Formalizing action and change in modal logic I: the frame problem

Marcos A. Castilho* Olivier Gasquet Andreas Herzig

Université Paul Sabatier — IRIT — Applied Logic Group
http://www.irit.fr/ACTIVITES/EQ_ALG
118, route de Narbonne
F-31062 – TOULOUSE CEDEX 4 – FRANCE
e-mail: {castilho, gasquet, herzig}@irit.fr
fax: +33 5.61.55.83.25

September 28, 1998

Abstract

We present the basic framework of a logic of actions and plans defined in terms of modal logic combined with a notion of dependence. The latter is used as a weak causal connection between actions and literals. In this paper we focus on the frame problem and demonstrate how it can be solved in our framework in a simple and monotonic way. We give the semantics, and associate an axiomatics and a decision procedure to it. The decision procedure is based on a sound and complete tableau method with single step rules to treat dependence. We show how it can be used to generate plans. Our solution is formally assessed by a translation of Gelfond and Lifschitz' logic $\mathcal{A}$. We briefly sketch the second part of the paper, showing how we can go beyond $\mathcal{A}$ by some examples involving nondeterminism and ramifications.

Keywords

Reasoning about actions, modal logic, frame problem, dependence, semantic tableaux

*On leave from Federal University of Paranà/Brazil. e-mail: marcos@inf.ufpr.br

Contents

1 Introduction

Up to now, the most prominent formalism in the reasoning about actions community has been the situation calculus. On the other hand, modal logics and in particular dynamic logic have been most prominent in two other fields, viz. the mathematical analysis of programs [28], and the philosophical analysis of actions [57]. It is the aim of this paper to give an analysis of the problems studied in the reasoning about actions community in terms of a simple multimodal logic close to Propositional Dynamic Logic (PDL).

If we want to undertake a systematic study of actions, we need a basic formalism we should agree on. Such a formalism should be as simple as possible. In this sense, the language of situation calculus is too rich. (The first reason is that it contains situations as first-class objects. Such entities do not seem to be basic in natural language, and we should try to do without. The second reason is that situation calculus is not propositional, but first-order 'right from the beginning', and consequently undecidable.) The recent trend to construct formalisms from a small number of concepts supports that claim. An example is the family of action logics based on $\mathcal{A}$ [19], which was explicitly designed to have a clear and simple language and semantics. Their basic ingredients are atomic actions and atomic formulas (propositional fluents), together with the classical connectives and the non-classical connectives λ, ; , after , and causes . Then complex actions and formulas can be built: λ is the empty action, $\alpha_1 ; \alpha_2$ is an action if α_1 and α_2 are, and A after α and α causes A are formulas is A if a formula and α is an action.[1] This can be compared to PDL: action composition takes the same form there, and A after α corresponds to the modal formula $[\alpha]A$, while the causes connective has no direct PDL counterpart. Therefore the PDL language is a good candidate for a basic language to talk about actions.

Many approaches to reasoning about actions presuppose that nonmonotonic features cannot be avoided if we want to solve the frame problem. Such features are tricky: first, nonmonotonic inference systems seem to be difficult to design in general (see the nonmonotonic reasoning community). Second, the definitions that have been proposed are very sophisticated ones, difficult to understand (see e.g. the different variants of circumscription). Third, and in consequence of the previous, there is a strong risk of getting unintuitive results, as exemplified by the classical Yale shooting scenario [27], and Sandewall's review of the domain [54].

If nonmonotonic mechanisms are required anyway, then this might be used as an argument against modal approaches: it should be simpler to add nonmonotonicity to classical logic than to a nonclassical logic. But in the last years there was a trend to monotonic solutions of the frame problem (see the work of Haas, Schubert, and most prominently Reiter [50]). Therefore, this argument cannot be used any longer. Significantly enough, it has been demonstrated recently that these solutions can be encoded in PDL [13].

We are convinced that such monotonic modal logic based approaches can make the way towards a proof-theoretic analysis of the domain. Moreover we can thus take profit of more than 30 years of research in the automated deduction community, where

[1] We have generalized here by allowing full compositionality. This allows us to identify the supplementary conditional *if* appearing in *causes* constructions with material implication: α causes A if B becomes $B \to (\alpha$ causes $A)$. Moreover, the empty action and action composition are noted here as in the extensions of $\mathcal{A}$, and not as in $\mathcal{A}$. We refer to section 8.1. for a precise definition of $\mathcal{A}$.

there has been done a lot of both theoretical and implementation work in particular on tableau theorem provers for modal logics. In this way we get an alternative to the existing automated methods of reasoning about actions, which mainly rely on either logic programming or circumscription.

Our basic logic, the logic of action and plans ($\mathcal{LAP}$), is slightly simpler than PDL. It is a propositional multimodal logic, with a modal operator $[\alpha]$ for every atomic action α, and a supplementary modal operator $\Box$ which is used to represent laws, i.e. formulas that are true in every situation. $\Box$ corresponds to the always connective in $\mathcal{AR}$ and $\mathcal{AR}_0$. The dual modal operator of possibility $\Diamond$ allows to express goals in plan generation tasks. Basically, the logic of each $[\alpha]$ is the modal logic K, while that of $\Box$ is $S4$. $\Box$ interacts with every $[\alpha]$ in the sense that $\Box A \rightarrow [\alpha]A$. The language of $\mathcal{LAP}$ is conceptually simpler than that of PDL, but it is sufficient to express most of the problems that have been investigated in the field.

The only connective of $\mathcal{A}$ that is lacking in $\mathcal{LAP}$ is causes. In $\mathcal{A}$, if α causes A holds then for every possible state of the world, A holds after the execution of α in that state. What is involved here is quantification over possible states. This aspect of the meaning of causes can be captured in $\mathcal{LAP}$ by means of the $\Box$ operator: if α causes A then we have $\Box[\alpha]A$. But the $\mathcal{LAP}$ formula $\Box[\alpha]A$ translates only part of the meaning of α causes A. The other aspect of the semantics of causes basically is that the execution of α does not change the truth values of all those atomic formulas not occurring in any causes statement for α. This is the way the frame problem is solved in $\mathcal{A}$. Indeed, this aspect could be captured in $\mathcal{LAP}$ by an appropriate set of formulas of the form $\Box(B \rightarrow [\alpha]B)$. But we clearly do not want to write down such frame axioms. Hence the frame problem puts the same type of difficulties in $\mathcal{LAP}$ as in the situation calculus. Our solution is different from the one adopted in the $\mathcal{A}$-family, and is somewhat closer in spirit to Sandewall's Features and Fluents framework [54], and related concepts.[2] We use an independence relation $\not\rightsquigarrow$ between actions and literals: $\alpha \not\rightsquigarrow L$ is read "L is independent of α", or "α may not cause L". In other words, if L has truth value false, it remains false after execution of α. This means that we represent a frame axiom $\Box(\neg L \rightarrow [\alpha]\neg L)$ by $\alpha \not\rightsquigarrow L$. But it follows from the principle of inertia that, for a given action, the number of literals that it does not affect is big. On the other hand, the number of literals affected by that action is small. Hence we shall not represent the independence relation itself, but its complement, the dependence relation $\rightsquigarrow$ between actions and literals (which can be supposed to be small).

Now, given a dependence relation $\rightsquigarrow$, we integrate it into $\mathcal{LAP}$, resulting in a new dependence-based logic $\mathcal{LAP}_{\rightsquigarrow}$. Its models are those models of $\mathcal{LAP}$ satisfying a single dependence-based condition, saying that if the execution of action α in a possible world (situation) w leads to world w', then for every literal L such that $\alpha \not\rightsquigarrow L$, if L is false in w then L is false in w'. In this way, we marry modal logic and dependence relations. We give a Hilbert style axiomatic system for that semantics, and show that standard proof-theoretic results for modal logics transfer in a straightforward way. We focus on theorem proving techniques, in particular tableau proof procedures. (We note that our solution could be used in the situation calculus as well, but technical results – in particular decidability – would be more difficult to obtain.)

[2] One of these being possibly changes of $\mathcal{AR}$ (as well as releases of $\mathcal{AR}_0$), in a sense we shall demonstrate that causes is not basic in $\mathcal{AR}$ and $\mathcal{AR}_0$.

The rest of the paper is as follows: in section 2 we define our object language, and in sections 3 and 4 we present the basic logic $\mathcal{LAP}$. In this logic the frame problem is not solved. In order to solve it, we introduce the notion of dependence, and for each dependence relation $\leadsto$ we obtain a new logic $\mathcal{LAP}_{\leadsto}$. This family of logics is presented in section 5. In section 6 we give a sound and complete tableau method with special stepback and stepforward rules. We present in section 7 an application of the tableau method in plan generation tasks. $\mathcal{LAP}_{\leadsto}$ is assessed in section 8, where we show a sound and complete translation from the logic $\mathcal{A}$ into $\mathcal{LAP}_{\leadsto}$. In section 9 we go beyond $\mathcal{A}$ and discuss some important points such as ramifications and nondeterminism. These latter points are treated in detail in the second part of this paper.

2 Language and notation

In our object language we have a family of operators $[\alpha]$, where α is the name of an atomic action, plus a single non-indexed operator $\Box$ that is used to write action laws as well as static laws.

Let $ACT = \{\alpha, \beta, \ldots\}$ be the set of *atomic actions* (like "*load*" and "*shoot*"), and let $ATM = \{P, Q, \ldots\}$ be the set of *atomic formulas*, or just *atoms* (like "*Loaded*" and "*Alive*"). Both sets may be infinite. The set $LIT = \{L, L_1, \ldots\}$ of literals is made up of all atoms and their negations. The set FOR of *formulas* is defined in the usual way as the smallest set such that

- $ATM \subseteq FOR$;
- $\top, \bot \in FOR$;
- if $A, B \in FOR$ and $\alpha \in ACT$ then $\neg A, A \wedge B, \Box A, [\alpha]A \in FOR$.

We read $[\alpha]A$ as "A holds after the execution of action α", $\Box A$ as "A is always true". In particular, $[\alpha]\bot$ means that α is not executable. A formula without modal operators is *classical*.

We use $\vee, \rightarrow, \leftrightarrow, \oplus$ as usual ($\oplus$ is the exclusive or). $\langle\alpha\rangle A$ abbreviates $\neg[\alpha]\neg A$ and $\Diamond A$ abbreviates $\neg\Box\neg A$. $\langle\alpha\rangle\top$ means that α is executable. $\Diamond A$ will be useful for planning: we read $\Diamond A$ as "there is a plan to bring about A". We also make use of some constructions familiar from dynamic logic: $[\alpha;\beta]A$ abbreviates $[\alpha][\beta]A$, $[\alpha \cup \beta]A$ abbreviates $[\alpha]A \wedge [\beta]A$, $[A?]B$ abbreviates $A \rightarrow B$, $[\lambda]A$ abbreviates A, and [if A then α else β]B abbreviates $(A \rightarrow [\alpha]B) \wedge (\neg A \rightarrow [\beta]B)$. We read λ as "do nothing", $A?$ as "continue if A holds", $\alpha \cup \beta$ as "do either α or β nondeterministically", $\alpha;\beta$ as "do α followed by β". With "λ", ";", "$\cup$", "?" we can thus construct complex actions that we call *plans*. We identify finite sets of formulas with the conjunction of its elements.

3 What is reasoning about actions?

In this section we state the terminology of reasoning about actions in terms of the above modal language. We illustrate our language and our logic throughout the paper by the classical Yale shooting scenario [27].

3.1 Belief bases, queries, and domain laws

We can describe in our modal language three main concepts: knowledge bases, queries, and domain laws.

Knowledge bases (KB) are finite sets of formulas without the $\Box$-operator. Usually they represent the current state of affairs, such as $\{Walking, Loaded\}$. They may as well contain information about results of action instances (alias observations), such as $[shoot]Alive$ in the Stanford murder mystery.

Queries are formulas whose derivability from the knowledge base under the domain laws is checked. An example is $[shoot]\neg Alive$ in the Yale shooting scenario. Particular queries are *goals*, which are formulas of the form $\Diamond A$, where A is classical. This kind of formula appears when we want to prove that there is a plan leading to a state of affairs where A is true. For example $\Diamond Dead$, $\Diamond(Alive \wedge \neg Loaded)$.

Domain laws (or non-logical axioms), noted LAW, are finite sets of formulas of the form $\Box A$, representing general laws about the world. We classify them into two categories: $\Box A$ is a *static law* (alias integrity constraint or state constraint) if A is a classical formula, and it is a *dynamic law* otherwise. Static laws are intended to state that A is always true, independently of any action occurrence. For example $\Box(Alive \leftrightarrow \neg Dead)$ and $\Box(Walking \rightarrow Alive)$. Dynamic laws are intended to describe actions laws.

We distinguish several sub-categories of dynamic laws:

- Laws of the form $\Box(B \rightarrow [\alpha]C)$ are called *effect laws*. B is the *precondition of* α and C is the *postcondition* (or the *effect*) of α. Examples of effect laws are $\Box(Loaded \rightarrow [shoot]Dead)$, $\Box(\top \rightarrow [shoot]\neg Loaded)$. (The latter is identified with $\Box[shoot]\neg Loaded$.)

  Particular effect laws of the form $\Box(A \rightarrow [\alpha]A)$ assert that α does not influence the truth of A. These are the so-called *frame axioms*. An example is $\Box(Alive \rightarrow [wait]Alive)$. *Conditional frame axioms* are of the form $\Box((B \wedge A) \rightarrow [\alpha]A)$, e.g. $\Box((\neg Loaded \wedge Alive) \rightarrow [shoot]Alive)$.

  Another particular case of effect laws are *unexecutability laws*, which take the form $\Box(A \rightarrow [\alpha]\bot)$. They state that α cannot be executed in the context A. For example $\Box(\neg HasGun \rightarrow [shoot]\bot)$.

- Laws of the form $\Box(B \rightarrow \langle\alpha\rangle\top)$ are *executability laws*. They guarantee the possibility to execute α in context B. For example $\Box(HasGun \rightarrow \langle shoot\rangle\top)$.

- Other form of dynamic laws exist, and can be written in our language. In [2] there are general statements linking several situations, independently of any action. For example, $\Box(\neg Alive \rightarrow \Box\neg Alive)$ (no action can make a dead man alive). They were called *immutable propositions* in [11]. As well, we can constrain the execution order. For example $\Box(Loaded \rightarrow [shoot]\Box[forcewalk]\bot)$ expresses that the action *forcewalk* can never be executed after *shoot* if the gun is loaded.

3.2 Main problems

If we try to reason about actions in a logical system, three main problems arise:

1. The *frame problem* [47] is that the overwhelming part of the dynamic laws are frame axioms, due to the hypothesis of inertia. For example after the action of loading the gun, the man is still alive. According to [47], we should not write down the frame axiom $\Box(Alive \rightarrow [load]Alive)$, but rather *deduce* it in an appropriate formal system.

2. The *ramification problem* [16] is that it is difficult to compute the indirect consequences of an action. For example after the *shoot* action, the direct consequences are $\neg Loaded$ and $\neg Alive$, but we must also conclude $\neg Walking$ as a ramification. It has been argued that it is undesirable to explicitly write down effect laws $\Box(A \rightarrow [\alpha]C)$ if C is an indirect effect of α, and that such indirect effects should be derived from the basic effect law $\Box(A \rightarrow [\alpha]C_0)$ together with some static laws relating C and C_0.

3. The *qualification problem* [46] is that it is difficult to write down executability laws. The reason is that the number of preconditions of an action is huge (e.g., the preconditions of the action of starting a car). Moreover, it may be undesirable or practically infeasible to check the truth of each of them. It is usually considered that we should derive the executability of an action by default, i.e., in the absence of information contradicting its executability. Hence mechanisms of nonmonotonic inference are mandatory here.

In this paper we mainly focus on the frame problem. Part II will be dedicated to the ramification problem. We stress that, contrarily to the qualification problem, these two problems do not require *a priori* nonmonotonic inference systems. Our solution will be monotonic, leaving the way open for a solution to the qualification problem by the addition of nonmonotonic features.

4 A basic modal logic of actions and plans

In this section we present the logic of actions and plans $\mathcal{LAP}$, which is a normal multimodal logic made up with a family of operators $[\alpha]$, where α is the name of an atomic action, plus a single non-indexed operator $\Box$. The logic of each $[\alpha]$ is the basic modal logic K, while that of $\Box$ is $S4$. $\Box$ interacts with every $[\alpha]$ in the sense that $\Box A \rightarrow [\alpha]A$. For further details about modal logic we refer the reader to [49].

4.1 Semantics of $\mathcal{LAP}$

Definition 1 We define models for $\mathcal{LAP}$ ($\mathcal{LAP}$-models) as quadruples of the form $\mu = \langle W, \{R_\alpha : \alpha \in ACT\}, R_\Box, \tau\rangle$, where W is a set of possible worlds (or possible states), $R_\Box$ and each R_α are binary relations on W (called accessibility relations), and $\tau : ATM \longrightarrow 2^W$ is an interpretation of atoms. We require $R_\Box$ to be reflexive and transitive, and $R_\alpha \subseteq R_\Box$, for every $\alpha \in ACT$.

The truth conditions are as usual:

- $\models^\mu_w \top$;
- $\not\models^\mu_w \bot$;

- $\models^{\mu}_{w} P$ iff $w \in \tau(P)$, for $P \in ATM$;
- $\models^{\mu}_{w} \neg A$ iff $\not\models^{\mu}_{w} A$;
- $\models^{\mu}_{w} A \wedge B$ iff $\models^{\mu}_{w} A$ and $\models^{\mu}_{w} B$;
- $\models^{\mu}_{w} [\alpha]A$ iff $\forall w' \in W : wR_{\alpha}w'$ implies $\models^{\mu}_{w'} A$;
- $\models^{\mu}_{w} \Box A$ iff $\forall w' \in W : wR_{\Box}w'$ implies $\models^{\mu}_{w'} A$.

A formula A is true in a model $\mu = \langle W, \{R_{\alpha} : \alpha \in ACT\}, R_{\Box}, \tau\rangle$ if $\models^{\mu}_{w} A$ for every $w \in W$. A is $\mathcal{LAP}$-valid (denoted by $\models_{\mathcal{LAP}} A$) if A is true in every $\mathcal{LAP}$-model. A is satisfiable if $\models^{\mu}_{w} A$ for some $\mu = \langle W, \{R_{\alpha} : \alpha \in ACT\}, R_{\Box}, \tau\rangle$ and $w \in W$.

Example 2 (The Yale Shooting Scenario (YSS) [27], part I)
Consider a scenario where *wait*, *load* and *shoot* represent the actions of waiting, loading and shooting. Let the atoms *Loaded* and *Alive* represent "the gun is loaded" and "the man is alive". The effect of the load action is that the gun is loaded, and the effects of the shoot action are that the gun is unloaded and that the person is killed if the gun is loaded. This action has no effect if the gun is unloaded. The wait action has no effect at all. Initially, the gun is unloaded and the man is alive. In $\mathcal{LAP}$ this scenario is written as:

$$LAW = \left\{ \begin{array}{l} \Box\langle wait\rangle\top, \\ \Box\langle load\rangle\top, \\ \Box\langle shoot\rangle\top, \\ \Box(Loaded \rightarrow [wait]Loaded), \\ \Box(\neg Loaded \rightarrow [wait]\neg Loaded), \\ \Box(Alive \rightarrow [wait]Alive), \\ \Box(\neg Alive \rightarrow [wait]\neg Alive), \\ \Box[load]Loaded, \\ \Box(\neg Alive \rightarrow [load]\neg Alive), \\ \Box(Alive \rightarrow [load]Alive), \\ \Box[shoot]\neg Loaded, \\ \Box(Loaded \rightarrow [shoot]\neg Alive), \\ \Box(\neg Alive \rightarrow [shoot]\neg Alive), \\ \Box((\neg Loaded \wedge Alive) \rightarrow [shoot]Alive) \end{array} \right\};$$

$$KB = \{\neg Loaded, Alive\}.$$

We have $\models_{\mathcal{LAP}} (KB \wedge LAW) \rightarrow [load][wait][shoot](\neg Loaded \wedge \neg Alive)$. Note that we must put $\Box\langle load\rangle\top$, $\Box\langle wait\rangle\top$, and $\Box\langle shoot\rangle\top$ into the set LAW in order to express that the actions *load*, *wait*, and *shoot* are always executable.

4.2 Axiomatics and theoretical complexity

Our axiomatization of $\mathcal{LAP}$ is made up of the following axioms schemas and inference rule schemas:

PL all tautologies of Classical Propositional Logic

$K([\alpha])$ $[\alpha](A \to B) \to ([\alpha]A \to [\alpha]B)$

$K(\Box)$ $\Box(A \to B) \to (\Box A \to \Box B)$

$T(\Box)$ $\Box A \to A$

$4(\Box)$ $\Box A \to \Box\Box A$

$I(\Box, [\alpha])$ $\Box A \to [\alpha]A$

MP from A and $A \to B$ infer B

$RN(\Box)$ from A infer $\Box A$

From $RN(\Box)$ and $I(\Box, [\alpha])$ we can derive the inference rule schema $RN([\alpha])$: from A infer $[\alpha]A$. Moreover, from $4(\Box)$ and $I(\Box, [\alpha])$ we can prove $\Box A \to [\alpha_1][\alpha_2]\ldots[\alpha_n]A$, for every $n \geq 1$.

Theorem 3 The axiomatics of $\mathcal{LAP}$ is sound and complete w.r.t. the semantics.

Proof $RN([\alpha])$ can be derived in $\mathcal{LAP}$. Hence $\mathcal{LAP}$ is a normal multimodal logic of the Sahlqvist family [53], and thus sound and complete w.r.t. the class of $\mathcal{LAP}$-models.

The theoretical complexity of $\mathcal{LAP}$ is the same as that of PDL:

Theorem 4 In $\mathcal{LAP}$, the problem of satisfiability is EXPTIME-complete.[3]

Proof See appendix A.

4.3 Discussion

Our language is similar to that of PDL. For the sake of comparison, suppose that the set of atomic actions is finite, and that $ACT = \{\alpha_1, \ldots, \alpha_n\}$. $\Box A$ is intended to simulate $[(\alpha_1 \cup \ldots \cup \alpha_n)^*]A$. But $\Box A$ is weaker, in the sense that the $\mathcal{LAP}$ counterpart

$$\Box(A \to [(\alpha_1 \cup \ldots \cup \alpha_n)^*]A) \to (A \to \Box A)$$

of the induction axiom is not valid in $\mathcal{LAP}$. Semantically, this means that $(R_{\alpha_1} \cup \ldots \cup R_{\alpha_n})^*$ is only included in $R_\Box$, but does not equal it.

This makes a difference only if we want to go beyond the reasoning tasks of prediction and planning, and prove liveliness properties, as done e.g. in GOLOG [35]. Consider an action theory with only one action α, containing an effect law $\Box[\alpha]P$, and let P hold. Then it would follow from the induction axiom that $\Box P$ holds, while in $\mathcal{LAP}$ it is not possible to prove that static law.

The original *raison d'être* of the PDL *-operator is to model loop sentences of computer programs. In what concerns reasoning about actions, it is certainly useful if we want to construct complex plans involving iterations (see e.g. [13, 35]), and might be considered in future elaborations of our approach. Our weaker $\Box$ operator has some advantages:

[3] As pointed out to us by an anonymous referee, even if we consider the fragment of the language containing just effect and executability laws as defined in section 3, $\mathcal{LAP}$ continues to be EXPTIME-complete. This can be established via the correspondence between description logics and propositional dynamic logics [55, 12]. Using static laws of the form $\Box A$, where A is a classical formula, and dynamic laws of the form $\Box(P \to [\alpha]Q)$ and $\Box(P \to \langle\alpha\rangle\top)$, where P, Q are atomic propositions, it is possible to encode the description logic $\mathcal{ALU}$ + Axioms, which is EXPTIME-complete [5].

- $\mathcal{LAP}$ is compact and strongly complete, while PDL is not. (This follows from theorem 43 in appendix B.)
- A tableau method can be defined straightforwardly by combining the tableau rules for the logics K and $S4$. We omit a presentation here, because it is a particular case of the method presented in section 6 (see [9] for details).

Note nevertheless that we have established by the above theorem 4 that $\mathcal{LAP}$ and PDL still have the same theoretical complexity.

The proof of the complexity result uses that deductions with global assumptions in modal logic K can be mapped into $\mathcal{LAP}$ proofs. Indeed, formulas such as effect and static laws (which are both of the form $\Box A$) behave just as global assumptions in deductions. (This corresponds to a well-known property of modal logic $S4$, viz. $A \vdash B$ iff $\vdash \Box A \rightarrow B$.) We have preferred the introduction of $\Box$ over deduction with global assumptions, because $\Box$ permits to express more complex laws, such as so-called immutable propositions (cf. section 2), and most importantly goals.

We could have given to our modal operator $\Box$ the semantics of the universal modality [48]. The latter has the truth condition

- $\models^{\mu}_{w} \Box A$ iff $\forall w' \in W : \models^{\mu}_{w'} A$

This means that the accessibility relation associated to $R_{\Box}$ is universal: $R_{\Box} = W \times W$. It has been shown in [48] that such an operator has the logic $S5$. Given that the accessibility relation property modally defined by S5 is that of $R_{\Box}$ being an equivalence relation, it follows that the universal accessibility relation is not modally definable: there is no modal formula able to force every possible world to be accessible. In our case, as we do not consider converse actions, a 'forward-looking' universal modality is sufficient for our purposes. Indeed, it is more convenient here e.g. because it permits to express immutable propositions by $\Box(\neg Alive \rightarrow \Box\neg Alive)$, or something like irreversible actions, such as $\Box(Loaded \rightarrow [shoot]\Box\neg Walking)$.

As expected, to effectively reason about actions and plans in $\mathcal{LAP}$ is difficult due to the huge number of frame axioms. In other words, within $\mathcal{LAP}$, the frame problem is unsolved. In [13] there is a monotonic solution to the frame problem in the style of [50] and others, using converse and complement of actions, but we shall present in the next section another monotonic solution to the frame problem that is based on the notion of dependence.

5 Modal logic plus dependence

How can we adapt $\mathcal{LAP}$ to solve the frame problem? In this section we present our solution in terms of a dependence relation between actions and literals. L depends on α means that α may cause L, i.e., after the execution of α, L may become true. We constrain models of $\mathcal{LAP}$ by a dependence-based condition saying that whenever L is independent of α, the falsehood of L is preserved under R_{α}.

5.1 Dependence relations

Our solution to the frame problem is based on the following considerations: suppose we have a complete description of a domain, containing all effect laws, in particular all frame axioms. For every action α and literal L, whenever there is a frame axiom $\Box(L \to [\alpha]L)$ then the execution of α cannot flip L from true to false, in any situation. In this case we say that *α may not cause $\neg L$*, or *$\neg L$ is independent of α*. (Note that we need the hypothesis here that all frame axioms operate on literals. This hypothesis is implicit in all the formalisms in the literature. We were unable to find examples where frame axioms on conjunctions or disjunctions appear in a natural way.) The set of such pairs (α, L) makes up an *independence relation* between actions and literals. Then we drop all those frame axioms. We thus have replaced the huge number of frame axioms by a huge independence relation: the hypothesis of inertia says that "almost all" of the literals L are independent of a given action α. On the other hand, it is reasonable to expect that the dependence relation (the complement of the independence relation) is much smaller that the independence relation. So, the trick is to represent the independence relation by the dependence relation.

With this motivation, we are ready for the definition of dependence.

Definition 5 A *dependence relation* is a binary relation between actions and literals $\leadsto \subseteq ACT \times LIT$. The complement of $\leadsto$ is noted $\not\leadsto$ and is called an *independence relation.*

It will be convenient to understand $\alpha \leadsto \neg\neg P$ as $\alpha \leadsto P$.

When we write $\alpha \leadsto L$ we want to say: "*α may cause L*", and this expresses that the execution of action α *may change* the truth value of L from false to true. For example $shoot \leadsto \neg Alive$: there is a state of affairs (viz. when the gun is loaded) where the execution of *shoot* flips the truth value of $\neg Alive$ from false to true.

Hence our dependence notion does not *cause* change, but it only *allows* change. This is similar in spirit to Sandewall's occlusion, and different from other causality-based approaches such as those of [39, 44].

Note that $\leadsto$ is not in the object language, but in the metalanguage. In the next section we show how dependence fits into the modal framework.

5.2 Semantics of $\mathcal{LAP}_{\leadsto}$

Suppose some dependence relation is given. In order to satisfy our intuitions, we must restrict our $\mathcal{LAP}$-models in the following way: suppose that we are in a particular situation (alias possible world) w where P does not hold. First, suppose $\alpha \leadsto P$ is the case. Then we know that the execution of α *may* cause P, but we don't know whether it will do so this time. Hence both are possible: the execution of α may lead to a situation where P continues to be false, but it may as well lead to a situation where P becomes true. Now suppose $\alpha \not\leadsto P$ is the case. Then we know that the execution of α *may never* make P true. Therefore the execution of α will leave P false.

Definition 6 Let $\leadsto$ be a dependence relation. A model for $\mathcal{LAP}_{\leadsto}$ (a $\mathcal{LAP}_{\leadsto}$-model) is a $\mathcal{LAP}$-model $\mu = \langle W, \{R_\alpha : \alpha \in ACT\}, R_\Box, \tau\rangle$, such that whenever $wR_\alpha w'$ then we have for every α:

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

In this way, a $\mathcal{LAP}_{\rightsquigarrow}$*-model* is just a $\mathcal{LAP}$-model with a supplementary condition to deal with dependence between actions and literals: if α may not cause L then the execution of α leaves unchanged the truth value of L.

Given a dependence relation $\rightsquigarrow$, we say that A is $\mathcal{LAP}_{\rightsquigarrow}$-valid (noted $\models_{\mathcal{LAP}_{\rightsquigarrow}} A$) if A is true in all $\mathcal{LAP}_{\rightsquigarrow}$-models.

Remark 7 In the case where $\rightsquigarrow = ACT \times LIT$, the set of $\mathcal{LAP}_{\rightsquigarrow}$-models is exactly the set of $\mathcal{LAP}$-models. In the other extreme (and less interesting) case where $\rightsquigarrow = \emptyset$, actions have no effects and the models collapse: they only contain a single world.

Example 8 (YSS, part II)
We rewrite example 2 in $\mathcal{LAP}_{\rightsquigarrow}$:

$$\rightsquigarrow = \left\{ \begin{array}{l} load \rightsquigarrow Loaded, \\ shoot \rightsquigarrow \neg Loaded, \\ shoot \rightsquigarrow \neg Alive \end{array} \right\};$$

$$LAW = \left\{ \begin{array}{l} \Box\langle wait\rangle\top, \\ \Box\langle load\rangle\top, \\ \Box\langle shoot\rangle\top, \\ \Box[load]Loaded, \\ \Box[shoot]\neg Loaded, \\ \Box(Loaded \rightarrow [shoot]\neg Alive), \\ \Box((\neg Loaded \wedge Alive) \rightarrow [shoot]Alive) \end{array} \right\};$$

$$KB = \{\neg Loaded, Alive\}.$$

Note that LAW still contains the conditional frame axiom $\Box((\neg Loaded \wedge Alive) \rightarrow [shoot]Alive)$. This point will be discussed in section 9.

As before, we have

$$\models_{\mathcal{LAP}_{\rightsquigarrow}} (KB \wedge LAW) \rightarrow [load][wait][shoot](\neg Loaded \wedge \neg Alive).$$

The independence $wait \not\rightsquigarrow \neg Loaded$ warrants that the gun remains loaded after waiting.

5.3 Axiomatics and theoretical complexity of $\mathcal{LAP}_{\rightsquigarrow}$

For a given dependence relation $\rightsquigarrow$, we axiomatize the class of $\mathcal{LAP}_{\rightsquigarrow}$-models by the axiomatics of $\mathcal{LAP}$, plus a dependence-based axiom schema

$Persistence([\alpha])$ $\neg L \rightarrow [\alpha]\neg L$, if $\alpha \not\rightsquigarrow L$.

Theorem 9 For every dependence relation $\rightsquigarrow$, the axiomatics of $\mathcal{LAP}_{\rightsquigarrow}$ is sound and complete w.r.t. the class of $\mathcal{LAP}_{\rightsquigarrow}$-models.

Proof See appendix B.

Theoretical complexity of $\mathcal{LAP}_{\leadsto}$is the same as that of $\mathcal{LAP}$:

Theorem 10 In $\mathcal{LAP}_{\leadsto}$, the problem of satisfiability is EXPTIME-complete.

Proof $\mathcal{LAP}$ is just $\mathcal{LAP}_{\leadsto}$ with $\leadsto = \emptyset$. Hence the complexity of $\mathcal{LAP}_{\leadsto}$ is at least EXPTIME. To establish the upper bound we exhibit a polynomial transformation t from $\mathcal{LAP}_{\leadsto}$ to $\mathcal{LAP}$. Let $Indep(A, \alpha)$ be the set of those independences occurring in A, i.e. $Indep(A, \alpha) = \{(\alpha, L) : \alpha$ and L appear in A, and $\alpha \not\leadsto L\}$. Note that $Indep(A, \alpha)$ is finite, and that its size is quadratic in the size of A. Then

$$t(A) = A \wedge \bigwedge_{(\alpha,L) \in Indep(A,\alpha)} \Box(\neg L \to [\alpha]\neg L)$$

We note in passing that this also illustrates that the frame problem has nothing to do with theoretical complexity.

6 Tableaux for $\mathcal{LAP}_{\leadsto}$

In this section we present a tableau proof procedure for $\mathcal{LAP}_{\leadsto}$.

The first tentative of solution is to add to the $\mathcal{LAP}$-tableau rules a new rule that takes into account the semantics of the dependence relation when a new node is generated: whenever L is in some node of the tableau, if $\alpha \not\leadsto \neg L$, then put the literal L in all worlds accessible from that node via α. We call this the "stepforward"-rule. But this is not enough: suppose, for example, that $\alpha \not\leadsto P$ and $\beta \not\leadsto \neg P$, for some atom P and actions α, β. Then the formula $\langle\alpha\rangle P \wedge \langle\beta\rangle\neg P$ is $\mathcal{LAP}_{\leadsto}$-unsatisfiable, but the above rule is not sufficient to make the tableau close. Our solution is to add a special "stepback"-rule, which permits to propagate independent literals backwards.[4] Such a rule is somewhat opposed to the usual top-dawn tableau construction (which mirrors Gentzen's sequent calculi). For a more general application of stepback rules see [43, 6].

The tableau calculus we are about to present is based on labelled formulas. We begin with some terminology:

Definition 11 A *labelled formula* is a pair (n, A), where A is a formula and n is an integer called the *label* of A.

A label denotes a possible world. (Of course, any infinite set of labels would do.)

Definition 12 A *skeleton*[5] Σ is a ternary relation $\Sigma \subseteq (ACT \cup \Box) \times \mathbb{N} \times \mathbb{N}$. Elements (α, n, n') of the relation are noted $n \stackrel{\alpha}{\to} n'$.

Definition 13 A *tree*[6] is a pair $\langle S, \Sigma\rangle$, where S is a set of labelled formulas and Σ is a skeleton.

[4] There is another option if dependence is defined between actions and atoms (rather than between actions and literals as in the present paper). This has been explored in [7], where it is shown that then there exists a normal form for formulas such that satisfiability for formulas in normal form reduces to satisfiability in $\mathcal{LAP}$.

[5] A skeleton intuitively corresponds to the accessibility relations.

[6] A tree intuitively corresponds to a model.

Tableau rules applicable to $\langle S, \Sigma\rangle$

rule $\bot$: If S contains (n, A) and $(n, \neg A)$, then add $(n, \bot)$ to it.

rule $\neg$: If S contains $(n, \neg\neg A)$, then add (n, A) to it.

rule $\wedge$: If S contains $(n, A \wedge B)$, then add (n, A) and (n, B) to it.

rule $\vee$: If S contains $(n, \neg(A \wedge B))$, then add $(n, \neg A)$ to it, and add $\langle S \cup \{(n, \neg B)\}, \Sigma\rangle$ to C^{i+1}.

rule K_α: If S contains $(n, [\alpha]A)$, and Σ contains $n \xrightarrow{\alpha} n'$, then add (n', A) to it.

rule $T_\Box$: If S contains $(n, \Box A)$, then add (n, A) to it.

rule $4_\Box$: If S contains $(n, \Box A)$, and Σ contains $n \xrightarrow{\Box} n'$ or $n \xrightarrow{\alpha} n'$ for some $\alpha \in ACT$, then add $(n', \Box A)$ to it.

rule $\langle\alpha\rangle$: If S contains $(n, \neg[\alpha]A)$, then add $(n', \neg A)$ to it, and add $n \xrightarrow{\alpha} n'$ to Σ, where n' is a fresh integer.

rule $\Diamond$: If S contains $(n, \neg\Box A)$, then add $(n', \neg A)$ to it and add $n \xrightarrow{\Box} n'$ to Σ, where n' is a fresh integer.

rule SF: If S contains (n, L), where L is a literal, and $n \xrightarrow{\alpha} n' \in \Sigma$, and $\alpha \not\rightsquigarrow \neg L$, then add (n', L) to it.

rule SB: If S contains (n', L), where L is a literal, and $n \xrightarrow{\alpha} n' \in \Sigma$, and $\alpha \not\rightsquigarrow L$, then add (n, L) to it.

Figure 1: Tableau rules

Definition 14 A *tableau* for A is the limit[7] of a sequence $C^0, \ldots, C^n, \ldots$ of forests (i.e., sets of trees) where the initial $C^0 = \{\langle\{0, A\}, \emptyset\rangle\}$, and every C^{i+1} is obtained from C^i by applying one of the tableau rules of figure 1 to some tree $\langle S, \Sigma\rangle \in C^i$.[8] This limit is noted C^∞.

Note that for each tree $\langle S, \Sigma\rangle$ in that forest, all the sets are downward closed in the usual sense [24].

Definition 15 A tree is $\langle S, \Sigma\rangle$ *closed* if $(i, \bot) \in S$ for some i. A forest is closed if all of its trees are closed. (Hence, a particular tableau is closed if all of its trees are closed.)

Remark 16 The last two rules SF and SB are associated with the dependence relation: literals whose negation is independent of an action α remain unchanged when α is executed. Let us recall that in a $\mathcal{LAP}_\rightsquigarrow$-model, if $(wR_\alpha w'$ and α may not cause $\neg L$ (i.e., if $\alpha \not\rightsquigarrow \neg L$) then L is true at w implies that L must be true at w', too (this is rule SF); and conversely, if $\alpha \not\rightsquigarrow L$ then L is true at w' implies that L must be true at w, too (this is rule SB).

Remark 17 As said in the introduction of this section, without the stepback rule SB there is no closed tableau for the unsatisfiable formula $\langle\alpha\rangle P \wedge \langle\beta\rangle\neg P$ if $\alpha \not\rightsquigarrow P$ and $\beta \not\rightsquigarrow \neg P$. Hence SB is necessary for completeness. But SB seems to be useless in almost all practical problems (and in particular in the standard examples), the reason

[7] This limit exists since as we will see, all the rules either increase the set of labelled formulas or increase the successor relations (of course, it may be infinite). It is obtained in the obvious way by pointwise union.

[8] We recall that the connectives of our language are $\top$, $\bot$, $\neg$, $\wedge$, $\Box$, and $[.]$, and that the other connectives are abbreviations.

label	states			skeletons	explanation
1	$(0, \neg((KB \wedge LAW) \to [load][wait][shoot](\neg Loaded \wedge \neg Alive)))$			$\Sigma_\alpha = \Sigma_\Box = \emptyset$	
2a	$(0, \Box[load]Loaded)$				$(\wedge)$ on 1
2b	$(0, \Box[shoot]\neg Loaded)$				$(\wedge)$ on 1
2c	$(0, \Box(Loaded \to [shoot]\neg Alive))$				$(\wedge)$ on 1
2d	$(0, \neg[load][wait][shoot](\neg Loaded \wedge \neg Alive))$				$(\wedge)$ on 1
3	$(0, [load]Loaded)$				$(T\Box)$ on 2a
4	$(1, \neg[wait][shoot](\neg Loaded \wedge \neg Alive))$			$0 \stackrel{load}{\to} 1$	$(\langle load \rangle)$ on 2d
5	$(2, \neg[shoot](\neg Loaded \wedge \neg Alive))$			$1 \stackrel{wait}{\to} 2$	$(\langle wait \rangle)$ on 4
6	$(3, \neg(\neg Loaded \wedge \neg Alive))$			$2 \stackrel{shoot}{\to} 3$	$(\langle shoot \rangle)$ on 5
7	$(1, Loaded)$				(K_{load}) on 3
8	$(2, Loaded)$				(SF) on 7
9	$(2, \Box[shoot]\neg Loaded)$				$(4\Box)$ on 2b
10	$(2, \Box(Loaded \to [shoot]\neg Alive))$				$(4\Box)$ on 2c
11	$(2, [shoot]\neg Loaded)$				$(T\Box)$ on 9
12	$(2, Loaded \to [shoot]\neg Alive)$				$(T\Box)$ on 10
13	$(2, \neg Loaded)$	$(2, [shoot]\neg Alive)$			$(\vee)$ on 12
14	$(2, \bot)$	$(3, \neg\neg Loaded)$	$(3, \neg\neg Alive)$		$(\bot)$ on 8 and 13; $(\vee)$ on 6
15	closed	$(3, \neg Loaded)$	$(3, \neg Alive)$		(K_{shoot}) on 11 ; (K_{shoot}) on 13
16		$(3, \bot)$	$(3, \bot)$		$(\bot)$ on 14 and 15 ; idem
17		closed	closed		all tableaux are closed

Figure 2: A $\mathcal{LAP}_{\rightsquigarrow}$-tableau proof of $(KB \wedge LAW) \to [load][wait][shoot](\neg Loaded \wedge \neg Alive)$.

being that it is not natural to describe an action α by formulas like $[\alpha]A$ or $\langle\alpha\rangle A$ if $\alpha \not\rightsquigarrow A$. In other words, we expect that in the set of domain laws LAW, all literals in the scope of some action α depend on α.

Remark 18 The rule application order is irrelevant in the sense that if at some stage some rule is applicable and is not applied, it remains applicable.

Example 19 (YSS, part III)
Continuing example 8, we give in figure 2 a tableau proof[9] of

$$\models_{\mathcal{LAP}_{\rightsquigarrow}} (KB \wedge LAW) \to [load][wait][shoot](\neg Loaded \wedge \neg Alive).$$

For an example where the query is a goal, see example 24 in the next section 7.

Theorem 20 (Completeness) If there is a fair open tableau for A then A is satisfiable.

Proof See appendix C.1.

Theorem 21 (Soundness) If all tableaux for A are closed then A is unsatisfiable.

[9] The information common to all the trees of the tableau is factorized for the sake of clarity (in particular, the common part of the skeletons). We denote "add (n, n') to Σ_α" by $n \stackrel{\alpha}{\to} n'$ in the 'skeletons'-column. Note how $Loaded$ is transported when rule SF is applied in line 8 due to $wait \not\rightsquigarrow \neg Loaded$.

Proof See appendix C.2.

Remark 22 (Termination and decidability) As the formulas generated by the tableau rules are subformulas of the initial formula, there is a finite number of distinct states. Using a loop test as for S4 [18] we can associate to our initial tableau a finite tableau from which a finite model can be extracted.

7 Plan generation

Suppose that $\Diamond G$ is some query (or goal), and suppose that we can prove $(KB \wedge LAW) \rightarrow \Diamond G$. Our reading of the $\Diamond$-operator being that there exists a plan to achieve G, we shall justify that claim in the present section by constructing a complex action π such that $\vdash_{\mathcal{LAP}_{\leadsto}} (KB \wedge LAW) \rightarrow \langle \pi \rangle G$.

This is not always possible. The reason is that the accessibility relation $R_\Box$ associated to $\Box$ may be bigger than $(\bigcup_{\alpha \in ACT} R_\alpha)^*$ (see the discussion in section 4.3). Intuitively, if there is some negative occurrence of the $\Box$-operator, then one can "jump" to another world without identifying any specific action. (This would be a 'magic action'.) We shall show that this cannot be the case if KB and LAW are of a particular form. What we need are formulas that do not contain negative occurrences of the $\Box$-operator: if all the transitions in tableaux are only triggered by $\neg[\alpha]$ (and not by $\neg\Box$), then a plan can be found by the composition of all actions that have been applied from the initial node to some closed node. This fragment is large enough for our reasoning tasks: there is no negative occurrence of $\Box$ in the categories of knowledge bases and domain laws defined in section 3. This gives supplementary support to our claim that the *-operator of PDL is not needed for the standard tasks in reasoning about actions and plans.

Let $\{\langle S^1, \Sigma^1 \rangle, \langle S^2, \Sigma^2 \rangle, \ldots, \langle S^n, \Sigma^n \rangle\}$ be a closed tableau for $KB \wedge LAW \wedge \Box\neg G$. (Remember that the tableau construction is finite.) Hence every tree in the tableau is closed. First of all we eliminate all those trees that can be closed without involving a subformula of $\Box\neg G$. For each of the remaining trees $\{\langle S^{i_1}, \Sigma^{i_1} \rangle, \ldots, \langle S^{i_n}, \Sigma^{i_n} \rangle\}$ we extract a finite path from the root to some closed node. This is the plan associated to that tree. Then the plan associated to the tableau is the nondeterministic choice $\pi_{i_1} \cup \ldots \cup \pi_{i_m}$ of all these plans.

Example 23 (The YSS, part IV)
With LAW and KB of example 19, we can as well prove that

$$\models_{\mathcal{LAP}_{\leadsto}} (KB \wedge LAW) \rightarrow \Diamond\neg Alive,$$

i.e., there exists a plan to achieve $\neg Alive$. The shortest proof of that gives us the plan $\pi = load; shoot$. The plan $\pi' = load; wait; shoot$ corresponds to a longer proof.

Example 24 (Nondeterministic plans)
Let Up mean that some switch is up, and let $toggle$ be an action described by

$$\leadsto = \left\{ \begin{array}{l} toggle \leadsto Up, \\ toggle \leadsto \neg Up \end{array} \right\};$$

label:	states		skeletons	explanation
1	$(0, \neg(LAW \to \Diamond Up))$		$\Sigma_\alpha = \Sigma_\Box = \emptyset$	
2a	$(0, \Box\neg[toggle]\bot)$			$(\wedge)$ on 1 (modulo normal forming)
2b	$(0, \Box(Up \vee [toggle]Up))$			$(\wedge)$ on 1 (modulo normal forming)
2c	$(0, \Box\neg Up)$			$(\wedge)$ on 1 (modulo normal forming)
3	$(0, \neg[toggle]\bot)$			$(T\Box)$ on 2a
4	$(0, Up \vee [toggle]Up)$			$(T\Box)$ on 2b
5	$(0, \neg Up)$			$(T\Box)$ on 2c
6	$(0, Up)$	$(0, [toggle]Up)$		$(\vee)$ on 4
7	$(0, \bot)$	$(1, \neg\bot)$	$0 \overset{toggle}{\to} 1$	$(\bot)$ on 5 and 6;$(\langle toggle\rangle)$ on 3
8	closed	$(1, Up)$		(K_{toggle}) on 6
9		$(1, \Box\neg Up)$		$(4\Box)$ on 2c
10		$(1, \neg Up)$		$(T\Box)$ on 9
11		$(1, \bot)$		$(\bot)$ on 8 and 10
11		closed		all tableaux are closed

Figure 3: A $\mathcal{LAP}_{\leadsto}$-tableau proof of $LAW \to \Diamond Up$.

$$LAW = \left\{ \begin{array}{l} \Box\langle toggle\rangle\top, \\ \Box(Up \to [toggle]\neg Up), \\ \Box(\neg Up \to [toggle]\, Up) \end{array} \right\}.$$

A tableau proof of $\models_{\mathcal{LAP}_{\leadsto}} LAW \to \Diamond Up$ is given in figure 3. The associated plan is $\pi = \lambda \cup toggle$.

This is an example of a nondeterministic plan. Generally, such plans seem to be of little use, because they do not allow us to effectively act in order to obtain the goal. Nevertheless it is possible to generate a deterministic plan from the tableau by generating an if-then-else construction when rule $(\vee)$ is applied: thus the plan if Up then λ else *toggle* can be found for the above example. This has been done in [7]. In this view, the plan our algorithm returns can be considered a first step towards the construction of a more elaborated plan. We shall not go into the details here.

Note also that in the above examples, the plan has been associated to the shortest tableau proof. There might be longer ones, which give us longer plans, such as $\pi' = load; wait; shoot$ or $\pi'' = load; wait; shoot; wait$ in the first YSS example, and $\pi' = toggle \cup (toggle; toggle)$ in the second example.

Theorem 25 Let neither KB nor LAW nor $\neg G$ contain negative occurrences of $\Box$. Let $KB \wedge LAW \wedge \Box\neg G$ be $\mathcal{LAP}_{\leadsto}$-unsatisfiable, and let π be a plan that has been found by the plan generation algorithm. Then $KB \wedge LAW \wedge [\pi]\neg G$ is $\mathcal{LAP}_{\leadsto}$-unsatisfiable.

Proof The proof is done by replacing $\Box\neg G$ by $[\pi]\neg G$ in the tableau. Then the same sequence of tableau rules as in the original tableau can be applied, and we can prove by induction that the latter leads again to a closed tableau. The only non-trivial case is that of rule $(\vee)$ decomposing $S = \{(n, \neg(A \wedge B))\} \cup S'$ into $S_1 = \{(n, \neg A)\} \cup S'$ and $S_2 = \{(n, \neg B)\} \cup S'$. Both of these trees must be closed. Things are easy if closure of one or both of these does not involve a subformula of $\Box\neg G$. Hence suppose both are closed because of $\Box\neg G$, and let π_1 and π_2 be the respective plans associated to them. We can suppose that S' contains $(n, \Box\neg G)$, because rule $4_\Box$ is always applicable. By induction hypothesis we can replace all occurrences of $\Box\neg G$ in S_1 by $[\pi_1]\neg G$, and in S_2 by $[\pi_2]\neg G$, while keeping closure. Hence we keep closure if we replace all occurrences of $\Box\neg G$ in S by $[\pi_1 \cup \pi_2]\neg G$.

8 Translation of action logic $\mathcal{A}$ into $\mathcal{LAP}_{\leadsto}$

In this section we give a sound and complete translation of the language $\mathcal{A}$ [19] into $\mathcal{LAP}_{\leadsto}$. Via the embeddings established in [32] we have thus a formal link to several other approaches.

8.1 Gelfond and Lifschitz' action logic $\mathcal{A}$

In $\mathcal{A}$ there are two kinds of propositions: value and effect propositions. A *value proposition* represents factual knowledge (observations). It gives the truth value of some atom after some sequence of actions, and has the form L after $\alpha_1; \alpha_2; \ldots; \alpha_m$, where L is a literal, and $\alpha_1, \alpha_2, \ldots, \alpha_m$ are atomic actions. For example to represent the fact that somebody was dead after execution of the shooting action, we write $\neg \mathit{Alive}$ after shoot.

Abbreviating the complex action $\alpha_1; \alpha_2; \ldots; \alpha_m$ by π, we shall write L after π for short. If $m = 0$ then the sequence of actions is λ, and L is true initially (also written $\mathsf{initially}$ L). VP is the set of all value propositions. For example, in the YSS (example 2) the set of all value proposition is:

$$VP = \left\{ \begin{array}{l} \mathsf{initially}\ \neg \mathit{Loaded}, \\ \mathsf{initially}\ \mathit{Alive} \end{array} \right\}.$$

Effect propositions describe the effects of actions and have the form α causes L if A, where α is an action name, L is a literal and A is a conjunction of literals. For example, we write shoot causes $\neg \mathit{Alive}$ if Loaded to state that shooting with a loaded gun kills someone.

α causes L if $\top$ is abbreviated α causes L. EP_α is the set of effect propositions for action α, and $EP = \bigcup_{\alpha \in ACT} EP_\alpha$ is the set of all effect propositions. For example, in the YSS the set of all effects propositions is:

$$EP = \left\{ \begin{array}{l} \mathit{load}\ \mathsf{causes}\ \mathit{Loaded}, \\ \mathit{shoot}\ \mathsf{causes}\ \neg \mathit{Loaded}, \\ \mathit{shoot}\ \mathsf{causes}\ \neg \mathit{Alive}\ \mathsf{if}\ \mathit{Loaded} \end{array} \right\}.$$

The set $EP \cup VP$ is called a *domain description.*

The semantics of $\mathcal{A}$ is in terms of a mapping Φ associating to every atomic action α a total transition function Φ_α on classical interpretations. Complex actions $\alpha_1; \ldots; \alpha_n$ are interpreted in the natural way as $\Phi_{\alpha_1;\ldots;\alpha_m} = \Phi_{\alpha_1} \circ \ldots \circ \Phi_{\alpha_m}$. The function Φ is an $\mathcal{A}$-model of EP_α if the following conditions are satisfied for every classical interpretation v and literal L:

- if there is an effect proposition α causes L if A in EP_α such that v satisfies[10] A, then $\Phi_\alpha(v)$ satisfies L (conditions (i) and (ii) in [19]);
- else v satisfies $\neg L$ implies $\Phi_\alpha(v)$ satisfies $\neg L$ (condition (iii) in [19]).

Φ is an $\mathcal{A}$-model of EP if Φ is an $\mathcal{A}$-model of EP_α, for every $\alpha \in ACT$. A value proposition L after π is $\mathcal{A}$-true at a classical interpretation v of Φ if $\Phi_\pi(v)$ satisfies L. A set of value propositions VP is $\mathcal{A}$-true at a classical interpretation v of Φ if every element of VP is true at v in Φ, According to [19], there can be at most one function Φ satisfying the two conditions above.

A domain description $EP \cup VP$ $\mathcal{A}$-entails a value proposition L after π if for every model Φ of EP and every classical interpretation v, if VP is $\mathcal{A}$-true at v then L after π is $\mathcal{A}$-true at v. For example the above YSS domain description $\mathcal{A}$-entails $\neg Alive$ after $shoot$.

8.2 The translation

Suppose EP and VP are given. We suppose that both are finite. VP is translated into a knowledge base KB_{VP}, and EP is translated into a set of effect laws LAW_{EP} and a dependence relation $\leadsto_{EP}$. (We drop the subscripts for convenience.)

1. The translation of a value proposition L after π is $[\pi]L$. KB is the set of all these formulas.
2. The dependence relation $\leadsto$ relates all those α and L which appear in some effect proposition: $\alpha \leadsto L$ iff there is α causes L if A in EP such that A is classically consistent.
3. The associated effect laws LAW are:

   (a) $\Box\langle\alpha\rangle\top$ for every action $\alpha \in ACT$;

   (b) $\Box(A \to [\alpha]L)$, for each α causes L if A in EP;

   (c) $\Box(\neg Pre(\alpha, L) \to (\neg L \to [\alpha]\neg L))$ for each α and L such that $Pre(\alpha, L)$ is different from $\bot$.

   $Pre(\alpha, L) = \bigvee_{(\alpha \textsf{ causes } L \textsf{ if } A_j) \in EP} A_j$ is the disjunction of all preconditions A_j for which L is the effect of α. (If there is no effect proposition for α with effect L then $Pre(\alpha, L)$ is $\bot$.)

Clause (a) says that in $\mathcal{A}$ actions are always executable.[11] (b) is straightforward. Clause (c) translates the inertia principle in $\mathcal{A}$: there are conditional frame axioms for all literals L depending on α, under the condition that none of the preconditions of L holds.

Remark 26 The size of the translation is linear in the cardinality of $EP \cup VP$.

[10] Satisfaction is understood here as satisfaction in classical logic.

[11] Without (a), there could be a $\mathcal{LAP}_{\leadsto}$-model for the translation of $\{P$ after $\alpha, \neg P$ after $\alpha\}$ although it is $\mathcal{A}$-false for all Φ and v. Liberatore [36] has investigated how unexecutability of actions could be integrated into $\mathcal{A}$.

Theorem 27 Let EP be a set of effect propositions such that there is an $\mathcal{A}$-model of EP, and let VP be a set of value propositions of $\mathcal{A}$. Let KB, $\leadsto$ and LAW be their translation. Then $EP \cup VP$ $\mathcal{A}$-entails a value proposition L after π if and only if $\models_{\mathcal{LAP}_{\leadsto}} (KB \wedge LAW) \to \langle \pi \rangle L$.

Proof See appendix D.

Remark 28 As actions in $\mathcal{A}$ are deterministic, we have as well that $EP \cup VP$ $\mathcal{A}$-entails L after π iff $\models_{\mathcal{LAP}_{\leadsto}} (KB \wedge LAW) \to [\pi] L$.

Remark 29 The theorem does not hold if there is no $\mathcal{A}$-model of EP: consider the set $EP = \{\alpha \text{ causes } Q \text{ if } P, \alpha \text{ causes } \neg Q \text{ if } P\}$. EP has no $\mathcal{A}$-model. Consequently EP $\mathcal{A}$-entails any value proposition, e.g., P after α. Nevertheless, its translation $LAW = \{\Box\langle\alpha\rangle\top, \Box(P \to [\alpha]Q), \Box(P \to [\alpha]\neg Q)\}$ is equivalent to $\Box\langle\alpha\rangle\top \wedge \Box\neg P$ and is satisfiable. In particular it does not imply $\langle\alpha\rangle P$.

Example 30 (YSS, part V)
The YSS domain (as described by VP and EP above) is mapped into $\mathcal{LAP}_{\leadsto}$ as follows:
The translation of VP is the knowledge base

$$KB = \{\neg Loaded, Alive\}.$$

The translation of EP gives the following dependences and domain laws:

$$\leadsto = \left\{ \begin{array}{l} load \leadsto Loaded; \\ shoot \leadsto \neg Loaded; \\ shoot \leadsto \neg Alive \end{array} \right\};$$

$$LAW = \left\{ \begin{array}{l} \Box\langle load\rangle\top, \\ \Box\langle wait\rangle\top, \\ \Box\langle shoot\rangle\top, \\ \Box[load]Loaded, \\ \Box[shoot]\neg Loaded, \\ \Box(Loaded \to [shoot]\neg Alive), \\ \Box((\neg Loaded \wedge Alive) \to [shoot]Alive)) \end{array} \right\}.$$

Note that in $\mathcal{A}$ nothing is stated about the wait action. Hence $wait \not\leadsto L$ for every literal L. But, as in $\mathcal{A}$ all actions are executable, we must add $\Box\langle wait\rangle\top$ to LAW. Thus we have the same set of dependences and domain laws as introduced in example 8.

The formula $(KB \wedge LAW) \to \langle load\rangle\langle wait\rangle\langle shoot\rangle(\neg Loaded \wedge \neg Alive)$ is valid in $\mathcal{LAP}_{\leadsto}$.

9 Discussion

We have formally assessed our solution to the frame problem w.r.t. Gelfond and Lifschitz' action logic $\mathcal{A}$. In this section we go beyond and briefly discuss the main characteristics of our approach and compare with previous work.

9.1 Monotonic solution

Why do we call our solution monotonic?

Shanahan argues in [58, section 1.10] that the distinction between monotonic and nonmonotonic approaches does not hold water (see also [15]). He points out that even the so-called monotonic approaches such as Reiter's make use of nonmonotonic reasoning, in the following sense: let e.g. $Exp(LAW)$ be the explanation closure of some action theory LAW. Then instead of considering the (clearly monotonic) derivation $Exp(LAW) \vdash A$, Shanahan rather examines the thus defined inference relation $LAW \rhd A$, and he shows that $\rhd$ is nonmonotonic: we may have $Exp(LAW) \vdash A$ without having $Exp(LAW \wedge B) \vdash A$.

Our solution to the frame problem is truly monotonic in Shanahan's sense: $LAW \rhd A$ is defined in our case as $\models_{\mathcal{LAP}_{\leadsto}} LAW \to A$. Then $LAW \rhd A$ implies $LAW \wedge B \rhd A$.[12]

9.2 Conditional frame axioms

In the preceding YSS example we were obliged to explicitly state

$$\Box((\neg Loaded \wedge Alive) \to [shoot]Alive).$$

Such a conditional frame axiom is not necessary in other formalizations of that scenario, such as in $\mathcal{A}$ (cf. section 8). Why do we need it here?

Suppose all you are told about some action α is that it always has effect P if A holds. This is written $\Box(A \to [\alpha]P)$ in our modal language. What do you conclude about the execution of α in situations where $\neg A$ holds? There are two possibilities:

- You do not conclude anything.
- You conclude that P can never be caused by α in this case, i.e. you conclude $\Box((\neg A \wedge \neg P) \to [\alpha]\neg P)$.

The second possibility relies on a particular hypothesis, that has been called 'explanation closure assumption' in Reiter's approach [50]. Indeed, his explanation closure axiom $(\neg F(\vec{x}, s) \wedge F(\vec{x}, do(a, s))) \to \gamma_F^+(\vec{x}, a, s)$ can be put into the form of a conditional frame axiom: $(\neg\gamma_F^+(\vec{x}, a, s) \wedge \neg F(\vec{x}, s) \to \neg F(\vec{x}, do(a, s)))$.

So we could automatically generate such axioms in a way similar to Reiter's. We chose not to build that in our logic, but rather to add conditional frame axioms 'by hand' in order to get a more flexible framework. In particular we think that our account is better suited when there are indeterminate and indirect effects. In the second part of the paper we shall study algorithms to semi-automatically compute conditional frame axioms.

The next example is a sibling of Sandewall's Russian Turkey Scenario. It illustrates that our choice is adequate at least in domains with incomplete knowledge, where the explanation closure assumption cannot be made.

[12] Well, the reason we stay monotonic is that we keep constant the underlying dependence relation. Indeed, in $\mathcal{LAP}_{\leadsto}$ knowledge representation is in terms of dependence, while reasoning is in terms of independence. As we use a sort of negation by failure (or rather a simple version of it, viz. nonmembership in a set) our inference relation is nonmonotonic w.r.t. dependences. Therefore, if we considered an inference relation $\langle \leadsto, LAW \rangle \rhd A$ instead of $LAW \rhd A$ then $\rhd$ would be nonmonotonic.

Example 31 (Toulouse shooting scenario (TSS))
Consider a revolver with a cylinder containing at most six bullets. If the cylinder is full then shooting results in killing the turkey. If the cylinder is not full then shooting may either kill or not. The formalization in $\mathcal{LAP}_{\leadsto}$ is:

$$\leadsto = \left\{ \begin{array}{l} load \leadsto Full, \\ shoot \leadsto \neg Full, \\ shoot \leadsto \neg Alive \end{array} \right\};$$

$$LAW = \left\{ \begin{array}{l} \Box\langle wait\rangle\top, \\ \Box\langle load\rangle\top, \\ \Box\langle shoot\rangle\top, \\ \Box[load]Loaded, \\ \Box[shoot]\neg Full, \\ \Box(Full \to [shoot]\neg Alive) \end{array} \right\}.$$

Here explanation closure of LAW would make us conclude

$$\Box((\neg Full \wedge Alive) \to [shoot]Alive),$$

which is not intended. (In fact, concepts such as possibly changes of $\mathcal{AR}$ have been proposed in the literature in order to block the automatic generation of such formulas.)

If we identify *Full* with *Loaded*, then the present scenario is the same as the YSS, except that in the YSS we know more in the case where the gun is not loaded. Our representation in $\mathcal{LAP}_{\leadsto}$ conforms to that: we get from the TSS to the YSS by adding

$$\Box(Full \to [shoot]\neg Alive).$$

9.3 Executability laws

Our solution forces us to add executability laws. (Consider e.g. example 8, $\Box\langle wait\rangle\top$, $\Box\langle load\rangle\top$ and $\Box\langle shoot\rangle\top$ explicitly appeared as domain laws.) We just recall that many approaches in the domain adopt the same solution, in particular, the situation calculus based approach of Reiter and colleagues, with their predicate $Poss$.

On the other hand, it might be desirable to automatically generate executability laws. This was done before, e.g. by Lifschitz, minimizing the predicate $Precond$ [37]. It amounts to assume that actions are maximally executable, as done explicitly in [21]. Under this hypothesis we can automatically generate executability laws from the static laws in conjunction with action laws. This will be explored in the second part of the paper.

9.4 Indirect effects

Dependence is a weak causal connection between actions and literals. We are thus close to causality-based approaches to the ramification problem. We refer here to recent work, where causality was formalized by means of a predicate [39], a modality [56], as inference rules [44], or by the addition of influence information [61].

Do we solve the ramification problem in $\mathcal{LAP}_{\leadsto}$? First of all, in $\mathcal{LAP}_{\leadsto}$ we can write down static laws, and when such laws can be handled then it is usually considered that the ramification problem is solved.

But, despite the satisfactory implementation of static laws, this is not the whole story in our case. Indeed, although we avoid writing down indirect effects, we are obliged to write down indirect dependences. For example in the YSS if there is a supplementary static law $\Box(\mathit{Walking} \rightarrow \mathit{Alive})$ [1], then we must add an indirect dependence $\mathit{shoot} \leadsto \neg \mathit{Walking}$.

Our account of the frame problem in terms of dependence is closely related with Haas and Schubert's explanation closure approach: if we look at dependence from right to left, then we can deduce that the only action that can flip the value of *Alive* from true to false is the action *shoot*. In this sense, the criticisms of Schubert's work also apply to ours.

One indication that things must be done in this way is the fact that approaches that automatically generate the explanation closure only apply to restricted classes of action theories. For example, Reiter's approach [50] is limited to domains with only deterministic actions without ramification, while that of Doherty and colleagues [15] is limited to domains without ramification (allowing thus for indeterminate effects).

Another argument is that explicit indirect dependences allow to handle examples which turn out to be problematic for other approaches to the ramification problem, as illustrated in the sequel.

Example 32 (Forcing a door)
We consider two actions, with the same effect of opening a door. The first action *open* has the implicit qualification [20, 41] that the door must not be barricaded (by say a cupboard). However, the second action *force* is stronger, in the sense that it always opens the door, even if it is barricaded. In this latter case, the action has the indirect effect of unblocking the door. We represent this scenario by:

$$\leadsto = \left\{ \begin{array}{l} \mathit{open} \leadsto \neg \mathit{Closed}, \\ \mathit{force} \leadsto \neg \mathit{Closed}, \\ \mathit{force} \leadsto \neg \mathit{Barricaded} \end{array} \right\};$$

$$\mathit{LAW} = \left\{ \begin{array}{l} \Box\langle \mathit{open} \rangle \top, \\ \Box\langle \mathit{force} \rangle \top, \\ \Box(\mathit{Barricaded} \rightarrow \mathit{Closed}), \\ \Box[\mathit{open}]\neg \mathit{Closed}, \\ \Box[\mathit{force}]\neg \mathit{Closed} \end{array} \right\}.$$

Suppose $\mathit{KB} = \{\mathit{Closed}, \mathit{Barricaded}\}$. Then we can easily show that:

$$\models_{\mathcal{LAP}_{\leadsto}} \mathit{KB} \wedge \mathit{LAW} \rightarrow [\mathit{force}](\neg \mathit{Closed} \wedge \neg \mathit{Barricaded}).$$

Hence we can derive the indirect effect of the *force* action. Moreover, we can also show that:

$$\models_{\mathcal{LAP}_{\leadsto}} \mathit{KB} \wedge \mathit{LAW} \rightarrow [\mathit{open}]\bot.$$

This means that in $\mathcal{LAP}_{\leadsto}$ we can derive the implicit qualification for the action *open*: it cannot be executed if the door is barricded.

This example is problematic e.g. for Thielscher's approach: due to the action *force* the pair $(\neg Closed, Barricaded)$ must appear in the influence relation. This makes it impossible to derive the implicit qualification of the action *open*. This example is also problematic for the causal approach of McCain and Turner [45], who are obliged to explicitly state the qualification $\neg Barricaded$ for the action *open* (which is what they would like to avoid). The key of our solution is that our dependence is indexed by actions. In this way we can have $open \not\rightsquigarrow \neg Barricaded$ while having $force \rightsquigarrow \neg Barricaded$.

The following example is also problematic for causal relations that are not action-indexed, like Thielscher's.

Example 33 (Toulouse suitcase)
This is an enhancement of Lin's suitcase example [39], where we want to express that if the locks are linked, then they are always in the same position. (Imagine e.g. the locks are attached by a rigid metal bar.) In $\mathcal{LAP}_{\rightsquigarrow}$ we can represent this by adding the static law $\Box(Linked \rightarrow (Up_1 \equiv Up_2))$ and stating appropriate conditional frame axioms for the context $\neg Linked$. The corresponding causal influence information in the approach of [61] is

$$\mathcal{I} = \{(Up_1, Opened), (Up_2, Opened), (Up_1, Up_2), (Up_2, Up_1)\},$$

which allows the generation of the unintended causal laws

$$Up_1 \textsf{ causes } \neg Up_2 \textsf{ if } \neg Opened,$$

and

$$Up_2 \textsf{ causes } \neg Up_1 \textsf{ if } \neg Opened.$$

It is object of ongoing work how to generate dependence in a semi-automatic way, by means of an interactive preprocessing algorithm. This will come in the second part of this paper.

9.5 Indeterminate effects

Dependence is similar to the notions *releases* and *possibly changes* in $\mathcal{AR}$ [22] and $\mathcal{AR}_0$ [33]. The latter are action logics extending $\mathcal{A}$ in particular in order to handle indirect effects of actions.[13] The official reading of α releases P and α possibly changes P is "when α is performed then fluent p is exempted from the law of inertia". If we take this serious then it seems that α causes P should in general entail α releases P and α possibly changes P. But this is not the case in these logics[14], and this is one of the differences with our dependence notion: in our case the law $\Box[\alpha]P$ in general comes with $\alpha \rightsquigarrow P$.

In our view, it is the complexity of the minimization-based semantics of $\mathcal{AR}_0$ and $\mathcal{AR}$ which made counterexamples possible: $\mathcal{AR}_0$ fails to give the intended solution

[13] Although it is difficult to formulate different readings, there are subtle differences in the semantics of releases and possibly changes that are due to different minimization policies.

[14] Indeed, the domain descriptions "α causes P" and "α causes P, α releases P" behave exactly the same, while the semantics of "α causes $P \vee Q$" is different from that of "α causes $P \vee Q, \alpha$ releases P, α releases Q"

to Lin's suitcase example, and both $\mathcal{AR}_0$ and $\mathcal{AR}$ fail to handle Thielscher's relay example [61]. We are convinced that due to its simplicity our account will better resist to counterexamples.

How can we deal with indeterminate effects of actions in $\mathcal{LAP}_{\leadsto}$? Consider the action *toss* of tossing a coin, whose result is either *Heads* or *Tails*.

We might consider that in any situation both outcomes are actual possibilities. I.o.w. it is always the case that the result of *toss* might be *Heads*, and that it might be *Tails*. Then we should describe *toss* by the dynamic law $\Box(\langle toss\rangle Heads \wedge \langle toss\rangle Tails)$. This view will not be suitable here, because it does not allow us to add observations. Indeed, suppose we have observed that *toss* has been executed, and that *Heads* showed up. If we tried to represent this by $\langle toss\rangle Heads$, we would add something which is already entailed by the laws, and if we tried $[toss]Heads$ then this would be inconsistent with $\langle toss\rangle Tails$ under the standard reading of the modal operator together with the obvious static law $\Box\neg(Heads \wedge Tails)$. But can we be sure that *Tails* is a possible outcome in any case? If we take a determinist's position (or at least admit it) we should allow for situations where *Heads* is the only possible outcome. Then we should rather describe our action by the laws $\Box[toss](Heads \vee Tails)$ and $\Box\langle toss\rangle\top$. This allows us to add the observation $[toss]Heads$ in a non-trivial way.[15]

Example 34 (Spinning and tossing a coin [2])
Suppose a coin is spun on a chessboard. Let *Black* mean that the coin touches a black square, and *White* that it touches a white square. After the spinning action the coin may touch either a white square, or a black, or both.

$$\leadsto = \left\{ \begin{array}{l} spin \leadsto Black,\ spin \leadsto \neg Black, \\ spin \leadsto White,\ spin \leadsto \neg White \end{array} \right\};$$

$$LAW = \{\Box\langle spin\rangle\top, \Box[spin](Black \vee White)\}.$$

Compare this to the action of tossing a coin (which can be seen as spinning a particular coin of surface 0). In this case, we keep $\leadsto$ and just replace in LAW the inclusive disjunction $\vee$ by an exclusive one:

$$LAW = \{\Box[toss](Black \oplus White)\}.$$

In the literature of the use of disjunctions to represent indeterminate effects have been criticized (see e.g. [4]): consider the above toss-a-coin example, but suppose the language contains only the atom *Heads* (and not *Tails*). In this scenario, so the argument goes, we can only write down tautologous effect laws of the form $\Box[toss](Heads \vee \neg Heads)$. This does not apply to $\mathcal{LAP}_{\leadsto}$ because dependence gives us another linguistic tool to express indeterminate effects, as demonstrated above.

In fact, there is a third option for the representation of indeterminate effects, viz. to view actions with indeterminate effects as nondeterministic actions that are obtained from simpler actions by the operator of nondeterministic choice $\cup$. This is closer to the standard dynamic logic account of nondeterministic programs. (This is

[15] This is why we chose to read $[\alpha]A$ as "A after α", and not as "A after every possible execution of α", which is the standard dynamic logic reading [28]. If we wanted to stay with the latter then we would have been obliged to consider a more expressive language such as that of Process Logic in order to represent observations, as proposed in [3].

also the way Reiter proposes to integrate indeterminate effects into GOLOG.) Such a representation is possible in $\mathcal{LAP}_{\leadsto}$ as well. Consider e.g. the above action of spinning a coin on a chessboard. *spin* is viewed as being identical to $spin_1 \cup spin_2 \cup spin_3$, with

$$LAW = \left\{ \begin{array}{l} \Box\langle spin_1\rangle\top, \\ \Box\langle spin_2\rangle\top, \\ \Box\langle spin_3\rangle\top, \\ \Box[spin_1](Black \wedge \neg White), \\ \Box[spin_2](\neg Black \wedge White), \\ \Box[spin_3](Black \wedge White) \end{array} \right\};$$

$$\leadsto = \left\{ \begin{array}{l} spin_1 \leadsto Black, spin_1 \leadsto \neg White, \\ spin_2 \leadsto \neg Black, spin_2 \leadsto White, \\ spin_3 \leadsto Black, spin_3 \leadsto White \end{array} \right\};$$

We do not want to open a debate here, but just observe that our official account in terms of disjunctions as described above is somewhat more concise and flexible. This is in particular the case when it comes to indeterminate effects that are indirect, like the following example.

Example 35 (Spinning a coin, part II)
Consider the same scenario as in example 34, but with a supplementary atom *OnTable* meaning that the coin is on the table. It seems reasonable to suppose that the direct effect of the *spin* action is *OnTable*, while *Black* and *White* are effects that are indirect.

In $\mathcal{LAP}$ this can be captured by the static law $\Box(OnTable \rightarrow (Black \vee White))$ together with the effect law $\Box[spin]OnTable$ and the same executability law as before. The dependence relation is now

$$\leadsto = \left\{ \begin{array}{l} spin \leadsto OnTable, \\ spin \leadsto Black, spin \leadsto \neg Black, \\ spin \leadsto White, spin \leadsto \neg White \end{array} \right\}.$$

The example illustrates that in order to treat indeterminate indirect effects of an action α in $\mathcal{LAP}_{\leadsto}$ the only thing we must add are the indirect dependences of α. We note that in other approaches such as [45, 40, 22], one also must state the indirect dependences explicitly in order to solve this example.

10 Conclusion

We have presented a *monotonic* solution to the frame problem which allows for actions with *indeterminate* and *indirect* effects. It is a *modal framework* with a simple *semantics*, an *axiomatics* and a *decision procedure* based on a simple *tableau theorem prover*, where we can express *indeterminate* and *indirect effects* of actions. Neither of these characteristics is novel alone, but as far as we know no approach in the literature unites all of them.

Modal logics (viz. PDL and extensions of it) have been used before [13, 52, 59]. In [59] frame axioms are computed in a way similar to our construction of dependence relations. Different monotonic solutions to the frame problem in the style of [50] and others are given in [13], using converse and complement of actions. Also, in [42], another formalism to reason about programs is used.

In what concerns dependence, similar notions of weak causality have been presented in the literature in the last five years to handle indeterminate and indirect effects of actions. In fact, our distinction between dependent and independent literals can be seen as a generalization of that between frame and non-frame fluents as proposed in [38], in the sense that we parametrize it by actions. Thielscher's relay example in [60] is an argument for such a generalization. As we have already pointed out in sections 1 and 9.5, our notion is similar to occluded [54] and releases and possibly changes [33, 22].

In relation with indirect effects of actions, causality is widely considered to be a fundamental concept, that has been used in [25, 39, 44, 60], among others. The analysis in [61] was deepened in section 9 by the observation that the influence of a fluent may depend on the context (example 32). Also, we have shown the need of action-indexed causation beyond the framework in [45] (example 33). These two examples cannot be solved with the notions of causality of the other approaches.

In the reasoning about actions community, only little attention has been paid up to now to mechanical proof procedures. By and large, the existing ones are either based on circumscriptive theorem provers [34, 25, 14] or on logic programming languages [19, 35, 51]. The former approaches use algorithms reducing second order logic to first order. While this is not possible in any case, Gustafsson and Doherty [25] have identified a fragment of second order logic where reduction is (theoretically) guaranteed. In any case, such approaches must cope with the undecidability of first order logic, while $\mathcal{LAP}_{\leadsto}$ is decidable.

In the logic programming approaches, action theories and knowledge bases are encoded as logic programs. These programs are then queried in order to do prediction and planning. This framework induces several limitations w.r.t. the nature of the action theories and knowledge bases. In the GOLOG implementation presented in [35, 51], action effect laws and the elements of the knowledge base must be definitional. This entails that there can be no disjunctions in effect laws and in knowledge bases. While Reiter succeeds in representing actions with indeterminate effects by means of nondeterministic actions that are implemented by macro expansion, the limitation of knowledge bases to definitional ones appears to be severe in open domains with incomplete information, as noted by Reiter himself. Moreover, as logical negation is identified with negation by failure, the implementation of GOLOG in [35, 51] requires complete knowledge about the initial situation. For these reasons, such an implementation of GOLOG cannot handle correctly e.g. Sandewall's deaf turkey scenario.

Finally, the semantics of several action logics such as the $\mathcal{A}$-family being in terms of finite state automata, it is a natural idea to apply to such structures model checking techniques (that are usually applied to prove properties of programs). As far as we know, the only place where this idea has been exploited is in [10].

$\mathcal{LAP}_{\leadsto}$ is based on tableau methods, and is not committed to the logic programming framework. It turned out that our stepback (SB) tableau rule can be generalized and successfully applied in order to define powerful modal tableaux, as we have recently demonstrated in [6]. The simplicity of our method supports our conviction that we can have an efficient implementation, e.g. taking profit of existing SAT-algorithms for classical logic [23], or of sophisticated strategies [30]. We plan to do this in future work.

It is common to give priority to the simplicity of writing domain descriptions. In this sense, the best example is certainly the logic $\mathcal{A}$ of [19]. As said in section 9, extensions of $\mathcal{A}$ to indirect and indeterminate effects of actions, while preserving this simplicity at the syntactic level, have a complex semantical machinery with built-in complete information hypotheses. In turn, we propose a simple language, with a clear semantics. We think that a simple framework is preferable, because we are much more immune against sophisticated toy examples which up to now successfully attacked most of the approaches that have been put forward – and we think the price to pay (viz. to write down indirect dependences, conditional frame axioms and executability laws) is not too high.

The generation of such information is one of the main themes of the second part of the paper (a first draft may be found in [8]).[16] An important aspect treated there is that interaction with a designer of an action theory is fundamental and must be considered. Our logic is relatively weak, but it can be supplemented in a semi-automatic way: the designer can decide whether some dependences, effect laws, and executability laws are the intended ones.

Acknowledgements

We are grateful to Luis Fariñas del Cerro, Patrick Doherty, Gerhard Lakemeyer, Erik Sandewall, and to the anonymous referees, whose comments helped to improve the presentation of this paper. We also thank Enrico Giunchiglia, Norman McCain, and Michael Thielscher for kindly explaining details of their respective approaches. Particular thanks are due to Fabio Massacci, who has pointed out an error in a previous version of the paper. This work was partially supported by CAPES/Brazil, under grants to the first author.

[16] In what concerns ramifications, Thielscher's solution presented in [61] is similar: he automatically derives causal relationships from the basic dynamic laws and the influence information.

A Complexity: proofs

Theorem 36 Complexity of $\mathcal{LAP}$-satisfiability is EXPTIME-complete.

The proof follows from the two lemmas below.

Lemma 37 Complexity of $\mathcal{LAP}$-satisfiability is at most EXPTIME.

Proof We define a polynomial transformation t from $\mathcal{LAP}$ into PDL (which is known to be in EXPTIME[17]). t recursively translates a given formula A_0 into $t(A_0)$:

- $t(p) = p$ for $p \in ATM$
- $t(\Box A) = [(\alpha_1 \cup \alpha_2 \cup \ldots \cup \alpha_n \cup \alpha_{n+1})^*]t(A)$, where $\alpha_1, \alpha_2, \ldots, \alpha_n$ are the atomic actions occurring in A_0, and α_{n+1} is an atomic action not occurring in A_0.
- and homomorphic else.

Clearly, t is a polynomial transformation.
Suppose that A_0 is $\mathcal{LAP}$-valid. By completeness there is a $\mathcal{LAP}$-proof of A_0. We transform that proof into a PDL-proof of $t(A_0)$. In order to do that it is sufficient to establish that t preserves validity of the axioms and inference rules of $\mathcal{LAP}$. For example every instance of the $\mathcal{LAP}$-axiom $\Box A \rightarrow [\alpha]A$ must be of the form $\Box A \rightarrow [\alpha_i]A$ and becomes

$$t(\Box A \rightarrow [\alpha_i]A) = [(\alpha_1 \cup \ldots \cup \alpha_n \cup \alpha_{n+1})*]A \rightarrow [\alpha_i]A$$

which is a theorem of PDL. Now suppose A_0 $\mathcal{LAP}$-satisfiable. Hence there is a $\mathcal{LAP}$-model $\mu = \langle W, \{R_\alpha : \alpha \in ACT\}, R_\Box, \tau\rangle$ and $w \in W$ such that $\models^\mu_w A_0$. (We have dropped from μ those R_α which do not occur in A_0.) Let $R_{\alpha_{n+1}} = R_\Box \setminus (R_{\alpha_1} \cup \ldots \cup R_{\alpha_n})*$. Then $(W, (R_{\alpha_1}, \ldots, R_{\alpha_n}, R_{\alpha_{n+1}}, \tau)$ is a PDL-model, and you can check by induction on the structure of A_0 that $\models^\mu_w A_0$.

Lemma 38 Complexity of $\mathcal{LAP}$-satisfiability is at least EXPTIME.

Proof We transform the deduction problem of modal logic K (which is known to be EXPTIME-complete [26]) into the validity problem of $\mathcal{LAP}$.
Let $\models_{K[\alpha]}$ *denote* consequence in the monomodal logic $K[\alpha]$ (K with a single modal operator $[\alpha]$). As the logic $K[\alpha]$ is compact [18], for every formula A and set of formulas S we have that $S \models_{K[\alpha]} A$ iff there is a finite subset $S' \subseteq S$ such that $S' \models_{K[\alpha]} A$. By completeness of tableaux for $K[\alpha]$, we have $S' \models_{K[\alpha]} A$ iff there is a closed tableau for $\neg A$ under global assumptions S' [18]. Now $K[\alpha]$-tableaux with global assumptions S' behave exactly as $\mathcal{LAP}$-tableaux with domain laws $\Box(\bigwedge S')$, where $\bigwedge S'$ is the conjunction of the elements of S'. In more formal words, there is a closed $K[\alpha]$-tableau for $\neg A$ under S' iff there is a closed $\mathcal{LAP}$-tableau for $\neg A \wedge \Box(\bigwedge S')$.

B Axiomatics of $\mathcal{LAP}_{\leadsto}$: soundness and completeness proofs

In this section we prove that the axiomatics of $\mathcal{LAP}_{\leadsto}$ is sound and complete w.r.t. the semantics given in section 5.2.

As usual, soundness is immediate: given the soundness of $\mathcal{LAP}$, the only thing to do is to verify that all instances of $Persistence([\alpha])$ are valid.

Completeness is easy to prove by the usual Henkin method (cf. for example [31]), and the rest of this section is devoted to it.

Following [29], we have:

Lemma 39 (Lindenbaum's lemma) Provided the language is denumerable, then any $\mathcal{LAP}_{\leadsto}$-consistent set S admits a maximal-$\mathcal{LAP}_{\leadsto}$-consistent extension,

Now, we are ready to prove the following properties about maximal $\mathcal{LAP}_{\leadsto}$-consistent sets:

Lemma 40 Let S be a maximal $\mathcal{LAP}_{\leadsto}$-consistent set, and A and B be formulas.

(i) If $\vdash_{\mathcal{LAP}} A$ then $A \in S$.

(ii) $A \in S$ iff $\neg A \notin S$.

(iii) $A \wedge B \in S$ iff $A \in S$ and $B \in S$.

(iv) $A \vee B \in S$ iff $A \in S$ or $B \in S$.

(v) if $A \to B \in S$ and $A \in S$ then $B \in S$.

The above is sufficient for canonical models for Classical (non-modal) Propositional Calculus. For modal logics we need further definitions and results.

Let S be a set of formulas of our language, we define $\Box^-(S)$ to be the set $\{A\colon \Box A \in S\}$ and $[\alpha]^-(S)$ to be the set $\{A\colon [\alpha]A \in S\}$.

Lemma 41 Let S be a $\mathcal{LAP}_{\leadsto}$-consistent set of formulas containing a formula $\Diamond A$ (respec. $\langle\alpha\rangle A$). Then the set $\Box^-(S) \cup \{A\}$ (respec. $[\alpha]^-(S) \cup \{A\}$) is $\mathcal{LAP}_{\leadsto}$-consistent.

Proof We give the proof only for the case $\Diamond A$. The It is similar for $\langle\alpha\rangle A$. Suppose $\Box^-(S) \cup \{A\}$ is $\mathcal{LAP}_{\leadsto}$-inconsistent, then there is some finite subset $\Box A_1, \ldots, \Box A_n$ of S such that: $\vdash_{\mathcal{LAP}} (A_1 \wedge \ldots \wedge A_n \wedge A) \to \bot$. Thus $\vdash_{\mathcal{LAP}} (A_1 \wedge \ldots \wedge A_n) \to \neg A$, this implies: (by necessitation rule and axiom $K(\Box)$ plus Modus Ponens) $\vdash_{\mathcal{LAP}} \Box(A_1 \wedge \ldots \wedge A_n) \to \Box\neg A$, i.e., $\vdash_{\mathcal{LAP}} \Box(A_1 \wedge \ldots \wedge A_n) \to \neg\Diamond A$ and by distributivity of $\Box$, $\vdash_{\mathcal{LAP}} (\Box A_1 \wedge \ldots \wedge BoxA_n) \to \neg\Diamond A$. But each $\Box A_i$ is in S, hence by lemma 40, $\neg\Diamond A$ would be in S leading to the conclusion that S is $\mathcal{LAP}_{\leadsto}$-inconsistent.

As usual, the canonical model for $\mathcal{LAP}_{\leadsto}$ is $\mu = \langle W, \{R_\alpha : \alpha \in ACT\}, R_\Box, \tau\rangle$ where:

- W is the set of all maximal-$\mathcal{LAP}_{\leadsto}$-consistent sets,
- $wR_\Box w'$ iff $\Box^-(w) \subseteq w'$,

- for every atomic actions α, $wR_\alpha w'$ iff $[\alpha]^-(w) \subseteq w'$,
- for every atom P, and every w of W, $w \in \tau(P)$ iff $P \in w$.

The fundamental theorem states that:

Theorem 42 For any formula A, $\models^\mu_w A$ iff $A \in w$.

Proof The proof is by induction on the structure of A. The theorem is immediate for atoms, by definition of τ. For the induction step, Boolean cases are trivial. We only prove the case $[\alpha]B$ (The case $\Box B$ is similar.)
(From left to right): Suppose $\models^\mu_w [\alpha]B$ and $[\alpha]B \notin w$. Hence, by lemma 40, $\neg[\alpha]B \in w$ (i.e., $\langle\alpha\rangle\neg B \in w$.) Let $S = [\alpha]^-(S) \cup \{\neg B\}$, by lemma 41 S is $\mathcal{LAP}_{\leadsto}$-consistent and (by lemma 39) can be extended to a maximal-$\mathcal{LAP}_{\leadsto}$-consistent set w'. By construction of μ, $w' \in W$ and $wR_\alpha w'$. As $\neg B \in w'$, by induction hypothesis, $\models^\mu_{w'} \neg B$ and thus $\not\models^\mu_w [\alpha]B$, contrary to the hypothesis.
(From right to left): Suppose $[\alpha]B \in w$. Then by definition of R_α we have $B \in w'$ for every w' such that $wR_\alpha w'$; by induction hypothesis $\models^\mu_{w'} B$. Hence by definition of satisfaction: $\models^\mu_w [\alpha]B$.

It remains to show that μ is a $\mathcal{LAP}_{\leadsto}$-model.

Theorem 43 The canonical model defined as above is a $\mathcal{LAP}_{\leadsto}$-model.

Proof The proof that $R_\Box$ is reflexive and $R_\Box$ is transitive is as usual.

(i) $R_\alpha \subseteq R_\Box$ for every $\alpha \in ACT$

(ii) If $\alpha \not\leadsto L$ and $wR_\alpha w'$ then $L \in w'$ implies $L \in w$.

(i) We have to prove that if $wR_\alpha w'$ then $wR_\Box w'$, i.e., $\Box^-(w) \subseteq w'$. Let $A \in \Box^-(w)$. This holds iff $\Box A \in w$, and as $\vdash_{\mathcal{LAP}} \Box A \to [\alpha]A$, we get by lemma 40 that $[\alpha]A \in w$. Thus, as $wR_\alpha w'$ we have $A \in w'$.

(ii) Let $\alpha \not\leadsto L$ and $wR_\alpha w'$. As $\vdash_{\mathcal{LAP}} \neg L \to [\alpha]\neg L$, by lemma 40 and 41, $\neg L \in w$ implies $\neg L \in w'$.

We can now state the completeness theorem:

Theorem 44 If a formula A is $\mathcal{LAP}_{\leadsto}$-valid then it is a $\mathcal{LAP}_{\leadsto}$-theorem.

Proof We prove the contrapositive: If a formula A is $\mathcal{LAP}_{\leadsto}$-consistent then it is $\mathcal{LAP}_{\leadsto}$-satisfiable: If A is $\mathcal{LAP}_{\leadsto}$-consistent, the set $\{A\}$ is so. By lemma 39 it can be extended to a maximal-$\mathcal{LAP}_{\leadsto}$-consistent set w. Of course, $A \in w$ and by the theorem 42 we have that $\models^\mu_w A$. As μ is a $\mathcal{LAP}_{\leadsto}$-model (by lemma 43) A is $\mathcal{LAP}_{\leadsto}$-satisfiable.

C Tableau for $\mathcal{LAP}_{\leadsto}$: soundness and completeness proofs

C.1 Completeness

In this subsection we prove the completeness of our tableau calculus. We show how, from any[17] open tableau for A we can construct a $\mathcal{LAP}_{\leadsto}$-model for A. For notational convenience we abbreviate $\bigcup_{\alpha \in ACT} \Sigma_\alpha$ by Σ_{ACT}.

Let $T = C^\infty$ be an open tableau (possibly infinite). Of course, there is an open tree $\langle S, \Sigma \rangle$ in T (since T is open). Let $\mu = \langle W, \{R_\alpha : \alpha \in ACT\}, R_\Box, \tau \rangle$ be the Kripke model defined as follows:

- W is the set of labels (integers) occurring in S;
- $R_\alpha = \Sigma_\alpha$ for $\alpha \in ACT$
- $R_\Box = (\Sigma_{ACT} \cup \Sigma_\Box)^*$ (reflexive and transitive closure of $\Sigma_{ACT} \cup \Sigma_\Box$)
- $n \in \tau(p)$ iff $(n, p) \in S$

Lemma 45 μ is a $\mathcal{LAP}_{\leadsto}$-model.

Proof [18]

(i) By construction of μ we have that for every $\alpha \in ACT$, $R_\alpha \subseteq R_\Box$; also by construction $R_\Box$ is reflexive and transitive.

(ii) Let $(n, n') \in R_\alpha$. We show that if $\alpha \not\leadsto P$ then $n \notin \tau(P)$ implies $n' \notin \tau(P)$, and if $\alpha \not\leadsto \neg P$ then $n \in \tau(P)$ implies $n' \in \tau(P)$.
Suppose $\alpha \not\leadsto \neg P$ and $n \in \tau(P)$.
$\Rightarrow (n, p) \in S$ by definition of τ;
$\Rightarrow (n', p) \in S$ (by rule SF);
$\Rightarrow n' \in \tau(P)$.
Suppose $\alpha \not\leadsto P$ and $n \notin \tau(P)$.
$\Rightarrow (n, p) \notin S$;
$\Rightarrow (n', p) \notin S$ (otherwise $(n, p) \in S$ by rule SB);
$\Rightarrow n' \notin \tau(P)$.

It remains to show that this model μ satisfies A.

Lemma 46 Let B be a subformula of A:
(*) if $(n, B) \in S$ then $\models^\mu_n B$.

Proof (By induction on the structure of B).
Induction initialization: let B be a literal then (*) holds by definition of τ.
Induction step: (We do not give the Boolean cases that are as usual.)

[17] We recall that we make the usual *fairness* assumption: any rule that can be applied will eventually be applied.

[18] In this and the following proofs, when we say "by rule R" we mean "by rule R and by the fairness assumption that rule R has been applied".

- B is $[\alpha]C$, $[\alpha]C \in Br(n)$
  $\Rightarrow (n', C) \in S$ by rule (K_α), for all n' such that $(n, n') \in \Sigma_\alpha = R_\alpha$
  $\Rightarrow \models^\mu_{n'} C$ (by induction hypothesis), for all n' such that $(n, n') \in R_\alpha = R_\alpha$
  $\Rightarrow \models^\mu_n [\alpha]C$.

- B is $\Box C$
  First, $(n, n') \in R_\Box \Rightarrow (n, n') \in (\Sigma_{ACT} \cup \Sigma_\Box)^*$
  $\Rightarrow \exists n_0 = n, n_1, \ldots, n_k = n'$ such that $(n_i, n_{i+1}) \in \Sigma_{ACT} \cup \Sigma_\Box$ for $0 \leq i < k$.
  Now suppose $(n, \Box C) \in S$
  $\Rightarrow (n_j, \Box C) \in S$ for all $0 \leq j \leq k$ by successive applications of rule $(4_\Box)$
  $\Rightarrow (n', C) \in S$ by rule $(T_\Box)$
  $\Rightarrow \models^\mu_{n'} C$ (by induction hypothesis)
  $\Rightarrow \models^\mu_n \Box C$.

- B is $\neg\Box C$
  $(n, \neg\Box C) \in S$
  $\Rightarrow$ There exists n' such that $(n, n') \in R_\Box$, and $(n', \neg C) \in S$ (by rule $\Diamond$)
  $\Rightarrow$ There exists n' such that $(n, n') \in R_\Box$, and $\models^\mu_{n'} \neg C$ (by induction hypothesis)
  $\Rightarrow \models^\mu_n \neg\Box C$.

- B is $\neg[\alpha]C$.
  The proof is similar to the case of $\neg\Box C$.

Corollary 47 $\models^\mu_0 A$.

It follows from the above that our tableau calculus is complete under the fairness assumption:

Corollary 48 If there is a fair open tableau for A then A is satisfiable.

C.2 Soundness

In this subsection, we prove the soundness of our tableau calculus: if all tableaux for A are closed then A is unsatisfiable. To this end we first state and prove a lemma.

Definition 49 Let S be a set of arbitrary labelled-formulas. A *state* $S(n)$ denotes the set of formulas A such that $(n, A) \in S$.

Lemma 50 [19] Let $\langle S, \Sigma\rangle$ be a finite tree and let
$\overline{S(n)} = S(n) \wedge \bigwedge_{\substack{\alpha \in ACT \\ (n,n') \in \Sigma_\alpha}} \langle\alpha\rangle S(n') \bigwedge_{(n,n') \in \Sigma_\Box} \Diamond S(n')$.
Then, for every rule: if for all possible $C^{i+1} = \{\ldots, \langle S^{i+1}, \Sigma^{i+1}\rangle, \ldots\}$ that can be obtained from $C^i = \{\ldots, \langle S^i, \Sigma^i\rangle, \ldots\}$ [20] there exists n such that $\overline{S^i(n)}$ is unsatisfiable, then so it is for C^i.

Proof This lemma is proved by considering the various rules. (We only consider the cases of the rules $\Diamond$ and $\langle\alpha\rangle$, and SB. We will write $\models$ instead of $\models_{\mathcal{LAP}_\rightsquigarrow}$).

[19] In this lemma, we identify a finite set of formulas with the conjunction of all its formulas, hence S also denotes the formula: $\bigwedge_{A \in S} A$.
[20] In fact, there is only one possibility, except in the case of rule $\vee$.

- Rule $\Diamond$: S^i contains $(\Diamond A, n)$, and applying rule $\Diamond$ adds (n, n') to $\Sigma^i_\Box$ and (A, n') to S^i. If all $S^i(n)$ were satisfiable, only $S^{i+1}(n')$ might have become unsatisfiable: but we have $\overline{S^{i+1}(n')} = \neg A$. Suppose $\neg A$ is unsatisfiable: $\models A$, hence $\models \Box\neg A$. Hence $S^i(n)$ is unsatisfiable too, since it contains $\Diamond A$. The same reasoning applies straightforwardly to rule $\langle\alpha\rangle$.

- Rule SB: S^i contains (L, n'), Σ^i_α contains (n, n'), L is a literal and $\alpha\not\rightsquigarrow L$, and applying rule SB adds (L, n) to S^i. Suppose $\overline{S^{i+1}(n)} = \overline{S^i(n) \cup \{L\}}$ is unsatisfiable, then $\models \overline{S^i(n)} \rightarrow \neg L$. But $\alpha\not\rightsquigarrow L$ implies $\models \neg L \rightarrow [\alpha]\neg L$; and $(n, n') \in \Sigma_\alpha$, implies $\models \overline{S^i(n)} \rightarrow \neg[\alpha]\neg\overline{S^i(n')}$, in particular: $\models \overline{S^i(n)} \rightarrow \neg[\alpha]\neg L$, and thus: $\models \overline{S^i(n)} \rightarrow [\alpha]\neg L \wedge neg[\alpha]\neg L$, hence $\overline{S^i(n)}$ is unsatisfiable.

Theorem 51 If a tableau for A is closed then A is unsatisfiable.

Proof The previous lemma ensures that all rules preserve $\mathcal{LAP}_\rightsquigarrow$-satisfiability. When transforming a forest into another one. Thus if A is $\mathcal{LAP}_\rightsquigarrow$-satisfiable, we will start with an $\mathcal{LAP}_\rightsquigarrow$-satisfiable forest $\{\langle\{(A, 0)\}, \emptyset\rangle\}$. Then, by the lemma, we cannot always obtain $\mathcal{LAP}_\rightsquigarrow$-unsatisfiable trees when rules are applied, and*a fortiori* we cannot always obtain closed trees.

D Translation from $\mathcal{A}$ to $\mathcal{LAP}_\rightsquigarrow$: soundness and completeness proof

Let EP be a set of effect propositions, VP a set of value propositions of $\mathcal{A}$, and KB, $\rightsquigarrow$ and LAW their translation. First we show that $\mathcal{A}$-models of EP correspond to particular $\mathcal{LAP}_\rightsquigarrow$-models satisfying LAW.

Lemma 52 Let Φ be an $\mathcal{A}$-model, and let $\mu = \langle W, \{R_\alpha : \alpha \in ACT\}, R_\Box, \tau\rangle$ be a $\mathcal{LAP}$-model such that

- $W = 2^{ATM}$ is the set of classical interpretations,
- $vR_\alpha v'$ iff $\Phi_\alpha(v) = v'$,
- $R_\Box = W \times W$, and
- $v \in \tau(P)$ iff $P \in v$.

Then Φ is an $\mathcal{A}$-model of EP iff μ is a $\mathcal{LAP}_\rightsquigarrow$-model and LAW is true in μ.

Proof Φ is an $\mathcal{A}$-model of EP if and only if the following hold for every EP_α and every classical interpretation v and literal L:

- If there is an effect proposition α causes L if A in EP_α such that v satisfies A, then $\Phi_\alpha(v)$ satisfies L,
- else v satisfies $\neg L$ implies $\Phi_\alpha(v)$ satisfies $\neg L$.

This is exactly the case if the following hold for μ:

1. For every α causes L if A in EP, $A \to [\alpha]L$ is true in μ.
2. For each α and every L, w, if there is no α causes L if A in EP such that $\models^{\mu}_{w} A$, then $\models^{\mu}_{w} \neg L \to [\alpha]\neg L$.

The last condition is the most delicate one. It can be split into

(i) $\forall \alpha, L$, if there is no A such that α causes L if A is in the set EP then we have $\models^{\mu}_{w} \Box(\neg L \to [\alpha]\neg L)$;

(ii) $\forall \alpha, L$, $\models^{\mu}_{w} \Box(\neg Pre(\alpha, L) \to (\neg L \to [\alpha]\neg L))$,

where $Pre(\alpha, L) = \bigvee_{(\alpha \text{ causes } L \text{ if } A_j) \in EP} A_j$. $Pre(\alpha, L)$ is the disjunction of all preconditions A_j for which L is the effect of α. If there is no effect proposition for α with effect L then $Pre(\alpha, L)$ is $\bot$. Condition (ii) corresponds to item (b) in the translation, and condition (i) means that μ is a $\mathcal{LAP}_{\leadsto}$-model: it is exactly in this that $\alpha \not\leadsto L$. It follows that LAW is true in μ, i.e., $\models^{\mu}_{w} LAW$ for every $w \in W$.

The lemma is useful to prove completeness of the translation:

Theorem 53 If $\models_{\mathcal{LAP}_{\leadsto}} (KB \wedge LAW) \to \langle \pi \rangle L$ then $EP \cup VP$ $\mathcal{A}$-entails the value proposition L after π.

Proof Suppose Φ is an $\mathcal{A}$-model of EP. The preceding lemma tells us that LAW is true in the associated $\mathcal{LAP}_{\leadsto}$-model μ.
In what concerns value propositions, it is easy to see that by the construction of μ, L after π is $\mathcal{A}$-true at a classical interpretation v of Φ iff $\models^{\mu}_{v} [\pi]L$ iff $\models^{\mu}_{v} \langle \pi \rangle L$.
Suppose that VP is $\mathcal{A}$-true at some classical interpretation v. Then we can prove from the above equivalence that $\models^{\mu}_{v} KB$. As LAW is true in μ and also by the hypothesis we have $\models_{\mathcal{LAP}_{\leadsto}} (KB \wedge LAW) \to \langle \pi \rangle L$, we must also have $\models^{\mu}_{v} \langle \pi \rangle L$. Again by the above equivalence, L after π is $\mathcal{A}$-true at v of Φ.

It remains to establish the soundness of the translation.

Theorem 54 If there is an $\mathcal{A}$-model of EP, and $EP \cup VP$ $\mathcal{A}$-entails the value proposition L after π, then $\models_{\mathcal{LAP}_{\leadsto}} (KB \wedge LAW) \to \langle \pi \rangle L$.

Proof Let Φ be the $\mathcal{A}$-model of EP. Supposing $\not\models_{\mathcal{LAP}_{\leadsto}} (KB \wedge LAW) \to \langle \pi \rangle L$. Hence there is some $\mathcal{LAP}_{\leadsto}$-model $\mu = \langle W, \{R_\alpha : \alpha \in ACT\}, R_\Box, \tau \rangle$ and some $w_0 \in W$ such that $\models^{\mu}_{w_0} KB \wedge LAW \wedge [\pi]\neg L$. We first show that $\models^{\mu}_{w} [\pi]L$ iff $\Phi_\pi(v_w)$ satisfies L, by induction on the length of π, where v_w is the classical interpretation associated to the possible world w: $v_w = \{P \in ATM : w \in \tau(P)\}$.
The equivalence holds for the base $\pi = \lambda$. For the induction step, we have $\models^{\mu}_{w} [\alpha; \pi']L$ iff $\models^{\mu}_{w'} [\pi']L$, for every w' such that $wR_\alpha w'$. By induction hypothesis, the latter is the case iff $\Phi_{\pi'}(v_{w'})$ satisfies L, for every w'. Now observe that on the one hand, there is at least one such w' (because $\Box\langle\alpha\rangle\top$ is in LAW), and on the other hand, the interpretation $v_{w'}$ is equal to $\Phi_\alpha(v_w)$.[21] Hence $\Phi_{\pi'}(\Phi_\alpha(v_w)) = \Phi_{\alpha;\pi(v_w)}$ satisfies L.

[21] This is due to the fact that (as noted in [19]) given a set of effect laws EP, there is at most one transition function Φ_α that satisfies all the conditions. The same is the case for LAW: whenever we have $wR_\alpha w_1$ and $wR_\alpha w_2$ then the classical interpretations v_{w_1} and v_{w_2} associated to w_1 and w_2 are the same, i.e., we must have $w_1 \in \tau(P)$ iff $w_2 \in \tau(P)$, for all P.

Let Φ be some $\mathcal{A}$-model of EP (which by hypothesis exists), and let v_{w_0} be the classical interpretation associated to w. As $\models^{\mu}_{w_0} KB \wedge [\pi]\neg L$, by the above equivalence we must have that both KB and $\neg L$ after π are $\mathcal{A}$-true at v_{w_0}. But then L after π cannot be true at v_{w_0} (because $\Phi_\pi(v_{w_0})$ cannot satisfy both L and $\neg L$). Hence $EP \cup VP$ cannot $\mathcal{A}$-entail L after π.

Finally, the soundness and completeness theorem 27 follows directly from theorems 53 and 54.

References

[1] A.B. Baker. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, 49(1–3):5–23, may 1991.

[2] C. Baral. Reasoning about actions: non-deterministic effects, constraints and qualification. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 2017–2023, Montreal, Ca, 1995.

[3] C. Boutilier and N. Friedman. Nondeterministic actions and the frame problem. In *Working notes of the AAAI-Spring Sysposium on Extending Theories of Actions*, 1995. Extended Abstract.

[4] G. Brewka and J. Hertzberg. How to do things with worlds: on formalizing actions and plans. *Journal of Logic and Computation*, 3(5):517–532, 1993.

[5] D. Calvanese. Reasoning with inclusion axioms in description logics: Algorithms and complexity. In *Proc. European Conference on Articifial Intelligence (ECAI'96)*, pages 303–307. John Wiley & Sons, Ltd., 1996.

[6] M.A. Castilho, L. Fariñas del Cerro, O. Gasquet, and A. Herzig. Modal tableaux with propagation rules and structural rules. *Fundamenta Informaticae*, 32(3/4), 1997.

[7] M.A. Castilho, O. Gasquet, and A. Herzig. Modal tableaux for reasoning about actions and plans. In S. Steel and R. Alami, editors, *European Conference on Planning (ECP'97)*, number 1348 in Lecture Notes in Artificial Intelligence, pages 104–116, Toulouse, France, 1997. Springer-Verlag.

[8] M.A. Castilho, O. Gasquet, and A. Herzig. A dependence-based framework for actions with indeterminate and indirect effects. Technical Report 98-04-R, IRIT/UPS, feb 1998.

[9] M.A. Castilho and A. Herzig. An alternative to the iteration operator of propositional dynamic logic. Technical Report 96-05-R, IRIT/UPS, jan 1996.

[10] A. Cimatti, E. Giunchiglia, F Giunchiglia, and P. Traverso. Planning via model checking: a decision procedure for $\mathcal{AR}$. In S. Steel and R. Alami, editors, *European Conference on Planning (ECP'97)*, number 1348 in Lecture Notes in Artificial Intelligence, pages 130–142, Toulouse, France, sep 1997. Springer-Verlag.

[11] M.-O. Cordier and P. Siegel. Prioritized transitions for updates. In *Proc. ECSQUARU'95*, number 946 in LNAI, pages 142–150. Springer-Verlag, 1995.

[12] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. Nat. (US) Conf. on Artificial Intelligence (AAAI'94)*, pages 205–212. AAAI Press/The MIT Press, 1994.

[13] G. De Giacomo and M. Lenzerini. *PDL*-based framework for reasoning about actions. In *Proc. 4th Congresss of the Italian Association for Artificial Intelligence (IA*AI'95)*, number 992 in LNAI, pages 103–114. Springer-Verlag, 1995.

[14] P. Doherty, W. Łukaszewicz, and A. Szałas. Computing circumscription revisited: Preliminary report. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1502–1508, Montreal, Canada, 1995.

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

[16] J.J. Finger. *Exploiting constraints in design synthesis.* PhD thesis, Stanford University, Stanford, CA, 1987.

[17] M.J. Fischer and R.E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[18] M. Fitting. *Proof methods for modal and intuitionistic logics.* D. Reidel, Dordrecht, 1983.

[19] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, pages 301–321, 1993.

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

[21] E. Giunchiglia. Determining ramifications in the situation calculus. In *Proc. Int. Conf. on Knowledge Representation and Reasoning (KR'96)*, pages 76–86, Cambridge, Massachussetts, 1996.

[22] E. Giunchiglia, G. N. Kartha, and V. Lifschitz. Representing action: indeterminacy and ramifications. *Artificial Intelligence*, 95:409–438, 1997.

[23] F. Giunchiglia and R Sebastiani. A SAT-based decision procedure for $\mathcal{ALC}$. In *Proc. Int. Conf. on Knowledge Representation and Reasoning (KR'96)*, pages 302–314, Cambridge, Massachussetts, 1996.

[24] R.P. Goré. *Cut-free sequent and tableau systems for propositional normal modal logics.* PhD thesis, University of Cambridge, England, 1992.

[25] J. Gustafsson and P. Doherty. Embracing occlusion in specifying the indirect effects of actions. In *Proc. Int. Conf. on Knowledge Representation and Reasoning (KR'96)*, pages 87–98, Cambridge, Massachussetts, 1996.

[26] J.Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[27] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In *Proc. Nat. (US) Conf. on Artificial Intelligence (AAAI'86)*, pages 328–333, Philadelphia, PA, 1986.

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

[29] L. Henkin. The completeness of the first-order functional calculus. *Journal of Symbolic Logic*, 14:159–166, 1949.

[30] A. Heuerding, M. Seyfried, and H. Zimmermann. Efficient loop-check for backward proof search in some non-classical propositional logics. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Proceedings of the 5th International Workshop TABLEAUX'96: Theorem Proving with Analytic Tableaux and Related Methods*, number 1071 in LNAI, pages 210–225. Springer-Verlag, 1996.

[31] G. Hughes and M.J. Cresswell. *A companion to modal logic.* Methuen & Co, 1984.

[32] G. N. Kartha. Soundness and completeness theorems for three formalizations of action. In *Proc. of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, pages 8660–871, Chambéry, France, 1993.

[33] G. N. Kartha and V. Lifschitz. Actions with indirect effects (preliminary report). In *Proc. Int. Conf. on Knowledge Representation and Reasoning (KR'94)*, pages 341–350, 1994.

[34] G. N. Kartha and V. Lifschitz. A simple formalizations of actions using circumscription. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1970–1975, Montreal, Canada, 1995.

[35] H. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 19,20:1–679, 1997. Special issue on reasoning about action and change.

[36] P. Liberatore. The complexity of the language $\mathcal{A}$. *Electronic Transactions in Artificial Intelligence*, 1(1–3):13–38, 1997.

[37] V. Lifschitz. Formal theories of action. In *Proceedings of the 1987 Workshop on the Frame Problem*, pages 35–57, 1987.

[38] V. Lifschitz. Frames in the space of situations. *Artificial Intelligence*, 46:365–376, 1990.

[39] F. Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1985–1991, Montreal, Canada, 1995.

[40] F. Lin. Embracing causality in specifying the indeterminate effects of actions. In *Proc. Nat. (US) Conf. on Artificial Intelligence (AAAI'96)*, 1996.

[41] F. Lin and R. Reiter. State constraints revisited. *Journal of Logic and Computation*, 4(5):655–678, 1994.

[42] W. Łukaszewicz and E. Madalińska-Bugaj. Reasoning about action and change using Dijkstra's semantics for programming languages: preliminary report. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1950–1955, Montreal, Canada, 1995.

[43] F. Massacci. Strongly analytic tableaux for normal modal logics. In Alan Bundy, editor, *Proc. Int. Conf. on Automated Deduction (CADE'94)*, number 814 in LNAI, pages 723–737. Springer-Verlag, 1994.

[44] N. McCain and H. Turner. A causal theory of ramifications and qualifications. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1978–1984, 1995.

[45] N. McCain and H. Turner. Causal theories of action and change. In *Proc. Nat. (US) Conf. on Artificial Intelligence (AAAI'97)*, pages 460–465, 1997.

[46] J. McCarthy. Epistemological problems of artificial intelligence. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI'77)*, pages 1038–1044, 1977.

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

[48] S. Passy and T. Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93:263–332, 1991.

[49] S. Popkorn. *First steps in modal logic*. Cambridge University Press, 1994.

[50] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. *Artificial Intelligence and Mathematical Theory of Computation*, Papers in Honor of John McCarthy:359–380, 1991.

[51] R. Reiter. Knowledge in action: Logical foundations for describing and implementing dynamical systems. Version of january, 5, 1998.

[52] S. Rosenschein. Plan synthesis: a logical approach. In *Proc. of the 8th International Joint Conference on Artificial Intelligence (IJCAI'81)*, pages 359–380, Academic Press, 1981.

[53] H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logics. In S. Kanger, editor, *Proc. 3rd Scandinavian Logic Symposium 1973*. North-Holland, 1973. Studies in logic 82 (1975).

[54] E. Sandewall. *Features and Fluents: the Representation of Knowledge about Dynamical Systems*, volume 1. Oxford University Press, 1994.

[55] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91)*, pages 466–471, Sydney, Australia, 1991.

[56] C. Schwind. A logic based framework for action theories. In J. Ginzburg, Z. Khasidashvili, C Vogel, J.-J. Lévy, and E. Vallduv, editors, *Logic, Language and Computation*, pages 275–291. CSLI publications, Stanford, USA, 1997.

[57] K. Segerberg. Getting started: Beginnings in the logic of action. *Studia Logica*, 51:347–378, 1992.

[58] M. Shanahan. *Solving the frame problem: a mathematical investigation of the common sense law of inertia*. The MIT Press, 1997.

[59] W. Stephan and S. Biundo. A new logical framework for deductive planning. In *Proc. of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, pages 32–38, 1993.

[60] M. Thielscher. Computing ramifications by postprocessing. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1994–2000, Montreal, Canada, 1995.

[61] M. Thielscher. Ramification and causality. *Artificial Intelligence*, 89:317–364, 1997.