Linköping Electronic Articles in
Computer and Information Science
Vol. 3(1998): nr 7

An Inductive Definition Approach to Ramifications

Marc Denecker
Daniele Theseider Dupré
Kristof Van Belleghem

Linköping University Electronic Press
Linköping, Sweden

http://www.ep.liu.se/ea/cis/1998/007/

Published on July 29, 1998 by
Linköping University Electronic Press
581 83 Linköping, Sweden

Linköping Electronic Articles in
Computer and Information Science
ISSN 1401-9841
Series editor: Erik Sandewall

Recommended citation:
<Authors>. <Title>. Linköping Electronic Articles in
Computer and Information Science, Vol. 3(1998): nr 7.
http://www.ep.liu.se/ea/cis/1998/007/. *July 29, 1998.*

This URL will also contain a link to the authors' home pages.

Marc Denecker[1], **Daniele Theseider Dupré**[2] **and Kristof Van Belleghem**[1]

[1] Department of Computer Science, K.U.Leuven,
Celestijnenlaan 200A, B-3001 Heverlee, Belgium.
[2] Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 Torino, Italy

E-mail: marcd@cs.kuleuven.ac.be, dtd@di.unito.it,
kristof@cs.kuleuven.ac.be

Abstract

In the current state of the art on the ramification problem, the purpose of causal laws is to restore the integrity of state constraints. In contrast with this view, we argue that causal laws should be seen as representations of how physical (or logical) forces and effects propagate through a dynamic system.

We argue that in order to obtain a natural and modular representation of the effect propagation process, a causal rule language is needed which allows for *recursion* to model effects on mutually dependent fluents, for *negation* to model effects which propagate in the *absence* of other effects, and with *complex fluent formulae* to model causality in a compact and natural way.

A fundamental property of the process of effect propagation in a dynamic system is that it is *constructive*: effects and change propagations do not spontaneously appear without an external cause. To adequately model the constructive nature of the physical change propagation process, we base the semantics of the formalism on the principle of inductive definition, the main mathematical constructive principle. We use a generalised inductive definition principle, generalising Clark completion and circumscription, to define a unique intended semantics for causal theories. Our approach can deal in particular with cyclic dependencies between effects, in such a way that to some definitions, in spite of syntactic cycles, we can assign a unique intended semantics in a constructive way, while other "actually cyclic" definitions are explicitly detected by the semantics as bad defi-

nitions.

Our formalism allows to express the effect propagation process with high precision. Evidence for this is found in the fact that in many applications, a representation is obtained which — almost as a side effect — correctly models the interacting effects of simultaneous actions. Our approach is presented independent of a specific time structure, such as Situation Calculus, $\mathcal{A}$ or Event Calculus, but we briefly discuss how it can be embedded in different time structures.

1 Introduction

Correctly representing indirect effects of actions (ramifications) is a fundamental issue in the research on reasoning about actions which has received a lot of attention lately, e.g. in [23, 21, 35, 15, 33, 14]. In this paper we introduce an approach which allows for much greater flexibility in the representation of indirect effects, both of individual actions and sets of simultaneous actions. Our approach is independent of a specific time structure and can therefore be embedded in most temporal reasoning formalisms. We offer the following contributions.

First of all, we argue that in order to represent general ramifications, independent effect propagation rules (*derived effect rules*[1]) are required which — contrary to the ones in existing approaches — are not necessarily coupled with state constraints: it is not hard to show that in the real world indirect effects are quite often not related to any actual state constraints, just as well as state constraints do not always lead to indirect effects. Up to now this has been neglected in the literature.

Second, we define a precise and general semantics for the proposed effect rules. There exist several approaches to defining a semantics for sets of rules, e.g. forms of circumscription in nonmonotonic reasoning, inductive definition semantics in mathematical logic, and several semantics for logic programs. We show that the approaches in these different research areas can all be seen as instances of the same idea, i.e. interpreting the set of rules as a generalised form of *inductive definition*, a notion that was elaborated in [9]. However, apart from the most general logic programming semantics, all approaches depend on different instances of a syntax-dependent notion of acyclicity. In reasoning about actions an acyclicity condition can be justified because of the acyclic nature of causality, but we argue that this notion should not be imposed on the syntactic level. This is because an evident requirement in knowledge representation is the modularity of representations: it should be possible to describe the behaviour of a system in terms of the behaviour of its components, where each component *type* has a specific representation which is used for different instances, in the same system or in different systems. The model of a system should only be dependent on the types of all components and on their connections. We will show that such a representation can give rise to apparently (i.e. syntactically but not semantically) cyclic definitions, even in the description of perfectly normal, well-behaved physical systems. Therefore we substitute the syntactic notion of acyclicity with a more flexible notion of well-founded, non-self-supported argument at the semantic level. This notion is investigated in detail in [9]. It is shown to correspond to the well-founded semantics of logic programs [40], but formalised in a way that explicitates the idea of

[1] roughly comparable to "causal laws" or "causal rules" in the literature

well-founded argument. The formalisation in the current paper is very similar to the one in [9]. By showing the appropriateness of this notion for formalising effect propagations, we complement the theoretical arguments of [9] with a correspondence to phenomena in the real world.

Third, we show that in many applications a *compact* and natural representation of direct and indirect effects of (possibly simultaneous) actions, with possibly interacting effects, can be obtained using one particular type of complex derived effect rules, i.e. effect rules stating that an effect is triggered by the change in truth value of a *complex fluent formula*. This is related to the observation that in a more low-level description of a problem domain the primitive derived effect rules tend to occur in particular combinations. We will show the relation between complex rules and combinations of primitive effect rules.

The focus of the paper is on the ramification problem, i.e. on determining the complete set of effects of a set of actions in a particular state. The semantics of a set of effect rules (an *effect theory*) is expressed by formally defining a *transition function* that maps any tuple (S, A) of a state of the world S (at some time point or situation) and a set of actions A which occur simultaneously in that state, to a new state in which all direct and indirect effects of the actions are realised. Inertia is tackled in the following way: a fluent may only change if a cause for the change can be derived from the effect theory (in the context (S, A)).

Other issues, such as temporal structure (branching versus linear time versus explicit representation of states as in Strips-like languages and A), the solution to the default qualification problem, etc., are not considered in detail. Also a complete solution to the inertia problem (including axioms stating that nothing changes as long as there are no actions) is not studied because a precise formalisation necessarily relies on a specific time structure and formalism, to which we do not want to commit ourselves. In other words we single out the ramification problem and study it independent of other issues in temporal reasoning. As a result the solution can be embedded in different temporal reasoning formalisms. As an example, in [37] and in more detail in [38] and [36], we present an embedding of the approach in an event-based language $\mathcal{ER}$. In addition we have addressed the issue of reasoning on the theory by mapping the language to Open Logic Programming, an extension of Logic Programming on which an abductive resolution procedure can be used for various forms of reasoning.

The rest of the paper is organised as follows. In the following section we introduce the types of information we will consider in reasoning about action, their relations and the representation methodology. The next sections are devoted to illustrating how different approaches to nonmonotonic reasoning, inductive definitions and logic programming semantics can be (and have been) used for dealing with some classes of problematic definitions relying on some form of acyclicity. Section 5.2 describes the general approach that only introduces acyclicity at the semantic level. In subsequent sections we discuss the issue of delays in effect propagations and introduce limited nondeterminism, based on which we compare our approach in detail with the one of Thielscher in [35]. Finally we show more relations to other recent work.

2 Motivation of the approach

In order to represent dependencies between fluents (or their changes) and actions or events in a system, three types of information are commonly used:

- *State constraints* represent properties that hold in each state, or at all times, in the dynamic system. These constraints describe the system from a static point of view.
- *Action preconditions* or *action qualifications* specify necessary conditions on the state of the world for particular actions to occur.
- *Effect rules* represent the effects of actions or events, and more in general all of the changes occurring in the system. One can distinguish between
  - *direct effect rules (action laws)*, representing the primitive effects of actions, i.e. the changes that are directly associated with actions;
  - *derived effect rules (causal laws)*, representing indirect effects or *ramifications*, i.e. how changes in certain fluents propagate to cause changes in other fluents. Using such rules in addition to direct effect rules, rather than trying to give all the effects of actions directly, is essential for the modularity of a representation.

Action preconditions and derived effect rules are often related to state constraints. For example, to the constraint that walking persons are necessarily alive, correspond both an implicit action precondition on the action $start\_walk$ ($alive$ must hold) and a derived effect rule that when someone dies, he/she also stops walking.

In the state of the art approaches to the frame problem, it is acknowledged that neither action qualifications nor ramifications can be derived automatically[2] from state constraints [12, 13, 20, 29] and must be explicitly represented. Yet, to date, action preconditions and especially causal laws are still considered to be tightly coupled with

[2] This is obvious e.g. in the above example, where the same state constraint represents a precondition on one action and leads to ramifications on other actions (those terminating $alive$).

state constraints. In some approaches, e.g. [21, 23, 16, 15], causal laws include an implicit state constraint component[3]. In the approach of [35], the semantics of causal rules is defined in terms of the state constraints, in the sense that they are seen as rules "restoring" the state constraints (therefore the same causal rule may have different semantics in contexts with different state constraints); causal rules are also (suggested to be) derived from state constraints augmented with additional *influence information*.

As mentioned in section 1, we argue in this paper that, in general, action qualifications and ramifications may be entirely independent of state constraints.

As far as qualifications are concerned, an obvious example of an action precondition which does not correspond to a state constraint is that in a chess game moving a piece from one position to another is only possible if the moved piece is initially on the starting position. On the other hand, the state constraint that a player may not be in check after his own move, represents an implicit precondition on possible moves; it would be very difficult and clumsy to express this action precondition explicitly, as a condition on the starting state from which the player makes the move.

As concerns ramifications, rather than focussing on their relation to state constraints and how these can be maintained through indirect effects, we model the physical forces or logical dependencies in the system; we consider causal laws to be representations of how changes propagate. As such, causal laws may be but do not need to be related to state constraints. Below is an example.

Example 1 Obvious examples of systems exhibiting indirect effects unrelated to state constraints are electronic counters in a digital network. Assume a network with many input signals, which can be directly modified through actions, an arbitrary internal structure, and one or more output lines. A counter on one of these output lines counts the number of times the output voltage changes from 0 to 1. It seems natural to represent this information by *derived effect rules* like

$$caus(count(n+1)) \leftarrow init(out), holds(count(n))$$
$$caus(\neg count(n)) \leftarrow init(out), holds(count(n))$$

Here $holds(l)$ means that l is true in the initial state (for a fluent literal l). $caus(l)$ means that there exists a cause for l to become true; it is possible that l is already true in the initial state, i.e. that $holds(l)$ is true; obviously, in that case l should remain true. $init(l)$ means that l changes from false to true: l is false in the initial state and there is a cause for l. For the time being, $init(l)$ is used as a shorthand notation for $holds(\neg l) \wedge caus(l)$.

[3] or a constraint relating different states in the case of delayed effect rules in [15].

The indirect effects described by the above rules are in no way related to a state constraint[4] : there is no relation between the contents of the counter and the current state of the network. What is happening here is just a propagation of effects. Indirect effects of this kind cannot be correctly modelled in state constraint based approaches.

We want to design an expressive language for describing effect propagations which satisfies the following conditions:

- expressivity: the language should cover a broad class of applications.
- robust representation of multiple effects: multiple effects may arise due to simultaneous actions or due to the presence of ramifications (even if any action has a single direct effect and there are no concurrent actions). If multiple changes are allowed to occur "at the same time" (at least, at the level of abstraction that assumes that direct and indirect effects occur with no delay), no matter if they are direct or indirect effects of a single or multiple actions, some problematic cases may arise if the different changes are not independent.
- modular representation: each *type* of component in the system should have a specific representation which can be reused in different instances, in the same system or in different systems.
- compact and intuitive representation.

As will be shown below, these requirements lead to a formalism in which effect propagations depend on the absence of other effects, and in which positive cycles may occur in the effect rules.

Example 2 Consider the suitcase example [21], where a suitcase with two locks opens if both locks are opened at the same time, or one lock is opened and the other one is *and remains* open.

The direct effect rules can be represented as follows:

$$caus(lock1open) \leftarrow act(open1)$$
$$caus(lock2open) \leftarrow act(open2)$$

for which the following high-level syntax can be given as in the $\mathcal{A}$ language and its variants:

$$open1 \ \textbf{causes} \ lock1open$$
$$open2 \ \textbf{causes} \ lock2open$$

Since the opening of the suitcase depends also on the openness of locks *not being terminated* at the considered time, and since we want

[4] To be precise, the second rule can be derived from the first one given the state constraint that count has only one value; strictly speaking only the first rule is then not related to a state constraint.

to take into account multiple changes at the same time, a possible set of derived effect rules is the following:

$$\begin{array}{lcl} caus(open) & \leftarrow & init(lock1open) \wedge holds(lock2open) \wedge \\ & & \neg init(\neg lock2open) \\ caus(open) & \leftarrow & init(lock2open) \wedge holds(lock1open) \wedge \\ & & \neg init(\neg lock1open) \\ caus(open) & \leftarrow & init(lock1open) \wedge init(lock2open) \end{array}$$

Note that intuitively, the above three rules can be represented more compactly as follows:

$$caus(open) \leftarrow init(lock1open \wedge lock2open)$$

or in a more $\mathcal{A}$-like syntax

$$\textbf{initiating } lock1open \wedge lock2open \textbf{ causes } open$$

where we introduced initiation of a complex formula, a conjunction in this case: the cases where this conjunction *becomes* true are exactly the three ones in the informal description and the three rules above.

The suitcase example illustrates two principles of our approach. The first is that effects may depend on the absence of other effects, for which reason the formalism should allow negative causality literals. On the semantic level, this may create problems when effects mutually depend on each others absence.

The second principle is the use of compact derived effect rules based on the initiation of complex fluents. Such complex effect rules can be reduced to sets of primitive effect rules (using only initiation of literals). This reduction is based on a simple case analysis of when complex fluent formulae are initiated given changes in the involved literals. We have evidence that most or all causal dependencies can be conveniently expressed in one compact form, and hence that there is no need for rules with arbitrary combinations of *init*, *caus*, *holds* and *act* literals.

Another potentially problematic form of dependency that could especially arise in the representation of physical systems is a cyclic dependency.

Example 3 Consider two connected gear wheels. Any force causing the first wheel to start or stop turning propagates to the second wheel and vice versa. The propagation of the effects in this system is represented by the cyclic rules:

$$\begin{array}{l} caus(turn_2) \leftarrow caus(turn_1) \\ caus(turn_1) \leftarrow caus(turn_2) \\ caus(\neg turn_2) \leftarrow caus(\neg turn_1) \\ caus(\neg turn_1) \leftarrow caus(\neg turn_2) \end{array}$$

Assume that there are the following direct effects:

$$caus(turn_1) \leftarrow act(start_1)$$
$$caus(turn_2) \leftarrow act(start_2)$$
$$caus(\neg turn_1) \leftarrow act(stop_1)$$
$$caus(\neg turn_2) \leftarrow act(stop_2)$$

The intended semantics of this set of rules is clear in all cases where there are no simultaneous occurrences of *stop* and *start* actions: both wheels start to turn iff there is at least one *start* action, and both wheels stop iff there is at least one *stop* action. What should be avoided is to allow for both wheels to start turning in the absence of any actions justifying this, by reasoning that the two changes justify each other.

Notice that such a cyclic dependency would not exist if the two fluents, which are apparently equivalent in the system (the state constraint $turn_1 \leftrightarrow turn_2$ holds) were replaced by a single fluent, which is caused by $start_1$ and by $start_2$. However, such a change in representation, which is easy in this case, cannot be seen as an instance of a systematic and modular representation methodology.

Several additional issues and choices need to be considered. One choice we make for reasons of conceptual clarity is to aim for a representation and a semantics such that, as long as nondeterminism is not *explicitly* introduced in effect rules, given a complete state of the world, the state resulting from an action or a set of concurrent actions is *uniquely* determined. More precisely, actions produce a unique set of direct effects, which lead to a unique set of additional (indirect) effects by propagation through the derived effect rules. We assign the semantics of effect rules constructively, corresponding to the actual physical or logical propagation[5].

If multiple changes at the same time are considered, we insist that only theories that do not entail multiple changes of the same fluent at the same time are considered as well-defined. In particular a fluent can not change from true to false and back again in the same batch of effects. This ensures the "same time" abstraction to be correct. In this respect our approach differs conceptually from the one in [35], where change propagations in one "simultaneous" batch of effects are assumed to happen consecutively and to take a very small amount of time. Hence in Thielscher's approach multiple changes of the same fluent in the same batch of effects are not considered problematic or inconsistent.

In order to achieve the goals described above and avoid the potential problems due to cyclic dependencies and dependencies on absence of effects, we introduce in the semantics a notion of well-founded, non-self-supported argument, which is especially appropriate for the semantics of causation. In fact, as remarked in [34], essential properties of causation are:

[5] State constraints do not take part in this constructive process. We return to the role of state constraints later.

- each effect should be caused, either by a primitive action or by another effect;
- there cannot exist self-supported effects;
- effect causation is well-ordered; there should not be an infinite chain of effects in which each effect is caused by the next effect in the chain.

In the literature of nonmonotonic reasoning, semantics of logic programming and inductive definitions, several approaches exist which can be reduced to the principles above or to some sort of "closure" principle for definitions. These approaches have been shown to be appropriate for different syntactic subclasses, some of them including — under certain conditions — negative literals in the body of rules (as for the $\neg init$ literals in the effect rules considered before), some of them including the possibility of positive loops as in the gear wheels example. In particular,

- predicate completion [7] is adequate for acyclic rule sets [2];
- circumscription [24] is adequate for positive rule sets;
- prioritized circumscription [18] is adequate for stratified rule sets;
- perfect model semantics is adequate for stratified rule sets [30];
- Iterated Inductive Definition semantics is adequate for stratified rule sets [1];
- the well-founded semantics of logic programs [40] is adequate for general rule sets, giving an "undefined" value to the atoms that do not have a well-founded, i.e. non-self-supported, argument in favor of their truth or falsity.

In the intersection of the syntactic classes, the different approaches coincide. This suggests that there is a uniform intuitive principle underlying these rules, which has been called the (generalised) principle of inductive definitions in [9]. The principle is related to the well-founded semantics, but will here be formulated in a way that explicitates the well-founded arguments.

In the following sections, we will investigate the semantics of theories of effect rules of the form

$$caus(l) \leftarrow \mathcal{L}_1, \ldots, \mathcal{L}_n$$

where l is a fluent literal and, for $i = 1, \ldots, n$, $\mathcal{L}_i$ is a literal of one of the forms

- $caus(l), \neg caus(l), holds(l), \neg holds(l)$ where l is a fluent literal. The $holds$ literals refer to the starting state of the transition.

- $act(a), \neg act(a)$ where a is an action constant; these literals refer to the actions that are part of the transition.

These rules will be called primitive effect rules.

In the most general case, the above definition is extended by also allowing literals $init(l)$ and $\neg init(l)$ in the body of rules and by allowing l in body literals to be a complex propositional fluent formula rather than a single fluent literal.

The rules involving complex fluent formulae and *init* literals can be reduced to (sets of) primitive rules. For *init* literals this reduction is straightforward using the equivalence $init(l) \leftrightarrow caus(l) \wedge holds(\neg l)$. Also the reduction of complex *holds* formulae to primitive ones is straightforward (*holds* can be moved inward through all operators). These extensions can be considered basic syntactic sugar and will be treated as such. The reduction of complex *caus* literals to primitive ones requires a more careful case analysis as illustrated in the suitcase example, and will be formalised later for the general case.

Finally, we choose to introduce some additional syntactic sugar with an $\mathcal{A}$-style look and feel for a particular subset of the effect rules which is sufficient for actual knowledge representation:

- a **causes** l **if** F where l is a fluent literal, a an action constant and F a complex fluent formula, is equivalent to the rule $caus(l) \leftarrow act(a) \wedge holds(F)$

- **initiating** F' **causes** l **if** F where l is a fluent literal and F, F' complex fluent formulae, is equivalent to the definition rule $caus(l) \leftarrow init(F') \wedge holds(F)$

If F is trivially true, we omit the **if** condition. In a later section we will discuss this choice of high-level effect rules.

The above description does not allow for variables in fluent formulae. Our formalisation can be extended to formulae involving variables and full quantification, but we do not address the issue in this paper. In some examples in the paper we use variables (without quantification): the intended meaning is that all variables are universally quantified with the entire rule as scope, and rules containing variables are considered as equivalent to the set of all their ground instances.

Given a theory Π_e of *primitive* effect rules[6] , our formal goal in the following sections is to define, for increasingly complex classes of theories, a transition function $Trans_{\Pi_e}$ that maps a pair consisting of

- a set S of fluent atoms, representing a state of the dynamic system,

- a set A of actions which happen simultaneously in state S.

[6]Later on we will define the semantics of *complex* effect rules by mapping them to an equivalent set of primitive rules.

to a new fluent set S' representing the state after the simultaneous occurrence of actions A in state S.

To this end we will first define a canonical interpretation M_{S,A,Π_e} determining which fluents are caused to be true or false given the rules in Π_e and a particular S and A. The problem of determining this interpretation is split up in several cases, corresponding to different syntactically restricted classes of theories. In sections 4, 3 and 5.1, we focus respectively on

- hierarchical effect theories: with negation but without recursion;
- positive recursive effect theories: no negation;
- stratified effect theories and general effect theories (with recursion and negation).

The canonical model defines the update set $U_{\Pi_e,S,A}$ for given S and A. This is basically the set of true *caus* literals in the canonical model, unless the theory yields a so-called bad definition.[7] The precise definition is as follows: if M_{S,A,Π_e} assigns a truth value $\mathbf{u}$ to any *caus* literal, then $U_{\Pi_e,S,A} = \{caus(l) \mid l \text{ is a fluent literal }\}$. Otherwise, $U_{\Pi_e,S,A} = \{caus(l) \mid M_{S,A,\Pi_e}(caus(l)) = \mathbf{t}\}$.

The transition function $Trans_{\Pi_e}$ is then defined as follows:

- if $U_{\Pi_e,S,A}$ contains a subset of the form $\{caus(f), caus(\neg f)\}$, then $Trans_{\Pi_e}(S,A) = \top$, where $\top$ denotes the inconsistent state.
- otherwise, $Trans_{\Pi_e}(S,A) = S \setminus \{f | caus(\neg f) \in U_{\Pi_e,S,A}\} \cup \{f | caus(f) \in U_{\Pi_e,S,A}\}$.

3 Predicate completion and explanation closure

In this section, we define the canonical model and hence the transition function for the class of hierarchical effect theories, in which effects may depend on the absence of other effects (i.e. negative *caus* literals may appear in the body of effect rules) but no recursive dependencies may exist. More formally, for a given effect theory Π_e, define the *immediate dependency relation* as the set of tuples of fluent literals (l, l') such that there is a rule $caus(l) \leftarrow ..caus(l')$ in Π_e. Define the dependency relation as the transitive closure of this immediate dependency. The class of hierarchical effect theories is the one in which no fluent literal depends on itself[8]. For this class of theories, a

[7] The possibility of bad definitions exists only in the most general case and will be discussed there.

[8] Note that only *caus* literals are taken in consideration to define dependency. The causation of a fluent may depend on the presence or absence of the same fluent in the initial state. This does not lead to recursion.

conceptually simple and intuitive approach in the literature can deal with some of the complexity involved in defining direct and indirect effects of actions.

As a base example, consider again the suitcase example (example 2). Defining a semantics for the set of rules requires being able to deal correctly also with the dependencies on absence of terminations. Since as usual in nonmonotonic reasoning (and in work on the frame problem in particular), we are not expecting reasons for the non-occurrence of effects to be provided explicitly, we should consider the rules above as completely *defining* all reasons for changes in the fluents involved in them. One of the early ways of formalising this definitional view in logic programming and nonmonotonic reasoning was to see a set of implication rules with a common "head" p as defining p, in the sense of reading them as "if and only if" rather than just "if". This is the predicate completion approach in [7].

Completion of the suitcase example at the ground term level yields the following theory[9] :

$$\begin{aligned}
caus(lock1open) &\leftrightarrow \; act(open1)\\
caus(lock2open) &\leftrightarrow \; act(open2)\\
caus(open) &\leftrightarrow \; init(lock1open) \wedge holds(lock2open) \wedge \\
&\qquad \neg init(\neg lock2open)\\
&\qquad \vee\, init(lock2open) \wedge holds(lock1open) \wedge \\
&\qquad \neg init(\neg lock1open)\\
&\qquad \vee\, init(lock1open) \wedge init(lock2open)
\end{aligned}$$

Here, a well-ordered way of assigning truth values is as follows: given a (real or hypothesized) complete state of the world (i.e. an assignment of truth values to fluents), and an action or a set of actions occurring (or hypothesized to occur) in that state,

- we first decide which changes occur to $lock1open$ and $lock2open$ given the complete definition of when such changes are caused and the current truth value of the fluents (this is important since, as argued before, ramifications usually involve changes driven by other changes, not just by something else being true or false);

- then we decide which change may affect $open$ given the complete definition of when such a change is caused.

In other words we first decide which direct effects occur, then which indirect effects occur. This construction can be made for all hierarchical effect theories and yields always the unique model of its completion.

This approach works fine in case of simultaneous actions. Assume that a person opens lock 1 and closes lock 2 simultaneously. In that case, the suitcase should remain closed; this is exactly what the completion semantics yields: in the first step of the construction, lock 1

[9] recall that $init(l)$ stands for $caus(l) \wedge \neg holds(l)$

is determined to be open, lock 2 to be closed. In the second stage of the construction, it is found that not both locks are closed, hence the rule for opening cannot be applied.

There is one important remark to be made here though. In the construction, it can be seen that the evaluation of effect rules is delayed until the existence or absence of effects at lower levels is determined. This hides a subtle assumption about the physical system; namely that in the physical system, the propagation of a higher level effect is delayed until the lower level fluents on which the effect depend have become stable in the physical system. An example illustrates the point.

Assume that the suitcase is opened, not be a mechanical spring but by an electrical motor which reacts to two hypersensitive nanosecond detectors connected to the locks. So, the locks do not mechanically block the opening of the suitcase. In such a system, it must be inpredictable whether the suitcase will open or not when a human opens one and *simultaneously* closes the other lock. This is simply because even when in the human's perception, the two actions occur simultaneously, the detectors will still discover some small delay between them, and if the opening action is detected first, the opening signal will be already sent to the motor before the closing of the other lock is detected.

In this alternative suitcase example, the completion is not correct because in a physical system the rule $caus(open) \leftarrow init(lock1open), holds(lock2open), \neg init(\neg lock2open)$ may fire before the system has stabilised on its premises. It is an underlying assumption of this representation which abstracts away delays, that the physical system stabilises on the premises of a rule before propagating the effect described by the rule. If this assumption is not satisfied, the completion semantics cannot be used, and in general we cannot expect any semantics to reason correctly on delays that have not been modelled (we will return to this issue in section 6).

In [2], it is shown that the completion of a hierarchical logic program has one unique model. It follows that given any state S and set of actions A, a hierarchical effect theory Π_e has exactly one model. This is the model M_{S,A,Π_e} used to define the update set and the transition function, using the definition given in the previous section.[10].

The completion is also able to deal with some potentially cyclic dependencies, as in the following example.

Example 4 Suppose that a person is happy if a person he hates is not happy (and this is the only reason to be happy), and the only cause of hatred is the rejection of papers. Suppose we start with

[10] Observe that no truth value **u** occurs anywhere as the completion yields a classical (2-valued) FOL theory; hence the update set is simply the set of true *caus* literals, as intended.

a state where John and Fred are unhappy and no one hates anyone; then John rejects a paper of Fred's, which surprisingly enough[11] makes Fred happy. A compact representation would be:

$$\textbf{initiating } hate(X,Y) \wedge \neg happy(Y) \textbf{ causes } happy(X)$$

$$reject\_paper(Y,X) \textbf{ causes } hate(X,Y)$$

which corresponds to the following low-level representation:

$$\begin{array}{lcll}
caus(happy(X)) & \leftarrow & init(hate(X,Y)) \wedge holds(\neg happy(Y)) & \\
 & & \wedge \neg init(happy(Y)) & (1) \\
caus(happy(X)) & \leftarrow & init(\neg happy(Y)) \wedge holds(hate(X,Y)) & \\
 & & \wedge \neg init(\neg hate(X,Y)) & (2) \\
caus(happy(X)) & \leftarrow & init(hate(X,Y)) \wedge init(\neg happy(Y)) & (3) \\
caus(hate(X,Y)) & \leftarrow & act(reject_paper(Y,X)) & (4)
\end{array}$$

We omit the corresponding rules for termination of happiness for reasons of brevity.

Notice that in rules (1), (2) and (3) happiness depends on happiness. The resulting completion is:

$$\begin{array}{lcll}
caus(happy(X)) & \leftrightarrow & init(hate(X,Y)) \wedge \neg holds(happy(Y)) & \\
 & & \wedge \neg init(happy(Y)) & \\
 & \vee & init(\neg happy(Y)) \wedge holds(hate(X,Y)) & \\
 & & \wedge \neg init(\neg hates(X,Y)) & \\
 & \vee & init(hate(X,Y)) \wedge init(\neg happy(Y)) & (5) \\
caus(hate(X,Y)) & \leftrightarrow & act(reject_paper(Y,X)) & (6)
\end{array}$$

Given the initial state:

$$\neg happy(john) \wedge \neg happy(fred) \wedge \forall X,Y \neg hate(X,Y)$$

and the action $reject\_paper(john, fred)$, i.e.

$$act(X) \leftrightarrow X = reject\_paper(john, fred)$$

the completion entails $init(hate(fred, john)$ and therefore also $init(happy(fred))$ because of the first disjunct of (5). Notice that $\neg init(happy(john))$ is true because, considering again (5) with $X = john$, the first and third disjuncts are false because no one rejected a paper of John's, and the second disjunct is false because John does not hate anyone.

The completion approach is a nice solution in some respects, since, after a conceptually simple syntactic transformation, it relies on the established semantics of classical (monotonic) logic. In reasoning about actions, this is essentially the approach introduced by Reiter [31], based on a monotonic solution to the frame problem which explicitates an "explanation closure" assumption. This approach, which

[11] Given real-world knowledge, but not given the rules provided in this example.

deals with actions with only direct effects, is further extended in [26] to deal with the ramification problem (restricted to the case of acyclic theories).

The shortcomings of this kind of solutions arise in case of positive cyclic dependencies, as shown in the next section.

4 Causation theories with positive cycles

Consider again the two gear wheels example (example 3). How can we model the semantics of this set of rules in correspondence with the intended one? The Clark completion does not correctly model these effects. For example, in case of an empty set of actions, the intended model is that all *caus* atoms are false, but one verifies that e.g. the interpretation $\{caus(turn\_1), caus(turn\_2)\}$ is also a model of this theory. Hence, according to Clark completion, the two gear wheels might start turning spontaneously.

To adequately model the effect propagation process in the above example, an appropriate semantics is the one used for inductive definitions [6, 27, 1]. By interpreting the above set of causal rules as an inductive definition, the intended semantics is obtained. The problem of completion semantics to model the propagation of the cyclic dependencies between effects, is entirely equivalent to its problem to model inductive definitions.

Positive Inductive Definitions (PID, i.e. definitions without negations in the body), have been formalised in various ways. Aczel [1] studies inductive definitions in an *abstract* representation with an obvious correspondence to definite logic programs [22]. A definition on a domain D of propositional symbols is represented as a possibly infinite set $\mathcal{D}$ of rules $p \leftarrow B$ with $p \in D, B \subseteq D$. Aczel gives three equivalent mathematical principles for describing the semantics of PID's. They are equivalent with the way the semantics of definite logic programs is defined [39]:

- The intended model of $\mathcal{D}$ can be defined as the least model of the implications. E.g. in [10], this unique minimal model semantics is expressed through a circumscription-like axiom (but expressing that the interpretation must be the least rather than a minimal model).

- The model can be expressed constructively as the least fixpoint of a step-wise operator[12] associated with the definition.

- The model can also be expressed as the interpretation in which each atom has a *proof tree*[13].

The PID semantics can be used to define the semantics of causal theories without negative literals. The above three principles and

[12] This operator is analogous to the T_P-operator for definite logic programs.
[13] Also this formalisation has been used in LP in [8].

its variants (e.g. circumscription) are mathematically equivalent and yield identical semantics. We have chosen the proof tree formalisation, for the reason that it is a constructive semantics which reflects the constructive nature of the effect propagation. A least model or minimal model semantics, though elegant and simple, is not constructive. A stepwise operator semantics is constructive as well but is not easily extended to the case of causal theories with negative cause literals.

We present the proof tree semantics for a slightly extended version of the formalism of [1][14].

Given is a domain D of propositional symbols including symbols $\mathbf{t}, \mathbf{f}$.

Definition 1 *A Positive Inductive Definition is a set of rules* $p \leftarrow B$ *with head* $p \in D$ *and body* B *a nonempty set of positive or negative literals of* D*. Let* $Defined(\mathcal{D})$ *be the set of atoms which occur in the head of a rule. For every rule* $p \leftarrow B \in \mathcal{D}$*, for every negative literal* $\neg q \in B$*, it should hold that* $q \notin Defined(\mathcal{D})$*.*

The set $Defined(\mathcal{D})$ is called the set of defined atoms. Its complement $Open(\mathcal{D}) = D \setminus Defined(\mathcal{D})$ is called the set of open atoms. We will assume always that $Open(\mathcal{D})$ includes the atoms $\mathbf{t}$ and $\mathbf{f}$, denoting true, false respectively.

Note that each symbol $p \in Defined(\mathcal{D})$ has at least one rule $p \leftarrow B \in \mathcal{D}$ (it may be the rule $p \leftarrow \{\mathbf{f}\}$) and the body B of a rule is never empty (B may be the singleton $\{\mathbf{t}\}$).

For any given interpretation M of the open symbols, $\mathcal{D}$ defines the truth value of all defined symbols of $\mathcal{D}$. The resulting model, which we denote $M^{\mathcal{D}}$, is the least model extending M and satisfying the rules of $\mathcal{D}$. The proof tree semantics is equivalent to this:

Definition 2 *A* $\mathcal{D}$*-proof-tree* $\mathcal{T}$ *of* $p \in D$ *is a tree of literals of* D *with* p *as root such that:*

- *all leaves of* $\mathcal{T}$ *are open literals (possibly* $\mathbf{t}, \mathbf{f}$*);*
- *for each non-leaf node* p *with set of immediate descendants* B*, it holds that* $p \leftarrow B \in \mathcal{D}$*; hence* p *is a defined symbol in* $\mathcal{D}$*;*
- $\mathcal{T}$ *contains no infinite branches (hence it is loop-free).*

Note that when $\mathcal{D}$ is a set of effect rules, a prooftree gives a visual picture of the effect propagation leading to the effect in the root of the tree.

Below is a constructive characterisation of the model of $\mathcal{D}$ given M.

[14] This extension facilitates the leap to the case of causal theories with negative literals in the next section.

Definition 3 *The model $M^{\mathcal{D}}$ is the set $\{p \in D | p$ has a proof tree $\mathcal{T}$ with only true leaves*[15] *in $M\}$.*

Using the above definitions, we define a semantics for the class of positive causal theories.

Definition 4 *A causal theory Π_e is positive iff all causation rules have no negative causation literals in the body*[16].

For any pair S, A, one defines the interpretation $M_{S,A}$ of holds and action atoms $M_{S,A}(holds(l)) = \mathbf{t}$ iff $l \in S$ and $M_{S,A}(act(a)) = \mathbf{t}$ iff $a \in A$. The canonical model M_{S,A,Π_e} used to define the update and transition functions is then ${M_{S,A}}^{\Pi_e}$.

Observe that — not surprisingly — for acyclic positive definitions, the results of the positive induction semantics and the completion semantics coincide. They capture the same intuition for different classes of definitions, and coincide in the intersection of these classes.

Consider again the gear wheels example, with effect rules

$$\begin{array}{l}
caus(turn_2) \leftarrow caus(turn_1) \\
caus(turn_1) \leftarrow caus(turn_2) \\
caus(\neg turn_2) \leftarrow caus(\neg turn_1) \\
caus(\neg turn_1) \leftarrow caus(\neg turn_2) \\
caus(turn_1) \leftarrow act(start_1) \\
caus(turn_2) \leftarrow act(start_2) \\
caus(\neg turn_1) \leftarrow act(stop_1) \\
caus(\neg turn_2) \leftarrow act(stop_2)
\end{array}$$

Assume a situation in which the wheels are not turning, i.e. $S = \{\neg turn_1, \neg turn_2\}$ and an action $start_1$. $caus(turn_1)$ has two (loop-free) finite proof trees: one in which it has as only immediate descendant the leaf $start_1$, and one in which it has $caus(turn_2)$ as immediate descendant and hence $start_2$ as only leaf. The former proof tree determines that $caus(turn_1)$ is true. Similarly, $caus(turn_2)$ is true due to the proof tree $caus(turn_2)$ — $caus(turn_1)$ — $start_1$. The finite proof trees of $caus(\neg turn_1)$ and $caus(\neg turn_2)$ all have false leaves (either $stop_1$ or $stop_2$), hence these literals are false. The update set is consistent and both wheels start turning. Observe that simultaneous $start_i$ and $stop_j$ actions would yield an inconsistent update set where both fluents and their negations would be initiated. This simultaneous occurrence is hence not allowed by the theory. Modelling what would happen in the real system if the actions were tried simultaneously, would require a greater level of detail.

[15] Note that an open atom p has only one prooftree, and this prooftree has one single node p. Hence, p has a prooftree with only true open literals in the leaves iff p is true in M. Hence, $M^{\mathcal{D}}$ is identical to M in $Open(\mathcal{D})$.

[16] We tacitly assume there exists a rule for every atom, possibly $caus(l) \leftarrow \{\mathbf{f}\}$.

5 A General Semantics

5.1 Stratified and Nonstratified Rule Sets

The positive induction semantics defined in the previous section allows for dealing with cyclic dependencies between effects. However, as opposed to the completion approach it can not deal with negations in effect rules, in other words with effects that depend on the absence of other effects. That both semantics coincide in their common range of applicability, suggests that they are approximations of the same more general concept in different specialised settings. It is this concept we are trying to identify.

In fact, as pointed out in [9], ways of extending the positive induction semantics to allow for negations in the rules have been studied as well in the area of Iterated Inductive Definitions (IID) as in Logic Programming and in the context of circumscription. In all three research areas similar extensions have been proposed which cover stratified rule sets. A rule set is stratified if atoms can be ordered in layers such that no atom depends on any atom in a higher layer, nor on the negation of an atom in a higher or equal layer. In terms of effect rules, this corresponds to the condition that effects can not depend in any way on effects in higher layers, and if they depend on the *absence* of another effect, this other effect should be in a strictly lower layer (since a negative effect literal evidently represents the absence of an effect).

Stratified rule sets are handled in IID by determining the effects one layer at a time. First it is determined which of the fluents in the lowest, most primitive layer are initiated, using only the effect rules for these fluents as a definition and only **t** and **f** as open atoms. These effect rules are not dependent on any effects in higher layers, so they can be dealt with independently. Moreover they do not depend on the absence of any effects, so they form a positive causal theory. Hence, the positive induction semantics of the previous section is applicable. When all effects in a layer are determined, in the next step these effects are added to the set of open literals and assigned their determined truth value. The effect rules for the next higher layer can then be tackled in exactly the same way, which is repeated until all effects are determined and a fixpoint is reached. For more formal details we refer to [1] or [27]. An alternative formalisation of this semantics in the logic programming setting is described in [30], and in the circumscriptive setting in [18]. Unsurprisingly, for definitions without positive cycles the approach again coincides with the completion approach.

Stratified definitions are the largest class for which a semantics is defined in IID. The stratification condition imposes that atoms (e.g. effects) can not depend on each other's or their own negation (e.g. absence): so-called *negative cycles* are not allowed. The fundamental reason for this is that negative cycles may make definitions nonsensical or ambiguous. They are much more problematic than positive cycles: as each effect needs a well-founded argument to occur, an effect that is only justified by a positive cycle does not occur. But the absence of an effect requires no argument, only that there is no argument for the effect's presence. Hence, if we consider the definition

$$caus(p) \leftarrow \neg caus(q)$$
$$caus(q) \leftarrow \neg caus(p)$$

there can be two possible readings: either we assume there is no argument for $caus(q)$, in which case $\neg caus(q)$ is an argument for $caus(p)$ and hence there is indeed no argument for $caus(q)$, or we apply the same reasoning with p and q switched. Obviously the arguments for both readings are self-supported, although on a higher level than in the case of a positive cycle. In this case there is no way around the problem, because rejecting both arguments, and hence concluding that p nor q are initiated, would lead to a violation of the rules read as simple implications. An even more extreme example is the definition

$$caus(p) \leftarrow \neg caus(p)$$

which can not be given a sensible reading based on non-self-supported arguments.

Since negative cycles lead to nonsensical or at least ambiguous definitions as illustrated above, they have been disregarded in IID. However, the notion of a cycle in the IID setting (and the LP or circumscriptive settings) is defined syntactically, whereas syntactic cycles do not necessarily correspond to actual cycles. Hence, a definition which contains negative cycles on the level of the syntax may nevertheless be a perfectly meaningful definition. In fact the "happiness" example of section 3, in which $happy(X)$ depends negatively on itself, is not stratified, whereas in most cases it is perfectly possible to determine who remains happy and who does not. Also propositional definitions may be syntactically nonstratified but have a clear meaning: it is possible to construct a physical system of which the natural representation contains negative cycles, but which exhibits nonproblematic and well-defined behaviour.

Assume for example the following electric circuit (a variation of the examples in [35])consisting of two interconnected sub-circuits. On one circuit there are two serially connected switches p and s and a relay $relay1$, on the other two switches r and q and a relay $relay2$. The first relay ensures that q is closed (i.e. true) if and only if there is current in the first circuit, whereas $relay2$ makes sure that p is open (false) if there is current in the second circuit. The following state constraints represent the two sub-circuits and the relays:

$$relay1 \leftrightarrow (p \wedge s)$$
$$q \leftrightarrow relay1$$
$$relay2 \leftrightarrow (r \wedge q)$$
$$\neg p \leftrightarrow relay2$$

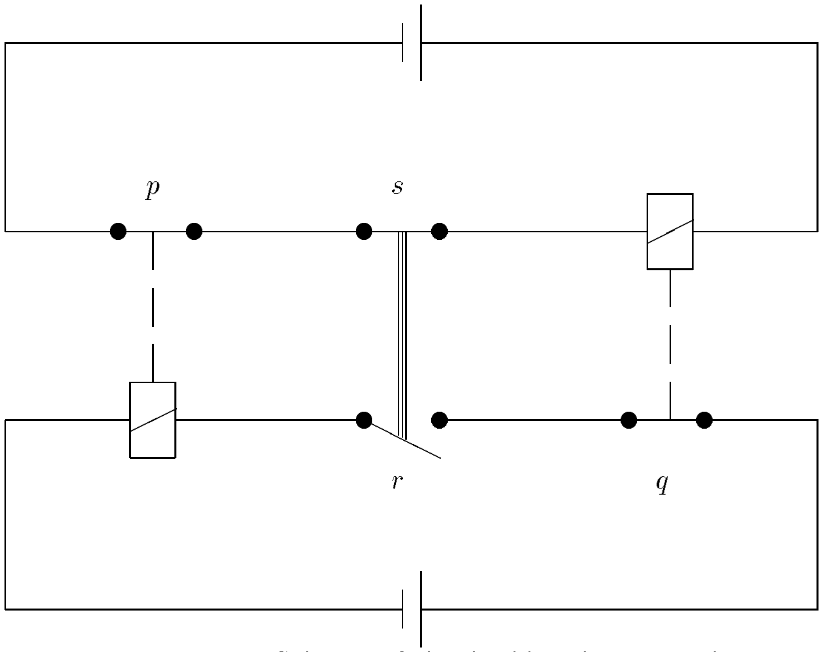

Figure 1: Schema of the double relay example

The corresponding effect rules can be represented (in short notation) as

initiating $p \wedge s$ **causes** $relay1$
initiating $relay1$ **causes** q
initiating $r \wedge q$ **causes** $relay2$
initiating $relay2$ **causes** $\neg p$

and their counterparts

initiating $\neg p$ **causes** $\neg relay1$
initiating $\neg s$ **causes** $\neg relay1$
initiating $\neg relay1$ **causes** $\neg q$
initiating $\neg r$ **causes** $\neg relay2$
initiating $\neg q$ **causes** $\neg relay2$
initiating $\neg relay2$ **causes** p

Now, assume that in the above circuit the switches r and s are mechanically connected, such that always exactly one of them is open. An additional state constraint is then $r \leftrightarrow \neg s$, and we assume that a change of either switch has an immediate impact on the other switch, as indicated by the effect rules

initiating $\neg r$ **causes** s
initiating r **causes** $\neg s$
initiating $\neg s$ **causes** r
initiating s **causes** $\neg r$

Observe that the positive cycles introduced by the last four rules make the completion approach inappropriate, like in the gear wheels example: completion yields $caus(\neg r) \leftrightarrow caus(s)$ and $caus(\neg s) \leftrightarrow caus(r)$, thus allowing for example the initiation of $\neg r$ and s (or $\neg s$ and r) at any time without external cause.

On the other hand, the rules

$$\begin{array}{l}
\textbf{initiating } p \wedge s \textbf{ causes } relay1 \\
\textbf{initiating } relay1 \textbf{ causes } q \\
\textbf{initiating } r \wedge q \textbf{ causes } relay2 \\
\textbf{initiating } relay2 \textbf{ causes } \neg p
\end{array}$$

can be expanded to

$$\begin{array}{l}
caus(relay1) \leftarrow init(p) \wedge init(s) \\
caus(relay1) \leftarrow init(p) \wedge holds(s) \wedge \neg init(\neg s) \\
caus(relay1) \leftarrow holds(p) \wedge \neg init(\neg p) \wedge init(s) \\
caus(q) \leftarrow init(relay1) \\
caus(relay2) \leftarrow init(q) \wedge init(r) \\
caus(relay2) \leftarrow init(q) \wedge holds(r) \wedge \neg init(\neg r) \\
caus(relay2) \leftarrow holds(q) \wedge \neg init(\neg q) \wedge init(r) \\
caus(\neg p) \leftarrow init(relay2)
\end{array}$$

which contains a negative cycle $caus(\neg p) \leftarrow caus(relay2) \leftarrow caus(q) \leftarrow caus(relay1) \leftarrow \neg caus(\neg p)$, i.e. the initiation of $\neg p$ depends on its own absence. The definition is not stratified, but the physical system can be built and will show quite simple behaviour. In particular, due to the construction of the circuit the switch p can never be open (which actually follows already from the state constraints). To see why $\neg p$ can not be initiated, observe that q and r can never be closed at the same time: if r gets closed then s will be opened, so $relay1$ will receive no more current and open q. As a result $relay2$ can never become active, and hence the only rule for initiating $\neg p$ is never applicable.

This example of a circuit which can be realised in practice shows that a syntactic notion of acyclicity is not sufficiently general for dealing with real-world ramifications. It is a fact that the cycles in the above formalisation are linked to some redundancy: there are fluents that are always true, fluents that are equivalent and rules which can never be applied. As in the gear wheels example, the cycles in the definitions can be removed by eliminating this redundancy. However, this is necessarily an ad hoc and non-modular approach. Tuning the representation to make a particular semantics applicable is not the way to go in knowledge representation. Moreover, observe that for some classes of nonstratified definitions (those without *actual* cycles), the simple completion approach still yields intuitive results, again suggesting that there is a more general principle underlying all these different semantics.

5.2 Formal semantics

There appears to exist a semantic notion of acyclicity, which guarantees that rule sets have a sensible and non-ambiguous reading. However, this notion does not appear to be easily captured syntactically. Now, rather than imposing more complex syntactic restrictions, we propose a general semantics which is applicable to all definitions satisfying the basic syntax. More specifically, this semantics should give the unique sensible interpretation to all definitions which have one, i.e. those for which each atom can be assigned a truth value based on a well-founded non-self-supported argument. For other definitions the semantics should indicate that no such unique assignment exists.

To define this more general semantics, we can look at existing work in Logic Programming. There two different semantics have been proposed for dealing with general definitions with negation: the stable and the well-founded semantics, which differ mainly in their treatment of ambiguous definitions. In the stable semantics an ambiguous definition is considered to have multiple models: in the ambiguous definition

$$\begin{array}{lcl} caus(p) & \leftarrow & \neg caus(q) \\ caus(q) & \leftarrow & \neg caus(p) \end{array}$$

either q or p would be initiated nondeterministically; both interpretations are consistent although self-supported through negation. Nonsensical definitions like $caus(p) \leftarrow \neg caus(p)$ have no stable model. The well-founded semantics is strictly constructive and allows no form of self-supported argument, not even through a negative cycle. Every definition has exactly one well-founded model. The semantics is 3-valued, and ambiguous as well as nonsensical definitions are marked by the presence of the third truth value "undefined" ($\mathbf{u}$) in the model: as we will show below, $\mathbf{u}$ is assigned to any atom for which all of the arguments are self-supported through a negative cycle. It has been proven that if the well-founded model is 2-valued, it coincides with the unique stable model of the definition, and for definitions without positive cycles also with the unique model of the completion. For stratified definitions the well-founded model is always 2-valued and coincides with the model defined in IID and the perfect model in LP.

We argue, like in [9], that the underlying principle of the well-founded semantics is the most natural and useful generalisation of the principle of inductive definition. The semantics is constructive, just like we expect physical effect propagations to be. It is not restricted to syntactical subclasses of definitions, but it detects and identifies non-constructive definitions. Observe that, as opposed to the stable semantics, it leaves no room for nondeterminism. This is in our view an important property for sets of rules that are apparently deterministic themselves: nondeterminism should not arise due to tricky interactions of deterministic rules, but should be modelled explicitly when intended. We will introduce nondeterministic effect rules in a later section as a natural extension of the deterministic rules.

Formally, the semantics for stratified definitions is generalised as follows. Since negative literals can occur in rule bodies, and since definitions are in general not stratified so that the definition can not be split up in positive definitions, we must slightly extend the notion of proof tree to allow for negative literals.

Definition 5 *A $\mathcal{D}$-proof-tree $\mathcal{T}$ of $p \in D$ is a tree of literals of D with p as root such that:*

- *all leaves of $\mathcal{T}$ are open literals (possibly $\mathbf{t}, \mathbf{f}$) or defined negative literals;*
- *for each non-leaf node p with set of immediate descendants B, it holds that $p \leftarrow B \in \mathcal{D}$; hence p is a defined symbol in $\mathcal{D}$;*
- *$\mathcal{T}$ contains no infinite branches (hence it is free of positive cycles).*

The truth value of an atom is determined by its best argument, i.e. its best proof tree. Like in the positive case, if any proof tree has only true leaves, the atom is true, and if all proof trees have a false leaf, the atom is false. However, for a nonstratified definition it can not be guaranteed that for a particular atom the truth values of all proof trees are known when one tries to determine the atom's truth. This can be solved by using the third truth value $\mathbf{u}$ to designate atoms of which the truth could not or has not yet been determined. We define $\mathbf{u}^{-1} = \mathbf{u}$. Clearly, initially all atoms are assigned $\mathbf{u}$. The notion of best argument can then be generalised as follows to 3-valued valuations.

Definition 6 ($\mathcal{V}_{\mathbf{P}}$,$\leq_F$) *Given a set of defined 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 $\forall l \in \mathbf{P} : I(l) \leq_F I(l')$. We use the symbol $\perp$ to denote the valuation which assigns $\mathbf{u}$ to all defined atoms.*

Given a valuation $I \in \mathcal{V}_{\mathbf{P}}$ and an interpretation O of the open atoms (i.e. in this paper the *act* and *holds* atoms, so $O = M_{S,A}$), for each $l \in \mathbf{P}$ we define its *supported value* w.r.t. I and O, denoted $SV_{I,O}(l)$, as the truth value proven by its best proof tree, i.e.

Definition 7 (supported value)

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

We now define the following operator which maps each atom to its supported value.

Definition 8 ($\mathcal{PI}_{\mathcal{D},O}$) *The* induction operator $\mathcal{PI}_{\mathcal{D},O} : \mathcal{V}_{\mathbf{P}} \rightarrow \mathcal{V}_{\mathbf{P}} : I \rightarrow I'$ *is defined such that* $\forall p \in \mathbf{P} : I'(p) = SV_{I,O}(p)$.

It can be proven that this operator is monotonic and hence always has a least fixpoint $\mathcal{PI}_{\mathcal{D},O}\!\uparrow$ for a given interpretation O of the open atoms. This allows us to define the model of a general definition $\mathcal{D}$ w.r.t. O, denoted $I_{\mathcal{D},O}$ as:

Definition 9 *Given* $< \mathbf{P}, \mathcal{D}, O >$, $I_{\mathcal{D},O} = \mathcal{PI}_{\mathcal{D},O}\!\uparrow$.

$I_{\Pi_e, M_{S,A}}$ is now the canonical model used to define the update sets and transition function given a general causal theory Π_e, as in section 2.

The generalised induction operator essentially mimics the truth assignment in one layer of a stratified definition: the atoms to which **t** or **f** have already been assigned are the atoms in lower layers. The atoms in higher layers still have the value **u**. Layer by layer the **u** values are replaced by **t** or **f**. In this sense, the operator can be interpreted as a more general implementation of the principle adhered to in stratified definitions: the partial interpretations generated by consecutive applications of the IID operator are now represented as full 3-valued interpretations where **u** indicates the atoms that have not yet been assigned a "real" truth value. Unsurprisingly for a stratified definition the least fixpoint of the generalised induction operator is then 2-valued and corresponds to the model defined using the IID operator. More in particular, observe that for a positive definition $\mathcal{D}$ and an arbitrary interpretation O of *holds* and *act* atoms, $SV_{\perp,O}(p)$ is 2-valued for all p and corresponds exactly to the model of the positive induction semantics. The results are formally summarised in section 5.5.

In definitions incorporating actual negative cycles[17], e.g.

$$\begin{array}{l} caus(p) \leftarrow \neg caus(q) \\ caus(q) \leftarrow \neg caus(p) \end{array}$$

both $caus(p)$ and $caus(q)$ retain the value **u**, as neither **u** can be eliminated until after the other one is removed. The resulting model contains **u**, which indicates that some atoms could not be given a real truth value or in other words that the definition is not a good constructive definition, i.e. it is either ambiguous or nonsensical. However, if the cycle is only apparent and not actual, as in the non-stratified definition

$$\begin{array}{l} caus(p) \leftarrow \neg caus(q), caus(r) \\ caus(q) \leftarrow \neg caus(p) \end{array}$$

we obtain a unique model in which $caus(r)$ and hence $caus(p)$ are false (the only proof tree for $caus(p)$ has a false leaf once $caus(r)$ is

[17] Observe that for this case we do not give a meaningful "real-world" example: it is our claim that such examples do not exist.

determined), and $caus(q)$ is true. So, the semantics detects actual cycles and does not forbid syntactical ones which are not problematic.

In [9], the least fixpoint of the induction operator has been proven to coincide with the least fixpoint of the well-founded operator of [40]. This is an argument for the position that the well-founded semantics represents a generalised inductive definition principle. The interpretation of logic programming under well-founded semantics as an inductive definition logic deviates considerably from other well-known knowledge theoretic interpretations given by the embeddings of logic programming in default and autoepistemic logic, for example in that in the inductive definition reading, negation is objective, whereas in the default and autoepistemic readings negation is seen as a modal operator. Our treatment of effect rules in this paper and the motivations given for it are therefore also an indication that this alternative perspective on logic programming provides a promising framework for general knowledge representation, which formalises important KR principles. However this issue is outside the scope of this paper.

5.3 Higher level effect rules

Above we have defined a semantics for sets of rules involving only primitive *caus* literals, plus *holds* and *act* literals which can be evaluated immediately in any $(state, action)$ pair. These rules form a very low-level language. Observe that in our examples, we have already introduced some syntactic sugar by using the predicate *init*, where $init(p)$ stands for $caus(p) \wedge \neg holds(p)$. This leads to (slightly) more intuitive rules, as we usually expect only actual changes to be a reason for further propagation. Another simplification is that we allow the parameter of *holds* to be a complex fluent formula, so that we can abbreviate e.g. $holds(a) \wedge \neg holds(b)$ to $holds(a \wedge \neg b)$. These notations have no practical implications.

A more substantial difference is due to the observation that in all examples we found in the literature, the primitive effect rules only occurred in particular forms and combinations, corresponding to one specific form of higher level, complex effect rules.

A first observation is that an effect is never triggered by the absence of other effects alone, e.g. the suitcase will not open just because nothing happens (which would be a serious violation of inertia). Each effect requires at least one other actual effect or action (e.g. the opening of a lock) as a cause, and the absence of other effects (e.g. the not simultaneously closing of the other lock), is only relevant when these other effects may counteract the causing effect.

A second observation, e.g. in applications like the suitcase example and the interlinked relays example, is that the interplay between effects and counter-effects reflects that the change in truth value of one particular combination of fluents (a fluent formula) is the actual cause of other changes: the suitcase opens as soon as it is no longer restrained, i.e. as soon as both locks are open. A relay is activated when it receives current, i.e. as soon as all the switches in its circuit become closed (no matter if they are closed at the same time or if the last one is closed when the others already were). This kind of examples is very common and can be modelled compactly by introducing complex initiations in effect rules.

Finally, effects or their propagation may be context dependent: in the example with the digital counter, the change of an output voltage triggers a new counter value which depends on the value at the time of change. In the Yale Shooting Problem the *shoot* action triggers the termination of *alive* if *loaded* holds.

Given these observations, we propose a high-level form of derived effect rules which represents that a change in one particular complex fluent formula, possibly in a particular context, is the cause of a new effect. Syntactically, as defined in section 2:

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

with l a fluent literal and F and F' complex fluent formulae. These rules can be read as

$$init(l) \leftarrow init(F'), holds(F).$$

In addition we use direct effect rules of the form

$$a \textbf{ causes } l \textbf{ if } F$$

with a an action constant and l and F as before, which represent

$$init(l) \leftarrow act(a), holds(F).$$

Observe that the direct effects of individual actions are represented independently, which is important for the modularity of the approach. Complex derived effect rules provide a compact and elegant way of describing the interaction of simultaneous actions. A nice example is the following, taken from [11]: a table is considered which can be lifted on the left and right sides; on top is a glass of water which will spill its contents as soon as the table is in a non-horizontal position.

The effects of the actions *lift_l* and *lift_r* (and similarly *drop_l* and *drop_r*) are represented by the direct effect rules

$$\begin{array}{c} lift\_l \textbf{ causes } up_l \textbf{ if } true \\ lift\_r \textbf{ causes } up_r \textbf{ if } true \end{array}$$

The rule that the glass spills its contents when the table is moved into a non-horizontal position is expressed through

$$\textbf{initiating } \neg(up_l \leftrightarrow up_r) \textbf{ causes } spilled \textbf{ if } on\_table$$

Assuming the table is initially on the floor, executing either one of the *lift* actions will cause the water to spill, but if they are executed at the same time there is no spilling.

We define a semantics for rules involving complex initiations by defining the set of primitive rules which is equivalent to them. This is a rather straightforward transformation, of which an example has already been given for the suitcase rule in section 2. We base the formal transformation on the notion of supporting sets: a supporting set of a complex fluent formula F is a consistent set of fluent literals which entails F. Formally this is defined as follows (we use the notation $\overline{l}$ for negation, i.e. for a fluent f, $\overline{f} = \neg f$ and $\overline{\neg f} = f$).

Definition 10 *Given a set of fluent literals L and a complex fluent formula F, we define when L is a supporting set of F (denoted $L\ sup\ F$) inductively as follows:*

- $L\ sup l$ *if* $l \in L$
- $L\ sup F_1 \wedge F_2$ *if* $L\ sup\ F_1\ \wedge\ L\ sup\ F_2$
- $L\ sup F_1 \vee F_2$ *if* $L\ sup\ F_1\ \vee\ L\ sup\ F_2$
- $L\ sup \neg F$ *if* $\forall L'\ sup\ F : \exists l \in L' : \overline{l} \in L$

L is a consistent supporting set of F, denoted $L\ sup_c\ F$, iff $L\ sup\ F$ and L does not contain a pair $\{f, \neg f\}$ for any f.

This definition — and thus the semantics defined below — can easily be extended to the predicate case, but such an extension is outside the scope of this paper. Only consistent supporting sets are of interest to us.

It can be shown that two formulae have the same supporting sets if and only if they are equivalent under 3-valued FOL semantics. This also implies that formulae which are equivalent under 2-valued FOL semantics do not have the same supporting sets (often not even the same consistent ones) if they are not 3-valued equivalent: e.g. given two fluents p and q, p has three consistent supporting sets $\{p\}, \{p, q\}$ and $\{p, \neg q\}$, whereas $(p \wedge q) \vee (p \wedge \neg q)$ has only the consistent supporting sets $\{p, q\}$ and $\{p, \neg q\}$.

A formula is true if and only if all literals of some (necessarily consistent) supporting set of it are true. It follows that F is *initiated* iff F is not already true and for some consistent 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. Based on this idea, we define the *grounding* of a complex derived effect rule, i.e. the set of primitive rules which is equivalent to it, as follows. We define $\mathbf{Caus}(L)$ as $\{\mathbf{Caus}(l) \mid l \in L\}$ and $\overline{L} = \{\overline{l} \mid l \in L\}$ for any set of literals L, and $\overline{P} = \{\neg p \mid p \in P\}$ for any set of **Caus** atoms P.

Definition 11 (grounding of effect rules)
*The grounding of a rule "a **causes** l **if** F" is the singleton*

$$\{caus(l) \leftarrow act(a), holds(F)\}$$

*The grounding of a rule "***initiating** F **causes** l **if** F'*" is the set*[18]

$$\{caus(l) \leftarrow caus(L_i), holds(L_p), \overline{caus(\overline{L_p})}, holds(\neg F), holds(F') \\ \mid L_i \cup L_p \ sup_c \ F \ \wedge \ L_i \neq \emptyset\}.$$

The grounding of a set of effect rules Π_e *is the union of the groundings of all rules in* Π_e*.*

In the propositional case we consider here, it can easily be proven that the above definition is equivalent to the one obtained by only considering minimal supporting sets (those that are not supersets of another supporting set), i.e. by imposing that $L_i \cup L_p$ is a minimal supporting set of F. This follows from the fact that the extra rules generated by nonminimal supporting sets lead to additional proof trees for literals, which are strict supersets of already existing proof trees for the same literal (generated by the corresponding minimal supporting set). Since the truth value of a proof tree is determined by its worst branch, the truth value of the smaller tree is never worse than the one of the larger tree. Hence the supported value of each literal in each iteration is the same, and the fixpoint is the same.

As an example, the formula $\neg(up_l \leftrightarrow up_r)$ has two minimal supporting sets $\{up_l, \neg up_r\}$ and $\{up_r, \neg up_l\}$. Hence, the grounding of

$$\textbf{initiating } \neg(up_l \leftrightarrow up_r) \textbf{ causes } spilled \textbf{ if } on\_table$$

is

$$\begin{array}{lcl}
caus(spilled) & \leftarrow & caus(up_l), holds(\neg up_r), \neg caus(up_r), \\
 & & holds(up_l \leftrightarrow up_r), holds(on_table). \\
caus(spilled) & \leftarrow & caus(up_r), holds(\neg up_l), \neg caus(up_l), \\
 & & holds(up_l \leftrightarrow up_r), holds(on_table). \\
caus(spilled) & \leftarrow & caus(\neg up_l), holds(up_r), \neg caus(\neg up_r), \\
 & & holds(up_l \leftrightarrow up_r), holds(on_table). \\
caus(spilled) & \leftarrow & caus(\neg up_r), holds(up_l), \neg caus(\neg up_l), \\
 & & holds(up_l \leftrightarrow up_r), holds(on_table). \\
caus(spilled) & \leftarrow & caus(up_l), caus(\neg up_r), \\
 & & holds(up_l \leftrightarrow up_r), holds(on_table). \\
caus(spilled) & \leftarrow & caus(\neg up_l), caus(up_r), \\
 & & holds(up_l \leftrightarrow up_r), holds(on_table).
\end{array}$$

which describes all the cases in which the water will be spilled. Observe that the last two rules will always have false bodies if we impose that a side can be lifted only if it is on the floor and dropped only if it is in the air.

[18] Rules in which $L_i = \emptyset$ have a necessarily false body since $holds(L_p), holds(\neg F)$ is unsatisfiable if L_p is a supporting set of F; hence these rules are not taken into account.

5.4 State constraints and preconditions

The question which remains to be answered is what the role of state constraints and explicit preconditions is in our proposal. These are completely orthogonal to the successor state calculation. They are interpreted as classical FOL formulae, in that any model of the theory must satisfy all preconditions and constraints. So, actions can not occur when their preconditions are not met. Likewise, if the successor state of a certain state after a particular action — which is determined by the effect rules alone — violates a state constraint, this successor state can not exist at any time and hence any action leading to it is impossible. In this way state constraints function as implicit action preconditions.

Observe that — if given — specific scenario information like action ocurrences and observed fluent values at particular times can be handled in the same way: as a FOL theory which restricts the possible scenarios. In this paper we do not discuss scenario information in depth as it is more dependent on the used topology of time and orthogonal to the transition function and the ramification problem.

5.5 Results

We now give some results concerning the semantics. For proofs we refer to [38].

Theorem 1 *Given a state S, a set of actions A and a set of direct and derived effect rules Π_e, the truth values of $caus(l)$ for all l are uniquely determined.*

The theorem guarantees that successor states generated by a set of deterministic effect rules are unique. In this respect our approach differs from the one in [35]. Unless nondeterminism is explicitly introduced, our theory leaves no room for ambiguity.

The following results give some alternative characterisations of the above semantics in special cases. They shed some light on the relation to existing approaches to the ramification problem.

Definition 12 (fluent dependency)
A fluent f occurring in a rule **initiating** F **causes** f **if** F' *or in a rule* **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 2 *If the derived effect rules are acyclic, i.e. if no fluent depends on itself, $I_{\mathcal{D},O}$ is always 2-valued. Moreover $I_{\mathcal{D},O}$ coincides with the unique model of the Clark completion of the definition rules.*

Theorem 3 *If the body of each derived effect rule is a single literal, $I_{\mathcal{D},O}$ is always 2-valued, and coincides with the unique model of the parallel circumscription of init and caus in the theory consisting of the definition rules read as implications.*

The above results formally show that for several classes of definitions for which other semantics are known to assign the intended meaning to all predicates, the generalised inductive definition semantics coincides with these semantics. However the generalised inductive definition semantics is more general, also dealing with cyclic definitions and with negations, even for nonstratified rule sets. Since for example completion and circumscription have often been used in some form or other in approaches to the ramification problem, our proposal can be seen as a high-level unification and generalisation of a number of existing approaches to this problem. Further on we will study some specific approaches at a more detailed level.

6 Delays

The semantics described in section 5.2 deals in a specific way with the problematic definitions where multiple interacting changes arise. In particular, it is adequate under the assumption that direct and indirect effects of an action, or a set of concurrent actions, occur with no delay, that is, the delays in the effect propagation occurring in the actual physical system are abstracted to zero.

Abstracting small delays is often convenient for several possible reasons:

- because the resulting model is simpler;
- because there is no actual knowledge (on the delays in this case) for providing an accurate model.

An abstraction is harmless when it does not affect the results we are interested in. The problem is to find out when this is the case for the zero-delays abstraction.

Consider the suitcase example and suppose we start with the state where the first lock is closed ($\neg lock1open$) and the second one is open ($lock2open$). As already discussed, to open the suitcase it is not only required that the first lock is opened, but also that the second one is not at the same time closed. Therefore if we concurrently switch the two locks, i.e. open the first one and close the second one, the intended conclusion is that the suitcase does not open. Intuitively, this is because in our semantics the rule

$$caus(open) \leftarrow init(lock1open) \wedge holds(lock2open) \wedge \neg init(\neg lock2open)$$

can only be used after $\neg init(\neg lock2open)$ has been established for the given batch of effects, i.e. when it is certain that the second lock will remain open. Since in the example $init(\neg lock2open)$ ultimately acquires a $true$ value, the rule cannot be used and $caus(open)$ remains $false$.

The above conclusion in the suitcase example is the intended one if there is no delay between various effects, or if the propagation to $open$

is slower than the immediate effect on $\neg lock2open$. In a real suitcase, the delays are of course non-zero. Which effects and propagations are faster, depends on the actual construction of the lock system and the precision of timing of the "simultaneous" actions. If the opening of the suitcase itself is rather slow, chances are good that it will remain closed if the actions are reasonably simultaneous.

The same reasoning can of course be applied to indirect effects: a device could be constructed — for whatever obscure reason — to simultaneously switch the two locks if a certain button is pushed, so that the effects on the locks are ramifications of another action. In this case, the effect on the suitcase depends on the relation between the propagation delays in this device and in the lock system. In an actual suitcase the result of pushing the button may always be that the suitcase opens, or always that it remains closed, or even that it behaves differently in different cases (if the delays are not constant).

A sufficient condition for the zero-delays abstraction to be safe is that in a stratified definition, effects in a lower layer occur before effects in higher layers. In fact, effects in a layer are computed by the semantics given the complete set of changes occurring in lower layers; such computation corresponds to the actual propagation of effects if the changes in the lower layers are already completely determined. In particular, observe that for a positive definition like the ones considered in section 4, the abstraction is always safe since there is a single layer.

When the behaviour of the actual system is dependent on knowledge that is not in the model, we can of course not expect any semantics to compensate for this. Any incompleteness in the knowledge should be dealt with in the representation. How this is to be done depends on the reason for the incompleteness:

- if relevant but existing knowledge has been abstracted away, it should be included in the model;
- if relevant knowledge is actually unavailable, the incompleteness must be explicit in the model, making it possible to derive the alternative possible evolutions of the system.

When the missing knowledge is knowledge about delays, the way to include it or to make the incompleteness explicit depends on the specific adopted time structure. An explicit representation of delays is therefore not discussed in this paper. However, in [38] we illustrate how to integrate delays with the contributions in this paper, assuming a linear time structure.

To deal with missing information (not necessarily only about delays) we can explicitly introduce actions with nondeterministic effects. This is discussed in the next section.

7 Nondeterminism

In this section we introduce a representation of nondeterministic actions and ramifications of a specific (limited) form, in which we can represent that some cause *may* have a certain effect. This extension helps in representing some forms of incomplete knowledge about the direct and indirect effects of actions, and is in addition useful for comparing our approach with Thielscher's further on.

The syntax of *nondeterministic direct effect rules* is

$$a \textbf{ possibly causes } l \textbf{ if } F$$

The meaning, for example, of

$$shoot \textbf{ possibly causes } \neg alive \textbf{ if } loaded$$

is that $\neg alive$ is a possible but not certain effect of the action, in other words here the hunter is less of an expert than in the usual formalisation of this problem, or we do not assume to have complete knowledge of the conditions under which the action is effective.

Similarly we extend derived effect rules: a *nondeterministic derived effect rule* has the form

$$\textbf{initiating } F \textbf{ possibly causes } l \textbf{ if } F'$$

meaning that l may become true when F becomes true at a time when F' holds.

The meaning of a set of rules which includes nondeterministic ones, is defined by extending the definition of grounding as follows: a nondeterministic rule has two groundings: the grounding of the corresponding deterministic rule (corresponding to the actual occurrence of the possible effect), and the empty grounding (corresponding to the effect's absence). Formally:

Definition 13 (nondeterministic grounding)

A grounding of a nondeterministic direct effect rule

$$a \textbf{ possibly causes } l \textbf{ if } F$$

is either the grounding of the direct effect rule

$$a \textbf{ causes } l \textbf{ if } F$$

or the empty set.

A grounding of a nondeterministic derived effect rule

$$\textbf{initiating } F \textbf{ possibly causes } l \textbf{ if } F'$$

is either the grounding of the derived effect rule

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

or the empty set.

A grounding of an effect theory $\Pi_e = \{r_k \mid 1 \leq k \leq l\}$ *is any set* $\bigcup_{k=1\ldots l} G_k$ *with each* G_k *a grounding of* r_k*.*

The semantics of a set Π_e of possibly nondeterministic rules is then defined applying the definitions in section 5.2 to any nondeterministic grounding Π'_e of Π_e. This leads to defining a set of possible models, rather than a canonical interpretation, and then to defining a transition *relation*, rather than function. We do not give these definitions explicitly.

Our rules for nondeterministic actions are similar to those in [14], although the semantics is defined in a different way and we also allow for nondeterministic ramifications. As we indicated, only a limited form of nondeterminism can be represented using these rules. A full treatment of nondeterminism, taking into account for example actions with alternative effects can be obtained by extending the principle used in the above semantics (defining a grounding for each possible effect), but is outside the scope of this paper. For more details, an extended motivation of the given formalisation and an integration of the approach in a specific time structure, we again refer to [38].

8 Nondeterminism and delays with respect to Thielscher's approach

The approach in [35] is also based on causal rules (syntactically uncoupled from state constraints; Thielscher also discusses a way of deriving causal rules from state constraints given additional influence information). Our approach however significantly departs from Thielscher's as regards the view on modeling systems with nondeterministic behaviour and delayed effects, which has considerable implications for the semantics and the theories that can be represented.

A first essential difference is that in Thielscher's approach *all* changes are optional: Thielscher's causal rules of the form

$$a \text{ causes } b$$

are exactly equivalent to our rules

$$a \textbf{ possibly causes } b \textbf{ if } true$$

as proven in [38]. The relation with rules a causes b if c is slightly more complex: as we shall see below, these rules are strictly stronger, i.e. applicable in more cases, than our a **possibly causes** b **if** c. However also the application of these rules is always optional.

Hence, any state obtained after applying any subset of applicable causal rules, and satisfying the state constraints, is a valid successor state according to Thielscher. As a result, effect propagations not related to state constraints are not correctly dealt with. A simple example is the digital network discussed before: assuming the presence of a causal rule like

$$out \text{ causes } count(n+1) \text{ if } count(n)$$

in Thielscher's approach, in the absence of a state constraint there is nothing enforcing application of this rule.

The second difference lies in the interpretation of the *if* condition in Thielscher's rules, as mentioned above. Thielscher proposes to apply causal rules sequentially, i.e. the if condition can be checked not only at the beginning of a batch of changes, but also after some changes have already taken place. It is due to the sequential application of rules that a causes b if c is strictly stronger than our non-deterministic rule: our corresponding rules can only be applied if the condition c holds in the starting state.

Thielscher's approach suggests that small delays can and do occur in change propagations, and can be relevant to them. On the other hand, they are not represented explicitly and their actual duration is abstracted away. This dual view is in some cases problematic.

One problematic case is the one where multiple *interacting* changes occur in the same batch of effects (due to multiple actions or to multiple effects of one action): if in the actual system the end result is dependent on the small delays involved in the effect propagations, the approach in [35] suggests that the semantics should provide alternative results, corresponding to all different orders of application of causal rules. Of course this is only correct if all orders correspond to a possible order of events in the real system: otherwise the intended set of possible results does not coincide with the set of possible results according to the semantics. In particular, if in the suitcase example an open lock and a closed lock *are* actually switched at the same time, or in the glass example both sides of the table are lifted at the same time, the sequential application of rules, in either order, does not give rise to the intended result. It may be obvious that a sequential semantics is not necessarily adequate for concurrent actions, but the problem also exists in the modified examples where the changes to openness of locks and heigth of the table sides occur as ramifications of a single action.

A variant of the above problem is especially apparent in the set of context-dependent rules

$$out \text{ causes } count(n+1) \text{ if } count(n)$$

given before. Here the partial abstraction of the delays and the sequential rule application in Thielscher's approach can give rise to an arbitrary increase of n: when *out* is initiated, the counter can take on any value, since as soon as a new value of *count* is reached, the next rule in the sequence can fire, ad infinitum. In the real system (if built correctly), the delays are such that only one increase will occur: the change in *out* triggers a change in *count*, and there it ends. The problem seems to be that there is only one triggering effect *out*, but due to the mixed view on delays it can be seen as occurring both before and after the effects (the change of *count*) it gives rise. Intuitively, it can take its own output as input, leading to all kinds of strange results. If the delays were represented explicitly (and correctly), only a unit increase would be inferred. If the delays are abstracted away entirely such that one set of simultaneous changes is imposed (as in our approach), there is also only a unit increase.

The above discussion motivates our preference for an approach where either delays are abstracted away entirely (which is safe for positive definitions or for delays respecting the strata of the definition), or where the knowledge about the delays is included, using explicit delayed effect rules or — in the absence of complete information — explicit nondeterminism.

Similar considerations apply to the approach in [4] where *implicit* as well as explicit nondeterminism is used for specifying the direct effects of concurrent actions. In that case, given a set of concurrent actions, if the union of the effects of single actions includes contradictory effects, e.g. f and $\neg f$, then f and $\neg f$ are interpreted as alternative effects of the set of actions. It turns out in that in this way modelling through explicit nondeterminism, with "alternatively causes" rules, or with the corresponding "causes" rules, interpreted with implicit nondeterminism, can give rise to the same interpretation. The authors of [4] comment that this is no problem, since the result of reasoning should be independent of whether the designer of the domain description is or is not aware of the uncertainty. We prefer to regard the case of contradictory effects as a bad definition, and encourage the representation of explicit nondeterminism together with appropriate modelling of the interrelations among fluents and their changes, possibly using causal rules with initiation of complex formulae.

9 Related work

Above we have already compared our work with the approach in [35], which is most similar to ours. For this reason we refer to [35] for a comparison with approaches not based on causal laws (e.g. categorisation based approaches like [19], [20], [5]): with respect to those approaches our proposal and Thielscher's are very similar. 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 it abstracts away delays at a macroscopic level but retains them at a microscopic level, whereas we argue to abstract them away entirely or represent them explicitly. Due to the former difference the full effect of syntactically uncoupling causal laws from state constraints is partially lost in [35]. For example, no state constraint independent effect propagations can be represented. However, the fact that influence information is represented independent of the state constraints is an important asset which also makes Thielscher's approach very appropriate for analysing other proposals using causal laws.

The approach to ramifications in the $\mathcal{E}$ language [17] 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.

In [23] the need for causal laws is clearly motivated and 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 [21] 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 our complex derived effect rules. However, the causal rules differ from ours 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 [15] contains expressions of the form $[t]\delta >> [s]\gamma$ 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 $[t]\delta >> [s]\gamma$ 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 $holds(\delta, t) \rightarrow holds(\gamma, s)$ and that if δ is 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, which is a considerable addition to the state of the art. As in the previously discussed approaches, however, we find that the formulae $[t]\delta >> [s]\gamma$ always entail the corresponding general constraint $holds(\delta, t) \rightarrow holds(\gamma, s)$, and hence in the case of immediate ramifications ($t = s$) they also entail the corresponding state constraint. Change propagation unrelated to a state or general constraint is also in this approach excluded. Finally, also in this approach cyclic dependencies in causal laws are not dealt with correctly, as indicated by the authors. As mentioned before, in our embedding of the approach to ramifications in a linear time structure in [38], we also deal with delayed effects of actions and delayed ramifications, as well as with general constraints like the aforementioned $holds(\delta, t) \rightarrow holds(\gamma, s)$. There is some similarity with the [15] approach, as indicated in the report.

As already mentioned, in [26] McIlraith presents an extension of the explanatory closure approach in [31] to the ramification problem, and in [25] provides an equivalent characterization in terms of prioritized circumscription. An acyclicity condition is imposed, defining "solitary stratified theories", which is actually a stronger condition than the usual stratification one, so that positive recursion as in our section 4 (e.g. the gear wheels example) is not allowed[19]. This approach is additional evidence that different approaches in the literature, such as syntactical transformations (completion, explanatory closure) and circumscription can be seen as instances of the same idea under different syntactic restriction.

In [28], Pinto investigates interacting effects of concurrent actions in the context of situation calculus, in the style of Reiter's axiomatization ([31]). He distinguishes between effect *synergy*, when concurrent actions have effects which are not present when the individual actiosn are performed individually, and effect *cancellation*, when effects of individual actions are deleted in the present of other concurrent actions. Pinto shows that a minimisation policy cannot properly model the intended semantics of a ramification rule in the presence of simultaneous actions. This is illustrated in the "glass on the table" example (although the author has preferred to use a soup bowl). He proposes a solution in which he introduces an intermediate natural *spilling* action [20], which is caused to happen with a delay if the table is lifted at one of its sides; it is this spilling action that causes *spilled* to become true.

Pinto also suggests an alternative approach based on causal rules which are intuitively and syntactically similar to our derived effect rules. Pinto shows how such causal rules should be translated in his framework using delayed actions. In our approach, these causal rules can be represented directly in the formalism. It would be interesting to further investigate the relation between both approaches in the future.

In [3], Baral and Gelfond define the language $\mathcal{A}_c$ to extend $\mathcal{A}$ with action qualifications and concurrent actions. The semantics of $\mathcal{A}_c$ is defined through a transition function, which maps tuples consisting

[19] From personal communication from the author of [26], the range of application of the approach seems wider than solitary stratified theories.

[20] Using the formalisation of natural actions of [32].

of a state and a *set* of concurrent actions to a new state; our transition function is of a similar sort. [3] expresses the effects in the soup bowl example as follows:

$$lift_l \textbf{ causes } spilled$$
$$lift_r \textbf{ causes } spilled$$
$$\{left_l, lift_r\} \textbf{ causes } \neg spilled \textbf{ if } \neg spilled$$

For a set of concurrent actions including $lift_l, lift_r$, the third rule overrides the two effect rules of the individual actions $lift_r$ and $lift_r$. The formalism of [3] is not designed to describe ramifications of actions.

At present, our language does not allow to directly express concurrent action effect rules of the type used in [3], although it is straightforward to integrate both formalisms. However, observe that in many applications (e.g. in the suitcase and the soup bowl example), our effect theories correctly model concurrent actions. This is because our theories aim to provide a precise account of the causal relationships in the dynamic system and thus lead to a more modular and concise representation. For example, the real cause of spilling is the fact that the table and hence the soup bowl is not horizontal. This is modelled directly by the derived effect rule

$$\textbf{initiating } up_l \leftrightarrow \neg up_r \textbf{ causes } spilled$$

In contrast, the $\mathcal{A}_c$ description requires 3 rules for modelling the effects of lifting actions on *spilling*. If there would be more types of actions which lift the table at one of its sides, then the number of concurrent action effect rules would increase dramatically, while the effect of all combinations of concurrent lifting actions on *spilled* is still correctly covered by the unique derived effect rule.

An interesting issue concerning ramifications was brought up in [33]. There 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.

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, and as such the issue is indeed problematic in our approach. 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; but in our proposal direct effect rules are the only constructs representing action laws. Hence, we cannot deal with the specification as given. Basically, our approach requires a much stricter representation methodology. In other words, we must determine precisely at what level of abstraction we are reasoning. One option is to consider the switches as nothing but operational details. In that case we can abstract them 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.

Alternatively, it may occur that the position of the switches is relevant in the domain, but that in a particular application we are 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. In that case, our approach requires that a strict distinction is made between representation and reasoning. The position of the switches is relevant, so we should model the domain using the usual direct and derived effect rules about switches and lamps. In the application at hand, we are not interested in the switches: it is only relevant that after a particular action or plan the lamp is on. Such a condition should be imposed explicitly (e.g. with a simple FOL axiom). It is then the task of the agent to find the primitive actions yielding the intended effect on the lamp (closing either of the switches), which it can achieve for example by abductive planning.[21] This approach adequately tackles the problem provided that the right level of abstraction is used in the representation. To arrive at this right level of abstraction, it would of course be useful to have appropriate tools available in the preceding analysis phase, just like the use of influence information for deriving effect rules from state constraints can be an important asset. The problem of correctly relating switches to lamps can be an issue in deriving the right representation-level effect rules. But in general, we argue that it is too complicated and not necessary for a representation language to deal with deriving causes from their effects, as this is a typical (abductive or — if the action law needs to be derived explicitly — inductive) reasoning task, to be dealt with either beforehand to arrive at the right representation, or after the representation is given to arrive at the correct conclusions, or most likely both.

10 Conclusion

We have presented a constructive, language-independent approach to the ramification problem. This approach contains several contributions:

[21] An alternative but equivalent view is that the turning on of the lamp is a macro-action 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.

- A complete uncoupling of ramifications from state constraints, so that a more general class of indirect effects can be tackled.
- A semantics able to deal with a very broad class of effect theories, involving negative and cyclic dependencies between causes, and generalising well-known principles like circumscription and completion.
- The introduction of a class of high-level rules which allow to write compact and modular effect theories.

In addition we have shown how to introduce a limited form of non-determinism in the above approach.

The semantics of an effect theory Π_e is defined by the function $Trans_{\Pi_e}$. As such, our approach is language-independent and can be easily integrated in various formalisms.

- Our approach can be straightforwardly integrated in the language $\mathcal{A}$ and would result in a language with simultaneous actions and with ramifications. Our effect theories extend the simple e-rules of $\mathcal{A}$. Our transition function $Trans_{\Pi_e}$ extends the transition function of $\mathcal{A}$ by allowing simultaneous actions rather than a single action.
- In a branching time calculus such as situation calculus, our effect theories could be integrated by imposing the following semantical condition on each model I, for each situation s and set of actions $a_1, .., a_n$:

  $$\begin{array}{c}\{f \mid I \models holds(f, do(a_1 + .. + a_n, s))\} = \\ Trans_{\Pi_e}(\{f|I \models holds(f, s)\}, \{a_1, ..a_n\})\end{array}$$

  Here, $do(a_1 + .. + a_n, s)$ denotes the situation resulting from simultaneous applications of $a_1, .., a_n$ to situation s, as in [11].
- In a linear time calculus such as event calculus, the formalism contains predicates *initiates* and *terminates*, representing the fact that at some time point some fluent atom is either initiated or terminated. Our formalism can be integrated by imposing the following condition on a model I of an event calculus: for each time point t in I, it should hold that:

  $$\begin{array}{c}\{caus(f) \mid I \models initiates(t, f)\} \cup \\ \{caus(\neg f) \mid I \models terminates(t, f)\} \\ = U_{\Pi_e, S, A}\end{array}$$

  with $S = \{f \mid I \models holds\_at(f, t)\}$, $A = \{a \mid I \models happens(a, t)\}$
- In [38] we have embedded the approach in a high-level language $\mathcal{ER}$, based on a linear time theory. In this language we integrate the above approach to ramifications with a more general

treatment of nondeterminism and with rules describing delayed effects, as well as with a general language suited for representing uncertainty about action occurrences and for example the initial state, and for representing observations at various time points.

In our proposal, the treatment of state constraints and action preconditions is independent of the successor state calculation. So, actions or simultaneous occurrences of actions can not occur when their preconditions are not met. Likewise, if a successor state S' of a certain state S after a particular set A of actions — would violate a state constraint, then this state constraint acts as an implicit action precondition forbidding the simultaneous occurrence A of actions in state S.

The fact that $Trans_{\prod_e}$ may be only partially defined may pose problems. Recall that the problem is due to the generality of our formalism, in which one can write nonsensical rules such as **initiating** f **causes** $\neg f$. Our view is that when $Trans_{\prod_e}(S, A)$ is undefined (because there occurs a **u** or an inconsistent set of initiations, the temporal theory should not allow $do(A, S)$ to be a legal state, e.g. the set of actions A should not be allowed to occur in the state S. This imposes a correctness criterion on the temporal theory. For example, in a theory containing the gear wheel rules, an action of starting the gear wheels and stopping the gear wheels may not occur simultaneously.

In case derived effect rules are related to state constraints, as in most examples seen in the literature, ways to derive them automatically from these state constraints can be developed similar to the method using influence information proposed in [35]. In [38] we show how to achieve this in the context of $\mathcal{ER}$, and we compare this approach with Thielscher's.

The theory we have presented and its embedding in $\mathcal{ER}$ have been mapped to OLP (open logic programming), an extension of logic programming with "open" predicates, i.e. predicates without a definition. An abductive resolution procedure supports reasoning — as well deductive as abductive and combinations of them — on the resulting theories, so that they can be used in practice.

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] K.R. Apt and M. Bezem. Acyclic programs. In *Proc. of the International Conference on Logic Programming*, pages 579–597. MIT press, 1990.

[3] C. Baral and M. Gelfond. Reasoning about effects of concurrent actions. *Journal of Logic Programming*, 31:85–118, 1997.

[4] S.E. Bornscheuer and M. Thielscher. Explicit and implicit indeterminism: reasoning about uncertain and contradictory specifications. *Journal of Logic Programming*, 31(1-3):119–155, 1997.

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

[6] 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.

[7] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and databases*, pages 293–322. Plenum Press, 1978.

[8] M. Denecker and D. De Schreye. Justification semantics: a unifying framework for the semantics of logic programs. Technical Report 157, Department of Computer Science, K.U.Leuven, 1992.

[9] Marc Denecker. The well-founded semantics is the principle of inductive definition. In *NM'98, Seventh International Workshop on Nonmonotonic Reasoning*, Trento, May 1998.

[10] S. Feferman. Formal theories for transfinite iterations of generalised inductive definitions and some subsystems of analysis. In A. Kino, J. Myhill, and R.E. Vesley, editors, *Intuitionism and Proof theory*, pages 303–326. North Holland, 1970.

[11] M. Gelfond, V. Lifschitz, and A. Rabinov. What Are the Limitations of the Situation Calculus. In S. Boyer, editor, *Automated reasoning, Essays in Honor of Woody Bledsoe*, pages 167–181. Kluwer Academic Publishers, 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] E. Giunchiglia, N. Kartha, and V. Lifschitz. Representing action: indeterminacy and ramifications. *Artificial Intelligence*, 95:409–443, 1997.

[15] 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.

[16] A. Kakas and R. Miller. A simple declarative language for describing narratives with actions. *Journal of Logic Programming, Special Issue on Reasoning about Action and Change*, 31(1-3), 1997.

[17] A. Kakas and R. Miller. A simple declarative language for describing narratives with actions. *Linkoping Electronic Articles in Computer and Information Science*, 2(12), 1997.

[18] V. Lifschitz. Computing Circumscription. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence, 1985*, pages 121–127, 1985.

[19] V. Lifschitz. Formal Theories of Action. In F. Brown, editor, *Proceedings of the 1987 Workshop on the Frame Problem in AI*, 1987.

[20] V. Lifschitz. Frames in the Space of Situations. *Artificial Intelligence*, 46:365–376, 1990.

[21] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In C.S. Mellish, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1985–1991, 1995.

[22] J.W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1987.

[23] Norman McCain and Hudson Turner. A causal theory of ramifications and qualifications. In C.S. Mellish, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1978–1984, 1995.

[24] J. McCarthy. Circumscription - a form of nonmonotonic reasoning. *Artifical Intelligence*, 13:89–116, 1980.

[25] S. McIlraith. A closed-form solution to the ramification problem (sometimes). In *Proceedings of IJCAI-97 Workshop on Nonmonotonic Reasoning, Action and Change*, 1997.

[26] S. McIlraith. Representing actions and state constraints n model-based diagnosis. In *Proceedings of AAAI97*, pages 43–49, 1997.

[27] Y. N. Moschovakis. *Elementary Induction on Abstract Structures.* North-Holland Publishing Company, Amsterdam- New York, 1974.

[28] J.A. Pinto. Concurrent actions and interacting effects. In A.G.Cohn, L. Schubert, and S.C. Shapiro, editors, *Proc. of the 5h International Conference on Principle of Knowledge Representation and Reasoning (KR'98)*, pages 292–303. Morgan Kaufman Publishers, Inc., 1998.

[29] Javier A. Pinto. Temporal reasoning in the situation calculus. Technical Report KRR-TR-94-1, Computer Science Dept., University of Toronto, 1994.

[30] T. Przymuzinski. Perfect Model Semantics. In *Proceedings of JICSLP 88*, pages 1081–1096, 1988.

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

[32] 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.

[33] Erik 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.

[34] Y. Shoham. Nonmonotonic reasoning and causation. *Cognitive Science*, 214:213–252, 1990.

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

[36] K. Van Belleghem. *Open Logic Programming as a Knowledge Representation Language for Dynamic Problem Domains.* PhD thesis, Department of Computer Science, K.U.Leuven, 1997.

[37] K. Van Belleghem, M. Denecker, and D. Theseider Dupré. Ramifications in an event-based language. In *Proc. of the Ninth Dutch Artificial Intelligence Conference, 1997*, 1997.

[38] K. Van Belleghem, M. Denecker, and D. Theseider Dupré. Representing ramifications in an event-based language. Technical Report CW257, Department of Computer Science, K.U.Leuven, 1997.

[39] M. van Emden and R.A Kowalski. The semantics of Predicate Logic as a Programming Language. *Journal of the ACM*, 4(4):733–742, 1976.

[40] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM*, 38(3):620–650, 1991.