 1999.

[15] R. S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 1992.

[16] L. 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. Kluwer Academic Publishers, 1990.

[17] I. Sommerville. *Software Engineering*. Addison Wesley, 1985.

[18] V. Sorge, S. Colton, M. Fisher, and J. Gow, editors. *Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*, Acapulco, 2003. Morgan Kaufmann Publishers.

[19] M. Thielscher. Computing ramifications by postprocessing. In Mellish [13], pages 1994–2000.

[20] D. Zhang, S. Chopra, and N. Y. Foo. Consistency of action descriptions. In *PRI-CAI'02, Topics in Artificial Intelligence*. Springer-Verlag, 2002.

[21] D. Zhang and N. Y. Foo. EPDL: A logic for causal reasoning. In B. Nebel, editor, *Proc. 17th Int. Joint Conf. on Artificial Intelligence (IJCAI'01)*, pages 131–138, Seattle, 2001. Morgan Kaufmann Publishers.

Andreas Herzig
Institut de Recherche en Informatique de Toulouse (IRIT)
118 route de Narbonne
F-31062 Toulouse Cedex 4 (France)
e-mail: `herzig@irit.fr`
`http://www.irit.fr/recherches/LILAC`

Ivan Varzinczak
Institut de Recherche en Informatique de Toulouse (IRIT)
118 route de Narbonne
F-31062 Toulouse Cedex 4 (France)
e-mail: `ivan@irit.fr`
`http://www.irit.fr/~Ivan.Varzinczak`