

Representing Ramifications in an Event-based Language

Kristof Van Belleghem Marc Denecker
Daniele Theseider Dupré

Report CW 257, December 1997



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Representing Ramifications in an Event-based Language

*Kristof Van Belleghem** *Marc Denecker†*
Daniele Theseider Dupré

Report CW257, December 1997

Department of Computer Science, K.U.Leuven

Abstract

In the last couple of years, several high-level languages have been proposed for modeling actions and change, following the example of Gelfond and Lifschitz's \mathcal{A} language. The goal of these languages is to analyse in a simplified context the fundamental issues in reasoning about actions and the constructs required for dealing with them. In this paper we present a narrative-based language \mathcal{ER} with a linear time structure, which is designed to deal with general fluent dependencies, in particular ramifications, both of consecutive and simultaneous actions. We argue that a combination of state constraints, effect propagation rules (causal laws) and action precondition rules is necessary to correctly represent all such dependencies. In particular we introduce causal laws involving *complex fluent formulae* and show such laws to be useful for a compact representation of ramifications of both single and simultaneous actions. We motivate and define a constructive semantics for causal laws based on the principle of inductive definitions. In addition, we design \mathcal{ER} to deal in a flexible way with complete and incomplete knowledge on action occurrences, action ordering and the initial state of the world. We present a mapping of \mathcal{ER} to Open Logic Programming using the Event Calculus, and prove its correctness. Two extensions to \mathcal{ER} introduce constructs for dealing with nondeterminism and delayed ramifications; the mapping to Open Logic Programming is extended accordingly. Finally we indicate how influence information as introduced by Thielscher can be used for deriving causal laws from state constraints (semi-)automatically in the \mathcal{ER} setting. We discuss how our approach compares with and differs from Thielscher's.

*Supported by Vlaams Instituut voor de Bevordering van het Wetenschappelijk-Technologisch Onderzoek in de Industrie (IWT)

†Supported by GOA 93/97-03

1 Introduction

Recently a number of high-level languages have been proposed for modeling actions and change, as a tool for studying the principles underlying time and causality in particular simplified settings. The first of these languages to emerge, the \mathcal{A} language of [10], models inertia and direct effects of actions in a branching time topology, with possible uncertainty on the initial state of the world. Extensions of \mathcal{A} tackle gradually more complex issues: for example \mathcal{AR}_0 ([16]) deals with indirect effects of actions (ramifications) and simple forms of non-determinism. Another formalism, the \mathcal{E} language ([15]), uses an event-based ontology modelled after a variant of the Event Calculus. \mathcal{E} allows for modeling uncertainty on the initial state of the world and includes an initial idea on dealing with some ramifications, which is currently being developed further. The authors have also devised extensions of \mathcal{E} to represent scenarios with incomplete knowledge on action ordering or action occurrences.

The language we design in this paper will be named \mathcal{ER} . Though it is based, like \mathcal{E} , on a variant of the Event Calculus, it has little formal correspondence with \mathcal{E} . The main goal of the language is to correctly represent a very general set of indirect effects (ramifications), both of consecutive and simultaneous actions. Moreover the language is intended to deal in a flexible way with complete and/or incomplete knowledge on action occurrences, action ordering or the initial state. The language is also further extended to deal with nondeterminism and delayed effects of actions.

We argue that to represent all ramifications and qualifications of actions, it is necessary to include in the language *state constraints* as well as effect propagation rules (*derived effect rules*¹) and explicit *action preconditions*: these three types of formulae are at least in part independent, i.e. there are derived effect rules that do not correspond to any state constraint and vice versa, and there are action preconditions not related to a state constraint. We motivate this and indicate differences between our approach and those in the recent literature.

We also argue that in many applications a clear and natural representation of indirect effects of actions in general and of the effects of simultaneous actions in particular can be obtained by using complex derived effect rules, i.e. effect rules stating that a change is triggered by the change in truth value of a *complex fluent formula*. \mathcal{ER} includes such complex derived effect rules. A complete treatment of simple and complex derived effect rules requires relying on a strong semantics. We base the semantics of \mathcal{ER} on the principle of inductive definitions. This principle yields at the same time the required expressiveness to deal with the frame and ramification problems even in the presence of negative and possibly cyclic dependencies between effects, while it has the advantage that the intuitions underlying the formal semantics (i.e. inductive definitions) are generally well-understood. Moreover we show that for restricted classes of definitions (like loop-free definitions or definitions without negations) for which simpler semantics have been proven adequate, the inductive definition semantics coincides with these semantics.

We assume in the first five sections of this paper that actions are deterministic, have no duration, and can be simultaneous, and that all changes are discrete. In this setting we intend to deal correctly with all immediate ramifications, i.e. all ramifications occurring at the time of the action(s) they are ramifications of. Moreover we should deal correctly with action qualifications that are entailed by the theory. We do not handle default qualifications, as in our view dealing with defaults is an entirely different problem than the inertia and ramification

¹similar to “causal laws” or “causal rules” in the literature; we will use these terms as synonyms except where otherwise indicated

problems and not typical for temporal domains.² Once the basic language is established, we further extend \mathcal{ER} to deal with delayed ramifications and with nondeterministic actions and ramifications.

A number of the issues tackled by \mathcal{ER} have already been addressed in one or more other approaches to representing actions. In \mathcal{ER} we want to tackle all of these issues in one coherent framework and at the same time address some unsolved problems. Meanwhile we want to keep the formalism and its intuitive (if not the formal) semantics simple. As we are dealing with a kind of “common sense” reasoning and representation, in our view the best way to meet these goals is to design the language with the general principle in mind to stay as close as possible to the intuitions about “real” time, actions, change and causality. This principle motivates, among others, the decisions of representing a large part of the theory as simple first order logic (there is no reason for any more complicated choice, and FOL is well-suited for representing incomplete knowledge), representing effects of actions as an inductive, constructive definition (which is how we intuitively interpret them) and choosing a time topology of actual events occurring on a single time line (as *real* time is most naturally seen as just one infinite line)³.

We provide a mapping of \mathcal{ER} to the Event Calculus in Open Logic Programming under justification semantics ([5]), an extension of the well-founded semantics to logic programs with undefined predicates and FOL axioms.

We extensively compare the \mathcal{ER} -approach to the ramification problem with the one in [31], which shows most similarities to it, and in somewhat less detail with most other recent approaches. Finally, we study the idea introduced in [31] of using influence information in a tool for (semi-)automatically deriving causal laws from state constraints. We propose an alternative approach to this problem in the setting of \mathcal{ER} and we illustrate the differences with Thielscher’s proposal.

In the next section, we present the syntax of \mathcal{ER} and motivate the design of the language in much detail. Section 3 discusses how \mathcal{ER} tackles the ramification problem and defines the semantics of \mathcal{ER} . Section 4 contains a couple of detailed examples and sheds some light on particular contributions. A mapping to OLP Event Calculus is presented and proven correct in section 5. In section 6 we extend the language to deal with nondeterministic actions and ramifications. Subsequently a comparison with Thielscher’s approach is provided in section 7, and a method for using influence information in section 8. Delayed effects of actions are introduced in section 9. In the two final sections we discuss more related work and conclude.

2 The Syntax of \mathcal{ER}

Basically, an \mathcal{ER} -theory consists of a set of effect rules determining direct and indirect effects of actions (a theory of causation), combined with a general first order theory describing the truth of fluents at certain times, the occurrence and order of actions, and general state constraints and action preconditions. Formally we define the following syntax :

Definition 2.1 (\mathcal{ER} -signature)

An \mathcal{ER} -signature Σ is a tuple $\langle \text{Sorts}, \text{Functors}, \text{Vars} \rangle$ with

- **Sorts** = $\{T, A, F, P\}$, representing the sorts time, action, fluent and atom.

²However, ideas to deal with the default qualification problem can be found in [30].

³However, for a discussion on the choice between a linear and a branching time topology we refer to [34], where we discuss this issue in much detail.

- **Functors** consists of
 - a set \mathbf{T} of constants of sort \mathcal{T} , denoted t, t_1, \dots , which includes all real numbers;
 - a set \mathbf{A} of constants of sort \mathcal{A} , denoted a, a_1, \dots ;
 - a set \mathbf{F} of constants of sort \mathcal{F} , denoted f, f_1, \dots ;
 - four typed predicate symbols⁴:
 - Happens** : $\mathcal{A} \times \mathcal{T} \rightarrow \mathcal{P}$;
 - \leq : $\mathcal{T} \times \mathcal{T} \rightarrow \mathcal{P}$;
 - Initially** : $\mathcal{F} \rightarrow \mathcal{P}$;
 - Holds** : $\mathcal{F} \times \mathcal{T} \rightarrow \mathcal{P}$.
- $\mathbf{Vars} = \mathbf{Vars}_{\mathcal{A}} \cup \mathbf{Vars}_{\mathcal{T}}$, disjoint infinite sets of variables of sort \mathcal{A} resp. \mathcal{T} , denoted as A, A_1, \dots resp. T, T_1, \dots

Definition 2.2 (terms, fluent formulae, general atoms) Terms are constants or variables. Terms of sort \mathcal{T} will be denoted by τ , action terms by α . A fluent literal l is either a fluent constant f or its negation $\neg f$. We define $\widehat{\mathbf{F}}$ as the set of fluent literals. A fluent formula F is any expression that can be constructed using fluent constants and the operators \neg, \wedge, \vee ($\rightarrow, \leftarrow, \leftrightarrow$ can also be used for convenience); in addition we assume that true and false are special fluent formulae). For any F, α, τ and τ' , the atoms **Holds**(F, τ), **Happens**(α, τ), $\tau \leq \tau'$ and **Initially**(F) are allowed general atoms.

Definition 2.3 (\mathcal{ER} -formulae) \mathcal{ER} formulae based on Σ are:

- **direct effect rules** of the form

$$a \text{ causes } l \text{ if } F'$$

representing that l becomes true whenever a is executed at a time when F' holds;

- **derived effect rules** of the form

$$\text{initiating } F \text{ causes } l \text{ if } F'$$

representing that l becomes true whenever F changes to true at a time when F' holds,⁵

- any sentence constructed in the usual way of **Holds**, **Happens**, **Initially** and \leq atoms and the connectives and quantifiers $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \forall$ and \exists .

Some classes of sentences are of particular importance:

- **state constraints** of the form

$$\forall T : \mathbf{Holds}(F, T)$$

- **action preconditions** of the form

$$\forall T : \mathbf{Happens}(a, T) \rightarrow \mathbf{Holds}(F, T)$$

⁴In addition, we assume an equality predicate for \mathcal{A} and \mathcal{T} and we assume $t, f \in \mathcal{P}$

⁵We will call a or F the body, l the head and F' the condition of an effect rule.

Other sentences may state complete or incomplete information about the truth of fluents at certain times, the occurrence and order of actions, or the initial state.

Definition 2.4 (\mathcal{ER} -theory) *An \mathcal{ER} -theory is a tuple $\langle \Sigma, \Pi_e, \Pi_p \rangle$ such that Σ is an \mathcal{ER} -signature, Π_e is a set of direct or derived effect rules based on Σ , Π_p is a set of sentences based on Σ .*

Now, let us explain and motivate the types of formulae proposed above. Direct effect rules are necessary constructs in even the simplest theories of action. They represent the simple immediate effects of all actions. Additional constructs are required when more complex issues are to be dealt with. Action preconditions for example are important when addressing the qualification problem, an issue to which we return later. They represent necessary and sufficient conditions that must be satisfied for an action to be able to occur.

Most importantly, we require constructs for addressing the ramification problem. A straightforward example of a ramification (i.e. an indirect effect) is that applying momentum to a gear wheel, which has the direct effect that the wheel starts turning, can also have the indirect effect that other wheels connected to it start turning. We need constructs allowing to correctly determine the complete set of direct and indirect effects an action gives rise to.

In the literature, ramifications have generally been considered strongly related to state constraints. In fact, the ramification problem has sometimes been defined as dealing with indirect effects *due to* state constraints. We prefer not to restrict ourselves to this subset, as we will argue that other indirect effects are just as important in practice. A state constraint is a fixed relation between fluents that needs to hold at all times. For example, if two gear wheels are connected, a state constraint is that at any time point they must be either both turning or both stationary. This relates to indirect effects as follows: if an action's direct effects result in a violation of a state constraint (like making one gear wheel turn in the above example), this may give rise to indirect effects restoring the validity of the constraint (e.g. the other gear wheel will start turning as well).

Evidently, state constraints should not always give rise to ramifications. For example, suppose a person can be a manager only if he/she has a PhD. Presumably, we do not want a person to get a PhD as a side effect of being promoted. Rather we intend promotion to be prohibited for a person who has no PhD. In this case, the state constraint imposes an implicit precondition on the promotion. So, state constraints play an important role both as preconditions of actions and as causes of indirect effects. However, they are not sufficient for dealing with either problem.

It has been argued convincingly in for example [22] that state constraints are insufficient to convey all of the information required to determine valid sets of effects: it is unclear in general if an action violating a state constraint will give rise to indirect effects or if this action is simply impossible ([13]). Also, as argued in [12], [19] and [24], there can be multiple sets of indirect effects able to restore the validity of a state constraint, and the intended set cannot be determined without additional information. There exists in other words no automated, domain-independent method to derive the ramifications and qualifications corresponding to an arbitrary set of state constraints. To cure this problem, *causal laws* in some form or other have been proposed ([22, 15, 31, 14, 20]) to represent ramifications. Causal laws are explicit rules describing that certain changes in fluents cause (or may cause) certain other changes: an example would be a rule stating that making one gear wheel turn results in the

turning of the other wheel. Thus causal laws provide a direct characterisation of the possible ramifications.

Interestingly, in all existing approaches incorporating them, causal laws are still tightly coupled with state constraints: in all aforementioned approaches they are used as a way of restoring integrity of some explicit or implicit state constraint.⁶ We argue that this is an unnecessary and undesirable limitation: there is no reason why indirect effects should always correspond to a state constraint, as state constraints are not the “cause” of ramifications. In our view ramifications are simply manifestations of effect propagations. They represent some physical or logical force causing particular effects when certain other effects occur: for example when one gear wheel starts turning, it will give rise to physical forces making other wheels turn.⁷ In this view state constraints, like in the above example that both wheels are turning or both are stationary, arise often as a *consequence* of particular effect propagation patterns. This does not in any way diminish the importance of state constraints, as they capture in a very concise and natural way a lot of information about a particular domain. However, it is wrong to assume the whole domain revolves around them: there is a lot going on which is not reflected in state constraints, these are only one high-level manifestation of the underlying mechanisms.

As an example, consider an alarm system that detects if somehow people enter a building. We assume the building has many possible entrances (doors, windows, and possibly unexpected ways of getting in). So, there are many actions able to bring someone in the building and these actions may not even all be known.⁸ We formalise the system using the fluents *in* (stating that there is someone inside), *active* (the alarm system is active) and *ring* (the alarm bell is ringing). While the system is active, anyone entering the building triggers the alarm: if *in* becomes true when *active* is already true, *ring* becomes true. In \mathcal{ER} this reads

initiating *in* causes *ring* if *active*

However, the corresponding state constraint $\forall T : \mathbf{Holds}(in \wedge active \rightarrow ring, T)$ is not valid, since activating the alarm system when someone is already in the building is not supposed to cause the bell to ring. Moreover we can assume that the proposed constraint may also be violated by shutting down the bell, without deactivating the alarm system. So, there is no state constraint related to this triggered effect, it is simply caused by a different change.

A maybe even more obvious example of a system incorporating indirect effects unrelated to state constraints, is a simple electronic counter. Let us say the events it counts are represented by a transition of a certain voltage from low to high. This could be represented by rules like

initiating *volt*₁ causes *count*(*n* + 1) if *count*(*n*)
initiating *volt*₁ causes \neg *count*(*n*) if *count*(*n*)

Also this indirect effect is in no way related to a state constraint. What is going on is just a propagation of effects, one effect triggering another one.

The above kinds of indirect effects cannot be correctly modelled by the existing approaches for dealing with ramifications. Therefore, we propose the following approach. We use independent effect propagation rules for representing ramifications. These look like the causal laws used in the literature, but

⁶We discuss the above approaches in more detail in section 10.

⁷However, the propagation is not necessarily a physical one: for example someone who stops being alive also starts being dead, which can be considered a “logical” effect propagation.

⁸Hence the use of an explicit rule relating the alarm bell to someone’s presence in the building cannot be circumvented by adding new direct effect rules for each action (which would be an undesirable approach in any decent knowledge representation system in any case).

differ essentially from them in that their semantics is independent of any state constraints in the theory (as opposed to the semantics of causal rules in [31]), and in that they do not include an implicit state constraint themselves (as opposed to causal laws in [20], [22], [15], and [14]). Thanks to this uncoupling of state constraints and derived effect rules, a wider range of indirect effects, including those in the above examples, can be modelled.

Given the direct and derived effect rules in an \mathcal{ER} -theory and a particular action or set of actions executed in a certain state, the resulting state after this action or set of actions is uniquely determined: actions produce some changes, which may lead to more changes by propagation through the derived effect rules. The role of state constraints is then reduced to filtering out models violating any state constraint in any state, which results in implicit action preconditions: if a state constraint would be violated by the combined direct and indirect effects of an action executed in a particular state, any interpretation in which that action occurs is not a model of the theory. Hence the action is impossible in that state.⁹ Given a particular state constraint, whether or not a state violating the constraint will arise as a result of a particular action occurrence depends on the presence or absence in the theory of derived effect rules related to the constraint (which may restore its validity). This “restorability” can in turn be dependent on how the constraint was violated in the first place. For example, we know a dead turkey cannot be walking:

$$\forall T : \text{Holds}(\textit{walking} \rightarrow \textit{alive}, T)$$

On the one hand, when a walking turkey dies, we know it will also stop walking. This is modelled by the derived effect rule :

initiating \neg alive causes \neg walking if true

On the other hand, one cannot resurrect a dead turkey by making it walk, so the rule **initiating walking causes alive if true** is not intended. In the absence of this rule, an action which makes *walking* true violates the state constraint if *alive* is false. Hence such an action, for example *start_walk*, is then impossible: the state constraint functions as an implicit precondition.¹⁰

Apart from state constraints, \mathcal{ER} also includes explicit action preconditions for dealing with qualifications. This is necessary because like indirect effects, also action preconditions are not necessarily related to state constraints. For example, in a chess game a move is only possible if the moved piece is initially on the starting position of the move. This cannot be represented by a state constraint, hence explicit preconditions are required. On the other hand, it is also undesirable to omit state constraints altogether, as they provide a very concise and natural way of representing information. To take the chess example again, a state constraint is that the player who has just made a move may not be in check. Compiling this constraint into a set of explicit move preconditions is certainly not the way to go.

The above discussion motivates the presence of state constraints, preconditions and derived effect rules in \mathcal{ER} . Now we still need to motivate the form of the derived effect rules, in particular why we need complex fluent formulae in the body of these rules. The first reason is conciseness: as we will illustrate,

⁹[21] describes an automated technique which, for a given set of direct effect rules of actions, derives from an arbitrary state constraint an equivalent explicit action precondition axiom. However, this technique does not take indirect effects into account.

¹⁰[31] describes an automatic way of deriving the intended causal rules related to a state constraint, using additional *influence information*. For derived effect rules corresponding to state constraints, we can use a variant of this method in \mathcal{ER} . We discuss this in section 8.

complex derived effect rules offer a very concise and natural way of representing indirect effects of actions. Strongly related to this is the observation that such rules, since they are triggered by combinations of effects, are perfectly suited for dealing with simultaneous actions. The issue of simultaneous actions will be discussed in section 4. Here we show the general applicability of complex derived effect rules.

As an example we present the suitcase domain from [20]. A suitcase is equipped with a spring mechanism which opens the suitcase when its two latches are open at the same time. This can happen in several different ways: both latches may be opened simultaneously, or one latch may already be open when the second one is opened. In the latter case, we need to ensure that the open latch is not just closed as the closed one is opened. Using a complex fluent formula, this set of possibilities can be represented by one derived effect rule

initiating $l_1 \wedge l_2$ causes *open* if *true*

where l_1, l_2 represent that latch 1 resp. 2 are open and *open* that the suitcase is open. Without complex fluent formulae, at least three rules would be needed, and they would need to be able to represent explicit absences of initiations.¹¹

With action preconditions, complex effect rules, and state constraints, we are able to characterise the general laws ruling a temporal domain. Apart from that, we need to represent scenario information in such a domain, like actual action occurrences, an initial state, known fluent values at certain time points. We choose to represent these as a standard first order theory, as a part of Π_p : first of all because this is the simplest approach, and second because first order logic allows us to deal easily with incomplete scenario information. We illustrate this in section 4.

3 The Semantics of \mathcal{ER}

In this section we discuss and define the semantics of \mathcal{ER} . As indicated, the most important concern is solving the frame and ramification problems. This is usually done by means of an inertia axiom stating that fluents persist unless they are changed, in combination with a representation of the closed world assumption, also described as a “minimisation of change”. This can be achieved by using a circumscription policy or techniques extending Clark completion.

It is unclear to us if a variant of circumscriptive minimisation can yield a general solution to the frame and ramification problems. The many increasingly complex variants proposed to date suggest that such a general solution is not evident, even though distinct variants yield solutions for particular restricted classes of theories. The reason for these problems is in our view that the idea that change has to be minimised in some way is only an approximation of the “inertia” we observe in the real world. It does not entirely correspond to our intuition.

Clark completion in turn formalises a simple and intuitive principle (“a change occurs if and only if we say so”), but is only applicable to a very restricted class of theories (i.e. theories without recursion in the effect rules). On the other hand, more powerful extensions of Clark completion exist, and are used in the definition of more advanced logic programming semantics like stable, well-founded or justification semantics. We will apply such an extension of Clark completion to \mathcal{ER} ’s effect rules.

¹¹Lin also utilises complex causal laws in [20]. However, these laws differ from our derived effect rules in that they incorporate a state constraint component.

Intuitively, the semantics we propose for a set of effect rules is to read them as an inductive definition of a predicate *causes*. Provided there are no cycles in the effect rules, this simply coincides with their completion. However, we argue that cyclic dependencies naturally occur in effect rules, and that there is a natural way to deal with them.

Consider two connected gear wheels. Any action which makes one gear turn, makes the other one turn as well, and any action which stops one gear, stops the other one. This can be represented by the following rules:

initiating *turning*₁ **causes** *turning*₂ **if true**
initiating \neg *turning*₁ **causes** \neg *turning*₂ **if true**
initiating *turning*₂ **causes** *turning*₁ **if true**
initiating \neg *turning*₂ **causes** \neg *turning*₁ **if true**

which introduce a cyclic dependency, though the example is certainly not far-fetched or unnatural. Evidently, given such mutually dependent effects, no effect should take place unless some exterior effect causes it (e.g. in the example a motor is started). In other words we comply with Shoham ([29]) who insists that causation is anti-reflexive, i.e. that causes for a fact should never include the fact itself. However, we claim that this condition should not be enforced by ruling out cycles in the causal rules on a syntactic level: the example shows that such cycles naturally arise in quite normal problem domains. Rather, cyclic dependencies should be given their intuitive meaning, which is that if and only if one of the mutually dependent effects has an “external cause”, both of them occur.

Negative dependencies (in the sense that the absence of a particular effect is a precondition for another effect to occur) do at first sight not occur in effect rules, but a closer look at the complex effect rules reveals that this is a false impression. Take the suitcase domain presented earlier, which contains the rule

initiating $l_1 \wedge l_2$ **causes** *open* **if true**.

The intended reading of this rule is that *open* is initiated if the conjunction $l_1 \wedge l_2$ becomes true. This can happen in three ways, intuitively

if l_1 is initiated and l_2 is initiated
if l_1 is initiated, l_2 holds and $\neg l_2$ is not initiated
if l_2 is initiated, l_1 holds and $\neg l_1$ is not initiated

In the last two cases, the initiation of $l_1 \wedge l_2$ depends on the absence of a primitive initiation, so there are negative dependencies. Below we will define the semantics of complex effect rules by mapping them to an equivalent set of primitive rules like those above and by interpreting these primitive rules as an inductive definition. These rules contain negative dependencies, yet giving them a natural semantics needs not be problematic: in the above example it is clear that when l_1 is initiated while l_2 holds, *open* is expected to be initiated rather than l_2 terminated¹². In general, the intended semantics is clear if the effects can be ordered in layers (*stratified*) such that each effect only depends negatively on more primitive effects, i.e. effects in lower layers.¹³

As we do not impose syntactic constraints ensuring stratifiability of a theory, in some cases a set of rules can have an ambiguous reading or be completely nonsensical. As an example of the first case, consider adding the rule

initiating $l_1 \wedge \neg open$ **causes** $\neg l_2$ **if true**

¹²We say a fluent f is terminated iff $\neg f$ is initiated.

¹³In the example, *open* is in a higher layer than l_1 and l_2 .

to the above example. Now the initiations of *open* and $\neg l_2$ depend negatively on each other. This leads to the following problem: assume $\neg open$, $\neg l_1$ and l_2 hold at a certain time, i.e. the suitcase is closed but one latch is open. Then the other latch is opened, i.e. l_1 is initiated. If now $\neg open$ would persist then according to the newly introduced rule $\neg l_2$ would be initiated. But if l_2 would persist, the original rule **initiating** $l_1 \wedge l_2$ **causes** *open* **if true** would force *open* to be initiated. Both possibilities are in accordance with the rules, and no non-ambiguous conclusion is possible. In this case one can at best argue that the effect is nondeterministic, but in our view nondeterminism, if intended, should be modelled explicitly and not follow from tricky combinations of rules which themselves do not hint at nondeterminism. We will introduce explicit nondeterministic rules in \mathcal{ER} in a later section.

The other case of a non-stratified definition, in which an effect negatively depends on itself, i.e. it occurs provided it does not occur, is clearly nonsensical. We handle both cases of non-stratified definitions by assigning an “undefined” truth value to effects that depend negatively on effects in the same layer. This truth value is interpreted as indicating an error in the definition, in the sense that the definition is not constructive. By dealing with non-constructive definitions in the indicated way, we avoid introducing complex syntactic restrictions ensuring stratification, and keep our approach general.

The above intuitions are formalised by the principle of inductive definition. This principle is well-suited for representing effect propagations due to its constructiveness: the truth of atoms propagates through definition rules like effects propagate due to physical or logical forces. Hence no atom can be true without a cause and a cause for a true atom can never depend on the atom itself.

3.1 Principle of Inductive Definition

The semantics and expressiveness of inductive definitions are studied in a sub-area of mathematical logic, the area of Iterated Inductive Definitions (IID) ([3, 23, 1]). We formalise this semantics in a different way and extend it to non-stratified definitions.

We need the following concepts.

Definition 3.1 ($\mathcal{V}_{\mathbf{P}, \leq_F}$) *Given a set of ground atoms \mathbf{P} , the set $\mathcal{V}_{\mathbf{P}}$ of (3-valued) valuations on \mathbf{P} is the set of all functions $\mathbf{P} \rightarrow \{\mathbf{t}, \mathbf{u}, \mathbf{f}\}$. On $\mathcal{V}_{\mathbf{P}}$, a partial order \leq_F is defined as the pointwise extension of the order $\mathbf{u} \leq_F \mathbf{t}$, $\mathbf{u} \leq_F \mathbf{f}$; more precisely, $\forall I, I' \in \mathcal{V}_{\mathbf{P}} : I \leq_F I'$ iff $\forall l \in \mathbf{P} : I(l) \leq_F I'(l)$.*

It is easy to prove that $\mathcal{V}_{\mathbf{P}, \leq_F}$ is a chain complete poset with least element \perp , the valuation which assigns \mathbf{u} to each atom.

Definition 3.2 (inductive definition) *Given a set of ground atoms \mathbf{P} , we define $\widehat{\mathbf{P}} = \mathbf{P} \cup \{\neg l \mid l \in \mathbf{P}\} \cup \{\mathbf{t}, \mathbf{f}\}$.¹⁴ Valuations can be naturally extended to $\widehat{\mathbf{P}}$. A definition rule in \mathbf{P} is an object $l \leftarrow B$ where $l \in \mathbf{P}$ and $B \subseteq \widehat{\mathbf{P}}$. l is called the head, B the body of the rule. A definition on \mathbf{P} is any set \mathcal{D} of rules in \mathbf{P} .*

Given \mathbf{P} and a definition \mathcal{D} on \mathbf{P} , we need to characterise a valuation $I_{\mathcal{D}}$ which defines the truth values of all atoms according to \mathcal{D} . In IID this involves stratifying the definition, but it has been argued in [5] that techniques inspired by logic programming semantics formalise the same intuitions in a more general and syntax independent way. We present this technique.

Definition 3.3 (proof tree) *A proof tree T for an atom $p \in \mathbf{P}$ is a tree of elements of $\widehat{\mathbf{P}}$ such that*

¹⁴ \mathbf{u} should never occur explicitly in a definition.

- the root of T is p
- for each non-leaf node n of T with immediate descendants B , “ $n \leftarrow B$ ” $\in \mathcal{D}$ or $B = \{\mathbf{f}\}$ (Hence, each atom has at least one (false) proof tree.)
- T is maximal, i.e. atoms occur only in non-leaf nodes. Leaf nodes then contain only \mathbf{t} , \mathbf{f} or negative literals.
- T is finite, i.e. contains no infinite branches

Given some valuation $I \in \mathcal{V}_{\mathbf{P}}$ of \mathbf{P} , for each $l \in \mathbf{P}$, we define its *supported value* w.r.t. I , denoted $SV_I(l)$, as the truth value proven by its “best” proof tree. Formally:

Definition 3.4 (supported value)

- $SV_I(l) = \mathbf{t}$ if l has a proof tree with all leaves containing true facts w.r.t. I ;
- $SV_I(l) = \mathbf{f}$ if each proof tree of l has a false fact w.r.t. I in a leaf;
- $SV_I(l) = \mathbf{u}$ otherwise; i.e. if each proof tree of l contains a non-true leaf, and some proof tree contains only non-false leaves.

For a definite definition \mathcal{D} , $I_{\mathcal{D}}$ is the valuation mapping each $p \in \mathbf{P}$ to $SV_{\perp}(p)$, i.e. each atom is mapped to its supported value (w.r.t. \perp). For non-definite definitions, $I_{\mathcal{D}}$ is obtained as a fixpoint of this operation:

Definition 3.5 ($\mathcal{P}I_{\mathcal{D}}$) The positive induction operator $\mathcal{P}I_{\mathcal{D}} : \mathcal{V}_{\mathbf{P}} \rightarrow \mathcal{V}_{\mathbf{P}} : I \rightarrow I'$ is defined such that $\forall p \in \mathbf{P} : I'(p) = SV_I(p)$.

It can be proven that this operator is monotonic and hence always has a least fixpoint $\mathcal{P}I_{\mathcal{D}} \uparrow$. This allows us to define $I_{\mathcal{D}}$ as:

Definition 3.6 Given $\langle \mathbf{P}, \mathcal{D} \rangle$, $I_{\mathcal{D}} = \mathcal{P}I_{\mathcal{D}} \uparrow$.

3.2 Tackling the Ramification Problem

We now show how to handle derived effect rules using an inductive definition based semantics. We present the approach in a language-independent way, such that it can be embedded in most currently existing formalisms like Situation Calculus, Event Calculus or \mathcal{A} style languages. We embed the approach in \mathcal{ER} in the next subsection.

The problem we want to deal with is the following: given the state of the world at a certain instant in time, a set of actions occurring at that time and a set of direct and derived effect rules, calculate the state of the world resulting after these actions. We denote a set of actions as A and a state as St . The set of all states is written as $\mathcal{S}t$.

We reduce direct and derived effect rules to an inductive definition of **Init** and **Causes**. These predicates intuitively denote strong and weak initiation, respectively: **Init**(A, St, l) means that l does not hold in St but holds in the successor state resulting after the application of the set of (simultaneous) actions A in St . **Causes**(A, St, l) means that l holds in the successor state of St after A , but possibly also already in St . The fluent literals l that are true in the successor state are the ones that are initiated and those that are true in St and of which the negation \bar{l} is not initiated.

The intended reading of an effect rule **initiating** F **causes** l **if** F' is that given F' , the change in truth value of F from false to true (strong initiation of F) causes l to become true if it was not already true (i.e. weakly initiates l). To formalise this for complex F we introduce the concept of a supporting set. This concept is based on a disjunctive normal form of the formula. Since a definition can have 3-valued interpretations, this normal form needs to be equivalence preserving under 3-valued FOL semantics. The 2-valued disjunctive normal form does not satisfy this property (since for example $F \wedge \neg F$ is not 3-valued equivalent to \mathbf{f}), but it is easy to derive a 3-valued variant:

Definition 3.7 (3-valued disjunctive normal form) *The 3-valued disjunctive normal form $3dnf(F)$ of a fluent formula F is obtained by applying the following rewriting rules to F or its constituents (if \leftrightarrow , \rightarrow or \leftarrow occur in F we assume they are rewritten in terms of \neg , \wedge , \vee as usual) until no further rules apply:¹⁵*

- replace $\neg\neg F$ by F
- replace $\neg(F \wedge G)$ by $\neg F \vee \neg G$
- replace $\neg(F \vee G)$ by $\neg F \wedge \neg G$
- replace $F \wedge (G \vee H)$ by $(F \wedge G) \vee (F \wedge H)$
- replace $F \wedge F$ by F
- replace $F \vee (F \wedge G)$ by F

The following properties can be proven. The rewriting process always terminates (since each step moves a \neg symbol inward, moves a \wedge symbol inward without moving a \neg outward, or eliminates \neg , \wedge or \vee symbols without moving \wedge or \neg outward). The resulting normal form $3dnf(F)$ is unique¹⁶, i.e. independent of the order in which subformulae are processed (this is easily verified by case analysis). $3dnf(F)$ is always a disjunction of conjunctions of literals (since otherwise one of the four first rules applies). $3dnf(F)$ is equivalent to F under 3-valued as well as 2-valued semantics (since all rewriting rules are equivalence preserving under either semantics). Finally, under either semantics, if F and G are equivalent then so are $3dnf(F)$ and $3dnf(G)$ (this follows immediately from the previous property).

We define the concept of supporting set as follows:

Definition 3.8 (supporting set) *Let F be a fluent formula and $F' = (l_1^1 \wedge \dots \wedge l_{n_1}^1) \vee \dots \vee (l_1^m \wedge \dots \wedge l_{n_m}^m)$ its 3-valued disjunctive normal form. A supporting set L of F is any set $\{l_1^i, \dots, l_{n_i}^i\}$, $\forall 1 \leq i \leq m$.*

A formula is true if and only if all literals of some supporting set of it are true. It follows that F is *initiated* iff F is not already true and for some supporting set L of F , all literals of some $L_i \subseteq L$ are initiated and all literals in $L_p = L \setminus L_i$ are true and not terminated. This leads to the formalisation below.

Definition 3.9 (notations)

In what follows $Ha(a, A)$ is the truth value of “ $a \in A$ ”, $Ho(l, St)$ is the truth value of “ $l \in St$ ”, and $Ho(F, St)$ for complex formulae is obtained from $Ho(l, St)$ for literals in the classical way. We define $\mathbf{Init}(A, St, L)$ as $\{\mathbf{Init}(A, St, l) \mid l \in L\}$ and $\mathbf{Causes}(A, St, L)$ as $\{\mathbf{Causes}(A, St, l) \mid l \in L\}$. Further, from now on we use the notation $\overline{f} = \neg f$, $\neg\overline{f} = f$, $\overline{L} = \{\overline{l} \mid l \in L\}$ for any set of literals L , and $\overline{P} = \{\neg p \mid p \in P\}$ for any set of \mathbf{Init} or \mathbf{Causes} atoms P .

¹⁵We assume commutativity and associativity are applied whenever needed.

¹⁶modulo commutativity and associativity

Definition 3.10 (definition induced by effect rules)

The definition induced by a rule “ a causes l if F ” is ¹⁷

$$\{\mathbf{Causes}(A, St, l) \leftarrow Ha(a, A), Ho(F, St) \mid A \subseteq \mathcal{A}, St \in St\}.$$

The definition induced by a rule “initiating F causes l if F' ” is

$$\begin{aligned} \{\mathbf{Causes}(A, St, l) \leftarrow & \overline{\mathbf{Init}(A, St, L_i)}, \overline{\mathbf{Init}(A, St, \overline{L_p})}, \\ & Ho(L_p, St), \neg Ho(F, St), Ho(F', St) \\ \mid & A \subseteq \mathcal{A}, St \in St \text{ and } L_i \cup L_p \text{ is a supporting set of } F\}. \end{aligned}$$

We define the definition induced by a set of effect rules Π_e as $\mathcal{D}_{init} = \mathcal{D}_g \cup \{\mathbf{Init}(A, St, l) \leftarrow \mathbf{Causes}(A, St, l), \neg Ho(l, St) \mid A \subseteq \mathcal{A}, St \in St, l \in \widehat{\mathbf{P}}\}$, where \mathcal{D}_g is the union of the definitions induced by all rules in Π_e .

\mathcal{D}_{init} is an inductive definition on the atom domain $\mathbf{P}' = \{\mathbf{Init}(A, St, l), \mathbf{Causes}(A, St, l) \mid A \subseteq \mathcal{A}, St \in St, l \in \widehat{\mathbf{F}}\}$, for which $I_{\mathcal{D}_{init}}$ is the least fixpoint of $\mathcal{PT}_{\mathcal{D}_{init}}$.

The rules $\mathbf{Init}(A, St, l) \leftarrow \mathbf{Causes}(A, St, l), \neg Ho(l, St)$ are the only rules for **Init**. Since the completion of the definition rules is entailed by the inductive definition semantics ([4]), the rules imply

$$\forall A, St, l : [\mathbf{Init}(A, St, l) \leftrightarrow \mathbf{Causes}(A, St, l) \wedge \neg Ho(l, St)]$$

and thereby capture the intended relation between strong and weak initiation.

The mutual recursion in the definitions of **Init** and **Causes** indicates that strong initiations may provide causes for literals to become true; only if these literals were not already true, i.e. if they were actually changed, they can themselves give rise to further ramifications. We have used the predicates **Causes** and **Init** to stress this important distinction between strong and weak initiation. However, given the above relation between **Causes** and **Init**, we could get rid of the **Init** predicate using the following theorem:

Theorem 3.1

Given that $\forall A, St, l : [\mathbf{Init}(A, St, l) \leftrightarrow \mathbf{Causes}(A, St, l) \wedge \neg Ho(l, St)]$, the definition $D_1 =$

$$\begin{aligned} \{\mathbf{Causes}(A, St, l) \leftarrow & \overline{\mathbf{Init}(A, St, L_1)}, \overline{\mathbf{Init}(A, St, \overline{L_2})}, \\ & Ho(L_2, St), \neg Ho(F, St), Ho(F', St) \\ \mid & A \subseteq \mathcal{A}, St \in St \text{ and } L_1 \cup L_2 \text{ is a supporting set of } F\}. \end{aligned}$$

is equivalent to $D_2 =$

$$\begin{aligned} \{\mathbf{Causes}(A, St, l) \leftarrow & \overline{\mathbf{Causes}(A, St, L_1)}, \overline{Ho(\overline{L_1}, St)}, \overline{\mathbf{Causes}(A, St, \overline{L_2})}, \\ & Ho(L_2, St), \neg Ho(F, St), Ho(F', St) \\ \mid & A \subseteq \mathcal{A}, St \in St \text{ and } L_1 \cup L_2 \text{ is a supporting set of } F\}. \end{aligned}$$

Intuitively, this is true because if at least one rule body for a particular literal is true in D_1 in a particular state, then at least one rule body is true for the same literal in D_2 in the same state, and vice versa.

Proof:

We first prove the first part: assume a rule in D_1 with a particular L_1 and L_2 and which has a true body in St . $L_1 \cup L_2$ is a supporting set of F . For the literals in L_1 , it holds that $\mathbf{Init}(A, St, l)$, hence $\mathbf{Causes}(A, St, l) \wedge \neg Ho(l, St)$, which entails $\mathbf{Causes}(A, St, l)$. For the literals in L_2 it holds that $Ho(l, St) \wedge \neg \mathbf{Init}(A, St, \bar{l})$, hence $(Ho(l, St) \wedge \neg \mathbf{Causes}(A, St, \bar{l})) \vee (Ho(l, St) \wedge Ho(\bar{l}, St))$,

¹⁷Recall that $Ha(a, A)$ and $Ho(F, St)$ are the *truth values* of “ $a \in A$ ” and “ $l \in St$ ”, not the formulae themselves.

which is equivalent to $(Ho(l, St) \wedge \neg \mathbf{Causes}(A, St, \bar{l}))$. Combining these results, we find that the rule in D_2 with the same L_1 and L_2 has a true body.

On the other hand, assume a rule in D_2 with particular L_1 and L_2 and which has a true body in a particular St . For the literals in L_1 , $\mathbf{Causes}(A, St, l)$ holds. We can partition L_1 into a set of literals which are true in St (L_{1a}) and a set of literals which are false in St (L_{1b}).¹⁸ For the literals in L_{1b} , $\mathbf{Causes}(A, St, l) \wedge \neg Ho(l, St)$ is true, which is equivalent to $\mathbf{Init}(A, St, l)$. For the literals in L_{1a} , $\mathbf{Causes}(A, St, l) \wedge Ho(l, St)$ is true, which immediately implies $\neg \mathbf{Causes}(A, St, \bar{l}) \wedge Ho(l, St)$. For literals in L_2 , $Ho(l, St) \wedge \neg \mathbf{Causes}(A, St, \bar{l})$ holds, which implies $Ho(l, St) \wedge \neg \mathbf{Init}(A, St, \bar{l})$. Combining these results, we find that a D_1 -rule constructed with L_{1b} as the new L'_1 and $L_2 \cup L_{1a}$ as the new L'_2 has a true body, with $L'_1 \cup L'_2 = L_1 \cup L_2$ the same supporting set. \square

Despite this equivalence result, we prefer to use both the \mathbf{Init} and \mathbf{Causes} predicates explicitly for reasons of clarity.

We now give some interesting results concerning the proposed semantics.

Theorem 3.2 *Given a state St , a set of actions A and a set of direct and derived effect rules, the truth values of $\mathbf{Init}(A, St, l)$ for all l are uniquely determined.*

Proof:

This can be proven as follows: the definition induced by a specific set of effect rules is a definition in which (by its construction) the set of rules for $\mathbf{Init}(A, St, l)$ and $\mathbf{Causes}(A, St, l)$ for each l depends only on the truth values $Ha(a, A)$ and $Ho(F, St)$. Given St and A , these truth values are uniquely determined. Hence, the set of rules for all atoms of the form $\mathbf{Init}(A, St, l)$ or $\mathbf{Causes}(A, St, l)$ is unique. Moreover, in the entire set of rules for a given St and A , the supported values of $\mathbf{Init}(A, St, l)$ and $\mathbf{Causes}(A, St, l)$ only depend directly or indirectly on the truth values of other atoms of the form $\mathbf{Init}(A, St, l')$ and $\mathbf{Causes}(A, St, l')$ for the same St and A . Hence, the set of rules for each particular St and A is a separate definition determining the values of all atoms in $\{\mathbf{Init}(A, St, l), \mathbf{Causes}(A, St, l) \mid l \in \bar{\mathbf{P}}\}$ independent of all other states or actions. $I_{\mathcal{D}_{init}}$ for this definition is unique, which proves the theorem. \square

The theorem guarantees that successor states generated by a set of deterministic effect rules are always unique. In this respect our approach differs from the one in [31]. Unless nondeterminism is explicitly introduced, the theory leaves no room for ambiguity.

The following results give some alternative characterisations of the above semantics in several special cases. Moreover they shed some light on the relation to existing approaches to the frame and ramification problems.

Definition 3.11 (fluent dependency)

*We say a fluent f occurring in **initiating F causes f if F' or in **initiating F causes $\neg f$ if F' depends on a fluent f' if f' occurs in F , or if a fluent which depends on f' occurs in F .*****

Theorem 3.3 *If the derived effect rules are acyclic, i.e. if no fluent depends on itself, $I_{\mathcal{D}_{init}}$ is always 2-valued. Moreover $I_{\mathcal{D}_{init}}$ coincides with the unique model of the Clark completion of the definition rules.*

Proof:

The rules in the definition induced by a rule **initiating F causes l if F' contain only literals $\mathbf{Init}(A, St, l')$ or $\neg \mathbf{Init}(A, St, l')$ for l' which are fluent literals**

¹⁸ L_{1b} is not empty: if it were, all literals in the supporting set $L_1 \cup L_2$ would already be true, which is in contradiction with the condition in the rule body that F should not hold.

occurring in F or negations of such fluent literals. The dependency relation is therefore the same for fluents in the effect rules as for literals containing them in the grounding. Hence, if the effect rules are acyclic, then so are the definition rules in the grounding. As we mentioned before, the positive induction operator has the same fixpoints as the well-founded operator, so an inductive definition is formally equivalent with a logic program under well-founded semantics. Under this reading, acyclic definitions correspond to acyclic logic programs, for which the well-founded semantics has been proven to be 2-valued and to coincide with the completion semantics. The theorem follows immediately. \square

Theorem 3.4 *If the body of each derived effect rule is a single literal, $I_{\mathcal{D}_{init}}$ is always 2-valued, and coincides with the unique model of the parallel¹⁹ circumscription of **Init** and **Causes** in the theory consisting of the definition rules read as implications.*

Proof:

If the derived effect rules have only single literals as bodies, the rules in \mathcal{D}_{init} contain no negative literals. Hence, they are equivalent to a definite logic program, for which the well-founded semantics has been proven to coincide with the perfect model semantics of [25] in [35]. The perfect model semantics is 2-valued, which proves the first result. Note moreover that $I_{\mathcal{D}_{init}}$ is unique, so there is one unique perfect model.

Further, in [25] it is proven that for definite programs the perfect models coincide with the minimal models of the program read as a set of implications. On the other hand, in [17] it has been shown that an interpretation is a model of $Circum(A; P; Z)$ (where A is the given theory, P the set of predicates to be circumscribed and Z a set of predicates allowed to vary) iff it is minimal in the class of models of A with respect to $\leq^{P;Z}$. $M_1 \leq^{P;Z} M_2$ iff M_1 and M_2 differ only in the predicates in $P \cup Z$ and the extension of each predicate of P in M_1 is a subset of its extension in M_2 . In our case, A is the set of rules read as implications, Z is empty and P contains both **Init** and **Causes**, the only predicates occurring in the definition. Then $M_1 \leq^{P;Z} M_2$ iff the set of true atoms in M_1 is a subset of that in M_2 . Hence, the minimal models with respect to $\leq^{P;Z}$ are the minimal models of the program read as a set of implications. It follows that for simple effect rules, the model $I_{\mathcal{D}_{init}}$ coincides with the unique perfect model, which in turn coincides with the model of the circumscription. This proves the theorem. \square

The above results show that for several classes of definitions for which other semantics are known to assign the intended meaning to all predicates, the inductive definition semantics coincides with these semantics. However the inductive definition semantics is more general, also dealing with definitions that cannot be dealt with by the more common semantics.

3.3 Formal semantics of \mathcal{ER}

We are now ready to define the semantics of \mathcal{ER} , embedding the semantics of effect rules given above in the specific \mathcal{ER} setting.

Definition 3.12 (temporal interpretation) *Given $\Pi = \langle \Sigma, \Pi_e, \Pi_p \rangle$ an \mathcal{ER} -theory, a temporal interpretation of Π is a structure $I = \langle \mathbf{P}, \mathcal{F}un, \mathcal{H} \rangle$ with:*

¹⁹Parallel circumscription is circumscription on multiple predicates at the same time, without priorities.

$$\begin{aligned}
\mathbf{P} = & \{t_1 \leq t_2 \mid t_1, t_2 \in \mathbf{T}\} \cup \\
& \{\mathbf{Initially}(l) \mid l \in \widehat{\mathbf{F}}\} \cup \\
& \{\mathbf{Happens}(a, t) \mid a \in \mathbf{A}, t \in \mathbf{T}\} \cup \\
& \{\mathbf{Holds}(l, t) \mid l \in \widehat{\mathbf{F}}, t \in \mathbf{T}\} \cup \\
& \{\mathbf{Init}(t, l) \mid t \in \mathbf{T}, l \in \widehat{\mathbf{F}}\} \cup \\
& \{\mathbf{Causes}(t, l) \mid t \in \mathbf{T}, l \in \widehat{\mathbf{F}}\}
\end{aligned}$$

$Fun : \mathbf{T} \rightarrow \mathbf{R}$, a mapping of time constants to reals such that each real number is mapped to itself (recall that $\mathbf{R} \subseteq \mathbf{T}$)

$\mathcal{H} : \mathbf{P} \rightarrow \{\mathbf{t}, \mathbf{f}\}$, a valuation.

\mathcal{H} defines relations interpreting **Happens**, **Holds**, \leq , **Initially**, **Init**, **Causes**; we denote them $\mathcal{H}a, \mathcal{H}o, \preceq, \mathcal{I}nitially, \mathcal{I}nit, \mathcal{C}auses$ respectively.²⁰ The predicates **Init** and **Causes**, which do not occur in Π_p , denote strong and weak initiation, respectively: **Init**(t, l) means that l does not hold at t but starts to hold immediately after t . **Causes**(t, l) means that l holds immediately after t , but can also hold at t . These predicates are determined by Π_e .

A temporal interpretation needs to satisfy the following conditions

- \preceq is the classical total order on \mathbf{R} .
- Well-founded event topology: the set $\mathbf{E} = \{t \mid \exists a : \mathcal{H}a(a, t)\}$ has a least element, denoted e_{start} and $\forall t \preceq t' : [t, t'] \cap \mathbf{E}$ is a finite set.²¹
- Consistency: $\forall t, \forall f : \neg \mathcal{C}auses(t, f) \vee \neg \mathcal{C}auses(t, \neg f)$
 $\forall f : \mathcal{I}nitially(f) \leftrightarrow \neg \mathcal{I}nitially(\neg f)$
 These formulae denote initiation consistency and initial state consistency. Consistency ($\forall t, \forall f : \mathcal{H}o(f, t) \leftrightarrow \neg \mathcal{H}o(\neg f, t)$) follows from these two formulae and the inertia axiom given a well-founded event topology.
- Definition of initial state: $\forall t \leq e_{start} : \mathcal{H}o(l, t) \leftrightarrow \mathcal{I}nitially(l)$;
- Inertia: $\forall t_1, t_2, \forall l :$
 $t_1 \prec t_2 \wedge (\neg \exists t_3 : t_1 \preceq t_3 \prec t_2 \wedge \mathcal{C}auses(t_3, \bar{l}))$
 $\rightarrow \mathcal{H}o(l, t_2) \leftrightarrow \mathcal{C}auses(t_1, l) \vee \mathcal{H}o(l, t_1)$

A temporal interpretation I is a model of an \mathcal{ER} -theory $\langle \Sigma, \Pi_e, \Pi_p \rangle$ iff it is a model of both Π_e and Π_p . To define whether I satisfies Π_p , we extend the truth function \mathcal{H} to all closed formulae F in the classical way. For complex **Holds** and **Initially** atoms, the interpretation of **Holds**(F, τ) is defined as the interpretation of the formula F' obtained from F by substituting each fluent atom f by **Holds**($f, Fun(\tau)$). The interpretation of **Initially**(F) is defined likewise. An interpretation I satisfies Π_p iff all formulae in Π_p are true in I .

Next we focus on the semantics of the effect theory Π_e . This semantics is based on the inductive definition semantics for effect rules given above. At a particular time t , if the state St is the set of literals true at t , the truth value $\mathcal{H}o(F, St)$ corresponds to the truth value $\mathcal{H}o(F, t)$. Likewise, if A is the set of actions occurring at t , $\mathcal{H}a(a, A)$ corresponds to $\mathcal{H}a(a, t)$. **Init**(A, St, l) and **Causes**(A, St, l) then correspond to **Init**(t, l) and **Causes**(t, l).

²⁰Implicitly, \mathbf{P} contains also all well-typed equality atoms and \mathbf{t}, \mathbf{f} . \mathcal{H} defines their natural interpretation; in particular $=$ is interpreted as the identity. Moreover for all t , $\mathcal{H}o(true, t) = \mathcal{I}nitially(true) = \mathbf{t}$ and $\mathcal{H}o(false, t) = \mathcal{I}nitially(false) = \mathbf{f}$.

²¹This condition plays the same role as the induction axiom in for example [21]. The first part of the condition ensures that there is no sequence of events extending infinitely into the past. The second part is our *non-intermingling principle* ([9]) : it disallows an infinite number of actions (and hence, changes in truth value of a fluent) in a finite period of time. Together the two parts guarantee that each time point is only preceded by a finite number of actions.

Definition 3.13 (grounding)

The grounding of a direct effect rule “ a causes l if F ” is²²

$$\{\mathbf{Causes}(t, l) \leftarrow \mathcal{H}a(a, t), \mathcal{H}o(F, t) \mid t \in \mathbf{T}\}.$$

The grounding of a derived effect rule “**initiating** F causes l if F' ” is

$$\{\mathbf{Causes}(t, l) \leftarrow \mathbf{Init}(t, L_i), \overline{\mathbf{Init}(t, \overline{L_p})}, \\ \mathcal{H}o(L_p, t), \neg\mathcal{H}o(F, t), \mathcal{H}o(F', t) \\ \mid t \in \mathbf{T} \text{ and } L_i \cup L_p \text{ is a supporting set of } F\}.$$

The grounding \mathcal{D}_{init} of a set of effect rules Π_e is $\mathcal{D}_g \cup \{\mathbf{Init}(t, l) \leftarrow \mathbf{Causes}(t, l), \neg\mathcal{H}o(l, t) \mid t \in \mathbf{T}, l \in \widehat{\mathbf{P}}\}$, where \mathcal{D}_g is the union of the groundings of all rules in Π_e .

\mathcal{D}_{init} is an inductive definition on the atom domain $\mathbf{P}' = \{\mathbf{Init}(t, l), \mathbf{Causes}(t, l) \mid t \in \mathbf{T}, l \in \widehat{\mathbf{F}}\}$, for which $I_{\mathcal{D}_{init}}$ is defined as the least fixpoint of $\mathcal{P}\mathcal{I}_{\mathcal{D}_{init}}$.

Taking everything together now, we obtain the following definition of a model of an \mathcal{ER} -theory.

Definition 3.14 (\mathcal{ER} -model)

Given an \mathcal{ER} -theory $\Pi_{\mathcal{ER}} = \langle \Sigma, \Pi_e, \Pi_p \rangle$, a temporal interpretation I is a model of $\Pi_{\mathcal{ER}}$, denoted $I \models \Pi_{\mathcal{ER}}$, iff $I \models \Pi_p$ and $I \models \Pi_e$, where

$I \models \Pi_p$ iff $\forall F \in \Pi_p : \mathcal{H}(F) = \mathbf{t}$.

$I \models \Pi_e$ iff $\forall t \in \mathbf{T}, l \in \widehat{\mathbf{F}} :$

$$\mathcal{I}nit(t, l) \leftrightarrow I_{\mathcal{D}_{init}}(\mathbf{Init}(t, l)) \text{ and } \mathcal{C}auses(t, l) \leftrightarrow I_{\mathcal{D}_{init}}(\mathbf{Causes}(t, l)).$$

Note that when $I_{\mathcal{D}_{init}}$ contains any truth value \mathbf{u} , the condition $I \models \Pi_e$ is unsatisfiable since $\mathcal{I}nit$ is a 2-valued relation. Recall that definitions with a 3-valued model are ambiguous (non-constructive) and are considered erroneous in our approach.

3.4 Properties of \mathcal{ER}

The following results carry over from the inductive definition semantics given before:

The rules $\mathbf{Init}(t, l) \leftarrow \mathbf{Causes}(t, l), \neg\mathcal{H}o(l, t)$ are the only rules for **Init**. Since the completion of the definition rules is entailed by the inductive definition semantics, the rules imply

$$\forall t, l : [\mathbf{Init}(t, l) \leftrightarrow \mathbf{Causes}(t, l) \wedge \neg\mathcal{H}o(l, t)]$$

We then find an immediate corollary of theorem 3.1:

Corollary 3.1 Given that $\forall t, l : [\mathbf{Init}(t, l) \leftrightarrow \mathbf{Causes}(t, l) \wedge \neg\mathcal{H}o(l, t)]$, the definition $D_1 =$

$$\{\mathbf{Causes}(t, l) \leftarrow \mathbf{Init}(t, L_1), \overline{\mathbf{Init}(t, \overline{L_2})}, \\ \mathcal{H}o(L_2, t), \neg\mathcal{H}o(F, t), \mathcal{H}o(F', t) \\ \mid t \in \mathbf{T} \text{ and } L_1 \cup L_2 \text{ is a supporting set of } F\}.$$

is equivalent to $D_2 =$

$$\{\mathbf{Causes}(t, l) \leftarrow \mathbf{Causes}(t, L_1), \overline{\mathcal{H}o(\overline{L_1}, t)}, \overline{\mathbf{Causes}(t, \overline{L_2})}, \\ \mathcal{H}o(L_2, t), \neg\mathcal{H}o(F, t), \mathcal{H}o(F', t) \\ \mid t \in \mathbf{T} \text{ and } L_1 \cup L_2 \text{ is a supporting set of } F\}.$$

²²Observe that $\mathcal{H}a(a, t)$ and $\mathcal{H}o(F, t)$ are the truth values of “**Happens**(a, t)” and “**Holds**(F, t)”.

Proof:

The proof follows the same reasoning as that of theorem 3.1, with truth values determined by t instead of by St and A . \square

Another result, adapted from theorem 3.2, is the following.

Theorem 3.5 *Given a set of direct and derived effect rules, a particular set of values for all fluents at a particular time t , and the set of actions occurring at t , the truth value of $\mathbf{Init}(t, l)$ is uniquely determined for all l .*

Proof:

The grounding of a specific set of effect rules is a definition in which (by its construction) the rules for $\mathbf{Init}(t, l)$ and $\mathbf{Causes}(t, l)$ only depend on the truth values $\mathcal{H}a(A, t)$ and $\mathcal{H}o(F, t)$. Given all fluents and actions at time t , these truth values are uniquely determined, hence the set of rules for all atoms of the form $\mathbf{Init}(t, l)$ or $\mathbf{Causes}(t, l)$ is unique. Moreover in this set of rules, $\mathbf{Init}(t, l)$ and $\mathbf{Causes}(t, l)$ only depend directly or indirectly on other atoms of the form $\mathbf{Init}(t, l')$ and $\mathbf{Causes}(t, l')$. Hence these rules form a complete definition on the atom domain $\{\mathbf{Init}(t, l), \mathbf{Causes}(t, l) \mid l \in \widehat{\mathbf{P}}\}$. $I_{\mathcal{D}_{init}}$ for this definition is unique, which proves the theorem. \square

The following statement is an immediate corollary:

Corollary 3.2 *Given a completely determined initial state, a complete list of action occurrences and a particular set of effect rules, the truth value of all fluents at all time points is uniquely determined.*

Proof:

The proof follows from the above theorem by induction on the well-founded event topology, since the truth value of each fluent at the time of each event is uniquely determined by the truth values and initiations at the time of the preceding event. \square

Finally, the results for acyclic rules and rules with only simple literals in the body also carry over immediately from the corresponding theorems above, with identical proofs. This illustrates how our semantics relates to existing approaches tackling the frame and ramification problems.

4 Examples of Various \mathcal{ER} -contributions

In this section, we formalise and discuss in more detail a few examples chosen to illustrate the claimed expressive power of \mathcal{ER} , in particular by illuminating some contributions we have not yet discussed in detail.

4.1 Simultaneous Actions

First of all, we have claimed that representing effects of simultaneous actions in \mathcal{ER} can be done in a very natural and concise way. A nice example is found in [11]: a glass on a table spills its contents as soon as the table is in a non-horizontal position. This can be straightforwardly represented using complex derived effect rules, as

$$\mathbf{initiating} \neg(up_l \leftrightarrow up_r) \mathbf{causes} \mathit{wet} \mathbf{if} \mathit{on_table}$$

where up_l , up_r represent that the left resp. right side of the table are lifted from the floor and wet that the table is wet. Recall that the grounding of this rule

contains

$$\begin{aligned}
\mathbf{Causes}(t, wet) &\leftarrow \mathbf{Init}(t, up_l), \mathbf{Holds}(\neg up_r, t), \neg \mathbf{Init}(t, up_r), \\
&\quad \mathbf{Holds}(up_l \leftrightarrow up_r, t), \mathbf{Holds}(on_table, t). \\
\mathbf{Causes}(t, wet) &\leftarrow \mathbf{Init}(t, up_r), \mathbf{Holds}(\neg up_l, t), \neg \mathbf{Init}(t, up_l), \\
&\quad \mathbf{Holds}(up_l \leftrightarrow up_r, t), \mathbf{Holds}(on_table, t). \\
\mathbf{Causes}(t, wet) &\leftarrow \mathbf{Init}(t, \neg up_r), \mathbf{Holds}(up_l, t), \neg \mathbf{Init}(t, \neg up_l), \\
&\quad \mathbf{Holds}(up_l \leftrightarrow up_r, t), \mathbf{Holds}(on_table, t). \\
\mathbf{Causes}(t, wet) &\leftarrow \mathbf{Init}(t, \neg up_l), \mathbf{Holds}(up_r, t), \neg \mathbf{Init}(t, \neg up_r), \\
&\quad \mathbf{Holds}(up_l \leftrightarrow up_r, t), \mathbf{Holds}(on_table, t). \\
\mathbf{Causes}(t, wet) &\leftarrow \mathbf{Init}(t, up_l), \mathbf{Init}(t, \neg up_r), \\
&\quad \mathbf{Holds}(up_l \leftrightarrow up_r, t), \mathbf{Holds}(on_table, t). \\
\mathbf{Causes}(t, wet) &\leftarrow \mathbf{Init}(t, up_r), \mathbf{Init}(t, \neg up_l), \\
&\quad \mathbf{Holds}(up_l \leftrightarrow up_r, t), \mathbf{Holds}(on_table, t).
\end{aligned}$$

for each t , where the last two rules can never have a true body since **Init** denotes strong initiation. Consider then two actions *lift_l* and *lift_r*:

$$\begin{aligned}
&lift_l \text{ causes } up_l \text{ if true} \\
&lift_r \text{ causes } up_r \text{ if true}
\end{aligned}$$

Assuming the table is initially on the floor, executing either one of the actions will cause the water to spill, but if they are executed at the same time there is no spilling. It can be checked easily by evaluating the bodies of all effect rules that the given inductive definition leads to this intended conclusion in all cases. In addition, the representation by a complex derived effect rule is very concise and close to the natural language formulation of the effect. It also has the advantage that it avoids the explicit use of absences of initiations in the language: these only occur at the level of primitive rules, in certain sensible combinations. Finally, the derived effect rule is entirely independent of the actions which can influence the fluents involved. This modularity is of course required, in particular if there can be multiple actions lifting a side of the table. On the basis of these observations, which can be extended to applications with simultaneous actions in general, we argue that complex derived effect rules provide a correct and concise natural way to represent effects of simultaneous actions.

4.2 Incomplete narrative information

Another issue in \mathcal{ER} which is orthogonal to the ramification issue we have discussed in much detail up to now, is the flexible representation of both complete and incomplete knowledge on the occurrence and order of actions and on the initial situation. To deal with this issue a simple first order logic theory is most appropriate: in general a FOL theory represents incomplete knowledge on all predicates. Hence, in an \mathcal{ER} -theory knowledge on all parts of the theory except on the effects of actions (which are represented by an inductive definition) is usually incomplete. However it is possible to explicitly state that knowledge about any part of the scenario is complete by using explicit Clark completion style axioms.²³ This approach offers maximal flexibility : it allows one to specify complete knowledge about very precise parts of the scenario (for example, about all occurrences of a certain action type, or about all action occurrences in a particular time interval), while leaving other parts partially specified.

²³Extensions of \mathcal{ER} , in particular for dealing with delayed ramifications, will incorporate stronger principles than explicit completion; in particular an inductive definition semantics will be applied to the then arising theory of actions. We elaborate on this in section 9.

As an example consider the well-known stolen car problem, formalised as follows:

$$\begin{aligned} & \textit{park causes parked if true} \\ & \textit{steal causes } \neg\textit{parked if true} \\ & \forall T : \mathbf{Happens}(\textit{steal}, T) \rightarrow \mathbf{Holds}(\textit{parked}, T) \\ & \mathbf{Happens}(\textit{park}, t_1) \wedge t_1 < t_2 \wedge \neg\mathbf{Holds}(\textit{parked}, t_2) \end{aligned}$$

This specification represents incomplete information on action occurrences, and entails $\exists T : t_1 < T < t_2 \wedge \mathbf{Happens}(\textit{steal}, T)$. We can assert complete knowledge on actions, i.e. that *park* is the only action, by adding

$$\forall A, T : \mathbf{Happens}(A, T) \leftrightarrow A = \textit{park} \wedge T = t_1$$

In that case the specification is inconsistent.

Intuitively, one expects the above statements to be true. However, let us work out this example to clarify the details of the formal semantics. First of all, the grounding of the effect rules is

$$\begin{aligned} & \{\mathbf{Causes}(t, \textit{parked}) \leftarrow \mathcal{H}a(\textit{park}, t) \mid t \in \mathbf{T}\} \cup \\ & \{\mathbf{Causes}(t, \neg\textit{parked}) \leftarrow \mathcal{H}a(\textit{steal}, t) \mid t \in \mathbf{T}\} \cup \\ & \{\mathbf{Init}(t, \textit{parked}) \leftarrow \mathbf{Causes}(t, \textit{parked}), \neg\mathcal{H}o(\textit{parked}, t) \mid t \in \mathbf{T}\} \cup \\ & \{\mathbf{Init}(t, \neg\textit{parked}) \leftarrow \mathbf{Causes}(t, \neg\textit{parked}), \mathcal{H}o(\textit{parked}, t) \mid t \in \mathbf{T}\}, \end{aligned}$$

i.e. at each time point the definition of **Causes** consists of just two rules. The rules depend only on the truth values $\mathcal{H}a(\textit{park}, t)$ and $\mathcal{H}a(\textit{steal}, t)$, which uniquely determine the values of **Causes**(*t, parked*) and **Causes**(*t, ¬parked*). We find that **Happens**(*park, t₁*) is true, hence so is **Causes**(*t₁, parked*): *parked* is weakly initiated. It then follows from the inertia axiom that

$$(\neg\exists T : t_1 \preceq T \prec t_2 \wedge (\mathbf{Causes}(T, \neg\textit{parked})) \rightarrow \mathcal{H}o(\textit{parked}, t_2).$$

Since it is given that $\neg\mathcal{H}o(\textit{parked}, t_2)$, we find that

$$\exists T : t_1 \preceq T \prec t_2 \wedge \mathbf{Causes}(T, \neg\textit{parked}).$$

Then, $\mathbf{Causes}(T, \neg\textit{parked}) \leftrightarrow I_{\mathcal{D}_{\textit{init}}}(\mathbf{Causes}(T, \neg\textit{parked}))$ by the second condition in definition 3.14, so it follows that

$$\exists T : t_1 \preceq T \prec t_2 \wedge I_{\mathcal{D}_{\textit{init}}}(\mathbf{Causes}(T, \neg\textit{parked})).$$

Since $I_{\mathcal{D}_{\textit{init}}}(\mathbf{Causes}(t, \neg\textit{parked}))$ is true if and only if $\mathcal{H}a(\textit{steal}, t)$ is true, it follows that

$$\exists T : t_1 < T < t_2 \wedge \mathbf{Happens}(\textit{steal}, T)$$

which is what we intended to prove. Given this result, it is also obvious that adding $\forall A, T : \mathbf{Happens}(A, T) \leftrightarrow A = \textit{park} \wedge T = t_1$ to the theory leads to inconsistency.

5 Mapping \mathcal{ER} to Open Logic Programming

In the previous years we have devoted a lot of research to the use of Event Calculus in Open Logic Programming as a general formalism for temporal reasoning (see e.g. [6], [32], [33], [7], [34]). This formalism provides the generality needed to represent a very wide range of theories of action. A problem is however that a user needs to take some non-trivial restrictions into account for the theory to remain correct, i.e. to entail the intended conclusions. The idea behind \mathcal{ER} was to provide a framework incorporating these necessary restrictions without giving up the expressive power of OLP Event Calculus for temporal reasoning.

In this section we close the circle by mapping \mathcal{ER} theories to OLP Event Calculus. An important consequence for \mathcal{ER} as a language is that it is mapped to a more general formalism for which sound reasoning procedures exist, allowing for a practical use of the language.

The mapping is not very complicated, as \mathcal{ER} and OLP Event Calculus are based on the same principles (inductive definitions, first order logic, and narratives). Of course we start by briefly presenting open logic programming.

5.1 Open Logic Programming

An open logic program $T = \langle P^O, C \rangle$ consists of

1. P : A set of Horn clauses augmented with negation in the body, i.e. formulae of the form $A \leftarrow B_1 \ \& \ \dots \ B_n \ \& \ \neg B_{n+1} \ \& \ \dots \ \neg B_{n+m}$ where A and all B_i are atoms.
2. O : A set of undefined (abducible, open) predicates.
3. C : A set of general first order logic formulae.

The set of clauses with a certain predicate p in the head is the definition of p . An undefined predicate is a predicate without a definition: its meaning is not determined by the program. FOL formulae can be used to express information on undefined predicates.

As semantics for open logic programs we adopt the justification semantics ([4]), formally an extension of well-founded semantics which allows for undefined predicates.

This semantics assigns, given any combination of values for the undefined predicates, unique truth values to the defined predicates by interpreting their clauses as an inductive definition. The resulting interpretations which moreover satisfy all of the FOL axioms, are the models of the theory. The justification semantics can be defined in several equivalent ways. We choose the following formalisation:

Definition 5.1 *A proof tree Tr for a ground atom p according to a particular open logic program $\langle P, O, C \rangle$ is a tree of ground literals such that*

- *the root of Tr is p*
- *for each non-leaf node n of Tr with a set of immediate descendants B , n is a defined atom and either $n \leftarrow \bigwedge_{b \in B} b$ is a ground instance of a program clause of P , or $B = \{\text{false}\}$.*
- *Tr is maximal, i.e. atoms of defined predicates do not occur in leaf nodes. In other words each leaf node contains true, false, an open atom or a negative literal.*
- *Tr is finite, i.e. contains no infinite branches*

Given some 3-valued interpretation I which is 2-valued in the open predicates, for each atom p in the language, we define its *supported value* w.r.t. I , denoted $SV_I(p)$, as the truth value proven by its best proof tree, i.e.

- $SV_I(p) = \text{t}$ if p has a proof tree with all leaves containing true facts w.r.t. I ;
- $SV_I(p) = \text{f}$ if each proof tree of p has a false fact w.r.t. I in a leaf;
- $SV_I(p) = \text{u}$ otherwise; i.e. if each proof tree of p contains a non-true leaf, and some proof tree contains only non-false leaves.

Note that open atoms have only one proof tree consisting of themselves alone, hence $I(p) = SV_I(p)$ for each atom of an open predicate.

The models of an open logic program $\langle P, O, C \rangle$ are now obtained as follows: take any interpretation I_O which assigns \mathbf{t} or \mathbf{f} to any open atom. $I_{O,\perp}$ is the interpretation which assigns \mathbf{u} to all defined atoms and the same value as I_O to all open atoms. Then construct $I_{O,P}$ as follows.

Definition 5.2 *The operator $\mathcal{P}I_{O,P} : \mathcal{J} \rightarrow \mathcal{J} : I \rightarrow I'$ is defined such that $\forall p : I'(p) = SV_I(p)$.*

This operator always has a least fixpoint $\mathcal{P}I_{O,P} \uparrow$ with respect to the specificity order on interpretations. Hence, we can define $I_{O,P}$ as:

Definition 5.3 *Given $\langle P, O, C \rangle$, $I_{O,P} = \mathcal{P}I_{O,P} \uparrow$.*

The least fixpoint of this operator can be proven to coincide with the least fixpoint of the well-founded operator of [35]. Details can be found in [5].

The models of an open logic program $\langle P, O, C \rangle$ are now all those interpretations $I_{O,P}$ obtained from an interpretation $I_{O,\perp}$ as above, and for which in addition $I_{O,P} \models C$ and $I_{O,P} \models \mathcal{F}\mathcal{E}Q(T)$, i.e. the models of the logic program in which in addition all FOL axioms and the Free Equality axioms are true.

The correspondence with the inductive definition semantics we have defined for $\mathcal{E}\mathcal{R}$ is obvious. They both formalise the inductive definition principle and do this in the same way. This of course greatly facilitates mapping $\mathcal{E}\mathcal{R}$ -theories to OLP.

5.2 Mapping predicates

First of all, we establish a relation between the predicates in the two formalisms. In the open logic program, we use the predicates *happens*, *holds*, \leq , *initially* and *causes*. The relation to the $\mathcal{E}\mathcal{R}$ predicates is the following, given a an action constant, t, t' time constants, l a fluent literal and F a fluent formula²⁴:

$$\begin{aligned} \text{happens}(a, t) &\leftrightarrow \mathcal{H}a(a, t) \\ \text{holds}(F, t) &\leftrightarrow \mathcal{H}o(F, t) \\ t \leq t' &\leftrightarrow t \preceq t' \\ \text{initially}(F) &\leftrightarrow \mathcal{I}nitially(F) \\ \text{causes}(t, l) &\leftrightarrow \mathcal{C}auses(t, l) \end{aligned}$$

In the open logic program, the predicates *causes*, *holds*, and the *initially* predicate for complex fluent formulae are defined predicates, while the other predicates are open. We deal with the complex *holds* and *initially* atoms right away, so that in the sequel we only need to handle *holds* and *initially* atoms containing fluent atoms. Defining the complex atoms can easily be done inductively, by the following clauses:

$$\begin{aligned} \text{holds}(F_1 \wedge F_2, T) &\leftarrow \text{holds}(F_1, T), \text{holds}(F_2, T). \\ \text{holds}(F_1 \vee F_2, T) &\leftarrow \text{holds}(F_1, T). \\ \text{holds}(F_1 \vee F_2, T) &\leftarrow \text{holds}(F_2, T). \\ \text{holds}(\neg F_1, T) &\leftarrow \neg \text{holds}(F_1, T). \\ \text{holds}(\text{true}, T) &. \\ \\ \text{initially}(F_1 \wedge F_2) &\leftarrow \text{initially}(F_1), \text{initially}(F_2). \\ \text{initially}(F_1 \vee F_2) &\leftarrow \text{initially}(F_1). \\ \text{initially}(F_1 \vee F_2) &\leftarrow \text{initially}(F_2). \\ \text{initially}(\neg F_1) &\leftarrow \neg \text{initially}(F_1). \\ \text{initially}(\text{true}) &. \end{aligned}$$

²⁴Note that the *Init* predicate has no Event Calculus counterpart, but is redundant since $\text{Init}(t, l) \leftrightarrow \text{Causes}(t, l) \wedge \mathcal{H}o(l, t)$

For simple *holds* atoms a definition is given below, while *initially* for fluent atoms is an open predicate. Observe that we need to add explicit clauses for the fluent formula *true*. This corresponds to the condition on \mathcal{ER} -interpretations that $\mathcal{H}o(true, t)$ and $\mathcal{I}nitially(true)$ must be true.

5.3 Dealing with the domain independent conditions

Next, let us have a look at the general conditions on \mathcal{ER} interpretations and how these relate to OLP Event Calculus. Concerning the time structure, we can impose like in \mathcal{ER} that time is isomorphic to the real numbers. In OLP Event Calculus, we usually impose the weaker condition that time points are linearly ordered, by the FOL axioms

$$\begin{aligned} \forall T_1, T_2 : (T_1 \leq T_2 \wedge T_2 \leq T_1) &\rightarrow T_1 = T_2 \\ \forall T_1, T_2, T_3 : (T_1 \leq T_2 \wedge T_2 \leq T_3) &\rightarrow T_1 \leq T_3 \\ \forall T_1, T_2 : T_1 \leq T_2 \vee T_2 \leq T_1. & \end{aligned}$$

Evidently the real numbers satisfy this condition. We can assume time to be isomorphic to them.

The well-founded event topology can be imposed as follows. We define the predicate *next* as

$$\begin{aligned} next(T_1, T_2) &\leftarrow happens(A_1, T_1), happens(A_2, T_2), T_1 < T_2, \\ &\quad \neg int_events(T_1, T_2). \\ int_events(T_1, T_2) &\leftarrow happens(A_3, T_3), T_1 < T_3, T_3 < T_2. \end{aligned}$$

and its transitive closure *before* as

$$\begin{aligned} before(T_1, T_2) &\leftarrow next(T_1, T_2). \\ before(T_1, T_2) &\leftarrow next(T_1, T_3), before(T_3, T_2). \end{aligned}$$

The axiom

$$[(T_1 < T_2) \wedge happens(A_1, T_1) \wedge happens(A_2, T_2)] \leftrightarrow before(T_1, T_2)$$

then imposes that there are only a finite number of events between each two events. In addition, the axiom

$$\exists E_{start} : \forall A, T : [happens(A, T) \rightarrow \neg(T < E_{start})]$$

ensures that there is a first event. Hence there is an order-preserving isomorphism between events and a subset of the natural numbers. This is the desired topology on events. In addition, we need to embed this topology in the time structure such that in case of an infinite number of events, there are no time points after the entire infinite sequence. This is imposed by the axiom

$$\begin{aligned} \forall T : ([\exists A, E : (happens(A, E) \wedge E < T \wedge \neg int_events(E, T))] \\ \vee [\neg exists A, E : (happens(A, E) \wedge E < T)]) \end{aligned}$$

Given that time is isomorphic to the real numbers, the well-founded event topology condition defined earlier is imposed by the above theory.

The initiation consistency condition is represented by the Event Calculus FOL axiom

$$\forall T, F : \neg causes(T, F) \vee \neg causes(T, \neg F)$$

which is equivalent to $\forall T, F : \neg Causes(T, F) \vee \neg Causes(T, \neg F)$. Then we still need to formalise the definition of the initial state and the inertia axiom. This

is achieved by providing the following definition for *holds*:

$$\begin{aligned}
\text{holds}(F, T) &\leftarrow T \leq e_{start}, \text{initially}(F). \\
\text{holds}(F, T) &\leftarrow \text{causes}(T', F), T' < T, \neg \text{clipped}(T', F, T). \\
\text{clipped}(T', F, T) &\leftarrow \text{causes}(T'', \neg F), T' \leq T'', T'' < T. \\
T' < T &\leftarrow T' \leq T, T' \neq T.
\end{aligned}$$

The first clause for *holds* defines the initial state, the other clause is the Event Calculus frame axiom, which will further on be proven equivalent to the inertia axiom in \mathcal{ER} given the other assumptions. Note that the event e_{start} is the first actual event. The initial state is the state before e_{start} (recall that time is considered unbounded in the past).

5.4 Mapping Π_p to FOL axioms

The Π_p part of an \mathcal{ER} theory contains general FOL formulae constructed from **Holds**, **Happens**, \leq and **Initially** atoms using connectives and quantifiers. We can map these formulae to FOL axioms straightforwardly by replacing *holds*(F, τ) for **Holds**(F, τ), *happens*(α, τ) for **Happens**(α, τ), $\tau_1 \leq \tau_2$ for itself and *initially*(F) for **Initially**(F). Given the mapping of predicates above it is easy to check that this mapping preserves the semantics.

5.5 Mapping Π_e to program clauses

As a guideline for mapping the effect rules to OLP, we can look at the definition of grounding in the semantics of \mathcal{ER} : as we had to do in the grounding, we need to deal with complex initiations, and this can be done in the same way:

A direct effect rule *a causes l if F* is mapped to the clause

$$\text{causes}(T, l) \leftarrow \text{happens}(a, T), \text{holds}(F, T).$$

A derived effect rule **initiating F causes l if F'** is mapped to the set of clauses

$$\begin{aligned}
\{\text{causes}(T, l) &\leftarrow \bigwedge_{l \in L_1} \text{causes}(T, l), \\
&\bigwedge_{l' \in L_2} (\text{holds}(l, T), \neg \text{causes}(T, l')) \\
&\neg \text{holds}(F, T), \text{holds}(F', T). \\
&| \quad L_1 \cup L_2 \text{ is a supporting set of } F\}
\end{aligned}$$

Both results of the mapping correspond to the respective groundings of the rules, but compacted to a finite number of clauses by using universal quantification over time points instead of different clauses for each t . A minor difference is that the predicates *happens* and *holds* occur explicitly in the clauses instead of their truth values. A more noteworthy difference is that in the body of the OLP clauses, we use the *causes* predicate which denotes weak initiation, whereas in \mathcal{ER} **Init**, i.e. strong initiation, is used in the inductive definition rules. However we have proven the equivalence of the two corresponding ways of defining *Causes* in \mathcal{ER} before.

Note that there is no predicate denoting strong initiation in the OLP theory. Therefore the \mathcal{ER} -rules defining **Init** in terms of **Causes** have no equivalent OLP clauses. They have been compiled into the clauses for *causes*.

5.6 A detailed example and some remarks

Taking everything together, we now show the mapping of the stolen car problem presented earlier:

$$\begin{aligned}
& \textit{park causes parked if true} \\
& \textit{steal causes } \neg\textit{parked if true} \\
\forall T : & \mathbf{Happens}(\textit{steal}, T) \rightarrow \mathbf{Holds}(\textit{parked}, T) \\
& \mathbf{Happens}(\textit{park}, t_1) \wedge t_1 < t_2 \wedge \neg\mathbf{Holds}(\textit{parked}, t_2)
\end{aligned}$$

First of all, we have the general formulae, i.e. the Event Calculus frame axiom

$$\begin{aligned}
\textit{holds}(F, T) & \leftarrow T < \textit{e_start}, \textit{initially}(F). \\
\textit{holds}(F, T) & \leftarrow \textit{causes}(T', F), T' < T, \neg\textit{clipped}(T', F, T). \\
\textit{clipped}(T', F, T) & \leftarrow \textit{causes}(T'', \neg F), T' \leq T'', T'' < T. \\
T' < T & \leftarrow T' \leq T, T' \neq T.
\end{aligned}$$

and the constraints on time

$$\begin{aligned}
\forall T_1, T_2 : & (T_1 \leq T_2 \wedge T_2 \leq T_1) \rightarrow T_1 = T_2 \\
\forall T_1, T_2, T_3 : & (T_1 \leq T_2 \wedge T_2 \leq T_3) \rightarrow T_1 \leq T_3 \\
\forall T_1, T_2 : & T_1 \leq T_2 \vee T_2 \leq T_1
\end{aligned}$$

The effect rules are mapped to clauses

$$\begin{aligned}
\textit{causes}(T, \textit{parked}) & \leftarrow \textit{happens}(\textit{park}, T). \\
\textit{causes}(T, \neg\textit{parked}) & \leftarrow \textit{happens}(\textit{steal}, T).
\end{aligned}$$

The other formulae are simply mapped to FOL formulae

$$\begin{aligned}
\forall T : & (\textit{happens}(\textit{steal}, T) \rightarrow \textit{holds}(\textit{parked}, T)) \\
& \textit{happens}(\textit{park}, t_1) \wedge t_1 < t_2 \wedge \neg\textit{holds}(\textit{parked}, t_2)
\end{aligned}$$

The predicates $<$, *initially* and *happens* are undefined, and we ignore complex *holds* and *initially* atoms as they do not occur in this example. The formula which was to be entailed is mapped to $\exists T : t_1 < T < t_2 \wedge \textit{happens}(\textit{steal}, T)$. Of course it is easy to check that this formula is entailed, following the same reasoning as in the \mathcal{ER} formalisation.

In \mathcal{ER} we also added an explicit completion on action occurrences to this formalisation, which led to inconsistency. We can do the same in the OLP formalisation, but note that another and simpler option is to make *happens* a defined predicate, defined by the fact *happens*(*park*, t_1). In general, if there is complete knowledge on *happens* or *initially* in any scenario, we can choose to make these predicates defined instead of adding explicit completions. This is only important for conciseness of representation and for reasoning performance: semantically there is no difference.

5.7 Equivalence proof

To prove equivalence of the two theories, we proceed as follows. First, observe that as indicated above, the initiation consistency, linear order and well-founded event topology conditions are properly axiomatised. Now, take any interpretation of the undefined predicates (*initially* for fluent atoms, *happens* and \leq), and the same interpretation for *Initially* for fluent atoms, $\mathcal{H}a$ and \preceq . From these we can compute the truth values of all other atomic formulae using on the one hand the program clauses in OLP, and on the other hand the inductive definitions and the inertia axiom in \mathcal{ER} . We will prove that this yields the same results in both formalisms, which leaves us with the same models of Π_e of each

theory. Finally, the FOL axioms and the corresponding theory Π_p and general axioms of \mathcal{ER} will be checked in all of those Π_e -models, retaining only those interpretations that satisfy all of the axioms. Given the trivial correspondence between Π_p formulae and general \mathcal{ER} axioms with FOL axioms in OLP, it is obvious that the same interpretations are retained in both formalisms by this step.

So what we still need to prove is that given *initially* for fluent atoms, *happens* and \leq , and their exact counterparts in \mathcal{ER} , we find the same truth value for all other atoms in both formalisms. For complex *initially* and *Initially* atoms this is trivial since we use the same inductive definition in terms of simple atoms in both formalisms. This leaves us with simple and complex *holds* atoms and *causes* atoms.

The clauses of *holds* and *causes* are mutually recursive. *holds*(f, t) depends on the truth values of *causes*(f, t') and *causes*($\neg f, t'$) for $t' < t$. *holds*(F, t) depends on atoms *holds*(f, t). *causes*(l, t) may depend on *holds*(F, t) (and *causes*(l', t)) for any F and l' . Given a well-founded event topology, we can prove equivalence of *holds*(f, t) with $\mathcal{H}o(f, t)$, of *holds*(F, t) with $\mathcal{H}o(F, t)$ and of *causes*(t, l) with $\mathcal{C}auses(t, l)$ by induction on events: first we prove that *holds*(F, t) coincides with $\mathcal{H}o(F, t)$ for all time points before the first event, then for each event e we prove consecutively that *causes*(e, l) \leftrightarrow $\mathcal{C}auses(e, l)$, that for all time points t between e and the next event (including it), *holds*(f, t) \leftrightarrow $\mathcal{H}o(f, t)$, and that for all these time points *holds*(F, t) \leftrightarrow $\mathcal{H}o(F, t)$. By induction on the well-founded event topology it then follows that *holds*(F, t) \leftrightarrow $\mathcal{H}o(F, t)$ for all time points and that *causes*(t, l) \leftrightarrow $\mathcal{C}auses(t, l)$ for all events (at non-event time points it is easy to see that *causes*(t, l) as well as $\mathcal{C}auses(t, l)$ are false from their definitions: no proof tree exists without *happens* atoms in one or more leaves).

We now proceed with the inductive proof.

1. Induction base:

For all F and for all $t \leq e_{start}$, in \mathcal{ER} it holds that *Initially*(F) \leftrightarrow $\mathcal{H}o(F, t)$. Likewise in OLP we know from the definition of *holds* that if f is a fluent atom and $t \leq e_{start}$, then *holds*(f, t) \leftrightarrow *initially*(f) since the second clause can never have a true body and the first clause simplifies to *holds*(f, t) \leftarrow *initially*(f). Hence, for all fluent atoms f , we know that *holds*(f, t) \leftarrow $\mathcal{H}o(f, t)$ for $t \leq e_{start}$. The definition of complex *holds* atoms in terms of simple ones is the same in OLP and \mathcal{ER} , so the result generalises to *holds*(F, t) \leftarrow $\mathcal{H}o(F, t)$ for all fluent formulae F and all $t \leq e_{start}$.

2. Assume *holds*(F, t) \leftrightarrow $\mathcal{H}o(F, t)$ for all F and a particular t . The definition of *causes*(t, l) for a certain t depends only on *happens*(a, t) and *holds*(F, t) and on itself (*causes*(t, l')), just like the definition of $\mathbf{C}auses(t, l)$ in \mathcal{ER} depends only on truth values $\mathcal{H}a(a, t)$ and $\mathcal{H}o(F, t)$. Moreover, the definitions are the same, as we have shown above. Hence we find that *causes*(t, l) \leftrightarrow $I_{\mathcal{D}_{init}}(\mathbf{C}auses(t, l))$.

3. Assume *causes*(t, l) \leftrightarrow $\mathbf{C}auses(t, l)$ for a particular event t . We need to prove that for all t' after t and before or equal to the next event t'' , *holds*(f, t') \leftrightarrow $\mathcal{H}o(f, t')$. Take an arbitrary f and t' . The inertia axiom in \mathcal{ER} yields for f the formula

$$\begin{aligned} t \prec t' \wedge \neg \exists t_3 : (t \preceq t_3 \prec t' \wedge \mathbf{C}auses(t_3, \neg f)) \\ \rightarrow \mathcal{H}o(f, t') \leftrightarrow \mathbf{C}auses(t, f) \vee \mathcal{H}o(f, t) \end{aligned}$$

which can be simplified to

$$\neg \text{Causes}(t, \neg f) \rightarrow (\mathcal{H}o(f, t') \leftrightarrow \text{Causes}(t, f) \vee \mathcal{H}o(f, t))$$

and then to

$$\begin{aligned} & [\mathcal{H}o(f, t') \wedge \neg \text{Causes}(t, \neg f)] \\ \leftrightarrow & [(\text{Causes}(t, f) \vee \mathcal{H}o(f, t)) \wedge \neg \text{Causes}(t, \neg f)] \end{aligned}$$

It follows that

$$\begin{aligned} & \mathcal{H}o(f, t') \leftarrow \\ & ([\text{Causes}(t, f) \wedge \neg \text{Causes}(t, \neg f)] \vee [\mathcal{H}o(f, t) \wedge \neg \text{Causes}(t, \neg f)]) \end{aligned}$$

Similarly, we find for $\neg f$ that

$$\begin{aligned} & \mathcal{H}o(\neg f, t') \leftarrow \\ & ([\text{Causes}(t, \neg f) \wedge \neg \text{Causes}(t, f)] \vee [\mathcal{H}o(\neg f, t) \wedge \neg \text{Causes}(t, f)]) \end{aligned}$$

hence because $\mathcal{H}o(\neg f, t) \leftrightarrow \neg \mathcal{H}o(f, t)$, that

$$\begin{aligned} & \neg \mathcal{H}o(f, t') \leftarrow \\ & ([\text{Causes}(t, \neg f) \wedge \neg \text{Causes}(t, f)] \vee [\neg \mathcal{H}o(f, t) \wedge \neg \text{Causes}(t, f)]) \end{aligned}$$

Contraposition of this formula yields

$$\mathcal{H}o(f, t') \rightarrow ([\neg \text{Causes}(t, \neg f) \vee \text{Causes}(t, f)] \wedge [\mathcal{H}o(f, t) \vee \text{Causes}(t, f)])$$

which can be simplified to

$$\mathcal{H}o(f, t') \rightarrow ([\neg \text{Causes}(t, \neg f) \wedge \mathcal{H}o(f, t)] \vee \text{Causes}(t, f))$$

Together with the formula obtained in the previous paragraph, which can be simplified to

$$\mathcal{H}o(f, t') \leftarrow ([\neg \text{Causes}(t, \neg f) \wedge \mathcal{H}o(f, t)] \vee \text{Causes}(t, f))$$

we obtain the equivalence

$$\mathcal{H}o(f, t') \leftrightarrow (\text{Causes}(t, f) \vee [\mathcal{H}o(f, t) \wedge \neg \text{Causes}(t, \neg f)])$$

which is equivalent to (given the assumption of the inductive step)

$$\mathcal{H}o(f, t') \leftrightarrow ([\text{causes}(t, f) \vee [\text{holds}(f, t) \wedge (\neg \text{causes}(t, \neg f))]]$$

On the other hand, the definition of *holds* entails

$$\begin{aligned} \text{holds}(f, t') & \leftrightarrow \exists t^* : [\text{causes}(t^*, f) \wedge t^* < t' \wedge \\ & \neg \exists t'' : (\text{causes}(t'', \neg f) \wedge t^* \leq t'' \wedge t'' < t')] \end{aligned}$$

since completion semantics is strictly weaker than justification semantics. A possible t^* in that formula can either be before t or equal to t , taking into account the fact that $t < t'$, that there is no event between t and t' and that time is a linear order. In the first case, the condition

$$\begin{aligned} & \text{causes}(t^*, f) \wedge t^* < t' \wedge \\ & \neg \exists t'' : (\text{causes}(t'', \neg f) \wedge t^* \leq t'' \wedge t'' < t') \end{aligned}$$

can be written as

$$\begin{aligned} & \text{causes}(t^*, f) \wedge t^* < t' \wedge t' < t \wedge \\ & \neg \exists t'' : (\text{causes}(t'', \neg f) \wedge t^* \leq t'' \wedge t'' < t) \wedge \neg \text{causes}(t, \neg f) \end{aligned}$$

which is equivalent to $holds(f, t) \wedge \neg causes(t, \neg f)$; in the second case that same condition simply reads $causes(t, f)$. We obtain

$$holds(f, t') \leftrightarrow [holds(f, t) \wedge \neg causes(t, \neg f)] \vee causes(t, f)$$

and this formula combined with the result for $\mathcal{H}o$ yields

$$holds(f, t) \leftrightarrow \mathcal{H}o(f, t)$$

4. The definition of complex *holds* atoms in terms of simple ones is the same in OLP and \mathcal{ER} , so if $holds(f, t) \leftrightarrow \mathcal{H}o(f, t)$ for all fluent atoms f and the above defined time points t , then $holds(F, t) \leftrightarrow \mathcal{H}o(F, t)$ for all fluent formulae F and all those t .

This completes the proof by induction, and thereby the entire equivalence proof of the OLP Event Calculus theory with the \mathcal{ER} theory.

6 Dealing with Nondeterminism in \mathcal{ER}

We now turn our attention to the representation of nondeterministic effects of actions. We assume in this discussion that the outcome of a nondeterministic action is one of a number of possible effects (possibly the empty effect), that the set of possible effects can be dependent on the state of the world, and that the actual effect can be different for different instances of the action.

As an example, consider a variant of the Yale Shooting Problem in which firing a loaded gun may nondeterministically hit a turkey in the head (killing it) or in the wing (breaking that wing). A representation of this effect could be a rule

$$\mathit{shoot} \text{ causes } \neg \mathit{alive} \vee \mathit{broken_wing} \text{ if } \mathit{loaded}$$

which is a generalisation of a direct effect rule with a disjunction in the head. In general, we choose the following syntax for nondeterministic effect rules:

$$a \text{ causes } D \text{ if } F$$

where a is an action, F a fluent formula, and D a disjunction of fluent literals.²⁵ For reasons that will be explained below we will use the symbol $|$ instead of \vee to denote disjunction in D .

The semantics of such a nondeterministic effect rule is that when the action a is executed while F holds, one of the disjuncts in D is (weakly) initiated. Let us try to define this more precisely for the above example: we can say that when a loaded gun is fired, *broken_wing* or *¬alive* should be initiated. But should this “or” be inclusive or exclusive? We will show that neither option is satisfactory.

Clearly an inclusive or is unintended: the bullet should hit the turkey only in one place, not both. This seems to leave the exclusive or as only plausible reading. However, the following examples show that this reading is also unintended. First, consider the ramification that a turkey dies when its wing gets broken while it is flying²⁶, as represented by the derived effect rule

$$\mathit{initiating} \ \mathit{broken_wing} \text{ causes } \neg \mathit{alive} \text{ if } \mathit{flying}$$

The intuitive result of shooting a flying turkey is that it always dies, either by the shot in the head or as a result of its broken wing. However, reading “or”

²⁵Below we further generalise the formulae allowed as D .

²⁶We ignore for now other evident relations between *flying*, *alive* and *broken_wing* as they are not relevant in this discussion.

in the nondeterministic effect rule as exclusive, we would reach the unintended conclusion that its wing can never get broken since it already dies.

A similar problem arises in the presence of simultaneous actions: assume that apart from the original hunter, there is now also an expert hunter who always kills the turkey when shooting. Then assume both hunters shoot at the same time. We expect the nondeterministic shot to either break the turkey's wing or to kill it, while the expert shot definitely kills the turkey. Yet reading "or" as exclusive leads to the conclusion that, since *alive* is already terminated by the expert shot, the nondeterministic shot cannot break the turkey's wing at the same time.

How should we read nondeterministic rules then? In the above examples, the intuitive reading is that the *shoot* action has exactly one effect, either killing the turkey or breaking its wing. However, other sources (either ramifications or effects of other simultaneous actions) may cause the second effect as well, so that it is not guaranteed that there is only one effect, even if only one may be caused by the given direct effect rule. In general, recall that we view effect rules as describing the propagation of effects due to some physical or logical force. The idea underlying nondeterministic actions is that this force may act in one of several possible ways. This boils down to saying that a rule

$$a \text{ causes } f_1 \mid f_2 \text{ if } F$$

is equivalent to

$$a \text{ causes } f_1 \text{ if } F \oplus a \text{ causes } f_2 \text{ if } F$$

or in other words a nondeterministic effect rule has at all times exactly the same effect as one rule obtained from it by retaining only one of its disjuncts. The thus obtained rules will be called the disjunctive components of the nondeterministic effect rules. Generalising this, we say an interpretation satisfies a definition \mathcal{D} at a particular time point iff it satisfies *any* definition obtained from \mathcal{D} by replacing each nondeterministic effect rule by one of its disjunctive components.

As an example, consider the rules formalising the effect of shooting a flying turkey:

$$\begin{aligned} & \textit{shoot causes } \neg\textit{alive} \mid \textit{broken_wing} \text{ if } \textit{loaded} \\ & \textit{initiating broken_wing causes } \neg\textit{alive} \text{ if } \textit{flying} \end{aligned}$$

This nondeterministic definition is satisfied if at each time point at least one of the deterministic definitions

$$\begin{aligned} & \textit{shoot causes } \neg\textit{alive} \text{ if } \textit{loaded} \\ & \textit{initiating broken_wing causes } \neg\textit{alive} \text{ if } \textit{flying} \end{aligned}$$

or

$$\begin{aligned} & \textit{shoot causes } \textit{broken_wing} \text{ if } \textit{loaded} \\ & \textit{initiating broken_wing causes } \neg\textit{alive} \text{ if } \textit{flying} \end{aligned}$$

is satisfied. In the first case, we get the effect that the turkey is killed by the shot, in the second case that its wing gets broken and it dies as a result.²⁷

Before formalising the above intuitive semantics, we want to extend the syntax of nondeterministic effect rules. In particular, we want to allow for a nondeterministic choice of sets of effects, rather than of single effects. This is achieved by defining nondeterministic effect rules as formulae

$$a \text{ causes } D \text{ if } F$$

²⁷It is essential that at different time points, different deterministic definitions can be satisfied: the nondeterministic action can have a different outcome each time. It can also occur by coincidence that both definitions are satisfied at a particular moment (if their effects happen to be the same). This is for example trivially the case when the gun is not loaded.

where a is an action, F a fluent formula, and D a disjunction of conjunctions of fluent literals. Note that “true” can be used to denote the empty effect, since true is always weakly initiated.

One may wonder why we do not allow D to be any propositional fluent formula F' , since any F' can be written in disjunctive normal form. The reason is that equivalent FOL formulae may lead to non-equivalent effect rules: for example p is equivalent to $(p \wedge q) \vee (p \wedge \neg q)$, but there is an important difference between the rules a **causes** p *if true* and a **causes** $(p \wedge q) \mid (p \wedge \neg q)$ *if true*: according to the first rule a initiates p and leaves q alone, while according to the second rule a initiates p and can nondeterministically initiate or terminate q . This is the reason why \mid should not be read as classical disjunction, and why we should not rely on reducing general propositional formulae to some normal form.

We are now ready to define the semantics of nondeterministic effect rules. To this end we extend the notion of grounding as follows : a grounding of a nondeterministic effect rule is obtained from the groundings of its disjunctive components by collecting, for each time point t , from the grounding of one arbitrary disjunctive component, all the rules containing t . Formally:

Definition 6.1 (nondeterministic grounding (direct))

The restriction G^t of a set of primitive definition rules G to a time point t is the set of all rules of G in which t occurs.

The grounding of a conjunctive direct effect rule

$$a \text{ causes } \bigwedge_{i=1..n} l_i \text{ if } F$$

is the set $\bigcup_{i=1..n} G_i$, where each G_i is the grounding of the direct effect rule a **causes** l_i *if* F .

A grounding of a nondeterministic direct effect rule

$$a \text{ causes } C_1 \mid \dots \mid C_m \text{ if } F$$

is any set $\bigcup_{t \in \mathbf{T}} G_{j_t}^t$, where $1 \leq j_t \leq m$ for each $t \in \mathbf{T}$, G_{j_t} is the grounding of the conjunctive effect rule a **causes** C_{j_t} *if* F , and $G_{j_t}^t$ is the restriction of G_{j_t} to t .

A grounding of $\Pi_e = \{r_k \mid 1 \leq k \leq l\}$ is any definition $\mathcal{D}_{init} = \mathcal{D}_g \cup \{\text{Init}(t, l) \leftarrow \text{Causes}(t, l), \neg \text{Ho}(l, t) \mid t \in \mathbf{T}, l \in \widehat{\mathbf{P}}\}$, where \mathcal{D}_g is any set $\bigcup_{k=1..l} G_k$ with each G_k a grounding of r_k .

$I \models \Pi_e$ iff for any grounding \mathcal{D}_{init} of Π_e , $\forall t \in \mathbf{T}, l \in \widehat{\mathbf{P}} :$
 $\text{Init}(t, l) \leftrightarrow I_{\mathcal{D}_{init}}(\text{Init}(t, l))$ and $\text{Causes}(t, l) \leftrightarrow I_{\mathcal{D}_{init}}(\text{Causes}(t, l))$.

The groundings of the above example definition are sets containing for each time point $t \in \mathbf{T}$, either

$$\begin{aligned} \text{Causes}(t, \neg \text{alive}) &\leftarrow \mathcal{H}a(\text{shoot}, t), \mathcal{H}o(\text{loaded}, t). \\ \text{Causes}(t, \neg \text{alive}) &\leftarrow \text{Init}(t, \text{broken_wing}), \mathcal{H}o(\text{flying}, t). \end{aligned}$$

or

$$\begin{aligned} \text{Causes}(t, \text{broken_wing}) &\leftarrow \mathcal{H}a(\text{shoot}, t), \mathcal{H}o(\text{loaded}, t). \\ \text{Causes}(t, \neg \text{alive}) &\leftarrow \text{Init}(t, \text{broken_wing}), \mathcal{H}o(\text{flying}, t). \end{aligned}$$

The syntax of derived effect rules can be extended in the same way as that of direct effect rules, which leads to rules

$$\text{initiating } F \text{ causes } D \text{ if } F'$$

representing nondeterministic ramifications. The semantics of such rules is defined by extending the definition of grounding in the same way as for direct effect rules. Formally:

Definition 6.2 (nondeterministic grounding (derived))

The grounding of a conjunctive derived effect rule

$$\text{initiating } F \text{ causes } \bigwedge_{i=1..n} l_{f_i} \text{ if } F'$$

is the set $\bigcup_{i=1..n} G_i$, where each G_i is the grounding of the derived effect rule **initiating** F **causes** l_{f_i} **if** F' .

A grounding of a nondeterministic derived effect rule

$$\text{initiating } F \text{ causes } C_1 \mid \dots \mid C_m \text{ if } F'$$

is any set $\bigcup_{t \in \mathbf{T}} G_{j_t}^t$, where $\forall t : 1 \leq j_t \leq m$, G_{j_t} is the grounding of the conjunctive derived effect rule **initiating** F **causes** C_{j_t} **if** F' , and $G_{j_t}^t$ is the restriction of G_{j_t} to t .

As an example, consider again the effect of shooting a flying turkey in the wing. Maybe it is unpredictable if the turkey will die as a result or not: it may for example depend on how high the turkey is flying. This can be represented by replacing the derived effect rule in the above example by

$$\text{initiating } \textit{broken_wing} \text{ causes } \neg \textit{alive} \mid \textit{true} \text{ if } \textit{flying}$$

where the disjunct *true* indicates the absence of any additional effect. The groundings of the resulting definition are now sets containing for each time point $t \in \mathbf{T}$, one of the four definitions

$$\begin{array}{ll} \mathbf{Causes}(t, \neg \textit{alive}) & \leftarrow \mathcal{H}a(\textit{shoot}, t), \mathcal{H}o(\textit{loaded}, t). \\ \mathbf{Causes}(t, \neg \textit{alive}) & \leftarrow \mathbf{Init}(t, \textit{broken_wing}), \mathcal{H}o(\textit{flying}, t). \\ \\ \mathbf{Causes}(t, \textit{broken_wing}) & \leftarrow \mathcal{H}a(\textit{shoot}, t), \mathcal{H}o(\textit{loaded}, t). \\ \mathbf{Causes}(t, \neg \textit{alive}) & \leftarrow \mathbf{Init}(t, \textit{broken_wing}), \mathcal{H}o(\textit{flying}, t). \\ \\ \mathbf{Causes}(t, \neg \textit{alive}) & \leftarrow \mathcal{H}a(\textit{shoot}, t), \mathcal{H}o(\textit{loaded}, t). \\ \mathbf{Causes}(t, \textit{true}) & \leftarrow \mathbf{Init}(t, \textit{broken_wing}), \mathcal{H}o(\textit{flying}, t). \\ \\ \mathbf{Causes}(t, \textit{broken_wing}) & \leftarrow \mathcal{H}a(\textit{shoot}, t), \mathcal{H}o(\textit{loaded}, t). \\ \mathbf{Causes}(t, \textit{true}) & \leftarrow \mathbf{Init}(t, \textit{broken_wing}), \mathcal{H}o(\textit{flying}, t). \end{array}$$

in which the rules for $\mathbf{Causes}(t, \textit{true})$ evidently have no effect.

The above definitions show that a syntax and semantics for nondeterministic ramifications can be obtained as a straightforward extension of our existing constructs. At first sight it may not be clear if nondeterministic ramifications are of great practical importance, but it should be noted that they occur at least implicitly in some approaches to the ramification problem, as we will discuss below.

6.1 Mapping nondeterministic effect rules to OLP

Dealing with nondeterministic effect rules can essentially be done in the same way as dealing with deterministic rules, basing the mapping on the groundings of the rules. A complication is that OLP does not contain disjunctive rules. We could of course extend the syntax of OLP with disjunctive rules, in which case the mapping would be straightforward. But this extension of the OLP syntax is not necessary: an OLP theory with the same models as the nondeterministic \mathcal{ER} theory can be cleanly constructed by introducing a number of auxiliary “degree of freedom” predicates. The technique has been discussed briefly in [34]

in the context of nondeterministic actions in Event and Situation Calculus. We here show how it applies to general effect rules, thereby also showing the precise semantics of theories containing such predicates (i.e. exactly the above defined semantics for nondeterministic effect rules).

As defined before, a nondeterministic rule has multiple groundings corresponding to its multiple possible outcomes. An interpretation is a model of a set of effect rules if at each time point it satisfies any set of definition rules containing exactly one grounding of each effect rule. The idea now is the following: for each effect rule, *all* of its groundings are converted to OLP like in section 5, but an additional literal is inserted in the body of each clause such that for any interpretation of the additional literals only one clause body is not trivially false. Moreover this one clause body then needs to be equivalent to the corresponding original clause (without the added literal). Which of the clause bodies is not trivially false, depends on the truth values of the additional literals. These literals should of course take on exactly those combinations of values needed to “select” all clauses once.

As an example, assume a simple nondeterministic action with two possible outcomes, described by a causes $f_1 \mid f_2$ if *true*. Its two disjunctive components would yield mappings

$$\begin{aligned} \text{causes}(T, f_1) &\leftarrow \text{happens}(a, T). \\ \text{causes}(T, f_2) &\leftarrow \text{happens}(a, T). \end{aligned}$$

We add literals to the rules such that only one rule body can be true in one interpretation:

$$\begin{aligned} \text{causes}(T, f_1) &\leftarrow \text{happens}(a, T), \text{choice}(c_r, 1, T). \\ \text{causes}(T, f_2) &\leftarrow \text{happens}(a, T), \text{choice}(c_r, 2, T). \end{aligned}$$

with FOL axioms

$$\begin{aligned} \forall T : \text{choice}(c_r, 1, T) \vee \text{choice}(c_r, 2, T). \\ \forall T, I, J : (\text{choice}(c_r, I, T) \wedge \text{choice}(c_r, J, T)) \rightarrow I = J. \end{aligned}$$

Depending on the values of the *choice* atoms, the definition at a particular time t is equivalent to either

$$\text{causes}(t, f_1) \leftarrow \text{happens}(a, t).$$

or

$$\text{causes}(t, f_2) \leftarrow \text{happens}(a, t).$$

Note that *choice* has three parameters: the first is to distinguish between rules (for each nondeterministic rule separate choices are required), the second to distinguish between choices for one rule, and the third is a time parameter (as the effect of a nondeterministic action may vary at different time points). Also of importance is that *choice* is an open predicate for which no definition and only the above FOL axioms exist. Thus *choice* atoms can take on all possible values such that for each c_r and T , exactly one atom $\text{choice}(c_r, I, T)$ is true. For each (I th) disjunct in a nondeterministic rule r , one separate atom $\text{choice}(c_r, I, T)$ is used. If a disjunct is itself a conjunction, clauses with equal bodies are generated for each conjunct.

We can formalise the above method as follows: a rule $r =$

$$a \text{ causes } \bigwedge_{j=1..n_1} l_{1,j} \mid \dots \mid \bigwedge_{j=1..n_m} l_{m,j} \text{ if } F$$

is mapped to the set of clauses

$$\bigcup_{i=1\dots m, j=1\dots n_i} \{ \text{causes}(T, l_{i,j}) \leftarrow \text{happens}(a, T), \text{holds}(F, T), \text{choice}(c_r, i, T). \}$$

and the FOL axioms

$$\begin{aligned} \forall T : \text{choice}(c_r, 1, T) \vee \dots \vee \text{choice}(c_r, m, T). \\ \forall T, I, J : (\text{choice}(c_r, I, T) \wedge \text{choice}(c_r, J, T)) \rightarrow I = J. \end{aligned}$$

with c_r a constant not occurring elsewhere in the theory. A rule $r' =$

$$\text{initiating } F' \text{ causes } \bigwedge_{j=1\dots n_1} l_{1,j} \mid \dots \mid \bigwedge_{j=1\dots n_m} l_{m,j} \text{ if } F$$

is mapped to the set of clauses

$$\begin{aligned} \bigcup_{i=1\dots m, j=1\dots n_i} \{ \text{causes}(T, l_{i,j}) \leftarrow & \bigwedge_{l \in L_1} \text{causes}(T, l), \\ & \bigwedge_{l' \in L_2} (\text{holds}(l', T), \neg \text{causes}(T, \bar{l}')), \\ & \neg \text{holds}(F', T), \text{holds}(F, T), \\ & \text{choice}(c_{r'}, i, T). \\ & \mid L_1 \cup L_2 \text{ is supporting set of } F' \} \end{aligned}$$

and the FOL axioms

$$\begin{aligned} \forall T : \text{choice}(c_{r'}, 1, T) \vee \dots \vee \text{choice}(c_{r'}, m, T). \\ \forall T, I, J : (\text{choice}(c_{r'}, I, T) \wedge \text{choice}(c_{r'}, J, T)) \rightarrow I = J. \end{aligned}$$

with $c_{r'}$ a constant not occurring elsewhere in the theory.

The interaction between the different forms of complexity makes for a harder to read notation, so let us give another example.

Consider the rule

$$\text{initiating } f_1 \vee f_2 \text{ causes } (g_1 \wedge g_2) \mid h \mid \text{true if true}$$

This rule is mapped to the clauses

$$\begin{aligned} \text{causes}(T, g_1) & \leftarrow \text{causes}(T, f_1), \neg \text{holds}(f_1 \vee f_2), \text{choice}(c, 1, T). \\ \text{causes}(T, g_1) & \leftarrow \text{causes}(T, f_2), \neg \text{holds}(f_1 \vee f_2), \text{choice}(c, 1, T). \\ \text{causes}(T, g_2) & \leftarrow \text{causes}(T, f_1), \neg \text{holds}(f_1 \vee f_2), \text{choice}(c, 1, T). \\ \text{causes}(T, g_2) & \leftarrow \text{causes}(T, f_2), \neg \text{holds}(f_1 \vee f_2), \text{choice}(c, 1, T). \\ \text{causes}(T, h) & \leftarrow \text{causes}(T, f_1), \neg \text{holds}(f_1 \vee f_2), \text{choice}(c, 2, T). \\ \text{causes}(T, h) & \leftarrow \text{causes}(T, f_2), \neg \text{holds}(f_1 \vee f_2), \text{choice}(c, 2, T). \\ \text{causes}(T, \text{true}) & \leftarrow \text{causes}(T, f_1), \neg \text{holds}(f_1 \vee f_2), \text{choice}(c, 3, T). \\ \text{causes}(T, \text{true}) & \leftarrow \text{causes}(T, f_2), \neg \text{holds}(f_1 \vee f_2), \text{choice}(c, 3, T). \end{aligned}$$

and the FOL axioms

$$\begin{aligned} \forall T : \text{choice}(c, 1, T) \vee \text{choice}(c, 2, T) \vee \text{choice}(c, 3, T). \\ \forall T, I, J : (\text{choice}(c, I, T) \wedge \text{choice}(c, J, T)) \rightarrow I = J. \end{aligned}$$

where the last two clauses can in fact be omitted (which is not done in the formalisation for simplicity reasons). The number of generated clauses is equal to the number of literals in the head times the number of supporting sets of the body of the derived effect rule.

We now prove that the extended mapping is correct, which is a slightly more complicated task than in the deterministic case. First, observe that with each \mathcal{ER} -interpretation I corresponds a class of OLP-interpretations which assigns the same value as I to all instances of predicates occurring in the \mathcal{ER} -theory, and an arbitrary value to each instance of the choice predicate. The correctness criterion we need to prove is then the following: an \mathcal{ER} -interpretation is a model of the \mathcal{ER} -theory if and only if at least one of its corresponding OLP-interpretations is a model of the OLP theory. More precisely this reads:

Theorem 6.1 *An \mathcal{ER} -interpretation is a model of Π_p and a model of at least one of the groundings of Π_e if and only if at least one of its corresponding OLP-interpretations is a model of the OLP theory.*

We first prove an important lemma.

Definition 6.3 *Given an \mathcal{ER} -interpretation I and an interpretation J_G of the choice predicate which satisfies the FOL axioms for that predicate. $J_{I,G}$ is the OLP-interpretation which corresponds to I (i.e. assigns the value corresponding to I to each atom corresponding to an \mathcal{ER} -atom) and which coincides with J_G on the choice predicate.*

Lemma 6.1 *To each grounding G of Π_e corresponds an interpretation J_G of the choice predicate such that an \mathcal{ER} -interpretation I is a model of Π_p and G if and only if $J_{I,G}$ is a model of the OLP theory.*

Proof:

To prove the lemma, we determine an interpretation of the *choice* predicate which allows us to use the proof of the deterministic case given in section 5. For a particular $J_{I,G}$, it is easy to see that all steps of the proof apply without modification, except for one of the inductive steps, i.e. the proof that at any time t , $causes(t, l) \leftrightarrow Causes(t, l)$ follows from $holds(F, t) \leftrightarrow Ho(F, t)$. For this to be valid it is required that G is equivalent to the definition of *causes*. Hence, if we can find an interpretation J_G which makes the OLP-definition of *causes* equivalent to the given grounding G of Π_e , it follows that I is a model of $G \cup \Pi_p$ if and only if $J_{I,G}$ is a model of the OLP theory.

Determining an appropriate interpretation of *choice* is not difficult: it follows from the definition of groundings and the construction of the mapping that, for each nondeterministic rule r and each time point t , the set of clauses in the mapping of r is equivalent to the k th grounding of r (i.e. the grounding corresponding to r 's k th disjunctive component), if $choice(c_r, k, t)$ is true and all other $choice(c_r, j, t)$ are false. Indeed, as indicated before, given these values for $choice(c_r, i, t)$, all clauses corresponding to other groundings than the k th one have a trivially false body at t whereas the clauses corresponding to the k th grounding are the clauses in the grounding of the k th disjunctive component with an additional true atom in the body.

So, given a grounding G corresponding to each time point t and for each rule r to the $k_{r,t}$ th disjunct of r , if we choose J_G such that $choice(c_r, i, t)$ is false if $i \neq k_{r,t}$ and true if $i = k_{r,t}$, then I is a model of $G \cup \Pi_p$ if and only if $J_{I,G}$ is a model of the OLP theory. Moreover this choice satisfies the given FOL axioms on *choice*. This proves the lemma. \square

Proof:

(of the theorem) The theorem follows immediately from the lemma if each interpretation J of the *choice* predicate corresponds to a grounding G . This is indeed the case: in each J exactly one atom, say $choice(c_r, i_{r,t}, t)$, is true for each r and t , due to the FOL axioms. By the construction of J_G , J then corresponds to the grounding G obtained by taking the $i_{r,t}$ th disjunct of each r at each t . \square

The above reasoning proves the correctness of the proposed mapping. We can then turn our attention to the relation between nondeterministic rules in \mathcal{ER} and the causal rules introduced in [31].

7 \mathcal{ER} Compared with the Approach of Thielscher

In this section we make a detailed comparison of \mathcal{ER} derived effect rules with Thielscher's approach to the ramification problem ([31]). The reason for this is

twofold: first of all Thielscher’s approach is one of the most recent ones, has been compared by Thielscher with a good deal of other recent approaches, and looks reasonably similar to ours. Therefore we think it is worth analysing the correspondence in detail. A second reason is that Thielscher has introduced the use of influence information for deriving certain causal rules from state constraints, which has struck us as a very interesting idea. Further on we will propose a different approach in \mathcal{ER} and show how it relates to Thielscher’s.

In what follows we will distinguish between *causal rules* (the constructs used by Thielscher) and *derived effect rules* (their counterpart in \mathcal{ER}). We will show a close correspondence between Thielscher’s causal rules and nondeterministic derived effect rules.

In [31], causal rules have the form

$$e \text{ causes } l \text{ if } F$$

where e and l are fluent literals and F a fluent formula. Intuitively, the semantics of such a rule is that strong initiation of e may cause weak initiation of l if F holds.²⁸ More precisely:

Definition 7.1 (concepts related to causal rules)

Given a state S and a set of effects E (i.e. a set of fluent literals known to be strongly initiated), the rule

$$e \text{ causes } l \text{ if } F$$

is applicable in (S, E) iff $e \in E$ and $S \models F \wedge e \wedge \bar{l}$.²⁹

Applying this rule results in a new state $(S \setminus \{\bar{e}\}) \cup \{e\}$ and a new set of effects $E \cup \{e\}$.³⁰

The successor states after the execution of an action A in a state S are obtained by applying sequences of applicable causal rules to the pair (S^*, E) , where E is the set of direct effects of A and $S^* = (S \setminus \bar{E}) \cup E$, i.e. the state obtained after applying the direct effects E to S . S' is a successor state of S after action A iff a pair (S', E') is obtained from (S^*, E) after any sequence of rule applications and S' satisfies all of the state constraints. In the sequel, we call the above E' the justifying set of effects of S' .

It is important to note that any sequence of applicable rules which leads to a state satisfying the state constraints, is valid. In particular, such a sequence does not need to contain *all* applicable rules, hence the intuitive reading that rules “may” cause effects. As a result, nondeterministic ramifications are obtained, as will be apparent in the comparison below.

There is a partial correspondence between Thielscher’s causal rules and nondeterministic derived effect rules in \mathcal{ER} . Let us first look only at causal rules of the form

$$a \text{ causes } b$$

with a and b fluent literals. Such a rule states that a strong initiation of a justifies the initiation of b , i.e. it may result in the initiation of b , but does not necessarily do so (since a rule is never forced to be applied). This corresponds at first sight to our nondeterministic derived effect rule

$$\text{initiating } a \text{ causes } b \mid \text{true if true.}$$

²⁸We say “may cause” rather than “causes” since rules never *need* to be applied in Thielscher’s approach.

²⁹The condition $S \models \neg l$ guarantees that all computed effects are strong initiations.

³⁰Observe that E may contain a subset of the form $\{e, \neg e\}$.

The rules are indeed equivalent under the condition of *initiation consistency* (a fluent cannot at the same time be initiated and terminated).³¹ To make this more precise, we introduce the following notation.

Definition 7.2 (initiation consistency) *A set of effects E is initiation consistent iff it does not contain a subset of the form $\{f, \neg f\}$.*

Definition 7.3 (applicability/justification for causal rules)

A fluent literal a is strongly initiated in (S, E) iff $a \in E$ and E is initiation consistent. Then, it follows from the above definition that a causal rule a causes b is applicable in (S, E) iff a is strongly initiated in (S, E) . We say the rule then justifies initiation of b , and moreover justifies strong initiation of b iff $b \notin S$.

Definition 7.4 (applicability/justification/(candidate) successor state in \mathcal{ER})

*A derived effect rule **initiating a causes b if true** or a nondeterministic derived effect rule **initiating a causes b | true if true** is applicable at t iff a is strongly initiated at t . The rule then justifies initiation of b , and justifies strong initiation of b iff b does not hold at t . Applying this rule to a set E of strongly initiated literals at t yields the set $E \cup \{b\}$.*

A set of fluent literals S' is a candidate successor state of S after A according to a set of effect rules Π_e iff there is a grounding \mathcal{D}_{init} of Π_e for which it follows from $\mathcal{H}a(A, t) \wedge \forall l \in S : \mathcal{H}o(l, t)$ that $S' \setminus S = \{l \mid I_{\mathcal{D}_{init}}(\mathbf{Init}(t, l')) = \mathbf{t}\}$ and $S' \cap S = \{l' \in S \mid I_{\mathcal{D}_{init}}(\mathbf{Init}(t, l')) = \mathbf{f}\}$. Intuitively a candidate successor state is a set of literals that can be true immediately after the action according to the effect rules.

A candidate successor state S' is a successor state iff it satisfies all the state constraints.

Note that the above definitions of applicability deviate from Thielscher's in that we don't impose $S \models a \wedge \neg b$. However, from the construction of (S, E) pairs, it follows that $E \subseteq S$ if E is initiation consistent. Hence $S \models a$ is trivially satisfied if $a \in E$. As for b , we say a rule is applicable even if b already holds. However, the rule then only justifies weak, rather than strong initiation of b .

We then have the following results:

Lemma 7.1 *Given a state S at time t and an initiation consistent set E of strongly initiated literals at t , the causal rule*

$$a \text{ causes } b$$

justifies strong initiation of a literal $b \notin (E \cup \overline{E})$ in $((S \setminus \overline{E}) \cup E, E)$ if and only if the derived effect rule

$$\text{initiating } a \text{ causes } b \text{ if true}$$

justifies the same strong initiation at t .

Proof:

Either rule is applicable if and only if a is strongly initiated. In that case, the causal rule justifies strong initiation of b iff $b \notin ((S \setminus \overline{E}) \cup E)$. Since $b \notin (E \cup \overline{E})$, this is equivalent to the condition $b \notin S$. On the other hand, the derived effect rule justifies strong initiation of b at t iff b does not hold at t . Since S is the state at t , this is also equivalent to $b \notin S$. \square

³¹ This condition is not imposed by Thielscher since in his approach change propagations are assumed to happen consecutively and to take a very small amount of time. Therefore there is no problem with initiations and terminations of fluents in the same batch of effects. However, we prefer not to adopt such a treatment of very small delays and for now we consider all propagations to be simultaneous and instantaneous, in which case initiation consistency is an essential condition. We return to the issue of delays in much detail in section 9.

Lemma 7.2 *Given a state S at time t and an initiation consistent set E of strongly initiated literals at t , applying the causal rule*

$$a \text{ causes } b$$

yields an initiation consistent set of strong initiations E' if and only if applying the derived effect rule

$$\text{initiating } a \text{ causes } b \text{ if true}$$

to E at t yields the same initiation consistent set E' .

Proof:

There are three cases. If $b \in E$ then $E = E'$. If $b \in \overline{E}$ then E' is not initiation consistent. If $b \notin (E \cup \overline{E})$, either both rules justify strong initiation of b , in which case $E' = E \cup \{b\}$, or neither justifies b , in which case $E' = E$. \square

Lemma 7.3 *Given a state S and set of effects E , and a causal rule mapping (S, E) to (S', E') . If E' is initiation consistent, any causal rule applicable in (S, E) is also applicable in (S', E') . Likewise, any nondeterministic derived effect rule applicable at t given E is also applicable at t given E' if E' is initiation consistent.*

Proof:

The lemma follows immediately from the observation that $E \subseteq E'$. \square

Lemma 7.4 *Given a state S at time t , an action A with direct effects E occurring at t , a set of causal rules $\{C_1, \dots, C_n\}$ and a corresponding set of derived effect rules $\{D_1, \dots, D_n\}$.*

1. *If (S', E') is obtained by applying a sequence of rules $(C_{s_1}, \dots, C_{s_m})$ to $((S \setminus \overline{E}) \cup E, E)$ and S' satisfies the state constraints, then*
 - (a) *If E' is initiation consistent, then S' is the successor state of S according to the set of derived effect rules $\{D_{s_1}, \dots, D_{s_m}\}$.*
 - (b) *If E' is not initiation consistent, S has no successor state after A according to $\{D_{s_1}, \dots, D_{s_m}\}$.*
2. *If S' is the successor state of S after A according to a subset of the derived effect rules $\{D_1, \dots, D_n\}$, it is a successor state according to the corresponding subset of causal rules and its justifying set of effects E' is initiation consistent.*

Proof:

1. From the previous lemmas it follows that applying the sequence of rules $(C_{s_1}, \dots, C_{s_m})$ to $(S_0, E_0) = ((S \setminus \overline{E}) \cup E, E)$ yields a sequence of pairs $((S_0, E_0), \dots, (S_m, E_m))$ such that D_{s_i} is applicable to (S_{i-1}, E_{i-1}) and yields (S_i, E_i) . Since each derived effect rule remains applicable in all (S_j, E_j) with $i < j$ if it is applicable in (S_i, E_i) , it follows that all derived effect rules are applicable in (S_m, E_m) , i.e. their conditions are satisfied. Since all derived effect rules are applied in the construction and the set of effects grows monotonically, the heads of all rules are also satisfied in (S_m, E_m) .

Now, assume S' is the candidate successor state of S according to the derived effect rules. Note that these rules form a definite definition, and there exists exactly one such S' . This S' is a successor state of S iff the initiation consistency condition in \mathcal{ER} is satisfied and S' satisfies the state constraints.

- (a) Assume E_m is initiation consistent. We then show that $S' = S_m$. Since only the heads of applicable rules are added to E and the number of rule applications is finite, each initiation of a literal in E_m has a true finite proof tree, hence all literals in E_m are true in S' . The initiation of any literal not in $E_m \cup S$ only has a false proof tree, since if it had been the head of any applicable rule, it would have been added to E . Therefore all literals not in $E_m \cup S$, i.e. all literals in $\overline{S \setminus E_m}$, are false in S' . Since S is a state, one of each pair $(f, \neg f)$ is true in S . Hence the true literals in S' are exactly those in $(S \setminus \overline{E_m}) \cup E_m$, which is equal to S_m since E_m is initiation consistent. By the construction $S_m = S'$ satisfies all the state constraints, so S' is a successor state of S .
- (b) Assume E_m is not initiation consistent. Then S' is not a successor state of S as the initiations leading to it violate the initiation consistency condition in \mathcal{ER} . Since S' is the unique candidate successor state for S , S has no successor state.
2. Assume S' is the successor state of S after A according to the direct effect rules for A and the derived effect rules $\{D_{t_1}, \dots, D_{t_k}\}$. Starting from $(S_0, E_0) = ((S \setminus \overline{E}) \cup E, E)$, we can build a sequence $((S_0, E_0) \dots (S_m, E_m))$ by randomly applying one applicable rule from the set $\{C_{t_1}, \dots, C_{t_k}\}$ in each step, until no more rules apply. Assume we order the C_{t_j} and D_{t_j} such that C_{t_i} is the rule applied in step i . E_m may now be initiation consistent or not.

Assume then E_m is initiation consistent. S_m is the successor state according to $\{D_{t_1}, \dots, D_{t_m}\}$. The rules $C_{t_{m+1}}, \dots, C_{t_k}$ are not applicable in (S_m, E_m) , so neither are $D_{t_{m+1}}, \dots, D_{t_k}$. Consider then adding these rules to the definition $\{D_{t_1}, \dots, D_{t_m}\}$. Since the rules $D_{t_{m+1}}, \dots, D_{t_k}$ are not applicable given E_m , their bodies are not in E_m , so the additional proof trees they give rise to can not contain true leaves. Hence, only finitely failed or infinite proof trees are added to the set of proof trees. Since such proof trees yield false as a truth value, the value of the best proof tree of each literal is unchanged. So S_m is also the successor state of S after A according to $\{D_{t_1}, \dots, D_{t_m}\}$, i.e. $S_m = S'$.

On the other hand, if E_m is initiation inconsistent, the candidate successor state $S' = S_m$ violates the initiation consistency condition. The addition of $\{D_{t_{m+1}}, \dots, D_{t_k}\}$ to the definition results only in additional proof trees for some effects, so any effect with a true proof tree also has this true proof tree according to the extended definition. Hence, the set of effects certainly contains E_m and therefore is initiation inconsistent. So according to $\{D_{t_1}, \dots, D_{t_k}\}$ there is no successor state to S .

□

Theorem 7.1 *Given a state S at time t , an action A with direct effects E occurring at t , a set of causal rules $\{C_1, \dots, C_n\}$ where $C_i = a_i$ causes b_i and a corresponding set of nondeterministic derived effect rules $\{ND_1, \dots, ND_n\}$ where $ND_i = \text{initiating } a_i \text{ causes } b_i \mid \text{true if true}$. S' is a successor state for S after A according to $\{ND_1, \dots, ND_n\}$ iff S' is a successor state of S after A according to $\{C_1, \dots, C_n\}$ and the justifying set of effects E' of S' is initiation consistent.*

Proof:

S' is a candidate successor state of S according to $\{C_1, \dots, C_n\}$ if it is obtained after applying any sequence of applicable causal rules. This is equivalent to the

condition that S' is obtained after application of a maximal sequence of causal rules of any subset of $\{C_1, \dots, C_n\}$ (since each sequence is a maximal sequence in any subset of $\{C_1, \dots, C_n\}$ containing all the applied rules and none of the applicable unapplied rules, and since vice versa any maximal sequence in a subset is also a sequence in that subset and hence a sequence in the entire set). An S' satisfying the above condition is a successor state for S iff it satisfies all of the state constraints.

In \mathcal{ER} the candidate successor states of S according to $\{ND_1, \dots, ND_n\}$ are those obtained by any grounding of this definition. Consider such a grounding: by definition, for a rule **initiating a_i causes b_i | true if true**, for each particular time point t the grounding of this rule is equivalent to the grounding of $D_i =$ **initiating a_i causes b_i if true** or to the grounding of the trivially satisfied rule $T_i =$ **initiating a_i causes true if true**. Hence, S'' is a successor state of S according to $\{ND_1, \dots, ND_n\}$ if and only if it is a candidate successor state of S according to any definition $\{D_{i_1}, \dots, D_{i_k}\} \cup \{T_{j_1}, \dots, T_{j_l}\}$ such that $(\{i_1 \dots i_k\}, \{j_1 \dots j_l\})$ is a partition of $\{1 \dots n\}$. Since the rules T_j only result in proof trees for *true*, omitting them from the definition does not influence the candidate successor state. Hence S'' is a candidate successor state of S according to $\{ND_1, \dots, ND_n\}$ iff it is the candidate successor state according to any subset $\{D_{i_1}, \dots, D_{i_k}\}$ of $\{D_1, \dots, D_n\}$. It is a successor state of S iff in addition it satisfies the state constraints and the set of effects leading to it is initiation consistent.

From the previous lemma it follows that the conditions on S' and S'' are equivalent if E' is initiation consistent, which proves the theorem. \square

The above comparison shows that Thielscher's causal rules **a causes b** have the same semantics as \mathcal{ER} rules **initiating a causes b | true if true** under the condition of initiation consistency.

The situation is more complicated for causal rules of the form

$$a \text{ causes } b \text{ if } F$$

which, roughly speaking, say that strong initiation of a causes strong initiation of b provided that F holds. Such rules do *not* correspond to our rules

$$\text{initiating } a \text{ causes } b \text{ | true if } F.$$

We will show that in \mathcal{ER} there is no equivalent for this complex type of causal rule. Closest to it is the combination of the two derived effect rules

$$\begin{aligned} &\text{initiating } a \text{ causes } b \text{ | true if } F \\ &\text{initiating } a \wedge F \text{ causes } b \text{ | true if } \neg a \end{aligned}$$

The first rule states that b may be initiated if a is strongly initiated, provided that F holds in the starting state S for this set of effects. The second rule states the same relation between changes in a and b , but now under the condition that F holds in the resulting state S' rather than the starting state (initiating $a \wedge F$ given that a was false, is equivalent to strongly initiating a while making sure that F becomes or stays true). Yet these two rules do not cover all the cases in which Thielscher's causal rule is applicable.

This is due to the fact that in Thielscher's approach, rules are applied consecutively, giving rise to a set of intermediate states. A rule **a causes b if F** can be applied at a certain point in such a sequence if F holds in the appropriate intermediate state. Thielscher shows some examples in which a fluent holds in one of the intermediate states, but neither in the starting nor the resulting state. In such a case, none of our derived effect rules would be applicable, but Thielscher's causal rule would. If F is a fluent, this difference is eliminated by

the initiation consistency condition, which prevents fluents from being initiated and terminated in the same batch of effects. However, as F can be a general formula, to obtain equivalence the initiation consistency condition would have to be extended in a way which is no longer reasonable: if no formula can be initiated and terminated in one batch of effects, we cannot allow even two consecutive effects. Indeed, assume we initiate a and b consecutively, then for example $a \wedge \neg b$ and $\neg a \wedge b$ are initiated and terminated in the same batch of effects, which is in no way problematic or counterintuitive.

For causal rules of this more complex form, we have no exact equivalent in \mathcal{ER} . The best approximation is given by two rules, as indicated above. One consequence is that any approach to using influence information in \mathcal{ER} will certainly deviate from Thielscher's. In the next section we present such a method and compare it to Thielscher's approach.

8 Influence Information

As we have extensively argued, in our approach ramifications are seen as manifestations of effect propagations. In other words, we assume that an expert modeling a dynamic domain models the effects existing in that domain and describes how these effects propagate. In \mathcal{ER} these propagations are represented by derived effect rules. However, in the literature state constraints have always been used in a high level description of dynamic domains: these state constraints are easily observable, and as they can arise as a result of particular combinations of effect propagations, they are often strongly related to these propagations.

For this reason, Thielscher ([31]) proposes a method for automatically deriving causal rules representing ramifications from state constraints, using influence information. The idea is that if two fluents f and g occur in the same state constraint, and f influences g , then a change in f which results in a violation of the constraint may cause an appropriate change in g which restores the validity of the constraint. The appropriate causal rules describing these change propagations are then derived.

Thielscher's causal rules thus have the explicit task of restoring the validity of state constraints when they are violated. Hence they are not descriptions of known effect propagations like derived effect rules in \mathcal{ER} : causal rules only need to be applied if there are violated state constraints and if they can solve that problem. This is reflected in the fact that they map to nondeterministic derived effect rules in \mathcal{ER} : they describe potential effect propagations that may occur if they are required.

Despite the different point of view on ramifications, the idea of automatically deriving ramification rules from state constraints and influence information also deserves consideration in \mathcal{ER} . For example, a formal characterisation of the relation between state constraints, influence information and derived effect rules can be useful to help an expert derive the precise effect propagation rules starting from a high-level description consisting only of state constraints and vague influence information.

Of course such an approach has limitations. First of all, as we have shown in section 2, not all effect propagations are related to state constraints. Hence it is not possible in general to generate all effect propagation rules using state constraints and influence information. Second, even in the case of state constraint related ramifications the addition of influence information may still leave the ramifications underspecified, as we will show. Finally, determining whether a set of rules is the intended set for given constraints and influence information is not trivial: the problem at hand shows some similarities to the tasks of machine learning and intentional database updating.

Despite these limitations, the above argument still stands. Therefore we provide in this section a method for generating a set of derived effect rules corresponding to given state constraints and influence information. In addition, the rule sets we generate give an indication of when the effect propagations are underspecified, thus warning the user that the generated rules are based on insufficient information and may require further attention. In this respect the method should be seen as a tool helping the user and not as a fully independent answer to the ramification problem. Moreover the examples should show that such an independent answer is not feasible, despite approximations by both Thielscher and ourselves.

8.1 Generating effect propagation rules

Our goal is to generate derived effect rules from a given set of state constraints and a given influence relation $Infl$ which is a binary relation on fluents: $(f, g) \in Infl$ if f can influence g . These derived effect rules should guarantee that whenever f is modified such that a particular state constraint containing f and g would be violated, an appropriate modification of g can occur which restores or helps to restore the validity of the constraint. On the other hand, indirect effects should also only be allowed if they satisfy the above condition.

We assume that the set of state constraints is written in conjunctive normal form, i.e. a conjunction of constraints where each constraint is a disjunction of fluent literals. We use the convention that in a constraint $l_1 \vee \dots \vee l_m$, each l_j is either f_j or $\neg f_j$ for a particular fluent f_j .

Since the state constraints and influence information may not provide sufficient information to determine the intended effect rules, it is not possible in general to give necessary and sufficient conditions for a set of effect rules to be correct. However, based on the intuitions about influence information sketched above, we propose the following correctness criterion, which is a necessary condition on the sets of effects that can be generated by the effect rules.

Definition 8.1 (weak correctness criterion)

A correct set of effects E with respect to given state constraints and influence information and a given starting state S , must satisfy the following conditions:

- For any state constraint C , $(S \setminus \overline{E}) \cup E \models C$
- Each effect in E is either
 - a direct effect, or
 - an indirect effect l_k such that for $E' = E \setminus \{l_k\}$ there exists some $C = \bigvee_{i=1 \dots n} l_i$ and some l_j ($1 \leq j, k \leq n$) such that $(f_j, f_k) \in Infl$, $\overline{l_j} \in E'$, and for some $E'' \subseteq E'$ with $\overline{l_j} \in E''$, $S \setminus \overline{E''} \cup E'' \not\models C$

In other words, resulting states must satisfy all state constraints, and each effect generated by the effect rules must be justified by some state constraint which would be violated due to other occurring effects and by the appropriate influence information. A stronger variant of the correctness criterion is obtained by imposing in addition that $E' = E''$:

Definition 8.2 (strong correctness criterion)

- For any state constraint C , $S \setminus \overline{E} \cup E \models C$
- Each effect in E is either

- a direct effect, or
- an indirect effect l_k such that for $E' = E \setminus \{l_k\}$ there exists some $C = \bigvee_{i=1\dots n} l_i$ and some l_j ($1 \leq j, k \leq n$) such that $(f_j, f_k) \in \text{Infl}$, $\bar{l}_j \in E'$, and $S \setminus \bar{E}' \cup E' \not\models C$

This variant requires that each effect only takes place if the state constraint it intends to restore would be violated by the combination of *all* other effects. This adds a strong minimality condition to the weak correctness criterion: if there are two ways to restore the validity of a state constraint, only one of them should be adopted, never both. It can be argued that the strong version of the correctness criterion is to be preferred, and in fact the rules generated by our approach satisfy the strong criterion. However we will show below that due to possible underspecification a violation of the strong criterion (which for example occurs in Thielscher's approach) is sometimes acceptable.

Our approach is modular, in that the derived effect rules are generated for each state constraint independently. First we study the case in which effect propagations follow unambiguously from a state constraint C and the influence information. It is easy to see that this is always the case when in C only one of the fluents, say f , can be influenced: then the only possible indirect effect is a modification of f . Hence, if such a modification is allowed by the influence information (i.e. if one of the fluents influencing f is changed) and if it would restore the constraint's validity, then f should always be changed: otherwise the state constraint would remain violated.

This propagation can be enforced as follows: for each state constraint $l_1 \vee \dots \vee l_m$ (with each l_j either f_j or $\neg f_j$ for a particular fluent f_j), which contains at most one l_k such that f_k can be influenced by any other fluent in the language, generate the derived effect rule

$$\text{initiating } \bigwedge_{t=1\dots m, t \neq k} \bar{l}_t \text{ causes } l_k \text{ if } \bigvee_{f_i \in I} l_i$$

where I is the set of fluents in the constraint that can influence f_k .

In this rule, the body $\bigwedge_{t=1\dots m, t \neq k} \bar{l}_t$ is the negation of the state constraint with l_k left out: the formula denotes that a necessary condition for l_k to be caused is that in the resulting state, the state constraint is violated unless l_k is true. The condition of the rule, $\bigvee_{f_i \in I} l_i$, denotes that at least one of the l_i which can influence l_k is true in the starting state, and hence is changed, so that its effect can propagate.

This case covers a very large class of applications, including the suitcase, table and flying turkey examples presented in earlier sections and nearly all examples studied in the literature. For example in the suitcase domain the influence information is that l_1 and l_2 may influence *open*, so certainly in any state constraint at most one fluent (*open*) can be influenced. The state constraint in disjunctive form in this example is $\neg l_1 \vee \neg l_2 \vee \text{open}$, which leads to the derived effect rule

$$\text{initiating } l_1 \wedge l_2 \text{ causes } \text{open} \text{ if } \neg l_1 \vee \neg l_2$$

Note that this rule is syntactically different from the one we used when modeling the example earlier. Indeed, the above rule can be simplified to

$$\text{initiating } l_1 \wedge l_2 \text{ causes } \text{open} \text{ if } \text{true}$$

since $l_1 \wedge l_2$ can be initiated only if its negation is true. This simplification can be generalised: if the one fluent in a constraint which can be influenced is

influenced by all other fluents in the constraint, we can generate the derived effect rule

$$\text{initiating } \bigwedge_{t=1\dots m, t \neq k} \bar{l}_t \text{ causes } l_k \text{ if } true$$

which is equivalent with the one generated using the general method.

Next we consider the case of constraints in which multiple fluents can be influenced. In general there is no unique set of indirect effects which can restore the constraint's validity in this case: the problem is underspecified and/or involves nondeterminism.

This is very clear in the following example: consider the state constraint $a \rightarrow (b \vee c)$, in disjunctive form $\neg a \vee b \vee c$, with the influence information $\{(a, b), (a, c)\}$. Assuming that we start with a state in which all three fluents are false, and then initiate a , the constraint's validity can be restored by initiating either b or c , or both. Since there is no reason to prefer either b or c , the rules we generate must be nondeterministic. However it is not clear if only the minimal changes (initiating only b or only c) are to be considered, or also the change in both b and c . Adhering to the strong version of the correctness criterion given above, only the minimal solutions would be acceptable: if b is initiated there is no justification for c and vice versa. The weaker version of the correctness criterion allows for a change in both b and c , since both are justified by the change in a . Here we can argue there is a problem of underspecification.

To make the discussion more concrete, assume a , b and c are courses taught at a university. The constraint represents that b and c are prerequisites for a , and the influence information that if a student wants to follow course a but has not followed b or c , his/her subscription for a can be allowed by subscribing him/her in addition to course b or c . The given information does not specify which of b or c is to be preferred. Real students would probably not be happy with two additional subscriptions and prefer the minimal adaptations, but there may be exceptions to this rule. Also, usually one would talk to the student to determine which additional course (s)he prefers, and not assign one nondeterministically. In other words, more information is clearly required to determine the best course of action.

Given the above considerations, in our view the preferable general solution is to propose only minimal sets of changes but to clearly indicate that there is a problem of underspecification.

This is achieved by the following formalisation: if multiple fluents, say $f_1 \dots f_s$, in one state constraint $l_1 \vee \dots \vee l_m$ can be influenced by other fluents in the theory, the nondeterministic derived effect rules

$$\text{initiating } \bigwedge_{t=1\dots m, t \neq k} \bar{l}_t \text{ causes } l_k \mid true \text{ if } \bigvee_{f_i \in I_k} l_i$$

are generated for $1 \leq k \leq s$, where I_k is the set of fluents in the constraint that can influence f_k . In other words, for each l_k we obtain a rule like the one we obtained in the previous case, except for the fact that the effect of the rule is now optional: the rule is nondeterministic. Of course since state constraints need to be satisfied at all times, it is required that if the state constraint is violated by a set of direct effects at least one of the rules is applied. In the above example, the obtained effect rules are

$$\begin{aligned} \text{initiating } a \wedge \neg b \text{ causes } c \mid true \text{ if } \neg a \\ \text{initiating } a \wedge \neg c \text{ causes } b \mid true \text{ if } \neg a \end{aligned}$$

At any particular time point, the semantics of the rules is given by one of four possible groundings, corresponding to the groundings of the following pairs of

deterministic rules.

initiating $a \wedge \neg b$ causes $true$ if $\neg a$
initiating $a \wedge \neg c$ causes $true$ if $\neg a$

initiating $a \wedge \neg b$ causes c if $\neg a$
initiating $a \wedge \neg c$ causes $true$ if $\neg a$

initiating $a \wedge \neg b$ causes $true$ if $\neg a$
initiating $a \wedge \neg c$ causes b if $\neg a$

initiating $a \wedge \neg b$ causes c if $\neg a$
initiating $a \wedge \neg c$ causes b if $\neg a$

In the case where a is initiated while b and c are false, the first pair of rules does not lead to a state satisfying the state constraint, and can be disregarded. The second and third pairs of rules yield minimal changes restoring the validity of the constraint, initiating either b or c . The fourth pair of rules yields a truth value “u” for the initiations of both b and c , so this is not a correct constructive definition.

Recall that a set of initiations satisfies a nondeterministic definition if it is a model of one of the definition’s groundings. Hence we find two sets of initiations consistent with both the definition and the state constraint: one in which b is initiated and one in which c is. These are precisely the minimal changes able to restore the state constraint’s validity. However, as we indicated we cannot be entirely certain if only the minimal changes should be considered: whether or not this is the case may be domain dependent, and the answer can in general not be given by state constraints and influence information alone. So we are not certain if the generated set of rules is the intended one, due to underspecification. Luckily, our proposal naturally incorporates a warning when such a problem of underspecification occurs: this warning is the presence of a bad definition in one of the groundings. Though this bad definition has no impact on the semantics (it is an additional grounding which yields no valid set of initiations), its presence warns the user that the problem may require further analysis. In this case the user may look into the details of the domain at hand, check which effect propagations are intended and decide to modify the generated rules if needed. By default the minimal change policy is maintained.

In the above example we had one fluent influencing two other fluents in the same constraint. The case in which two different fluents influence a third resp. fourth fluent in the same constraint, as in $\neg a \vee b \vee \neg c \vee d$ with influence information $\{(a, b), (c, d)\}$, is entirely similar: if a and c are initiated when b and d are false, the constraint may be restored by initiating either b or d . We again obtain two nondeterministic effect rules of which one grounding does nothing and yields a state violating the constraint, two groundings yield the minimal changes, and one grounding is a bad definition indicating a problematic specification.

A third possibility is that two fluents in a constraint C can be influenced, but only one of them by another fluent in C and the second one by a fluent outside C . For example, assume the constraints

$$\begin{aligned} \neg a \vee b \vee \neg c \\ \neg d \vee c \end{aligned}$$

with influence information $\{(a, b), (d, c)\}$. For the first constraint, due to the

influence of a on b , we obtain a rule

initiating $a \wedge c$ causes b | true if $\neg a$

As should be expected, the influence of d on c does not lead to effect rules for the first constraint. It does however generate an — in this case deterministic, as only c can be influenced — rule for the second constraint:

initiating d causes c if $\neg d$

Assuming a state in which all fluents are false and a and d are initiated, the second rule will cause c to be initiated and as a result, since now $a \wedge c$ is initiated, the first rule may or may not initiate b depending on which of the two groundings is considered. However, the case in which b is not initiated leads to a state violating the first constraint, so only the other grounding yields a model. Hence there is no nondeterminism in this case, even though the nondeterministic derived effect rule suggests there is.³² Also, there is no grounding which yields a bad definition. Indeed there is no problem of underspecification in this example: the effect rules determine a unique set of effects which restores all state constraints.

We are now ready to prove that the proposed method generates rules that satisfy the correctness criterion given at the beginning of this section:

Theorem 8.1 *Any set of effects E obtained by applying to a particular valid \mathcal{ER} -state S a given set of direct effect rules and a set of derived effect rules generated from state constraints and influence information by the above method, and which yields a valid \mathcal{ER} -state S' , satisfies the strong correctness criterion, i.e.*

1. For any state constraint C , $S \setminus \overline{E} \cup E \models C$
2. Each effect in E is either
 - (a) a direct effect, or
 - (b) an indirect effect l_k such that for $E' = E \setminus \{l_k\}$ there exists some $C = \bigvee_{i=1..n} l_i$ and some l_j ($1 \leq j, k \leq n$) such that $(f_j, f_k) \in Infl$, $\overline{l_j} \in E'$ and $S \setminus \overline{E'} \cup E' \not\models C$

Proof:

1. Since $S' = S \setminus \overline{E} \cup E$ is a valid \mathcal{ER} -state, it entails all state constraints.
2. Any occurring effect l_k must occur in the head of a rule with a true body. This can either be a direct effect rule, or a derived effect rule obtained from the state constraints and influence information by our method. In the former case condition 2(a) is satisfied. In the latter case, the derived effect rule is of the form

$$\text{initiating } \bigwedge_{t=1..m, t \neq k} \overline{l_t} \text{ causes } D_k \text{ if } \bigvee_{f_i \in I_k} l_i$$

where D_k is either l_k or l_k | true. This rule is necessarily obtained from the state constraint $C = l_1 \vee \dots \vee l_m$ and the influence information that all fluents in I_k can influence f_k . For this rule to be applicable it is required

³²One may wonder why we do not generate deterministic rules in this case. We show the difference in the next section when comparing deterministic and nondeterministic rules.

that some l_j that can influence f_k is true in S , and that $\bigwedge_{i=1\dots m, i \neq k} \bar{l}_i$ is false in S' . Since $(S \setminus \bar{E}') \cup E' = (S' \setminus l_k) \cup \bar{l}_k$, the second of these conditions implies that $S \setminus \bar{E}' \cup E' \neq C$. Also, the two conditions together imply that $\bar{l}_j \in E'$. Hence, condition 2(b) is satisfied.

□

8.2 A more uniform notation

The above method for dealing with influence information generates deterministic or nondeterministic effect rules depending on the number of fluents which can be influenced in each particular constraint. In this way we obtain clear deterministic rules when possible, and nondeterministic rules if there are multiple options. Moreover we obtain only one rule for each constraint and each fluent which can be influenced in it, thus keeping the number of rules relatively low.

Formally we can devise a simpler, equivalent characterisation of the generated effect rules, in which not only each constraint but also each item of influence information is dealt with independently, and moreover each (state constraint, influence item) pair is handled in exactly the same way. We achieve this by applying two simplifications.

Our first simplification consists of handling each (state constraint, influence item) pair independently: for each state constraint $l_1 \vee \dots \vee l_m$ and for each influence information item (f_i, f_k) , we generate the derived effect rule

$$\text{initiating } \bigwedge_{t=1\dots m, t \neq k} \bar{l}_t \text{ causes } l_k \text{ if } l_i$$

if l_k is the only fluent in the constraint which can be influenced, and

$$\text{initiating } \bigwedge_{t=1\dots m, t \neq k} \bar{l}_t \text{ causes } l_k \mid \text{true if } l_i$$

otherwise.

The set of rules obtained in this way is equivalent to the one obtained by our above method:

Theorem 8.2 *For a particular state constraint $l_1 \vee \dots \vee l_m$ and a set of fluents $I_k = \{f_i \mid (f_i, f_k) \in \text{Infl}\}$, the derived effect rule*

$$\text{initiating } \bigwedge_{t=1\dots m, t \neq k} \bar{l}_t \text{ causes } l_k \text{ if } \bigvee_{f_i \in I_k} l_i$$

is equivalent to the set of derived effect rules

$$\{\text{initiating } \bigwedge_{t=1\dots m, t \neq k} \bar{l}_t \text{ causes } l_k \text{ if } l_i \mid f_i \in I_k\}.$$

Likewise, the nondeterministic derived effect rule

$$\text{initiating } \bigwedge_{t=1\dots m, t \neq k} \bar{l}_t \text{ causes } l_k \mid \text{true if } \bigvee_{f_i \in I_k} l_i$$

is equivalent to the set of derived effect rules

$$\{\text{initiating } \bigwedge_{t=1\dots m, t \neq k} \bar{l}_t \text{ causes } l_k \mid \text{true if } l_i \mid f_i \in I_k\}.$$

The proof is as follows. First consider the deterministic case: the grounding of a derived effect rule

initiating F causes l if F'

is

$$\{\mathbf{Causes}(t, l) \leftarrow \mathbf{Init}(t, S_i), \overline{\mathbf{Init}(t, S_p)}, \\ \mathcal{Ho}(S_p, t), \neg \mathcal{Ho}(F, t), \mathcal{Ho}(F', t) \\ | t \in \mathbf{T} \text{ and } S_i \cup S_p \text{ is a supporting set of } F'\}.$$

Since the derived effect rules mentioned in the theorem differ only in their condition, each yields a set of clauses of the form

$$\begin{aligned} \mathbf{Causes}(t, l_k) &\leftarrow B_1, \mathcal{Ho}(F', t). \\ &\vdots \\ \mathbf{Causes}(t, l_k) &\leftarrow B_n, \mathcal{Ho}(F', t). \end{aligned}$$

with F' the condition of the particular rule. Hence, the theorem follows if we can prove that each clause

$$\mathbf{Causes}(t, l_k) \leftarrow B_j, \mathcal{Ho}\left(\bigvee_{f_i \in I_k} l_i, t\right).$$

is equivalent to the set of clauses

$$\{\mathbf{Causes}(t, l_k) \leftarrow B_j, \mathcal{Ho}(l_i, t). \mid f_i \in I_k\}$$

Now, the first clause is equivalent to

$$\mathbf{Causes}(t, l_k) \leftarrow B_j.$$

if $\mathcal{Ho}(\bigvee_{f_i \in I_k} l_i, t)$ is true, and to

$$\mathbf{Causes}(t, l_k) \leftarrow \text{false}.$$

otherwise. On the other hand, the set of “primitive” clauses is equivalent to

$$\mathbf{Causes}(t, l_k) \leftarrow B_j.$$

if one of the $\mathcal{Ho}(l_i, t)$, and hence $\bigvee_{f_i \in I_k} \mathcal{Ho}(l_i, t)$, is true, and to

$$\mathbf{Causes}(t, l_k) \leftarrow \text{false}.$$

otherwise. The equivalence then follows from the fact that $\mathcal{Ho}(\bigvee_{f_i \in I_k} l_i, t) = \bigvee_{f_i \in I_k} \mathcal{Ho}(l_i, t)$.

The proof for nondeterministic rules is obtained by applying the above reasoning to each grounding independently. \square

A second simplification eliminates the distinction between constraints in which only one fluent is influenced and the other constraints. We can always use the following unique rule for deriving effect rules, regardless of the number of influenced fluents: for each state constraint $l_1 \vee \dots \vee l_m$ and for each influence information item (f_i, f_k) , we generate the nondeterministic derived effect rule

$$\mathbf{initiating} \quad \bigwedge_{t=1\dots m, t \neq k} \overline{l_t} \text{ causes } l_k \mid \text{true if } l_i.$$

In other words, we can always generate nondeterministic rules. The resulting definition is equivalent with the one using deterministic rules for constraints in which only one fluent is influenced.

Theorem 8.3 *Given a state constraint $l_1 \dots l_m$ in which only l_k can be influenced. The derived effect rule*

$$D = \text{initiating} \quad \bigwedge_{t=1 \dots m, t \neq k} \bar{l}_t \text{ causes } l_k \text{ if } l_i.$$

is equivalent to, i.e. leads to the same successor states as, the nondeterministic derived effect rule

$$ND = \text{initiating} \quad \bigwedge_{t=1 \dots m, t \neq k} \bar{l}_t \text{ causes } l_k \mid \text{true if } l_i.$$

Proof:

ND has two groundings: one which corresponds to the grounding of D and one which only contains a rule for $true$ (which corresponds to an omission of D from the rule set). Hence, it suffices to prove that this second grounding does not yield successor states not generated by the first grounding. Turning this around, it is sufficient to prove that any successor state of the rule set without D is also a successor state of the rule set with D .

Now, assume S' is a successor state of the rule set without D . If in the starting state S the fluent l_i is false, then bodies of rules derived from D are always false, so D has no effect. Otherwise, first observe that D 's addition cannot introduce new proof trees for any of the l_t , due to the fact that none of the l_t can be influenced. In other words, D cannot introduce cycles (in particular over negation), so the truth value of $\bigwedge_{t=1 \dots m, t \neq k} \bar{l}_t$ remains invariable with or without D . We then have the following cases. If $\bigwedge_{t=1 \dots m, t \neq k} \bar{l}_t$ is false in S' , then D is not applicable so its addition has no effect. Otherwise, $\bigwedge_{t=1 \dots m, t \neq k} \bar{l}_t$ is true (undefined truth values are not possible in a state). In that case, if l_k is true in S' , D has no additional effect, and if l_k is false in S' , then S' is no successor state since the state constraint is violated. Hence, we find that the addition of D does not eliminate any successor states of the rule set without D , which due to the above reasoning ensures that D and ND are equivalent. \square

One might expect that also for rules generated from constraints in which only one fluent can be influenced from within the constraint (but in which other fluents can be influenced from without the constraint, as in the last example of section 8.1), we would obtain a similar equivalence result between deterministic and nondeterministic rules. However, the following example shows that there is no equivalence in that case: take two constraints

$$a \vee c \vee d \quad b \vee c \vee d$$

and influence information $\{(a, c), (b, d)\}$. In both constraints, one fluent can be influenced from within and one from without the constraint. The rules generated from these data would be

$$\begin{aligned} &\text{initiating } \neg a \wedge \neg d \text{ causes } c \mid \text{true if } a \\ &\text{initiating } \neg b \wedge \neg c \text{ causes } d \mid \text{true if } b \end{aligned}$$

which for a starting state $\{a, b, \neg c, \neg d\}$ and direct effects $\neg a, \neg b$ would generate two valid successor states $\{\neg a, \neg b, c, \neg d\}$ and $\{\neg a, \neg b, \neg c, d\}$ obtained with minimal changes, and in addition a bad grounding indicating that there is insufficient information. On the other hand, the corresponding deterministic rules

$$\begin{aligned} &\text{initiating } \neg a \wedge \neg d \text{ causes } c \text{ if } a \\ &\text{initiating } \neg b \wedge \neg c \text{ causes } d \text{ if } b \end{aligned}$$

would only yield a bad definition and no successor states at all.

We have now obtained a uniform and modular characterisation of the derived effect rules corresponding to state constraints and influence information, which by the above theorems is equivalent to our original proposal. This original proposal can be seen as a more intuitive approach, in which sets of constraints obtained by the uniform method are contracted into single constraints, and in which apparent but non-existing nondeterminism has been eliminated. One advantage of the uniform method is that it will help us establish a correspondence with the approach in [31].

8.3 Comparing our method with Thielscher's

We study the relation between our approach and the one in [31]. Formally, the causal rules in Thielscher's approach are computed as follows: assume $D_1 \wedge \dots \wedge D_n$ is the conjunctive normal form of the conjunction of all state constraints, and each $D_i = l_1 \vee \dots \vee l_{m_i}$ with all l_j fluent literals f_j or $\neg f_j$. Then for each D_i and for each (f_j, f_k) in the influence relation with l_j and l_k in D_i , the causal rule

$$\overline{l_j} \text{ causes } l_k \text{ if } \bigwedge_{t=1 \dots m_i, t \neq j, t \neq k} \overline{l_t}$$

is generated.

For the same constraint and influence information, we generate the rule

$$\text{initiating } \bigwedge_{t=1 \dots m_i, t \neq k} \overline{l_t} \text{ causes } l_k \mid \text{true if } l_j$$

Recall from the previous section that this derived effect rule, together with

$$\text{initiating } \overline{l_j} \text{ causes } l_k \mid \text{true if } \bigwedge_{t=1 \dots m_i, t \neq k, t \neq j} \overline{l_t}$$

is the closest approximation in \mathcal{ER} of the above causal rule. In other words, there is an immediately clear correspondence between the generated rules in both approaches, with our derived effect rules being strictly and considerably weaker than Thielscher's causal rules.

Consider a state constraint $a \vee b \vee \bigvee_i l_i$ with (a, b) in the influence relation. This gives rise to the causal rule

$$\neg a \text{ causes } b \text{ if } \bigwedge_i \overline{l_i}.$$

and to the derived effect rule

$$\text{initiating } \neg a \wedge \bigwedge_i \overline{l_i} \text{ causes } b \mid \text{true if } a$$

The derived effect rule is applicable if $\neg a \wedge \bigwedge_i \overline{l_i}$ becomes true and a is strongly terminated. In that case b should be initiated for the constraint to remain satisfied. The corresponding causal rule is applicable in a sequence of causal rules if a is strongly terminated and $\bigwedge_i \overline{l_i}$ is true in some appropriate intermediate state. So the causal rule is applicable in a number of cases where the derived effect rule is not. In these cases however, the application of the rule is not required in order to restore the validity of the constraint it is derived from: the state constraint is only violated if all of its literals are false in the generated successor state, so if $\bigwedge_i \overline{l_i}$ holds in some intermediate (or the initial) state but not in the generated successor state, there is no need for b to be initiated. In other words, Thielscher's rules do not satisfy the strong correctness criterion: they sometimes generate unnecessary, non-minimal ramifications.

As an example, assume we have state constraints $a \vee \neg d$, $b \vee \neg d$ and $c \vee b \vee \neg a$. The influence relation is $\{(d, a), (d, b), (a, c)\}$. We then get the causal rules

d causes a
 d causes b
 a causes c if $\neg b$

Assume then we have a state $\{\neg a, \neg b, \neg c, \neg d\}$ and an action with direct effect d . Using the first two rules we find the successor state $\{a, b, \neg c, d\}$, which satisfies all of the state constraints. Also, using the first, third and second rule in that order we obtain a state $\{a, b, c, d\}$, which also satisfies the state constraints. Using the derived effect rules obtained from the state constraints,

initiating d causes a | true if $\neg d$
initiating d causes b | true if $\neg d$
initiating $a \wedge \neg b$ causes c | true if $\neg a$

we only find the resulting state $\{a, b, \neg c, d\}$: the first two rules must be applied to restore the first two constraints, and then the condition of the third rule is not satisfied.

So in this case the causal rules give rise to an additional successor state which is reached by a non-minimal set of changes (the change in c is unnecessary). This is to be expected in general given the fact that the causal rules are applicable under weaker conditions than derived effect rules. More precisely we can prove that the set of valid successor states according to the \mathcal{ER} theory is a subset of the set according to Thielscher's corresponding theory:

Theorem 8.4 *A successor state S' of a state S and action A with direct effects E according to the nondeterministic derived effect rules obtained from given influence information and state constraints using the above method, is also a successor state of S after A according to the set of causal rules derived using the same influence information and state constraints by Thielscher's method.*

Proof:

It is sufficient to prove that in a particular successor state (which we will construct) S'' of S according to the causal rules, each literal true in S' is also true. Since no causal rule is ever forced to be applied, assume we will not apply any rules leading to the initiation of the negation of any literal in S' . So any literal already true in S or an intermediate state and still true in S' can be assumed to be true in S'' . Likewise, we can assume any initiation atom false in the transition between S and S' to be false in the effects leading to S'' . It remains to be proven that strongly initiated literals in S' can be made true in E'' and S'' . This can be proven by induction on the depth of the best proof tree of each such strong initiation:

1. Assume the initiation, say of literal l , has a proof tree of depth 1. Then there is a direct effect rule **A causes l if F** in the theory with F true in S , and so an equivalent effect description is in Thielscher's formalisation: l is simply a direct effect of A , so it is true in $S_0 = (S \setminus \overline{E}) \cup E$ and all subsequent states.
2. Assume all initiations with a true proof tree of depth k have been derived by causal rules, leading to (S_{n_k}, E_{n_k}) , and l has a proof tree of depth $k+1$. Then there is a rule

initiating $l' \wedge \bigwedge_i l_i$ causes l | true if $\overline{l'}$

in Π_e such that each l_i is either true in S or has a proof tree of depth at most k , and l' also has a proof tree of depth at most k . There is also a corresponding causal rule

$$l' \text{ causes } l \text{ if } \bigwedge_i l_i$$

By the induction hypothesis, l' and each l_i which was not already true in S is true in S_{n_k} , and by the assumptions made above also the other l_i are still true in S_{n_k} . Hence the causal rule is applicable and l can be derived. Moreover by the above assumptions this will not cause any other applicable rule to become inapplicable. Assuming then $l_1 \dots l_j$ are all of the literals with a best proof tree of depth $k+1$, their corresponding causal rules can be applied in any sequence starting from (S_{n_k}, E_{n_k}) , leading to $(S_{n_{k+1}}, E_{n_{k+1}})$. This proves the inductive step.

The theorem follows from this inductive argument. \square

Related to this, we should mention that in some cases the derived effect rules generated by our method only lead to states in which some of the state constraints are still violated, whereas Thielscher's rules yield sets of indirect effects leading to a valid state. In other words, in some cases our rules lead to an action qualification where Thielscher's lead to ramifications. This occurs intuitively speaking when a particular constraint is restored in a non-minimal way, which has the side effect of restoring another violated constraint. An example is the following: assume the constraints

$$\begin{array}{lll} \neg b \vee a & \neg b \vee e & d \vee c \\ \neg a \vee c & \neg a \vee \neg d \vee e & \end{array}$$

and the influence information

$$\{(b, a), (b, e), (a, d), (d, c)\}$$

From these data, we can derive the causal rules

1. b causes a
2. b causes e
3. a causes $\neg d$ if $\neg e$
4. $\neg d$ causes c

and on the other hand the nondeterministic derived effect rules

1. **initiating** b causes a | true if $\neg b$
2. **initiating** b causes e | true if $\neg b$
3. **initiating** $a \wedge \neg e$ causes $\neg d$ | true if $\neg a$
4. **initiating** $\neg d$ causes c | true if d

Assume then a state S in which $\neg a, \neg b, \neg c, d$ and $\neg e$ are true, and an action A which initiates b . Using the causal rules, we can obtain a candidate successor state by applying rules 1 and 2 in any order (none of the intermediate states satisfies the state constraints), which leads to the state $\{a, b, \neg c, d, e\}$. No other rules are applicable in that state, but it still violates the state constraint $\neg a \vee c$. A second candidate successor state can be obtained by consecutively applying rules 1 and 3, followed by 2 and 4 in any order. Again, no intermediate state satisfies the constraints, but the final state $\{a, b, c, \neg d, e\}$ does, and therefore is a successor state.

Using the derived effect rules to compute a successor state, we need to apply the first two rules to satisfy the state constraints $\neg b \vee a$ and $\neg b \vee e$, and the

other rules are not applicable as a result. Hence, we obtain Thielscher’s first candidate successor state $\{a, b, \neg c, d, e\}$, which violates $\neg a \vee c$ so is rejected. No other candidate successor state satisfies both $\neg b \vee a$ and $\neg b \vee e$, so we obtain no valid successor state: the action is impossible under the given circumstances.

In summary, we find that the method we have developed only generates effect rules that restore the constraints they are derived from with a minimal set of changes, whereas Thielscher’s method also allows for non-minimal changes. It is interesting to note that in the setting proposed by Thielscher, i.e. if causal rules are derived from state constraints with the explicit task of restoring the validity of these constraints, minimal change solutions (adhering to the strong correctness criterion) are to be preferred. This remains the case even when this approach sometimes prohibits a particular action rather than generating a set of ramifications which “by accident” restores some constraint’s validity: after all, we have to keep in mind that our goal is not generating ramifications restoring the state constraints at all costs. Rather we want to derive exactly those ramifications that are justified by the domain knowledge. Unjustified ramifications are to be avoided, as inertia is still the most fundamental law governing temporal domains.

On the other hand, if the rules are seen as effect propagation rules only roughly related to state constraints, the non-minimal solutions obtained using Thielscher’s method need not be rejected. In general we should simply not assume that the state constraints and influence information provide sufficient information to determine the precise intended effect rules, so neither should we impose a strong preference for minimal changes with respect to these state constraints. In our approach this issue is dealt with by an indication of possible underspecification (i.e. the presence of a bad grounding in the semantics of the effect rules), in addition to the generation of rules prescribing minimal changes.

One way to deal with the problem of underspecification could be making influence information more fine-grained. For example, one can imagine saying that a can influence b or c , but not both, or only in certain circumstances. However, if influence information needs to get this detailed, there is little or no reason for not immediately writing the intended effect rules instead.

Due to the fact that Thielscher was the first and up to now only researcher to use influence information, we have extensively compared our approach to his, even though our viewpoints on ramifications show some fundamental differences. In the next section, we address one of the most fundamental of these differences: the different views on delays in change propagations.

9 Delayed Causation

Effect propagations, especially in physical systems, usually incorporate very small delays. For all practical purposes these delays can usually be abstracted away and the effects assumed to be simultaneous and instantaneous. This abstraction yields a significant simplification and is adopted in most temporal reasoning approaches, including the one presented here.

Evidently, this abstraction is no longer valid if the presence of delays has macroscopic effects. In this case they must be represented explicitly. The approach of Thielscher we have discussed above is a kind of mixed approach in this respect: on the one hand, delays are abstracted away in the sense that the successor states of each state are obtained by applying a complete batch of effects to this starting state, and nothing can interrupt this batch of effects. On the other hand, the delays are assumed to really exist and to have possible macroscopic effects. In particular contradictory effects, like initiating and terminating a par-

ticular fluent in the same batch of effects, are allowed in Thielscher’s approach, and between these two changes the momentary value of the fluent may have effects that remain visible, like in Thielscher’s light detector example which we will discuss later.

We prefer a different approach for two reasons: first of all, the assumption that ramifications incorporate a small delay is not always valid. This assumption implies for example that all state constraints which are restored by ramifications, are violated for very small periods of time. While for some state constraints this is acceptable, in other cases it leads to counterintuitive results. As an example, assume we have fluents *on* and *off* representing the states of a switch. Assume $on \leftrightarrow \neg off$ is a state constraint, then we expect that rules stating that any effect is caused by the switch being *on* and *off* at the same time, would never have any effect. However if we assume that the state constraints are violated for small periods of time, these counterintuitive effects can occur any time the switch is toggled.

A second problem is in our view that if there are actual delays that are so pronounced as to have macroscopic effects, the assumption that they can nevertheless be abstracted away is not necessarily valid. For example, it is not clear why no other actions would be allowed to occur during these delays.

For these reasons, we adopt the following approach: usually we assume ramifications to be instantaneous, either because they really are, or because the delays involved are entirely irrelevant. Of course then we assume real instantaneity, and disallow contradictory effects or effects generated by hypothetical intermediate states. This has been our approach up to now.

On the other hand, if delays are relevant and have macroscopic effects, we model them explicitly in a theory of delayed causation. There are plenty of options for such a theory, which we study in this section.

The basic idea is that an action or event *a* at a certain time *t* may cause another action or event *b* at a later time $t + d$. Possibly this depends on some conditions *F* at the time of *a*. This could be represented as

***a* dcauses *b* after *d* if *F*.**

Another option is to let a particular combination of fluents be the trigger of a later event, as in

initiating *F'* dcauses *b* after *d* if *F*.

with *F'* a fluent formula. Similarly, the delayed effect may not be an event but a fluent change:

***a* ecauses *l* after *d* if *F*.**

or

initiating *F'* ecauses *l* after *d* if *F*.

with *l* a fluent literal (which in turn might be extended to a disjunction of conjunctions to represent nondeterminism).

Another issue is that a delayed effect may be cancelled if some conditions are changed before it actually occurs. To represent this, we would need to distinguish conditions that need to hold at the time of initial “causation” of the delayed effect and conditions that need to persist until it takes place. For example:

***a* if *F* dcauses *b* if *F'* persists after *d*.**

In all the above cases, it is some action or the initiation of some fluent formula which causes the delayed effect. Related to this but slightly different is the

issue of natural events, discussed for example in [26], where events may be (immediately) triggered as soon as some fluent formula holds (rather than as soon as it is initiated). Clearly in the language \mathcal{ER} presented here, this would not make sense as formulae hold only immediately after their initiation, all changes are discrete and in the real numbers there is no time point immediately after another one. The importance of natural events is of course strongly linked to continuous change. As continuous change is outside the scope of this paper, we do not discuss natural events further here, though a treatment of them would be very similar to that of delayed effects.

Whatever the exact form of delayed effect rules, it seems natural to us to read them as a definition of the predicate **Happens**, or as part of the inductive definition of **Causes**. In the former case, it might also be wise to distinguish between “primitive” events (actions performed by some agent) and “delayed effect” events, if only for the reason that the former may be arbitrary while the latter are uniquely determined by what preceded them.

It is not easy to keep the syntax of delayed effect theories simple while having the needed expressive power. The rules are quite complex, requiring many parameters representing different conditions. One thing we can do in the interest of simplicity is restricting the syntax to only one type of rule. To achieve this we need to decide whether the rule body will be an event or an initiation and whether its head will be an event or an initiation. Since in \mathcal{ER} up to now initiations follow from actions at the same time point, the most flexible approach is to have initiations be the cause of later events: the other three cases can then be dealt with by combining immediate *event* \rightarrow *initiation* propagations with delayed *initiation* \rightarrow *event* propagations. Hence we propose rules of the form

initiating F if F' ecauses e if F'' persists after d

with the intended reading that if at any time point t , F is strongly initiated while F' holds, and if F'' remains true throughout $]t, t + d]$, then event e occurs at time $t + d$.

As far as the formal semantics is concerned, the idea is to read these rules as part of a (definite) inductive definition on the predicate **Happens**. These rules describe the “caused” events, and an additional set of rules which we will introduce further on defines the occurrence in terms of **Happens** of primitive events.

Note that in the definition of **Happens** no cycles can occur: a delayed event occurrence is uniquely determined by what happened before and what holds at the time of occurrence t , which can be evaluated without referring to any occurrences at t or later time points.

We now define the proposed extension, which we will call \mathcal{ERD} , more precisely. First of all, we introduce a new sort \mathcal{PA} of primitive actions, which is a subset of \mathcal{A} , and a set $\mathbf{PA} \subseteq \mathbf{A}$ of constants of sort \mathcal{PA} . We replace the predicate **Happens** : $\mathcal{A} \times \mathcal{T} \rightarrow \mathcal{P}$ by **P_Happens** : $\mathcal{PA} \times \mathcal{T} \rightarrow \mathcal{P}$.

Definition 9.1 (*\mathcal{ERD} -signature*)

An \mathcal{ERD} -signature Σ is a tuple $\langle \mathbf{Sorts}, \mathbf{Functors}, \mathbf{Vars} \rangle$ with

- **Sorts** = $\{T, \mathcal{A}, \mathcal{PA}, \mathcal{F}, \mathcal{P}\}$, representing the sorts time, action, primitive action, fluent and atom.
- **Functors** consists of
 - a set **T** of constants of sort T , denoted t, t_1, \dots , which includes all real numbers;
 - a set **A** of constants of sort \mathcal{A} , denoted a, a_1, \dots ; **PA** is a subset of **A** of constants of sort \mathcal{PA}

- a set \mathbf{F} of constants of sort \mathcal{F} , denoted f, f_1, \dots ;
- four typed predicate symbols $\leq: T \times T \rightarrow \mathcal{P}$; $\mathbf{P_Happens}: \mathcal{PA} \times T \rightarrow \mathcal{P}$; $\mathbf{Initially}: \mathcal{F} \rightarrow \mathcal{P}$; $\mathbf{Holds}: \mathcal{F} \times T \rightarrow \mathcal{P}$.
- $\mathbf{Vars} = \mathbf{Vars}_A \cup \mathbf{Vars}_T$, disjoint infinite sets of variables of sort A resp. T , denoted as A, A_1, \dots resp. T, T_1, \dots

Definition 9.2 (\mathcal{ERD} -formulae)

Given Σ , the formulae of \mathcal{ERD} are:

- direct effect rules of the form

a causes D if F

- derived effect rules of the form

initiating F causes D if F'

- delayed effect rules of the form

initiating F if F' dcauses a if F'' persists after d

- any sentence (i.e. formula without free variables) constructed in the usual way of \mathbf{Holds} , $\mathbf{P_Happens}$, \leq , $\mathbf{Initially}$ atoms and the connectives and quantifiers $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \forall, \exists$.

where d is a positive real number ($0 < d$), a an action, D a disjunction of conjunctions of fluent literals, and F, F', F'' general fluent formulae.

Definition 9.3 (\mathcal{ERD} -theory)

An \mathcal{ERD} -theory is a tuple $\langle \Sigma, \Pi_e, \Pi_d, \Pi_p \rangle$ such that Σ is an \mathcal{ERD} -signature, Π_e is a set of direct or derived effect rules based on Σ , Π_d is a set of delayed effect rules based on Σ , Π_p is a set of sentences based on Σ .

The semantics of \mathcal{ERD} can be defined as follows.

Definition 9.4 (\mathcal{ERD} -interpretation)

Given an \mathcal{ERD} -signature Σ , a temporal (\mathcal{ERD} -)interpretation of Σ is a structure I

$\langle \mathbf{P}, \mathit{Fun}, \mathcal{H} \rangle$ with:

$$\begin{aligned} \mathbf{P} = & \{t_1 \leq t_2 \mid t_1, t_2 \in \mathbf{T}\} \cup \\ & \{\mathbf{Initially}(l) \mid l \in \widehat{\mathbf{F}}\} \cup \\ & \{\mathbf{Happens}(a, t) \mid a \in \mathbf{A}, t \in \mathbf{T}\} \cup \\ & \{\mathbf{P_Happens}(pa, t) \mid pa \in \mathbf{PA}, t \in \mathbf{T}\} \cup \\ & \{\mathbf{Holds}(l, t) \mid l \in \widehat{\mathbf{F}}, t \in \mathbf{T}\} \cup \\ & \{\mathbf{Init}(t, l) \mid t \in \mathbf{T}, l \in \widehat{\mathbf{F}}\} \cup \\ & \{\mathbf{Causes}(t, l) \mid t \in \mathbf{T}, l \in \widehat{\mathbf{F}}\} \end{aligned}$$

$\mathit{Fun}: \mathbf{T} \rightarrow \mathbf{R}$, a mapping of time constants to reals
such that each real number is mapped to itself

$\mathcal{H}: \mathbf{P} \rightarrow \{\mathbf{t}, \mathbf{f}\}$, a truth assignment function.

\mathcal{H} defines relations interpreting $\mathbf{Happens}$, $\mathbf{P_Happens}$, \mathbf{Holds} , $\mathbf{Initially}$, \leq , \mathbf{Init} and \mathbf{Causes} ; we denote them $\mathcal{H}a, \mathcal{PH}a, \mathcal{H}o, \mathit{Initially}, \preceq, \mathit{Init}$ and Causes respectively. An \mathcal{ERD} -interpretation needs to satisfy the same general conditions as an \mathcal{ER} -interpretation (see section 3.3). An \mathcal{ERD} -interpretation I is a model of an \mathcal{ERD} -theory $\langle \Sigma, \Pi_e, \Pi_d, \Pi_p \rangle$ iff it is a model of Π_e , Π_d and Π_p . Whether I is a model of Π_p and Π_e or not is defined like in \mathcal{ER} , extended with nondeterministic rules as in the previous section. The only thing left to be defined is when I is a model of Π_d . This is done as follows:

Definition 9.5 (grounding of Π_d)

The grounding of a delayed effect rule

initiating F if F' dcauses e if F'' persists after d

is the set:

$$\{\mathbf{Happens}(e, t) \leftarrow \mathcal{H}o(F', \mathcal{F}un(t) - d), \mathcal{I}nit(\mathcal{F}un(t) - d, F), \\ \mathcal{P}ersists(F'', \mathcal{F}un(t) - d, \mathcal{F}un(t)) \mid t \in \mathbf{T}\}.$$

where $\mathcal{H}o(F, t)$ is defined as before, $\mathcal{I}nit(t, F)$ is the truth value of

$$\neg \mathcal{H}o(F, t) \wedge \bigvee_{L_i \cup L_p = \text{supp. set of } F} (\mathcal{I}nit(t, L_i) \wedge \overline{\mathcal{I}nit(t, L_p)} \wedge \mathcal{H}o(L_p, t))$$

and $\mathcal{P}ersists(F, t', t)$ is the truth value of

$$\forall T' : (t' < T') \wedge (T' \preceq t) \rightarrow \mathcal{H}o(F, T')$$

The grounding $\mathcal{D}_{\text{delay}}$ of Π_d is the union of the groundings of all rules of Π_d .

Finally, we combine delayed effect events with primitive actions in one definition, as follows:

Definition 9.6 (effect definition)

The effect definition $\mathcal{D}_{\text{event}}$ of Π_d is

$$\mathcal{D}_{\text{delay}} \cup \{\mathbf{Happens}(pa, t) \leftarrow \mathbf{P_Happens}(pa, t) \mid t \in \mathbf{T}, pa \in \mathbf{PA}\}.$$

$\mathcal{D}_{\text{event}}$ is a (definite) inductive definition on the atom domain $\mathbf{A}' = \{\mathbf{Happens}(a, t) \mid t \in \mathbf{T}, a \in \mathbf{A}\}$, for which $I_{\mathcal{D}_{\text{event}}}$ is defined as $\mathcal{P}\mathcal{I}_{\mathcal{D}_{\text{event}}} \uparrow$.

We can then complete the definition of model of an \mathcal{ERD} -theory:

Definition 9.7 (\mathcal{ERD} -model)

Given an \mathcal{ER} -theory $\Pi_{\mathcal{ER}} = \langle \Sigma, \Pi_e, \Pi_d, \Pi_p \rangle$, a temporal interpretation I is a model of $\Pi_{\mathcal{ER}}$, denoted $I \models \Pi_{\mathcal{ER}}$, iff $I \models \Pi_p$, $I \models \Pi_d$ and $I \models \Pi_e$, where

$$I \models \Pi_p \text{ iff } \forall F \in \Pi_p : \mathcal{H}(F) = \mathbf{t}.$$

$$I \models \Pi_d \text{ iff } \forall t \in \mathbf{T}, a \in \mathbf{A} : \mathcal{H}a(a, t) \leftrightarrow I_{\mathcal{D}_{\text{event}}}(\mathbf{Happens}(a, t)).$$

$$I \models \Pi_e \text{ iff } \forall t \in \mathbf{T}, l \in \widehat{\mathbf{F}} :$$

$$\mathcal{I}nit(t, l) \leftrightarrow I_{\mathcal{D}_{\text{init}}}(\mathbf{Init}(t, l)) \text{ and } \mathcal{C}auses(t, l) \leftrightarrow I_{\mathcal{D}_{\text{init}}}(\mathbf{Causes}(t, l)).$$

As an example, we present the relay problem of [31], which is in that paper represented without explicit delays as an example of Thielscher's view on and approach to indirect effects. As we indicated before, in our approach this example should be modelled with explicit delays. The example is an electric circuit with a light, several switches and a relay, as shown in Figure 9.1.

One switch s_1 is serially connected to a pair of parallel wires. On one of these wires we find a switch s_2 and a light, on the other we find a switch s_3 and a relay which operates s_2 . The state constraints are $(s_1 \wedge s_2) \leftrightarrow \text{light}$ and $(s_1 \wedge s_3) \leftrightarrow \text{relay}$, with influence information $\{(s_1, \text{light}), (s_2, \text{light}), (s_1, \text{relay}), (s_3, \text{relay})\}$. We obtain the following derived effect rules:

$$\begin{aligned} &\mathbf{initiating } s_1 \wedge s_2 \text{ causes } \text{light} \mid \text{true if true} \\ &\mathbf{initiating } \neg s_1 \vee \neg s_2 \text{ causes } \neg \text{light} \text{ if true} \\ &\mathbf{initiating } s_1 \wedge s_3 \text{ causes } \text{relay} \mid \text{true if true} \\ &\mathbf{initiating } \neg s_1 \vee \neg s_3 \text{ causes } \neg \text{relay} \mid \text{true if true} \end{aligned}$$

In addition, we need to model the effect of the relay. The relay operates s_2 , and we assume there is some delay involved between the activation of the relay and

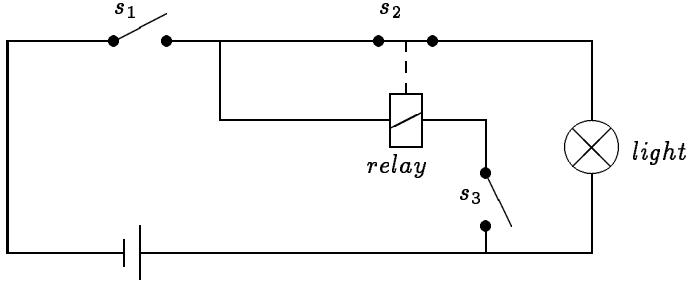


Figure 1: Schema of the relay example

its effect. So we write (omitting the “if true” conditions and assuming the state of the relay needs to persist until the effect occurs):

initiating *relay* dcauses $\neg s_2$ if *relay* persists after d
initiating $\neg relay$ dcauses s_2 if $\neg relay$ persists after d

Then, in case s_2 and s_3 are closed (i.e. true) but s_1 is open (false), the closing of s_1 will have the effect of turning on the light and activating the relay. Then, after a time period of length d and unless other actions occur which influence the relevant fluents, the relay will cause s_2 to open and the light to dim again.

Thielscher has extended this example to incorporate a detector, assumed to turn (and then remain) on as soon as light shines on it. We can model the behaviour of this detector by the rule

initiating *light* dcauses *detect* if *light* persists after d'

assuming that there is a threshold duration d' for light to be detected. Thielscher uses this example to illustrate nondeterminism arising from the causal rules. Indeed, whether or not the detector will detect light in the above situation, depends on which of d or d' is greater, which is left implicit in Thielscher’s approach. If we assume d and d' to be unknown positive time constants, we also obtain nondeterminism. On the other hand, if d and d' are known (which we assume), the outcome of closing s_1 is uniquely determined.

9.1 Mapping delayed effect rules to OLP

Finally, we need to map delayed effect rules to OLP. This can be done as follows. First of all, as the introduction of delayed effect rules required a modification to the syntax of \mathcal{ER} (introducing a new predicate **P_Happens** which replaces **Happens** in Π_p), we need to modify the mapping accordingly. This is done by treating **P_Happens** exactly as we treated **Happens** before, i.e. by mapping each atom **P_Happens**(α, τ) in Π_p formulae to *phappens*(α, τ) in the corresponding FOL axiom and declaring *phappens* an open predicate. *happens* now no longer occurs in FOL axioms (since **Happens** does not occur in Π_p). It is a defined predicate of which the definition follows below. To distinguish between primitive and non-primitive actions, we simply introduce a predicate *primitive*/1 defined by enumeration. Moreover we add a FOL axiom *phappens*(A, T) \rightarrow *primitive*(A) to the theory.

Then, recall that the effect definition D_{event} of a set of delayed effect rules Π_d is $D_{delay} \cup \{\mathbf{Happens}(pa, t) \leftarrow \mathbf{P_Happens}(pa, t) \mid t \in \mathbf{T}, pa \in \mathbf{PA}\}$. Hence, the mapping of Π_d to OLP is a definition of *happens* which consists of some

clauses for each delayed effect rule, plus one general clause. This general clause is simply

$$happens(A, T) \leftarrow phappens(A, T).$$

The other clauses are obtained as follows: a rule

initiating F if F' dcauses a if F'' persists after d

is mapped to

$$\begin{aligned} \{happens(a, T) \leftarrow & T = T' + d, \bigwedge_{l \in L_1} causes(T', l), \\ & \bigwedge_{l \in L_2} (holds(l, T'), \neg causes(T', \bar{l})), \\ & \neg holds(F, T'), holds(F', T'), persists(T', F'', T). \\ | & L_1 \cup L_2 \text{ is a supporting set of } F\} \end{aligned}$$

where *persists* is defined as

$$\begin{aligned} persists(T', F, T) & \leftarrow holds(F, T'), \neg clipped(T', F, T). \\ persists(T', F, T) & \leftarrow causes(T', F), \neg clipped(T', F, T). \end{aligned}$$

and *clipped* as defined before.

The correctness proof of this mapping can largely be based on the proof of the case without delays in section 5. As before, it is clear that Π_p and the FOL axioms in OLP are equivalent. Following the same reasoning as in section 5, we find that it suffices to prove that, given *initially* for fluent atoms, *phappens* and \leq , and their exact counterparts in \mathcal{ER} , we find the same truth value for all other atoms in both formalisms. For complex *initially* and *Initially* atoms this is trivial since we use the same inductive definition in terms of simple atoms in both formalisms. This leaves us with simple and complex *holds* atoms, *causes* atoms and *happens* atoms.

We can again use induction on events. First of all, note that if the set of primitive events satisfies the well-founded topology requirement, then so does the set of all time points that possibly give rise to an event, provided that all delays are finite positive constants and that there is a finite number of delayed effect rules. This set of possible events contains all t' of the form $t + \sum_i k_i d_i$, where t is a primitive event, the d_i are the delay constants occurring in delayed effect rules and the k_i are natural numbers. No event can occur at any other time.

That this set has a well-founded topology follows from the following observations. First, there is a first element, which is the first primitive event as there are no negative delays. Second, the number of possible events in a finite time interval is finite, which we prove by induction on time intervals of length d_{min} , with d_{min} the minimal delay constant in the theory, as follows. There is only one event in $[e_{start}, e_{start} + d_{min}[$, and if there is a finite number of events in $[e_{start}, t[$ then there is only a finite number of events in $[t, t + d_{min}[$. The latter statement follows from the fact that each non-primitive event in $[t, t + d_{min}[$ must be caused by an event in $[e_{start}, t[$, which is a finite set in which each event can only cause a finite number of new events. In addition, $[t, t + d_{min}[$ may contain primitive events, but since these form a well-founded topology their number is also finite. Hence, the set of all time points that are possible events is well-founded.

Using this well-founded topology we prove the equivalence of *holds*(f, t) with $\mathcal{H}o(f, t)$, *holds*(F, t) with $\mathcal{H}o(F, t)$, *happens*(a, t) with $\mathcal{H}a(a, t)$, and *causes*(t, l) with $\mathcal{C}auses(t, l)$ by induction on possible events. The equivalence of *holds*(F, t) with $\mathcal{H}o(F, t)$ for time points before the first event is proven as in section 5. We then prove consecutively for each possible event e that *happens*(a, e) \leftrightarrow

$\mathcal{H}a(a, e)$, that $\text{causes}(e, l) \leftrightarrow \text{Causes}(e, l)$ and that for all time points t between e and the next possible event $\text{holds}(f, t) \leftrightarrow \mathcal{H}o(f, t)$ and $\text{holds}(F, t) \leftrightarrow \mathcal{H}o(F, t)$.

If we can prove the first step for a particular event, the proof of the other three steps is exactly the same as in section 5, so we only need to prove that $\text{happens}(a, e) \leftrightarrow \mathcal{H}a(a, e)$ if all predicates in OLP and \mathcal{ER} coincide for all $t < e$.

This result is obtained as follows. It follows from the inertia axiom and the initiation consistency condition that

$$[\mathcal{H}o(F, T) \vee \text{Causes}(T, F)] \wedge \neg \exists T' : [\text{Causes}(T'', \neg F) \wedge T \leq T' \wedge T' < e]$$

is equivalent to

$$\forall T' : [(T < T' \wedge T' \leq e) \rightarrow \mathcal{H}o(F, T')]$$

Since all predicates in OLP and \mathcal{ER} coincide for all time points before e , the former formula is equivalent to $\text{persists}(T, F, e)$, while the truth value of the latter is $\text{Persists}(T, F, e)$. Hence, the truth value of $\text{persists}(T, F, e)$ is $\text{Persists}(T, F, e)$ for all F and all $T < e$. In addition, the entire definitions of happens and **Happens** only refer to time points before e , so it follows that they are equivalent. The correctness of the mapping is thereby proven.

10 Related work

We have already compared our work in detail with the approach in [31], which is most similar to ours. For this reason we can refer to [31] for a comparison with approaches not based on causal laws (e.g. categorisation based approaches like [18], [19], [2]): with respect to those approaches \mathcal{ER} and Thielscher's proposal have the same advantages. The most important differences between Thielscher's approach and ours are that Thielscher's causal rules are strongly coupled with (derived from and used in combination with) state constraints, and that Thielscher abstracts away all delays at a macroscopic level but retains them at a microscopic level, whereas we either abstract delays away entirely or represent them explicitly. Due to the former difference the full effect of syntactically uncoupling causal laws from state constraints (which is only achieved in Thielscher's approach and ours) is partially lost. For example, no state constraint independent effect propagations can be represented. Nevertheless, the fact that influence information is represented independent of the state constraints makes Thielscher's approach very appropriate for analysing other proposals using causal laws, which we will do.

The approach to ramifications in the \mathcal{E} language ([15]) can be interpreted as a more coarse-grained variant of Thielscher's: it uses formulae A *whenever* C , with A a fluent and C a set of fluent literals to be read as a conjunction. Such a formula corresponds to a combination of the state constraint $A \leftarrow \bigwedge_{c \in C} c$ with influence information stating that each fluent in C influences the fluent A . As a result of this tight coupling of influence information and state constraint, it is not possible to represent some of the more fine-grained influences that can be represented in Thielscher's approach. However, it should be noted that dealing with ramifications was not a major goal of \mathcal{E} . In other respects, the \mathcal{E} language is closer to \mathcal{ER} than Thielscher's approach, in particular in its use of an event-based time structure modelled after the Event Calculus. Apart from less stress on ramifications, an interesting point of difference with \mathcal{ER} is also that \mathcal{E} is mapped to standard logic programming rather than OLP, using an autoepistemic view on LP rather than the definitional view we prefer. One consequence of this is that the mapping of \mathcal{ER} (to OLP) is sound and complete, whereas the mapping of \mathcal{E} (to LP) is sound but not complete.

Returning to the issue of ramifications, we should also consider the approach in [22], where the need for causal laws is clearly motivated and where causal laws are presented as so-called *S-conditionals*, i.e. formulae $\phi \Rightarrow \psi$ with ϕ and ψ propositional formulae, in an extension of *S5* modal logic. The reading of such a law is that ϕ determines the truth of ψ : it entails the state constraint $\neg\phi \vee \psi$, plus in case ψ is a literal the influence information that literals in ϕ influence ψ . If ψ is not a literal, the picture gets more complicated: then all literals in ϕ can influence all literals in ψ and all literals in ψ can influence each other. In this respect the proposal is more general than the \mathcal{E} approach. In any case, it is clear that like in \mathcal{E} the causal laws entail the corresponding state constraints, which is the most essential difference with our approach.

Another similar approach is the proposal in [20] based on the situation calculus. Lin introduces a new predicate *caused*(p, v, s) meaning that proposition p is *caused* to have truth value v in state s . This predicate is circumscribed to minimise change. Ramifications are represented by formulae using the *caused* predicate, e.g. for the suitcase example $up(l_1, s) \wedge up(l_2, s) \rightarrow caused(open, true, s)$ represents that if both latches are open, then the suitcase is caused to be open. The above formula entails the state constraint $up(l_1, s) \wedge up(l_2, s) \rightarrow open(s)$, and incorporates moreover the influence information that l_1 and l_2 may influence *open*. Note that the condition of the rule is a complex formula, making it similar to a complex derived effect rule in \mathcal{ER} . However, the causal rules differ from the ones in \mathcal{ER} in that they also entail the corresponding state constraint and in the fact that the minimisation policy does not allow for cyclic dependencies.

The approach in [14] is based on the *Features and Fluents* framework ([27]), and in that sense differs considerably from ours as far as basic concepts are concerned. However, there are some interesting correspondences at a higher level. As an example, the circumscription policy consisting of a combination of minimising and filtering corresponds to our use of inductive definitions for effect rules (minimising changes) and first order logic for observations, action preconditions and state constraints (used to filter interpretations). An important difference with our approach is (apart from the formal details of syntax and semantics) the fact that actions are considered to have duration. Another difference is that time is considered to be isomorphic to the natural numbers. In a sense this corresponds to our condition of well-founded event topology, i.e. *events* in \mathcal{ER} could be mapped to the natural numbers in an order-preserving way, but we consider it more natural to see *time* itself as a full real line. For dealing with ramifications, [14] contains expressions (and formulae incorporating these) of the form $[t]\delta \gg [s]\gamma$ where δ and γ are fluent formulae and t and s temporal expressions such that $t \leq s$. This allows for dealing with both immediate (if $t = s$) and delayed ramifications. The formula $[t]\delta \gg [s]\gamma$ is defined to mean $[[t]\delta \rightarrow [s]\gamma] \wedge ((([t-1]\neg\delta \wedge [t]\delta) \rightarrow [s]X(\gamma))]$, which basically says that $\mathbf{Holds}(\delta, t) \rightarrow \mathbf{Holds}(\gamma, s)$ and that if δ is strongly initiated at t then γ is allowed to change value at s . This is similar to a state constraint plus the influence information that δ may influence γ , with of course the important generalisation that t and s need not be equal, so that one does not only obtain state constraints but also constraints relating fluents at different time points. Hence, the rules represent not only immediate but also delayed ramifications. This is a considerable advantage of [14]'s approach with respect to nearly all other recent proposals.³³ As in the previously discussed approaches, however, we find that the formulae $[t]\delta \gg [s]\gamma$ always entail the corresponding general constraint $\mathbf{Holds}(\delta, t) \rightarrow \mathbf{Holds}(\gamma, s)$, and hence in the case of immediate ramifications

³³But note that both the more general constraints between fluents at different time points and explicit delayed ramification rules can be represented in \mathcal{ER} .

($t = s$) they also entail the corresponding state constraint. Change propagation unrelated to a state or general constraint is also in this approach excluded, unlike in \mathcal{ER} . Finally, a disadvantage of the approach in [14] with respect to \mathcal{ER} is that cyclic dependencies in causal laws are not dealt with correctly, as indicated by the authors.

In [28] it is argued that approaches to the ramification problem should be able to deal with so-called *downstream* indirect effects. Sandewall gives the example of a lamp connected to two parallel switches, such that closing either switch turns on the lamp. He argues that one should be able to specify the main effect of an action (for example of turning on the lamp) without specifying the operational details of how this is accomplished (by closing either of the switches). An approach to the ramification problem should then be able to use this description and derive the direct effects of the action from the indirect effect that the lamp is turned on. It is argued that this is a problem for causality-based approaches like ours and most of the above ones.

As stated, the issue is indeed problematic. Clearly we can not write the action law as a direct effect rule, since it would then imply that the lamp is turned on while the switches are untouched. And in \mathcal{ER} , direct effect rules are the only constructs representing action laws. However, we argue that the indirect effect should not be explicitly specified by some action law. We agree with Sandewall’s argument that it is often interesting to worry only about the main effect of an action and not about how it is achieved, but this issue deserves closer attention. Given Sandewall’s problem specification, we see two possibilities: on the one hand, the switches may be considered nothing but operational details. In that case they can be abstracted away altogether, and the turning on of the lamp can be modelled as an action with the plain direct effect that the lamp is on. On the other hand, it may occur that the position of the switches is relevant in the domain, but that in a particular application one is only interested in the state of the lamp. This can happen when one gives an agent the task to turn on the lamp, or when turning on the lamp is a necessary step in a plan. But in that case, one can simply model the domain using the usual direct and derived effect rules, and impose in the application at hand that the lamp should be on after a particular action or after a particular plan (imposing this can be done with a simple FOL axiom). Then the agent can find the primitive actions yielding the intended effect on the lamp (closing either of the switches) simply by abductive planning.³⁴ This approach adequately tackles the given problem, and we find there is no need to modify the domain representation. In general, we argue that there is no need for the representation to deal with deriving causes from their effects, nor is it even desirable that it should do so: this is a typical (abductive or — if the action law needs to be derived explicitly — inductive) reasoning task and not a representation issue.

11 Conclusion

We have presented an narrative-based language able to deal — in a setting of instantaneous actions and discrete change — with all immediate ramifications and known action qualifications, and with delayed ramifications, possibly in the presence of nondeterministic and simultaneous actions and of incomplete knowledge on action occurrences, action ordering or the initial state of the world. The language allows for change propagations not related to state (or

³⁴An alternative but equivalent view is that the turning on of the lamp is a macro-action (as defined in [8]) which can consist of either of the primitive actions. Deriving a primitive action which satisfies the conditions of the macro-action is also a typical abductive task.

more general) constraints between fluents, and for recursion and cycles in the rules describing change propagation. We have discussed and motivated the types of constructs used in the language to reach all of those goals, and presented a semantics based on first order logic and the principle of inductive definitions. This semantics was chosen due to its closeness to the intuitions underlying in particular effect propagations.

We have mapped the language to OLP Event Calculus and proven the correctness of this mapping. The language provides constructs offering the expressive power of the OLP Event Calculus for representing temporal domains, while restricting the latter formalism in such a way that a correct representation methodology is imposed, avoiding unintended and counterintuitive models.

The presented language has been compared with recent proposals for dealing with ramifications. It is intended to deal with the various problems tackled by previous proposals in one coherent language, and to deal in addition with some unaddressed problems, like cycles in derived effect rules and change propagation unrelated to constraints. The extensions for nondeterministic actions and delayed ramifications deal with some other less basic issues in novel ways. Finally we have illustrated how influence information can be used to help derive some of the constructs in our language (a subset of the derived effect rules) directly from state constraints, adapting and improving the method introduced by Thielscher.

References

- [1] P. Aczel. An Introduction to Inductive Definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 739–782. North-Holland Publishing Company, 1977.
- [2] 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.
- [3] W. Buchholz, S. Feferman, W. Pohlers, and W. Sieg. *Iterated Inductive Definitions and Subsystems of Analysis: Recent Proof-Theoretical Studies*. Springer-Verlag, Lecture Notes in Mathematics 897, 1981.
- [4] M. Denecker. *Knowledge Representation and Reasoning in Incomplete Logic Programming*. PhD thesis, Department of Computer Science, K.U.Leuven, 1993.
- [5] M. Denecker. Inductive Definitions, Logic Programming, Knowledge Representation. Technical report, K.U. Leuven, 1996.
- [6] M. Denecker, L. Missiaen, and M. Bruynooghe. Temporal reasoning with abductive event calculus. In *Proceedings of ECAI 92, Vienna*, 1992.
- [7] M. Denecker, K. Van Belleghem, G. Duchatelet, F. Piessens, and D. De Schreye. A realistic experiment in knowledge representation in open event calculus : Protocol specification. In M. Maher, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming, 1996*, pages 170–184, 1996.
- [8] C. Evans. The Macro-Event Calculus: Representing Temporal Granularity. In *Proceedings of PRICAI, Tokyo*, 1990.
- [9] A. Galton. An investigation of non-intermingling principles in temporal logic. *Journal of Logic and Computation*, 6:271–294, 1996.

- [10] M. Gelfond and V. Lifschitz. Describing Action and Change by Logic Programs. In *Proc. of the 9th Int. Joint Conf. and Symp. on Logic Programming*, 1992.
- [11] M. Gelfond et al. What Are the Limitations of the Situation Calculus. In *Automated reasoning, Essays in Honor of Woody Bledsoe*. Kluwer, 1991.
- [12] M. Ginsberg and D. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35, 1988.
- [13] M. Ginsberg and D. Smith. Reasoning about action II: The qualification problem. *Artificial Intelligence*, 35, 1988.
- [14] J. Gustafsson and P. Doherty. Embracing occlusion in specifying the indirect effects of actions. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning*, pages 87–98. Morgan Kaufman, 1996.
- [15] A. Kakas and R. Miller. A simple declarative language for describing narratives with actions. *to appear in : Journal of Logic Programming, Special Issue on Reasoning about Actions*, 1996.
- [16] G. N. Kartha and V. Lifschitz. Actions with indirect effects (preliminary report). In *Proc. of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, pages 341–350, 1994.
- [17] V. Lifschitz. Computing Circumscription. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence, 1985*, pages 121–127, 1985.
- [18] V. Lifschitz. Formal Theories of Action. In F. Brown, editor, *Proceedings of the 1987 Workshop on the Frame Problem in AI*, 1987.
- [19] V. Lifschitz. Frames in the Space of Situations. *Artificial Intelligence*, 46:365–376, 1990.
- [20] F. Lin. Embracing causality in specifying the indirect effects of actions. In C. Mellish, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1985–1991, 1995.
- [21] F. Lin and R. Reiter. State constraints revisited. *J. of Logic and computation, special issue on actions and processes*, 4:655–678, 1994.
- [22] N. McCain and H. Turner. A causal theory of ramifications and qualifications. In C. Mellish, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1978–1984, 1995.
- [23] Y. N. Moschovakis. *Elementary Induction on Abstract Structures*. North-Holland Publishing Company, Amsterdam- New York, 1974.
- [24] J. A. Pinto. Temporal reasoning in the situation calculus. Technical Report KRR-TR-94-1, Computer Science Dept., University of Toronto, 1994.
- [25] T. Przymuzinski. Perfect Model Semantics. In *Proceedings of ICLP/SLP 1988*, pages 1081–1096, 1988.
- [26] R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning*, pages 2–13. Morgan Kaufman, 1996.

- [27] E. Sandewall. *Features and Fluents. A Systematic Approach to the Representation of Knowledge about Dynamical Systems. Volume I.* Oxford University Press, 1994.
- [28] E. Sandewall. Assessments of ramification methods that use static domain constraints. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning*, pages 99–110. Morgan Kaufman, 1996.
- [29] Y. Shoham. Nonmonotonic reasoning and causation. *Cognitive Science*, 214:213–252, 1990.
- [30] M. Thielscher. Causality and the qualification problem. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning*, pages 51–62. Morgan Kaufman, 1996.
- [31] M. Thielscher. Ramification and causality. *to appear in : Artificial Intelligence*, 1997.
- [32] K. Van Belleghem, M. Denecker, and D. De Schreye. Representing continuous change in the abductive event calculus. In P. V. Hentenrijck, editor, *Proc. of the International Conference on Logic Programming, 1994*, pages 225–240, 1994.
- [33] K. Van Belleghem, M. Denecker, and D. De Schreye. The Abductive Event Calculus as a General Framework for Temporal Databases. In *Proc. of the International Conference on Temporal Logic*, pages 301–316, 1994.
- [34] K. Van Belleghem, M. Denecker, and D. De Schreye. On the relation between situation calculus and event calculus. *Journal of Logic Programming, Special Issue on Reasoning about Action and Change*, 31(1-3):3–37, 1997.
- [35] A. Van Gelder, K. Ross, and J. Schlipf. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM*, 38(3):620–650, 1991.