Domain descriptions should be modular

Andreas Herzig **Ivan Varzinczak***
Institut de Recherche en Informatique de Toulouse (IRIT)
118 route de Narbonne, F-31062
Toulouse Cedex 4, France
e-mail: {herzig,ivan}@irit.fr

Abstract

We address the problem of what a good domain description for reasoning about actions should look like. We state some metatheoretic 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.

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. In order to distinguish them from logical axioms, we prefer to speak of effect laws, executability laws, and static laws, respectively. Moreover we single out those effect laws whose effect is $\bot$, and call them inexecutability 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, and it is natural to expect that it follows from the static laws alone. By means of examples we show that if this is not the case, then unexpected conclusions might follow from Σ.

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 will turn out that in all existing accounts which allow for these four kinds of laws (Lin 1995; McCain & Turner 1995; Thielscher 1995; Castilho, Gasquet, & Herzig 1999; Zhang & Foo 2001), 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 flawed domain descriptions can be made easier.

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

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. 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 (McCarthy & Hayes 1969). *PFOR* $= \{A, B, \ldots\}$ is the set of all classical formulas.

A *static law*[1] 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 (Thielscher 1995).

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) (Harel 1984). The formula $[\alpha]A$ expresses that A is true after every possible execution of α. $[\alpha][\beta]A$ should be interpreted in the usual way as $[\alpha;\beta]A$.

An *effect law*[2] *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

$$\textit{Loaded} \rightarrow [\textit{shoot}]\neg \textit{Alive},$$

saying that whenever the gun is loaded, after shooting the turkey is dead. Another one is [*tease*]*Walking*: in every circumstance, the result of teasing the turkey is that it starts walking.

[1] 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.

[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.

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.

All our postulates can be stated as well for other frameworks, in particular for action languages such as $\mathcal{A}$, $\mathcal{AR}$ and others, and for Situation Calculus based approaches. In action languages one would write $shoot$ causes $\neg$*Alive* if *Loaded*, and in the Situation Calculus formalism it would be

$$\forall s(\mathit{Holds}(\mathit{Loaded}, s) \rightarrow \neg \mathit{Holds}(\mathit{Alive}, \mathit{do}(\mathit{shoot}, s)))$$

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.

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*[3] *for* α is of the form $A \rightarrow \langle\alpha\rangle\top$, where $A \in$ *PFOR* is classically consistent. For instance *HasGun* $\rightarrow \langle$*shoot*$\rangle\top$ says that shooting can be executed whenever the agent has a gun, and $\langle$*tease*$\rangle\top$ establishes that the turkey can always be teased.

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 (Schubert 1990; Doherty, Łukaszewicz, & Szałas 1996; McCain & Turner 1995; Thielscher 1995) more or less tacitly consider that executability laws should not be made explicit but rather inferred by the reasoning mechanism. Others (Lin 1995; Zhang & Foo 2001) have executability laws as first class objects one can reason about.

It seems strange to us to just state information about necessary conditions for execution of an action (inexecutabilities) without saying anything 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. And if we have a plan such as *load*; *shoot* of which we know that it achieves the goal $\neg$*Alive* then we would like to be sure that it is executable in the first place! In any case, allowing for executability laws gives us more flexibility and expressive power.

[3]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.

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

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 $M = \langle W, R, I\rangle$,

- $w \models_M P$ if $w \in I(P)$;
- $\models_M \Phi$ 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

$$\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\mathsf{PDL}} \mathit{HasGun} \rightarrow [\mathit{load}]\mathit{HasGun}$$

(We write $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}$ instead of $\mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I}$.)

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 (Lin 1995; McCain & Turner 1995), several authors have argued that some notion of causality is needed. We here opt for the dependence based approach presented in (Castilho, Gasquet, & Herzig 1999), where dependence information has been added to PDL. In (Demolombe, Herzig, & Varzinczak 2003) it has been shown how Reiter's solution to the frame problem can be recast in PDL.

$\alpha \rightsquigarrow L$ denotes that the execution of action α *may change* the truth value of the literal L. In our example we have

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

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

meaning of these independences is that the frame axioms *HasGun* $\rightarrow$ [*load*]*HasGun*, $\neg$*Alive* $\rightarrow$ [*tease*]$\neg$*Alive* and *Alive* $\rightarrow$ [*tease*]*Alive* hold.

We assume $\rightsquigarrow$ is finite. A 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 $\models_{\rightsquigarrow}$. In our example we obtain

$$\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_{\rightsquigarrow} \textit{HasGun} \rightarrow [\textit{load}]\textit{HasGun}$$

Our dependence relation based approach thus solves the frame problem. However, it does not entirely solve the ramification problem: while indirect effects such as *Loaded* $\rightarrow$ [*shoot*]$\neg$*Walking* can be deduced with $\models_{\rightsquigarrow}$, we still have to state indirect dependences such as *shoot* $\rightsquigarrow \neg$*Walking*. Nevertheless, and as it has been argued in (Castilho, Herzig, & Varzinczak 2002; Herzig & Varzinczak 2004), our approach complies with the state of the art because none of the existing approaches can handle actions with both indeterminate and indirect effects.

Postulates

Our central hypothesis is that the different types of laws should be neatly separated and only interfere in one sense: static laws together with action laws for α may have consequences that do not follow from the action laws for α 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**: $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\rightsquigarrow} \bot$

A domain description should be logically consistent.

P1. **No implicit executability laws**:

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

If an executability 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**:

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

If a classical formula 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**:

if $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\rightsquigarrow} A \rightarrow [\alpha]\bot$, 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 $\mathcal{X}$.[4]

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.

No implicit inexecutability laws

Consider the following domain description, where $\rightsquigarrow$ is as above:

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

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

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

(For the time being, executability does not matter.) From [*tease*]*Walking* it follows with $\mathcal{S}_1$ that [*tease*]*Alive*, i.e., in every situation, after teasing the turkey is alive: $\mathcal{S}_1, \mathcal{E}_1 \models_{\mathsf{PDL}}$ [*tease*]*Alive*. Now as *tease* $\not\rightsquigarrow$ *Alive*, the status of *Alive* is not modified by *tease*, and we have $\mathcal{S}_1, \mathcal{E}_1 \models_{\rightsquigarrow} \neg$*Alive* $\rightarrow$ [*tease*]$\neg$*Alive*. From the above, it follows

$$\mathcal{S}_1, \mathcal{E}_1, \mathcal{X}_1, \mathcal{I}_1 \models_{\rightsquigarrow} \neg\textit{Alive} \rightarrow [\textit{tease}]\bot,$$

i.e., a dead turkey cannot be teased. But

$$\mathcal{S}_1, \mathcal{I}_1 \not\models_{\mathsf{PDL}} \neg\textit{Alive} \rightarrow [\textit{tease}]\bot,$$

hence Postulate P2 is violated. The formula $\neg$*Alive* $\rightarrow$ [*tease*]$\bot$ is an example of what we call an *implicit inexecutability law*.

In the literature, such laws are also known as *implicit qualifications* (Ginsberg & Smith 1988), and it has been argued that it is a positive feature of frameworks to leave them implicit and provide mechanisms for inferring them (Lin 1995; Thielscher 1997). The other way round, one might argue as well that implicit qualifications indicate that the domain has not been described in an adequate manner: the form of inexecutability laws is simpler than that of effect

[4] We nevertheless would like to point out that maximizing executability is not always intuitive. Suppose we know that if we have the ignition key, the tank is full, ..., and the battery tension is beyond 10V, then the car (necessarily) will start. Suppose we also know that if the tension is below 8V, then the car will not start. What should we conclude in situations where we know that the tension is 9V? Maximizing executabilities makes us infer that it will start, but such reasoning is not what we want if we would like to be sure that all possible executions lead to the goal.

laws, and it might be reasonably expected that it is easier to exhaustively describe them.[5] Thus, all 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 $NewCons_A(B)$ computes the set of strongest clauses that follow from $A \wedge B$, but do not follow from A alone (cf. e.g. (Inoue 1992)). It is known that $NewCons_A(B)$ 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 $NewCons_P((\neg P \vee Q) \wedge (\neg P \vee R \vee T)) = \{Q, R \vee T\}$. And for our example, $NewCons_{\mathit{Walking} \rightarrow \mathit{Alive}}(\mathit{Walking}) = \{\mathit{Alive}, \mathit{Walking}\}$.

Algorithm 1 (Finding implicit inexecutability laws)
input: $\mathcal{S}, \mathcal{E}, \mathcal{I}, \rightsquigarrow$
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**
      $A_J := \bigwedge\{A_i : A_i \rightarrow [\alpha]C_i \in J\}$
      $C_J := \bigwedge\{C_i : A_i \rightarrow [\alpha]C_i \in J\}$
      if $\mathcal{S} \cup \{A_J\}$ is classically consistent **then**
        for all $\bigvee L_i \in NewCons_{\mathcal{S}}(C_J)$ **do**
          if $\forall i,\ \alpha \not\rightsquigarrow L_i$ **and** $\mathcal{S}, \mathcal{I} \not\models_{\mathsf{PDL}} (A_J \wedge \bigwedge \neg L_i) \rightarrow [\alpha]\bot$ **then**
            $\mathcal{I}^I := \mathcal{I}^I \cup \{(A_J \wedge \bigwedge \neg L_i) \rightarrow [\alpha]\bot\}$

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

Note that the algorithm terminates because we have assumed $\rightsquigarrow$ finite.

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

The proof of this theorem relies on a sort of interpolation theorem for multimodal logic, which basically says that if $\Phi \models \Psi$ and Φ and Ψ have no action symbol in common, then there is a classical formula A such that $\Phi \models A$ and $A \models \Psi$.

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 $NewCons_{\mathcal{S}}(C_J)$ might result in exponential growth. While we might expect $\mathcal{E}_\alpha$ to be reasonably small in practice, the size of $NewCons_{\mathcal{S}}(C_J)$ is more difficult to control.

The algorithm not only decides whether the postulate is satisfied, its output $\mathcal{I}^I$ also provides information on how to "repair" the domain description. Basically there are three options, that we illustrate with our example:

1. Add $\neg \mathit{Alive} \rightarrow [\mathit{tease}]\bot$ to $\mathcal{I}_1$;
2. Add the (unintuitive) dependence $\langle \mathit{tease}, \mathit{Alive} \rangle$ to $\rightsquigarrow$;
3. Weaken the effect law $[\mathit{tease}]\mathit{Walking}$ to $\mathit{Alive} \rightarrow [\mathit{tease}]\mathit{Walking}$.

It is easy to see that whatever we opt for, the new domain description will satisfy P2.

Now we turn to another type of implicit laws.

No implicit static laws

Executability laws increase expressive power, but might conflict with inexecutability laws. For instance, let $\mathcal{S}_2 = \mathcal{S}_1$, $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{X}_2 = \{\langle \mathit{tease} \rangle \top\}$, and $\mathcal{I}_2 = \{\neg \mathit{Alive} \rightarrow [\mathit{tease}]\bot\}$. (Note that Postulate P2 is satisfied.) We have the unintuitive $\mathcal{X}_2, \mathcal{I}_2 \models_{\mathsf{PDL}} \mathit{Alive}$: 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 $\mathcal{I}$, which can be obtained by running Algorithm 1 in a first stage.

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 $A \rightarrow [\alpha]\bot \in \mathcal{I}$ and $A' \rightarrow \langle \alpha \rangle \top \in \mathcal{X}$ **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 $\mathcal{S}^I = \{\mathit{Alive}\}$.

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 (At_i means we are at number i):

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

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

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

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

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

In order to satisfy P2, we run Algorithm 1 and get $\mathcal{I}_3 = \{(\mathit{At}_1 \wedge \mathit{At}_2) \rightarrow [\mathit{goleft}]\bot\}$. Now applying Algorithm 2 to this domain description gives us the implicit static law $\neg(\mathit{At}_1 \wedge \mathit{At}_2)$, 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 this concerns the necessary conditions for executability, and thus it is not related to the qualification problem, which basically says that it is difficult to state all the sufficient conditions for executability.

What shall we do with an implicit static law? Again, there are several options: whereas in the latter example the implicit static law should be added to $\mathcal{S}$, (and more generally $\neg(At_i \wedge At_j)$ for $i \neq j$) 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.

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 under the hypothesis of complete knowledge of the preconditions for executability of actions.

For our running example, letting $\mathcal{X}_4 := \mathcal{X}_4 \cup \mathcal{X}^I$ establishes maximal executability for such a domain description.

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 Postulate P3, then $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_\rightsquigarrow \bot$ 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 Postulate P3, then $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_\rightsquigarrow A \rightarrow [\alpha]C$ iff $\mathcal{S}, \mathcal{E}_\alpha, \mathcal{I}_\alpha \models_\rightsquigarrow A \rightarrow [\alpha]C$.

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:

Theorem 6 If $\langle\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}\rangle$ satisfies Postulate P3, then $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_\rightsquigarrow A \rightarrow \langle\alpha\rangle\top$ iff $\mathcal{S}, \mathcal{X}_\alpha \models_\rightsquigarrow A \rightarrow \langle\alpha\rangle\top$.

Theorem 7 If $\langle\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}\rangle$ satisfies Postulates P2 and P3, then $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_\rightsquigarrow A \rightarrow [\alpha]\bot$ iff $\mathcal{S}, \mathcal{I}_\alpha \models_\rightsquigarrow A \rightarrow [\alpha]\bot$.

Although in the present paper concurrency is not taken into account, we conjecture that Theorems 5, 6 and 7 hold when we have concurrent action execution.

Theorem 8 There exist domain descriptions $\langle\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I}\rangle$ not satisfying P3 such that $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_\rightsquigarrow A \rightarrow [\alpha]C$ and $\mathcal{S}, \mathcal{E}_\alpha, \mathcal{I}_\alpha \not\models_\rightsquigarrow A \rightarrow [\alpha]C$.

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

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 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. This means we should have:

P5. **No implicit effect laws**:

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

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 $\rightsquigarrow$ of Example 1. It satisfies Postulates P1, P2, P3, and P4, but does not satisfy P5. Indeed:

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

and

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

but

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

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**:

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

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)**:

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

Indeed, for the above domain description we have

$$\mathcal{E}_5 \models_{\rightsquigarrow} (\neg \mathit{HasGun} \wedge \mathit{Loaded}) \rightarrow [\mathit{shoot}]\neg \mathit{Alive},$$

but

$$\mathcal{S}_5, \mathcal{E}_5, \mathcal{X}_5, \mathcal{I}_5 \models_{\rightsquigarrow} (\neg \mathit{HasGun} \wedge \mathit{Loaded}) \rightarrow [\mathit{shoot}]\bot.$$

This is certainly too strong. Our example also illustrates that it is sometimes natural to have 'redundancies' or 'overlaps' between $\mathcal{I}$ and $\mathcal{E}$.

Related work

Pirri and Reiter have investigated the metatheory of the situation calculus (Pirri & Reiter 1999). 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.* (Zhang, Chopra, & Foo 2002) have also proposed an assessment of what a good domain description should look like. They develop the ideas in the framework of EPDL (Zhang & Foo 2001), 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, [6] 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 (in our terms) every logically possible world is accessible. $\Sigma \models_{\mathsf{EPDL}} A$ implies $\models_{\mathsf{EPDL}} A$.

Furthermore, two assumptions are made to preclude the existence of implicit qualifications. Satisfaction of such assumptions means the domain description 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 domain descriptions.

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:

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

The causal statement $[\mathit{On}]\neg \mathit{Off}$ means that *On* causes $\neg$*Off*. Such a domain description satisfies each of the consistency requirements (in particular it is uniformly consistent, as $\Sigma \not\models_{\mathsf{EPDL}} \bot$). Nevertheless, Σ is not safe for the static law $\neg(\mathit{On} \wedge \mathit{Off})$ cannot be proved.[7]

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 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 domain description, and then every time something is wrong with it this is taken as a global problem inherent to the domain description 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 a domain description), 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.* (Lang, Lin, & Marquis 2003) address consistency in the causal laws approach (McCain & Turner 1995), 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

[6] But not as expressive as one might think: For instance, in modeling the nondeterministic 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.

[7] A possible solution could be considering the set of static constraints explicitly in the domain description (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 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 has yet to be analyzed.

description for *toggle* is:

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

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 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.

Another criticism that could be made about both these approaches concerns the assumption of full executability they rely on. We find it too strong to require all actions to be always executable, and to reject as bad a domain description admitting situations where some action cannot be executed at all. As an example, consider the very simple domain description given by $\mathcal{S}_6 = \mathcal{S}_1$, $\mathcal{E}_6 = \{[tease]Walking\}$, $\mathcal{X}_6 = \mathcal{X}_1$ and $\mathcal{I}_6 = \mathcal{I}_1$, and consider $\rightsquigarrow = \{\langle tease, Walking\rangle\}$. Observe that, with our approach, it suffices to derive the implicit inexecutability law $\neg Alive \rightarrow [tease]\bot$, change $\mathcal{I}$, and the system will properly run in situations where $\neg Alive$ is the case.

On the other hand, if we consider the equivalent representation of such a domain description in the approach of Lang *et al.*, after computing the completion of Σ_{tease}, if we test its executability we will get the answer "no", the reason being that *tease* is not executable in the possible state where $\neg Alive$ holds. Such an answer is correct, but note that with only this as guideline we have no idea about where a possible modification in the action theory should be carried on in order to achieve full executability for *tease*. The same holds for Zhang and Chopra's proposal.

Just to see how things can be even worse, consider the domain description $\langle \mathcal{S}'_6, \mathcal{E}'_6, \mathcal{X}'_6, \mathcal{I}'_6\rangle$, with $\mathcal{S}'_6 = \mathcal{S}_6$, $\mathcal{E}'_6 = \mathcal{E}_6$, $\mathcal{X}'_6 = \{Alive \rightarrow \langle tease\rangle\top\}$ and $\mathcal{I}'_6 = \{\neg Alive \rightarrow [tease]\bot\}$, with the same $\rightsquigarrow$, obtained by the correction of $\langle \mathcal{S}_6, \mathcal{E}_6, \mathcal{X}_6, \mathcal{I}_6\rangle$ above with the algorithms we propose. Observe that $\langle \mathcal{S}'_6, \mathcal{E}'_6, \mathcal{X}'_6, \mathcal{I}'_6\rangle$ satisfies all our postulates. It is not hard to see, however, that the representation of such an domain description in the above frameworks, when checked by their respective consistency tests, is still considered to have a problem.

This problem arises because Lang *et al.* do not allow for executability laws, thus they cannot make the distinction between $\mathcal{X} = \{\langle tease\rangle\top\}$, $\mathcal{X} = \{Alive \rightarrow \langle tease\rangle\top\}$ and $\mathcal{X} = \emptyset$. By their turn, Zhang and Chopra allows for specifying executabilities, but their consistency definitions do not distinguish the cases $Alive \rightarrow \langle tease\rangle\top$ and $\langle tease\rangle\top$.

Conclusion

We have tried to point out some of the problems that arise if 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.[8]

We have put forward some postulates, and in particular tried to demonstrate that when there are implicit inexecutability and static laws then one has slipped up in designing the domain description in question. As shown, a possible solution comes into its own with Algorithms 1 and 2, 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 $\mathcal{I}$.

In 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.

The present paper is also a step toward the solution of the problem of indirect dependences: indeed, if the indirect dependence $shoot \rightsquigarrow \neg Walking$ is not in $\rightsquigarrow$, then after running Algorithm 1 we get an indirect inexecutability $(Loaded \wedge Walking) \rightarrow [shoot]\bot$, i.e., *shoot* cannot be executed if $Loaded \wedge Walking$ holds. Such an unintuitive inexecutability is not in $\mathcal{I}$ and thus indicates the missing indirect dependence.

The general case is nevertheless more complex, and it seems that such indirect dependences cannot be computed automatically in the case of indeterminate effects. We are currently investigating this issue.

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

References

Castilho, M. A.; Gasquet, O.; and Herzig, A. 1999. Formalizing action and change in modal logic I: the frame problem. *J. of Logic and Computation* 9(5):701–735.

Castilho, M. A.; Herzig, A.; and Varzinczak, I. J. 2002. It depends on the context! a decidable logic of actions and

[8] 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.

plans based on a ternary dependence relation. In Benferhat, S., and Giunchiglia, E., eds., *Proc. Int. Conf. on Non-Monotonic Reasoning (NMR'02)*, 343–348.

Demolombe, R.; Herzig, A.; and Varzinczak, I. 2003. Regression in modal logic. *J. of Applied Non-classical Logics (JANCL)* 13(2):165–185.

Doherty, P.; Łukaszewicz, W.; and Szałas, A. 1996. Explaining explanation closure. In *Proc. Int. Symposium on Methodologies for Intelligent Systems*.

Ginsberg, M. L., and Smith, D. E. 1988. Reasoning about actions II: The qualification problem. *Artificial Intelligence* 35(3):311–342.

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

Herzig, A., and Varzinczak, I. 2004. An assessment of actions with indeterminate and indirect effects in some causal approaches. Technical report, Institut de recherche en informatique de Toulouse (IRIT), Université Paul Sabatier. `http://www.irit.fr/ACTIVITES/LILaC/`.

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

Lang, J.; Lin, F.; and Marquis, P. 2003. Causal theories of action – a computational core. In Sorge, V.; Colton, S.; Fisher, M.; and Gow, J., eds., *Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*, 1073–1078. Acapulco: Morgan Kaufmann Publishers.

Lin, F. 1995. Embracing causality in specifying the indirect effects of actions. In Mellish (1995), 1985–1991.

McCain, N., and Turner, H. 1995. A causal theory of ramifications and qualifications. In Mellish (1995), 1978–1984.

McCarthy, J., and Hayes, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B., and Mitchie, D., eds., *Machine Intelligence*, volume 4. Edinburgh University Press. 463–502.

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

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

Schubert, L. K. 1990. Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In Kyberg, H. E.; Loui, R. P.; and Carlson, G. N., eds., *Knowledge Representation and Defeasible Reasoning*. Kluwer Academic Publishers. 23–67.

Thielscher, M. 1995. Computing ramifications by postprocessing. In Mellish (1995), 1994–2000.

Thielscher, M. 1997. Ramification and causality. *Artificial Intelligence* 89(1–2):317–364.

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

Zhang, D.; Chopra, S.; and Foo, N. Y. 2002. Consistency of action descriptions. In *PRICAI'02, Topics in Artificial Intelligence*. Springer-Verlag.