A Constructive Approach to the Ramification Problem

Kristof Van Belleghem* Marc Denecker†
Daniele Theseider Dupré
Department of Computer Science , K.U. Leuven,
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
E-mail: {kristof,marcd}@cs.kuleuven.ac.be
Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 Torino, Italy
E-mail: dtd@di.unito.it

May 29, 1998

Abstract

In the state of the art on the ramification problem, it is acknowledged that ramifications of actions are not logical consequences of state constraints and require explicit representation through causal laws. Yet, causal laws are still tightly coupled with state constraints in that their syntactic form is derived from state constraints or in that they merely serve to restore integrity of state constraints. In this paper, we cut this strong connection. Our focus is not on state constraints but on the propagation of physical or logical forces and effects through the dynamic system. Causal laws are considered representations of how these forces and effects propagate. As such, causal laws may be but do not need to be related to state constraints. We define a semantics for causal laws which allows for representing cyclic dependencies between effects and combined effects on *complex fluent formulae*. The effect propagation process is constructive. To adequately model this process, we propose a semantics for causal laws based on the main mathematical constructive principle: inductive definitions. Our approach is language-independent and can be embedded in Situation Calculus, Event Calculus, A-type or STRIPS-type languages. We discuss the relation with other recent approaches based on a restricted notion of nondeterminism.

*Supported by the Vlaams Instituut voor de Bevordering van het Wetenschappelijk-Technologisch Onderzoek in de Industrie (IWT)
†Supported by GOA 93/97-03

1 Introduction

In reasoning about actions in dynamic systems, at least three types of information are important:

- *State constraints* represent properties that hold at each state in the dynamic system. In this paper they will be represented by formulae of the form:

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

  where F is a fluent formula and S a variable of the sort representing time (e.g. situation or time point, depending on the adopted time structure).[1]

- *Effect rules* represent the effects that may occur in the world. One can distinguish between direct effect rules (action laws), representing the primitive effects of actions, and derived effect rules (causal laws), representing indirect effects or *ramifications*, i.e. how the initiation of certain fluents or fluent expressions can cause other fluents to become true or false. The first sort of rules will be represented by rules of the form:

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

  with a an action constant, l a fluent literal (a fluent or its negation) and F a fluent formula. Derived effect rules will be represented by rules of the form:

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

  with F, F' fluent formulae, and l a fluent literal. The *ramification problem* is defined as the problem of determining all direct and derived effects of certain actions.

- *Action preconditions* or *action qualifications* specify necessary conditions on the states in which an action may occur. An action precondition will be represented by formulae:

  $$\forall A, S : \mathbf{Act}(A, S) \wedge a \in A \rightarrow \mathbf{Holds}(F, S)$$

  with a an action constant, A a term representing a set of actions and F a fluent formula. The *qualification problem* is in this paper defined as the problem of determining the preconditions of actions entailed by a theory.[2]

[1] In what follows, we will use a slightly simplified terminology and talk about S as a *state*. More precisely we then mean the state of the world at time S. Only in the discussion on nondeterminism will an explicit distinction between a time and the state at that time become relevant.

[2] Observe that we do not handle the general qualification problem which is concerned with default reasoning on such conditions.

Often action preconditions and effect rules are related to state constraints. The state constraint $\forall S : \mathbf{Holds}(walking \rightarrow alive, S)$ stating that one must be alive to be walking, corresponds to an implicit action precondition on the action $start\_walk$: $\forall A, S : \mathbf{Act}(A, S) \wedge start\_walk \in A \rightarrow \mathbf{Holds}(alive, S)$, and to a derived effect "dying ends walking": **initiating** $\neg alive$ **causes** $\neg walking$ **if** $true$.

In the state of the art approaches to the frame problem, it is acknowledged that neither action qualifications nor ramifications can be derived automatically[3] from state constraints ([6], [5], [11], [15]) and require explicit representation through action preconditions and causal laws. Yet, to date, action preconditions and especially causal laws are still considered to be tightly coupled with state constraints. In some approaches, e.g. [12], [13], [10] and [8], causal laws include an implicit state constraint component[4] . In the approach of [17], the semantics of causal rules is defined in terms of the state constraints (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.*

We argue in this paper that in general, action preconditions and causal laws may be entirely independent of state constraints.

With respect to qualifications, 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 the player who has just made a move may not be in check represents an implicit precondition on moves which is not easily represented explicitly.

With respect to ramifications, rather than focusing on state constraints and how they can be maintained through indirect effects, the problem is to model the reality of the physical or logical forces and effects in the dynamic system and how they propagate through the system. We consider causal laws to be representations of how these forces and effects propagate. As such, causal laws may be but do not need to be related to state constraints. An obvious example of a system exhibiting indirect effects unrelated to state constraints, is an electronic counter in a digital network. Assume the network has many input signals, which can be directly modified through actions, an arbitrary internal structure, and one or more outputs 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

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

[3]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 $alive$).

[4]or a constraint relating different states in the case of delayed effect rules in [8]

These indirect effects are in no way related to a state constraint: 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 modeled in state constraint based approaches.

The direct and derived effect rules we consider have a very general form and allow for representing cyclic dependencies between effects as well as for effects dependent on the initiation of *complex fluent formulae*. Derived effect rules with complex fluent formulae allow for a natural and concise representation of many quite common effect propagations; moreover as an interesting side-effect such rules turn out to be remarkably suitable and robust for representing the combined effects of simultaneous actions. This is illustrated by the following problem from [4]: 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

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

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

$$\mathbf{initiating}\ \neg(up_l \leftrightarrow up_r)\ \mathbf{causes}\ spill\ \mathbf{if}\ on\_table$$

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. A representation without complex fluent formulae in the derived effect rule would require a careful case analysis of different possible states and actions leading to at least four different rules, like "the water is *spilled* if up_l is initiated while up_r is false and up_r is not itself initiated".[5]

A natural example of cyclic effect rules is the following. Consider two connected gear wheels. Any action which makes one of them turn, makes the other one turn as well, and any action which stops one wheel, stops the other one. The propagation of the forces executed by the gear wheels on each other, can be represented by the rules:

$$\begin{array}{l} \mathbf{initiating}\ turn\_1\ \mathbf{causes}\ turn\_2\ \mathbf{if}\ true \\ \mathbf{initiating}\ \neg turn\_1\ \mathbf{causes}\ \neg turn\_2\ \mathbf{if}\ true \\ \mathbf{initiating}\ turn\_2\ \mathbf{causes}\ turn\_1\ \mathbf{if}\ true \\ \mathbf{initiating}\ \neg turn\_2\ \mathbf{causes}\ \neg turn\_1\ \mathbf{if}\ true \end{array}$$

Despite the natural appearance and the intuitive clarity of effect rules, defining their semantics is a non-trivial mathematical problem, in which the following issues need to be addressed:

[5] Lin utilises complex causal laws in [12]. These laws differ from our derived effect rules in that they incorporate a state constraint component, as most other approaches.

- With cyclic dependencies, the semantics should not derive *self-supported effects*. E.g. in the gear wheel example, it should not be derived that the first gear wheel is caused to turn because the second one is caused to turn and vice versa. Without any exterior cause, the state of the gear wheels should not change. Typically, a Clark completion-style semantics would run into the problem of allowing such self-supported effects. This problem calls for a strategy of minimisation of the set of causations; therefore a technique like circumscription would be more appropriate. However, also circumscription runs into problems, due to the next problem.

- Derived effect rules with complex fluent formulae hide implicit negative dependencies between causations, as in the table example above. According to the rule **initiating** $\neg(up_l \leftrightarrow up_r)$ **causes** *spill* **if** *on_table*, *spill* will be caused when up_l is initiated and up_r is false, provided that in the same state up_r is not initiated, i.e. provided that there is *no* cause for up_r (otherwise, $\neg(up_l \leftrightarrow up_r)$ would not be initiated after all). The effect on *spill* thus depends on the absence of an effect on up_r. Applying circumscription to rules incorporating such negative dependencies runs into the same sort of problems as the (naive) application of circumscription to the Yale Turkey Shooting problem ([9]).

- Due to the absence of syntactical constraints on effect rules, theories of effect rules can be constructed which cannot be given a consistent semantics. A simple example is the effect rule **initiating** a **causes** $\neg a$ **if** *true*. In other cases, contradictions may appear in more subtle ways. A mathematical technique is needed to detect these contradictions and to deal with them. It is unclear to us if a more refined circumscription policy might solve these problems. The many increasingly complex variants of circumscription proposed to date suggest that a general solution is not evident, even though distinct variants yield solutions for particular classes of theories.

The solution we propose is the following. The effect propagation process is a constructive process. As remarked in [16], essential properties are :

- 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.

To adequately model this propagation process, a constructive semantics with the same features as the process itself seems appropriate. We base our approach on the main mathematical constructive principle: the principle of inductive definition ([2, 14, 1]). We define a conservative extension of this principle which allows for dealing with non-stratified definitions, using techniques inspired by those used in the definition of logic programming semantics.

In the following section, we formally define the principle of inductive definition and show how to use it to give a natural semantics to effect rules. The introduction of limited nondeterminism will be addressed in a third section, and be used as the basis for a comparison with related work in the fourth and final section.

2 Formal Semantics

In this section we define the semantics of the formulae introduced above, as a state transition function dependent on the occurring actions and the effect rules. As such the approach is not restricted to one particular action language. In particular embeddings in e.g. Situation Calculus and Event Calculus can be readily obtained.

Intuitively, the semantics we propose for a set of effect rules is to read them as an inductive definition of predicates **Caus** and **Init**. This is formalised below.

2.1 Principle of Inductive Definition

The semantics and expressiveness of inductive definitions are studied in a subarea of mathematical logic, the area of Iterated Inductive Definitions (IID) ([2, 14, 1]). The semantics proposed there require the definitions to be stratified. For our purposes this is not sufficient, as indicated above. However, as it appears, the problems of defining the semantics of non-stratified inductive definitions are analogous to the problems of defining the semantics of non-stratified logic programs. Consequently, techniques from logic programming semantics can be used to design a conservative extension of the IID semantics, extending it to non-stratified definitions.

We need the following concepts.

Definition 1 **($\mathcal{V}_\mathbf{P}$,$\leq_F$)** *Given a set of ground atoms* $\mathbf{P}$*, the set* $\mathcal{V}_\mathbf{P}$ *of (3-valued) valuations on* $\mathbf{P}$ *is the set of all functions* $\mathbf{P} \rightarrow \{\mathbf{t}, \mathbf{u}, \mathbf{f}\}$*. On* $\mathcal{V}_\mathbf{P}$*, a partial order* $\leq_F$ *is defined as the pointwise extension of the order* $\mathbf{u} \leq_F \mathbf{t}, \mathbf{u} \leq_F \mathbf{f}$*; more precisely,* $\forall I, I' \in \mathcal{V}_\mathbf{P} : I \leq_F I'$ *iff* $\forall l \in \mathbf{P} : I(l) \leq_F I(l')$*.*

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

Definition 2 (inductive definition) *Given a set of ground atoms* $\mathbf{P}$*, we define* $\widehat{\mathbf{P}} = \mathbf{P} \cup \{\neg l | l \in \mathbf{P}\} \cup \{\mathbf{t}, \mathbf{f}\}$*.*[6] *Valuations can be naturally extended to* $\widehat{\mathbf{P}}$*.*

[6] $\mathbf{u}$ should never occur explicitly in a definition.

A definition rule *in* $\mathbf{P}$ *is an object* $l \leftarrow B$ *where* $l \in \mathbf{P}$ *and* $B \subseteq \widehat{\mathbf{P}}$. l *is called the head,* B *the body of the rule. A definition on* $\mathbf{P}$ *is any set* $\mathcal{D}$ *of rules in* $\mathbf{P}$.

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

Definition 3 (proof tree) *A proof tree* T *for an atom* $p \in \mathbf{P}$ *is a tree of elements of* $\widehat{\mathbf{P}}$ *such that*

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

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

Definition 4 (supported value)

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

Consider for example the rules for the two gear wheels above, plus appropriate direct effect rules (we rewrite the rules as a definition of **Caus**; below we will elaborate on how to do this in general):

$\mathbf{Caus}(A, S, turn_2) \leftarrow \mathbf{Caus}(A, S, turn_1)$
$\mathbf{Caus}(A, S, turn_1) \leftarrow \mathbf{Caus}(A, S, turn_2)$
$\mathbf{Caus}(A, S, \neg turn_2) \leftarrow \mathbf{Caus}(A, S, \neg turn_1)$
$\mathbf{Caus}(A, S, \neg turn_1) \leftarrow \mathbf{Caus}(A, S, \neg turn_2)$
$\mathbf{Caus}(A, S, turn_1) \leftarrow Ha(start_1, A)$
$\mathbf{Caus}(A, S, turn_2) \leftarrow Ha(start_2, A)$
$\mathbf{Caus}(A, S, \neg turn_1) \leftarrow Ha(stop_1, A)$
$\mathbf{Caus}(A, S, \neg turn_2) \leftarrow Ha(stop_2, A)$

In these rules, $Ha(a, A)$ can for now be considered as atoms denoting $a \in A$. Assume the wheels are at first not turning, i.e. $S = \{\neg turn_1, \neg turn_2\}$ and a single action $start_1$. $\mathbf{Caus}(A, S, turn_1)$ has two loop-free finite proof trees: one in which it has as only immediate descendant the leaf $Ha(start_1, A)$, and one in which it has $Ha(turn_2, A)$ as immediate descendant and hence $Ha(start_2, A)$ as only leaf. The former proof tree determines that the supported value of $\mathbf{Caus}(A, S, turn_1)$ is $\mathbf{t}$. Similarly, $\mathbf{Caus}(A, S, turn_2)$ has supported value $\mathbf{t}$ due to the proof tree $\mathbf{Caus}(A, S, turn_2)$ — $\mathbf{Caus}(A, S, turn_1)$ — $Ha(start_1, A)$. The finite proof trees of $\mathbf{Caus}(A, S, \neg turn_1)$ and $\mathbf{Caus}(A, S, \neg turn_2)$ all have a false leaf (either $Ha(stop_1, A)$ or $Ha(stop_2, A)$), hence these literals have $\mathbf{f}$ as supported value.

For a definite (or positive) definition $\mathcal{D}$ like the above one, we define $I_{\mathcal{D}}$ as the valuation mapping each $p \in \mathbf{P}$ to $SV_{\perp}(p)$, i.e. each atom is mapped to its supported value (w.r.t. $\perp$). E.g. we find that in the given example both wheels start turning. For non-definite definitions, $I_{\mathcal{D}}$ is obtained as a *fixpoint* of the above operation:

Definition 5 **($\mathcal{PI}_{\mathcal{D}}$)** *The* positive induction operator $\mathcal{PI}_{\mathcal{D}} : \mathcal{V}_{\mathbf{P}} \rightarrow \mathcal{V}_{\mathbf{P}} : I \rightarrow I'$ *is defined such that* $\forall p \in \mathbf{P} : I'(p) = SV_I(p)$.

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

Definition 6 *Given* $< \mathbf{P}, \mathcal{D} >$, $I_{\mathcal{D}} = \mathcal{PI}_{\mathcal{D}}\!\uparrow$.

For a stratified definition, the fixpoint construction basically mimics a layer by layer truth assignment: each iteration determines the truth values of the initiations in the lowest layer that has not yet been dealt with. After a number of steps equal to the number of layers, all initiations are determined to be true or false. As an example, assume we add a rule to the gear wheel domain stating that an alarm should go off whenever one wheel starts turning but the other does not.[7]

$$\textbf{initiating } turn_1 \wedge \neg turn_2 \textbf{ causes } alarm \textbf{ if } true$$

As we will show below, this rule roughly corresponds to the following low-level definition rules for **Caus**:

$$\begin{array}{lcl}
\mathbf{Caus}(A, S, alarm) & \leftarrow & \mathbf{Caus}(A, S, turn_1), \mathbf{Caus}(A, S, \neg turn_2). \\
\mathbf{Caus}(A, S, alarm) & \leftarrow & \mathbf{Caus}(A, S, turn_1), \mathbf{Holds}(\neg turn_2, S), \\
 & & \neg\mathbf{Caus}(A, S, turn_2). \\
\mathbf{Caus}(A, S, alarm) & \leftarrow & \mathbf{Caus}(A, S, \neg turn_2), \mathbf{Holds}(turn_1, S), \\
 & & \neg\mathbf{Caus}(A, S, \neg turn_1).
\end{array}$$

[7] Clearly it is impossible for this to occur given the other domain knowledge, but of course nothing prevents us from modeling this rule, while a more "useful" example would force us to complicate the domain description, which is not particularly helpful.

This definition is stratified with $alarm$ in a higher level than $turn_1$ and $turn_2$ ($alarm$ depends negatively on $turn_1$ and $turn_2$, which depend only positively on each other). Starting with a truth value **u** for all **Caus** literals, in a first iteration the three rules for $alarm$ yield proof trees with undefined leaves, so that $\mathbf{Caus}(A, S, alarm)$ remains **u**. However, as indicated above, in this same iteration $\mathbf{Caus}(A, S, turn_1)$ and $\mathbf{Caus}(A, S, turn_2)$ get truth values **t**. Then, in the second iteration the proof trees for $\mathbf{Caus}(A, S, alarm)$ have only true or false leaves, and $\mathbf{Caus}(A, S, alarm)$ is assigned a new truth value **f**. At this point the fixpoint is reached.

It follows from the construction that the truth value **u** occurs in the fixpoint of a definition only (but not necessarily) if some atoms negatively depend on each other, or one atom on itself, i.e. only if the definition is not stratified.[8] We interpret this truth value as indicating an error, in the sense that the definition is not constructive. Thus nonsensical and ambiguous definitions are detected and dealt with.

The least fixpoint of the positive induction operator has been proven to coincide with the least fixpoint of the well-founded operator of [19]. 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 very promising framework for general knowledge representation, which formalises important KR principles. However this issue is outside the scope of this paper.

2.2 Tackling the Ramification Problem

We now show how to handle effect rules using an inductive definition based semantics. The problem we want to deal with is the following: given the state of the world at a certain time (described using the **Holds** predicate in the usual way), a set of actions occurring at that time and a set of direct and derived effect rules, calculate the state of the world resulting after this set of actions. We denote a set of actions as A and a time as S. The set $\mathcal{S}$ is the set of all times.

We reduce direct and derived effect rules to an inductive definition of **Init** and **Caus**. These predicates intuitively denote strong and weak initiation, re-

[8] If there is a mutual negative dependency and yet the fixpoint contains no **u**, this means that the dependency is only of a syntactic, not a semantic nature. For an example and a more in-depth discussion of this phenomenon we refer to [3].

spectively: $\mathbf{Init}(A, S, l)$ means that l does not hold at S but holds immediately after the application of the set of actions A at S. $\mathbf{Caus}(A, S, l)$ means that l holds immediately after the occurrence of A, but possibly also already at S. The successor state of a state S after A contains those fluent literals l that are initiated (by A) and those that are true in S and of which the negation $\bar{l}$ is not initiated.

The intended reading of a derived effect rule **initiating** F **causes** l **if** F' is that given F', strong initiation of F causes weak initiation of l. To formalise this for complex F we introduce the concept of a supporting set. This concept is based on a disjunctive normal form of the formula. Since a definition can have 3-valued interpretations, this normal form needs to be equivalence preserving under 3-valued FOL semantics. The 2-valued disjunctive normal form does not satisfy this property (since e.g. $F \wedge \neg F$ is not 3-valued equivalent to **f**), but it is easy to derive a 3-valued variant:

Definition 7 (3-valued disjunctive normal form) *The 3-valued disjunctive normal form $3dnf(F)$ of a fluent formula F is obtained by applying the following rewriting rules to F or its constituents (if $\leftrightarrow$, $\rightarrow$ or $\leftarrow$ occur in F we assume they are rewritten in terms of $\neg$, $\wedge$, $\vee$ as usual) until no further rules apply:*[9]

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

The following properties can be proven. The rewriting process always terminates. The resulting normal form $3dnf(F)$ is unique[10] , i.e. independent of the order in which subformulae are processed. $3dnf(F)$ is always a disjunction of conjunctions of literals. $3dnf(F)$ is equivalent to F under 3-valued as well as 2-valued semantics. Finally, under either semantics, if F and G are equivalent then so are $3dnf(F)$ and $3dnf(G)$.

We define the concept of supporting set as follows:

Definition 8 (supporting set) *Let F be a fluent formula and $F' = (l_1^1 \wedge .. \wedge l_{n_1}^1) \vee ... \vee (l_1^m \wedge .. \wedge l_{n_m}^m)$ its 3-valued disjunctive normal form. A supporting set L of F is any set $\{l_1^i, .., l_{n_i}^i\}$, $\forall 1 \leq i \leq m$.*

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

[9] We assume commutativity and associativity are applied whenever needed.
[10] modulo commutativity and associativity

Definition 9 (notations) *In what follows* $Ha(a, A)$ *is the truth value of "*$a \in A$*" and* $Ho(F, S)$ *is the truth value of "*$\mathbf{Holds}(F, S)$*". We define* $\mathbf{Init}(A, S, L)$ *as* $\{\mathbf{Init}(A, S, l) \mid l \in L\}$ *and* $\mathbf{Caus}(A, S, L)$ *as* $\{\mathbf{Caus}(A, S, l) \mid l \in L\}$*. Further, from now on we use the notation* $\overline{f} = \neg f$*,* $\overline{\neg f} = f$*,* $\overline{L} = \{\overline{l} \mid l \in L\}$ *for any set of literals* L*, and* $\overline{P} = \{\neg p \mid p \in P\}$ *for any set of* **Init** *or* **Caus** *atoms* P*.*

Definition 10 (grounding of effect rules)
The grounding of a rule "a **causes** l **if** F*" is*

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

*The grounding of "***initiating** F **causes** l **if** F'*" is*

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

The grounding of a set of effect rules Π_e *is* $\mathcal{D}_{init} = \mathcal{D}_g \cup \{\mathbf{Init}(A, S, l) \leftarrow \mathbf{Caus}(A, S, l), \neg Ho(l, S) \mid A \subseteq \mathcal{A},\ S \in \mathcal{S}, l \in \widehat{\mathbf{P}}\}$*, where* $\mathcal{D}_g$ *is the union of the groundings of all rules in* Π_e*.*

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

In case the interpretation of **Caus** contains any truth value **u** or in case this interpretation is contradictory in that for some fluent f both f and $\neg f$ are caused, the resulting successor state is considered invalid. The set of actions leading to this state is then disallowed, like in the case of violated state constraints discussed below.

The rules $\mathbf{Init}(A, S, l) \leftarrow \mathbf{Caus}(A, S, l), \neg Ho(l, S)$ are the only rules for **Init**. Since the completion of the definition rules is entailed by the well-founded and therefore also the inductive definition semantics, the rules imply

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

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

The mutual recursion in the definitions of **Init** and **Caus** indicates that strong initiations may provide causes for literals to become true; only if these literals are not already true, i.e. if they are actually changed, they can themselves give rise to further ramifications.

As an example of the grounding definition, consider the rule

$$\mathbf{initiating}\ turn_1 \wedge \neg turn_2\ \mathbf{causes}\ alarm\ \mathbf{if}\ true$$

The formula $turn_1 \wedge \neg turn_2$ is in disjunctive normal form and has one supporting set $\{turn_1, \neg turn_2\}$. Hence, the grounding of this rule is exactly

$$\begin{array}{lcl}
\mathbf{Caus}(A, S, alarm) & \leftarrow & \mathbf{Init}(A, S, turn_1), \mathbf{Init}(A, S, \neg turn_2), \\
& & \neg Ho(turn_1 \wedge \neg turn_2, S). \\
\mathbf{Caus}(A, S, alarm) & \leftarrow & \mathbf{Init}(A, S, turn_1), Ho(\neg turn_2, S), \\
& & \neg \mathbf{Init}(A, S, turn_2), \neg Ho(turn_1 \wedge \neg turn_2, S). \\
\mathbf{Caus}(A, S, alarm) & \leftarrow & \mathbf{Init}(A, S, \neg turn_2), Ho(turn_1, S), \\
& & \neg \mathbf{Init}(A, S, \neg turn_1), \neg Ho(turn_1 \wedge \neg turn_2, S). \\
\mathbf{Caus}(A, S, alarm) & \leftarrow & Ho(\neg turn_2, S), \neg \mathbf{Init}(A, S, turn_2), \\
& & Ho(turn_1, S), \neg \mathbf{Init}(A, S, \neg turn_1), \\
& & \neg Ho(turn_1 \wedge \neg turn_2, S).
\end{array}$$

Observe that the last rule has a trivially false body due to the conflicting Ho conditions.

The question which remains to be answered is what the role of state constraints and preconditions is in this context. As suggested by their syntax, these are interpreted as classical FOL formulae. In other words, they are orthogonal to the successor state calculation. In the general theory of action, which may contain (complete or incomplete) information about occurring actions and observed fluent values, they will be formulae that need to be satisfied. 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 paper we do not aim at introducing a specific time structure (and therefore a formalisation of inertia), but rather at defining a state transition function that can be incorporated in different time structures.

We now give some interesting results concerning the semantics. For proofs we refer to [18].

Theorem 1 *Given a state S, a set of actions A and a set of direct and derived effect rules, the truth values of* $\mathbf{Init}(A, S, l)$ *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 [17]. Unless nondeterminism is explicitly introduced, the 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 11 (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,* $I_{\mathcal{D}_{init}}$ *is always 2-valued. Moreover* $I_{\mathcal{D}_{init}}$ *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,* $I_{\mathcal{D}_{init}}$ *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 show that for several classes of definitions for which other semantics are known to assign the intended meaning to all predicates, the inductive definition semantics coincides with these semantics. However the inductive definition semantics is more general, also dealing with definitions that cannot be dealt with by the more common semantics.

3 Nondeterminism

In this section we introduce a representation of nondeterministic actions and ramifications where there is a possible effect rather than a definite one. This extension is useful for representing incomplete knowledge about the direct and indirect effects of actions, and will also be used 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.

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.

Our rules for nondeterministic actions are similar to those in [7], although the semantics is defined in a different way and we also allow for nondeterministic ramifications. Observe that 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[11] can be obtained by

[11] A representation

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

is only an approximation for an action with alternative effects l and l' since it allows for none or both of l and l' to be directly caused by a.

extending the principle used in the semantics below, but is outside the scope of this paper.

The meaning of a set of rules, including nondeterministic ones, is defined by modifying definition 10 as follows: the grounding of a nondeterministic rule is obtained considering, for each time S, either the grounding of the corresponding deterministic rule, or nothing[12] .

Definition 12 (nondeterministic grounding)
The restriction G^S of a set of primitive definition rules G to a time S is the set of all rules of G in which S occurs.

A grounding of a nondeterministic direct effect rule

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

is any set $\bigcup_{S\in\mathcal{S}} G'^S$, where G'^S is either empty or it is the restriction G^S of the grounding G of the corresponding direct effect rule a **causes** *l* **if** *F.*

A grounding of a nondeterministic derived effect rule

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

is any set $\bigcup_{S\in\mathcal{S}} G'^S$, where G'^S is either empty or it is the restriction G^S of the grounding G of the corresponding derived effect rule **initiating** *F* **causes** *l* **if** *F'.*

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

4 Discussion

We have presented a new language-independent approach to the ramification problem, which allows for ramifications unrelated to state constraints, and deals with cyclic and negative dependencies between effects. We have discussed the types of constructs required to reach those goals, and presented a semantics based on the principle of inductive definitions. This semantics was chosen due to its closeness to the intuitions underlying effect propagations (e.g. constructiveness).

The approach most similar to ours is the one described in [17], which at least syntactically uncouples causal rules from state constraints. However, a

[12] Observe that here the distinction between time point/situation and state becomes relevant: a separate choice is made for each time, not only for each state. This represents that at different times a nondeterministic action can have different effects, even if the state of the world at those times is the same. To allow for hypothetical reasoning in branching time, separate choices would be required also for each set of actions at each time.

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

$$a \textbf{ causes } b$$

are exactly equivalent to our rules

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

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 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

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

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. This is consistent with Thielscher's view that small delays are involved in all change propagations. (In our view, such delays should be modeled explicitly if present, using delayed effect rules, but this issue is outside the scope of this paper). It is due to this sequential application of rules that a **causes** b **if** c is strictly stronger than our nondeterministic rule: our corresponding rules can only be applied if the condition c holds in the starting state.

There is no room here for a detailed discussion of the impact of sequential vs simultaneous application. However, it is interesting to note that a set of rules of the form

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

is problematic in a sequential approach, as they give rise to an arbitrary increase of n: when *out* is initiated, the counter can take on any value as the rule can be applied ad infinitum. Hence our point of view that one needs to distinguish explicitly between immediate and delayed effect propagations.

As far as other approaches using causal laws are concerned, we have observed that these can be mapped to a particular subset of Thielscher's framework: in Thielscher's terms, the causal laws in these approaches can be seen as state constraints with hard-wired *influence information*, e.g. in [13] a causal law $\phi \Rightarrow$

ψ represents the state constraint $\phi \rightarrow \psi$ plus the influence information that all literals in ϕ can influence all literals in ψ and all literals in ψ can influence each other.

With respect to approaches not based on causal laws, we refer to the arguments presented in [17]: Thielscher's and our approach differ in the same essential aspects from these approaches.

Finally, we want to add a few words on further (and earlier) work on this framework. We have embedded the proposal in a more complete action language using a linear time structure, which we have called $\mathcal{ER}$. In this theory we deal with complete or partial information about action occurrences, action order, initial state and given observations at arbitrary time points. These issues are completely orthogonal to the approach to ramifications presented here. In addition we have generalised the approach to nondeterminism described above, and included rules for dealing with delayed ramifications. More details can be found in [18].

We have provided a mapping of $\mathcal{ER}$ theories into Open Logic Programming, an extension of logic programming with open predicates, i.e. predicates which do not have a definition (and which correspond to abducibles in abductive logic programming). This mapping is roughly based on the Event Calculus, and has been proven sound and complete. The OLP theory can be used for various types of reasoning — e.g. deductive, abductive or combinations thereof, with as most important instances temporal projection, diagnosis and planning — using an abductive resolution procedure.

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] 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.

[3] M. Denecker, D. Theseider Dupré, and K. Van Belleghem. A constructive approach to the ramification problem. Submitted, 1998.

[4] 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.

[5] M. Ginsberg and D. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35, 1988.

[6] M. Ginsberg and D. Smith. Reasoning about action II: The qualification problem. *Artificial Intelligence*, 35, 1988.

[7] E. Giunchiglia, N. Kartha, and V. Lifschitz. Representing action: indeterminacy and ramifications. *Artificial Intelligence*, 95:409–443, 1997.

[8] 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.

[9] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logic, and the frame problem. In *Proceedings of the National Conference on Artificial Intelligence, Philadelphia*, pages 328–333, 1986.

[10] 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.

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

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

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

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

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

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

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

[18] 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.

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