Available online at www.sciencedirect.com

ScienceDirect

Artificial Intelligence 171 (2007) 332–360

Artificial Intelligence

www.elsevier.com/locate/artint

Inductive situation calculus

Marc Denecker [a,*], Eugenia Ternovska [b]

[a] *Department of Computer Science, KU Leuven, Belgium*
[b] *School of Computing Science, Simon Fraser University, Canada*

Received 15 November 2006; received in revised form 7 February 2007; accepted 8 February 2007

Available online 20 February 2007

Abstract

Temporal reasoning has always been a major test case for knowledge representation formalisms. In this paper, we develop an inductive variant of the situation calculus in ID-logic, classical logic extended with inductive definitions. This logic has been proposed recently and is an extension of classical logic. It allows for a uniform representation of various forms of definitions, including monotone inductive definitions and non-monotone forms of inductive definitions such as iterated induction and induction over well-founded posets. We show that the role of such complex forms of definitions is not limited to mathematics but extends to commonsense knowledge representation. In the ID-logic axiomatization of the situation calculus, fluents and causality predicates are defined by simultaneous induction on the well-founded poset of situations. The inductive approach allows us to solve the ramification problem for the situation calculus in a uniform and modular way. Our solution is among the most general solutions for the ramification problem in the situation calculus. Using previously developed modularity techniques, we show that the basic variant of the inductive situation calculus *without* ramification rules is equivalent to Reiter-style situation calculus.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Knowledge representation; Inductive definitions; Situation calculus

1. Introduction

ID-logic[1] [5,8,10] is an extension of classical logic with inductive definitions (ID). In mathematical texts, inductive definitions are usually represented as collections of rules, which represent the base case and inductive cases. Inductive rules may be *monotone* or *non-monotone*. An example of the latter is the following rule in the definition of the truth relation $\models$:

$$I \models \neg\psi \quad \text{if} \quad I \not\models \psi,$$

which states that I satisfies $\neg\psi$ if I does *not* satisfy ψ. It is well known that in general, inductive definitions cannot be represented in first-order logic (FO). ID-logic extends classical logic with a construction that allows for a uniform

* Corresponding author.
E-mail address: Marc.Denecker@cs.kuleuven.be (M. Denecker).

[1] The term ID-logic was introduced by the first author in [5] to denote a logic of sets of classical first-order logic sentences and definitions. This logic was extended to its current definition in [8] and in [7], where it was called NMID-logic in order to emphasize that the logic deals with non-monotone inductive definitions (NMIDs).

0004-3702/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.artint.2007.02.002

representation of different sorts of inductive definitions; moreover, this representation preserves the rule-based nature of definitions in mathematical texts. The semantics of this new construct is based on the well-founded semantics of logic programming [41]. This semantics correctly formalizes the semantics of different types of definitions that can be found in mathematics, e.g. recursion-free definitions, monotone inductive definitions, and non-monotone inductive definitions such as inductive definitions over well-founded orders and iterated inductive definitions [4,6].

ID-logic occupies an interesting place in the spectrum of logics used in mathematics, computer science and knowledge representation. As an extension of classical logic with a fixpoint semantics for inductive definitions, it can be viewed as a new element in the family of fixpoint logics. Monotone fixpoint logics have their origin in the logical study of monotone inductive definitions [1,28]. The contribution of ID-logic is that it formalizes two non-monotone inductive principles (i.e., inductive definition over a well-founded order and iterated inductive definition), which differ from the non-monotone principle based on the inflationary fixpoint studied in the inflationary fixpoint logic IFP [16]. ID-logic is similar to description logics [2] in its separation of definitional and assertional knowledge, but it allows definitions for n-nary predicates and non-monotone inductive definitions. In addition, ID-logic is formally an extension of Logic Programming and its variants such as Abductive Logic Programming and Datalog. In this way, ID-logic induces an alternative informal semantics for logic programming, solidly based on the mathematical principle of inductive definitions. As such, the study of semantical and computational aspects of ID-logic can lead to synergy and integration of all these different areas.

On the computational level, ID-logic has recently been proposed as the underlying language for a constraint programming framework [27]. This framework is based on ideas from descriptive complexity theory and is similar in some respects to Answer Set Programming [20,29]. A problem instance is a finite structure, and a problem specification is an ID-logic formula defining the relationship between an instance and its solutions. Solving a problem amounts to expanding the structure with new relations to satisfy the formula. Depending on the expressiveness allowed, the framework captures various complexity classes, including P and NP. Several ID-logic solvers have been developed [21,30].

The focus of this paper is on *knowledge representation* and *modeling* in ID-logic. Although diverse forms of inductive definitions occur frequently in mathematics, there is little awareness in the logic and KR community of non-monotone forms of inductive definitions and of the potential role of inductive definitions for knowledge representation. Thus, a central aim of this paper is to clarify and illustrate these types of definitions. We provide examples of monotone definitions, definitions by induction over well-founded order and iterated inductive definitions and relate these to other knowledge representation principles such as completion and circumscription. Moreover, we show that the role of these complex forms of definitions is not limited to mathematics but extends to commonsense knowledge representation.

A second major purpose of the paper is to illustrate the use of a "tool set" from [8,42] for analyzing definitions, consisting of different *modularity theorems*, *totality theorems* and *translation theorems*. Our experiment demonstrates the effectivity of the tool set for breaking up large complex definitions into conjunctions of smaller and simpler ones, for translating definitions into classical logic, and for proving consistency and correctness of ID-logic theories.

The domain of application selected for our study is *temporal reasoning*. Since the early days of AI, temporal reasoning, in particular the *situation calculus*, has been a major test case for knowledge representation languages. In [25], McCarthy and Hayes exposed the famous *frame problem*, showing the difficulty of axiomatizing actions and causation in classical logic. This problem has (partly) motivated the development of the area of *non-monotonic reasoning*, leading to non-monotone logics such as default logic [32] and non-monotone reasoning techniques in classical logic such as circumscription [23] and completion [3]. Many different temporal reasoning formalisms were developed. Currently, the most widely adopted formalization of situation calculus is the one in classical logic developed by Reiter and his collaborators in the nineties [18,31,34]. Other well-known solutions are Event calculus [35], Fluent calculus [39], non-monotonic logic approaches such as the (many extensions of) the language $\mathcal{A}$ [14] and non-monotonic causal theories [15,22].

We present here a formalization of situation calculus in ID-logic, which we call the *inductive situation calculus*. Temporal reasoning is a natural application for using inductive definitions on the set of situations. In Reiter's situation calculus for example, the description of the initial state may be viewed as the base case, and successor state axioms may be seen the inductive case. By axiomatizing situation calculus in ID-logic, we explicitate the definitional structure underlying situation calculus. The main component of the inductive situation calculus will be a definition of fluent and causality predicates by simultaneous, non-monotone, iterated induction in the well-founded set of situations. This definition and its components are natural illustrations of each of the above mentioned types of inductive definitions.

A first benefit in explicitating the definitional structure of situation calculus, is that we considerably gain on the representational level. In particular, the inductive situation calculus is more expressive and modular than Reiter's classical logic version. On the level of modularity, in the inductive situation calculus it is possible to represent effects of specific actions on specific fluents in specific circumstances by individual effect rules. It is well-known that increased modularity may improve *elaboration tolerance* [24]. On the level of expressivity, the inductive situation calculus can handle *recursive ramifications*, where an effect to one fluent may cause an effect to another fluent and vice versa. The challenge in handling such recursive ramifications is to avoid erroneous models in which the related fluents "cause" each other and become true simultaneously without external cause. By interpreting effect rules as definitional rules, such *spontaneous generation* of effects is avoided. As a consequence, the inductive situation calculus currently provides the most general solution of the *ramification problem*. It also provides the most general solution for *defining fluents* in situation calculus, since fluents can be defined by monotone or non-monotone inductive definitions.

We also prove a range of *correctness* results of the inductive situation calculus, which are obtained using the above mentioned tool set. Our strategy will be to break up the large simultaneous inductive definition of all fluents in a conjunction of small component definitions, to prove their totality and to translate them into classical logic. The main results are the following:

- We use the tool set to prove equivalence between Reiter's situation calculus and a subformalism of the inductive situation calculus. More precisely, our techniques allow us to translate this subformalism into classical first-order logic theories, closely related and provably equivalent to Reiter style situation calculus.
- Extending the previous result, we show that a much broader class of inductive situation calculus theories can be translated into extensions of Reiter's situation calculus in first- or second-order logic, without using the inductive definition construct of ID-logic. However, the advantage of using explicit inductive definitions is that different effect and ramification theories, which can be modeled in a *uniform way* in the inductive situation calculus, require different translation policies, using different combinations of *predicate completion* and *circumscription*.
- We will prove an *initial state expansion property* for the inductive situation calculus. This theorem guarantees that for each model satisfying the subtheory that axiomatizes the initial situation, and each extension of this model interpreting the action symbols, there is a unique way to extend this structure into a model of a complete inductive situation calculus theory. To demonstrate the importance of this property, we discuss two of its implications. First, the inductive situation calculus satisfies the well-known property of *relative satisfiability* [31]: a theory in it is satisfiable if and only if the subtheory of the initial state is satisfiable. Satisfiability of a first-order or ID-logic theory is, in general, an undecidable property. The initial state expansion property thus reduces the problem of proving satisfiability of an inductive situation calculus to the smaller problem of proving satisfiability of the theory of the initial situation.

  Second, this proposition shows that the inductive situation calculus correctly solves the frame problem. Recall from [25] that, put simply, the *frame problem* is the problem of representing what *does not change* as a result of performing actions. In first order logic, it turned out that a "naive" situation calculus theory, consisting merely of effect and inertia formulas, provides only a weak, highly incomplete axiomatization of the temporal reasoning domain, in the sense that the theory accepts many *unintended models* which differ from the intended models by the fact that fluents become true spontaneously or are caused by the wrong type of action. To find elegant and general solutions for this problem turned out to be challenging. Recall Hank and McDermot's famous Turkey Shooting experiment [17], in which all early non-monotone approaches to temporal reasoning were shown to be too weak, in the sense of accepting unintended models. Ultimately, it took the knowledge representation community about two decennia to come up with sufficiently general theories that avoid these unintended models. In the inductive situation calculus, this problem is excluded from the start, because successor states are *defined* in terms of the initial state; hence, an initial state, together with a choice of actions, can be extended in exactly one model. This model is constructed using the effect rules and correctly captures the state transitions of the dynamic system. Thus, the initial state expansion property guarantees that there are no unintended models of an inductive situation calculus unless its initial situation is an unintended model of the subtheory of the initial situation. The initial state expansion property is, in this respect, an important correctness property of formalizations of situation calculus.

- Our results imply that the *initial state expansion property* and all its implications are satisfied by Reiter's situation calculus as well, since this formalism is (equivalent to) a subformalism of the inductive situation calculus. To our knowledge, this is the first time that such a theorem was proven for situation calculus.

There are other topics in this paper which are of wider interest for the history of knowledge representation and the philosophy of logic.

- ID-logic achieves a coherent and conceptually clean *integration* of classical monotonic logic and logic programming, two knowledge representation paradigms which so far seemed to be incompatible. Our experiment illustrates the different roles of these formalisms. In particular, integrating (extended forms of) logic programs with classical logic compensates for the latter's representational weakness on inductive definability.
- In a similar spirit, ID-logic is also a clean and coherent integration of monotone and non-monotone logic and displays both monotone and non-monotone behaviour. ID-logic extends classical logic with a new type of atomic formulae, the *definition*, which is a set of definitional implications. As such, ID-logic is a monotone logic, in the sense that adding an ID-logic formula to an ID-logic theory preserves all logical consequences of the original theory. On the other hand, extending an ID-logic definition with a new definitional implication has a non-monotone effect and does not preserve logical consequences. For instance, adding the rule $\forall x(p(x) \leftarrow x = b)$ to the definition $\{\forall x(p(x) \leftarrow x = a)\}$ deletes the logical consequence $\neg p(b)$. Consequently, the definitional implications are the non-monotone modules of ID-logic.[2]
- The definitional implication is an interesting, non-standard sort of *conditional* which, to the best of our knowledge, has not been studied from a philosophical logic perspective. The properties of this non-monotone, non-truth-functional connective are nicely illustrated in the inductive situation calculus, where definitional implications are used to represent effects of specific actions on specific fluents in specific circumstances. Interestingly, these rules are very similar to the so called *effect rules* in Reiter's situation calculus. Semantically however, there is an important difference between these rules in the two formalisms. In Reiter's situation calculus, they are interpreted as *material implications*, and must be completed by additional axioms to obtain the final axiomatization in terms of *successor state axioms*. In the inductive situation calculus, effect rules are interpreted as definitional implications, which entail the corresponding material implications, but are not equivalent to them. As a consequence, we obtain an axiomatization which is equivalent to Reiter's solution to the frame problem in a natural way, *without adding anything to our effect rules*. In this sense, we claim that *Reiter's situation calculus makes hidden use of inductive definitions*.
- The natural and effective use of definitional rules for representing effects (including recursive ramifications!) also suggests an unexpected and intriguing relationship between inductive definitions and *causality*. In retrospect, this is not so surprising at all, since an inductive definition can be understood as a description of a mathematical construction process, where the definitional implications represent atomic operations executed during the process. But this is exactly the role of an effect rule in a causal theory. This topic deserves a deeper investigation, which might lead to a better understanding of the role of inductive definitions and logic programming in knowledge representation and philosophical logic.

The paper is structured as follows. In the next section, we define ID-logic and present the modularity, totality and translation theorems that form the tool set using which we will analyze the inductive situation calculus. In Section 3, we review the more traditional variant of the situation calculus, which is similar to [34]. In the rest of the paper,

[2] Be aware that in this paper the term *monotonicity* is used in two different meanings, one stemming from inductive definability, the other from knowledge representation. A "monotone" inductive definition is—roughly—a definition without negation in rule bodies but the subformalism of "monotone" inductive definitions is non-monotone in the knowledge representation sense, since adding (monotone) definitional rules to such a definition is a non-monotone operation, as illustrated by the example.

we present the formalism of the inductive situation calculus, address the ramification problem and consider several detailed examples.

This paper extends the conference version [7].

2. Preliminaries

2.1. Preliminaries from logic

We begin by fixing notation and terminology for the basic syntactic and semantic notions related to first- and second-order logic.

We assume an infinite supply of distinct symbols, which are classified as follows:

1. Logical symbols:
   (a) Parentheses: (,);
   (b) Logical connectives: $\wedge$, $\neg$;
   (c) Existential quantifier: $\exists$;
   (d) Binary equality symbol: $=$ (optional);
   (e) Two propositional symbols: **t** and **f**.
2. Non-logical symbols:
   (a) countably many object symbols. Object symbols are denoted by low-case letters;
   (b) for each positive integer $n > 0$, countably many n-ary function symbols of arity n. Function symbols are denoted by low-case letters;
   (c) for each positive integer n, countably many n-ary relation symbols, also called predicate or set symbols of arity n. We use upper-case letters to denote predicates.

As usual, we identify object symbols with 0-ary function symbols and propositional symbols with predicate symbols of arity 0.

Remark 1. In most parts of this paper, we do not make a formal distinction between variable and constant symbols. Symbols occurring free in a formula can be viewed as constants. Symbols in the scope of a quantifier are viewed as variables. In examples, we tend to quantify over x, y, X, Y, and leave c, g, f and P, Q free and treat them as constants. This convention allows us to simplify the exposition by considering several cases at once.

We define a vocabulary as any set of non-logical symbols. We denote vocabularies by $\tau, \tau_\Delta^{o}, \ldots$. We use σ, σ_1, σ_2 etc., to refer to an arbitrary symbol of the vocabulary. We write $\bar{\sigma}$ to denote a sequence of symbols $(\sigma_1, \sigma_2, \ldots)$ or, depending on the context, the set of symbols $\{\sigma_1, \sigma_2, \ldots\}$. Likewise, $\bar{X}$ denotes a sequence or a set of relational symbols (i.e., set variables or constants), and $\bar{x}$ is used to denote a sequence or a set of object symbols, etc.

A *term* is defined inductively as follows:

– an object symbol is a term;
– if $t_1, \ldots, t_n$ are terms and f is an n-ary function symbol, where $n \geqslant 1$, then $f(t_1, \ldots, t_n)$ is a term.

A formula is defined by the following induction:

– if P is an n-ary predicate constant or variable, and $t_1, \ldots, t_n$ are terms then $P(t_1, \ldots, t_n)$ is a formula, called an *atomic formula* or simply an *atom*;
– if ϕ, ψ are formulas, then so are $\neg\phi$, $\phi \wedge \psi$;
– if x is an object symbol, f a function symbol, X is a predicate symbol and ϕ is a formula, then $\exists x\ \phi$, $\exists f\ \phi$ and $\exists X\ \phi$ are formulas.

A bounded occurrence of symbol σ in formula ϕ is an occurrence of σ in a sub-formula $\exists\sigma\psi$ of ϕ. A free occurrence of σ in ϕ is an unbounded occurrence. The set of symbols which occur free in ϕ is denoted $\mathit{free}(\phi)$. This set can also be defined inductively:

– If ϕ is atomic, say of the form $A(t_1,\dots,t_n)$ then the set $\mathit{free}(\phi)$ is the set of all object, relational and functional symbols occurring in ϕ;
– $\mathit{free}(\neg\phi) := \mathit{free}(\phi)$;
– $\mathit{free}(\phi\wedge\psi) := \mathit{free}(\phi)\cup\mathit{free}(\psi)$;
– $\mathit{free}(\exists\sigma\phi) := \mathit{free}(\phi)\setminus\{\sigma\}$.

A relation symbol X has a negative (positive) occurrence in formula F if X has a free occurrence in the scope of an odd (even) number of occurrences of the negation symbol $\neg$.

A formula ϕ is a formula *over vocabulary* τ if its free symbols belong to τ ($\mathit{free}(\phi)\subseteq\tau$). We use SO[$\tau$] to denote the set of all formulas over τ; and we use FO[τ] to denote the set of first-order formulas over τ, that is those without quantified predicate or function variables.

We use $(\phi\vee\psi)$, $(\phi\supset\psi)$, $(\phi\equiv\psi)$, $\forall x\phi$, $\forall f\phi$ and $\forall X\phi$, in the standard way, as abbreviations for the formulas $\neg(\neg\phi\wedge\neg\psi)$, $\neg(\phi\wedge\neg\psi)$, $\neg(\phi\wedge\neg\psi)\wedge\neg(\psi\wedge\neg\phi)$, $\neg\exists x(\neg\phi)$, $\neg\exists f(\neg\phi)$, $\neg\exists X(\neg\phi)$, respectively.

Having defined the basic syntactic concepts, we define the semantic concepts. Let A be a non-empty set. A *value* for an n-ary relation (function) symbol σ of vocabulary τ in A is an n-ary relation (function) in A. A value for a 0-ary function symbol, i.e., an object constant or variable, is an element of the domain A. A value for a 0-ary relation symbol Y is either $\emptyset$ or $\{()\}$, the singleton of the empty tuple. We identify these two values with *false*, respectively *true*. The value of the equality symbol is always the identity relation on A. The value of **t** is $\{()\}$ (true) and the value of **f** is $\emptyset$ (false).

A *structure* I for a given vocabulary τ (in short, a τ-*structure*) is a tuple of a domain $\mathit{dom}(I)$, which is a non-empty set, and a mapping of each symbol σ in τ to a value σ^I in $\mathit{dom}(I)$. If $\sigma\in\tau$ and I is a τ-structure, we say that I *interprets* σ. We also use letters J, K, L, M to denote structures. Given I, τ_I denotes the set of symbols interpreted by I.

Let us introduce notation for constructing and modifying structures with a shared domain A. Let I be a τ-structure, and $\bar\sigma$ be a tuple of symbols not necessarily in τ. Structure $I[\bar\sigma:\bar v]$ is a $\tau\cup\bar\sigma$-structure, which is the same as I, except symbols $\bar\sigma$ are interpreted by values $\bar v$ in $\mathit{dom}(I)$. Given a τ-structure I and a sub-vocabulary $\tau'\subseteq\tau$, the restriction of I to the symbols of τ' is denoted $I|_{\tau'}$. Vice versa, I is called an extension of I_o if $I|_{\tau_{I_o}} = I_o$.

Let t be a term, and let I be a structure interpreting each symbol in t. We define the *value* t^I of t under I by the usual induction:

– if t is an object symbol σ, then t^I is σ^I, the value of σ in I;
– if $t = f(t_1,\dots,t_n)$, then $t^I := f^I(t_1^I,\dots,t_n^I)$.

Next we define the *satisfaction* or *truth* relation $\models$. Let I be a structure and let ϕ be a formula such that each free symbol in ϕ is interpreted by I. We define $I\models\phi$ (in words, *ϕ is true in I*, or *I satisfies ϕ*) by the following standard induction:

– $I\models X(t_1,\dots,t_n)$ if $(t_1^I,\dots,t_n^I)\in X^I$;
– $I\models\psi_1\wedge\psi_2$ if $I\models\psi_1$ and $I\models\psi_2$;
– $I\models\neg\psi$ if $I\not\models\psi$;
– $I\models\exists\sigma\psi$ if for some value v of σ in the domain $\mathit{dom}(I)$ of I, $I[\sigma:v]\models\psi$.

Note that the truth of a formula ϕ is only well-defined in a structure interpreting each free symbol of ϕ. We shall denote the truth value of ϕ in I by ϕ^I, i.e., if $I\models\phi$ then ϕ^I is true and otherwise, it is false.

We use notation $\phi(x_1,\dots,x_n)$ to emphasize that symbols $x_1,\dots,x_n$ are distinct and are free in ϕ. Sometimes, we wish to investigate the truth value of a formula ϕ as a function of the values assigned to a specific tuple of symbols $\bar\sigma$. We then call these symbols the *parameters* of ϕ and denote the formula by $\phi(\bar\sigma)$. Let I be some structure and let $\bar v$ be a tuple of values for $\bar\sigma$ in the domain $\mathit{dom}(I)$. We often write $I\models\phi[\bar v]$ to denote $I[\bar\sigma:\bar v]\models\phi$. If X is an n-ary relation symbol and $\bar{d}$ is an n-tuple of elements of some domain A, then $X[\bar{d}]$ is *a domain atom in* A. For I a structure with domain A, the value of $X[\bar{d}]$ in I is true if $\bar{d} \in X^I$; otherwise it is false. For a vocabulary τ, we define At_A^τ as the set of all domain atoms in domain A over relation symbols in τ.

For a set $\bar{X}$ of relation symbols, we define $At_A^{\bar{X}}$ as the set of all domain atoms in domain A over relation symbols in $\bar{X}$. In general, for a given vocabulary τ with predicates $\bar{X}$, we denote $At_A^{\bar{X}}$ as At_A^τ.

2.2. ID-logic

In this subsection, we describe ID-logic [10]. ID-logic was introduced by the first author in [5] as a logic of sets of first order logic sentences and definitions. In [8] and in [7], this logic was extended to its current definition by allowing arbitrary boolean combinations of atoms and definitions.

Let us fix some vocabulary τ. A new binary connective $\leftarrow$ is called the *definitional implication*. A *definition* Δ is a set of rules of the form

$$\forall \bar{x}(X(\bar{t}) \leftarrow \varphi),$$

where $\bar{x}$ is a tuple of object variables, X is a predicate symbol (i.e., a predicate constant or variable) of some arity r, $\bar{t}$ is a tuple of terms of length r of the vocabulary τ, φ is an arbitrary first-order formula of τ, which may contain free object or relational variables. The definitional implication $\leftarrow$ must be distinguished from material implication denoted $\supset$.

Note that in front of rules, we allow only universal quantifiers. In the rule $\forall \bar{x}(X(\bar{t}) \leftarrow \varphi)$, $X(\bar{t})$ is called the *head* and φ is the *body* of the rule. A *defined symbol* of Δ is a relation symbol that occurs in the head of at least one rule of Δ; other relation, object and function symbols are called *open*. Let τ be a vocabulary including all free symbols of Δ. The subset of defined symbols of definition Δ is denoted τ_Δ^{d}. The set of open symbols of Δ in τ is denoted τ_Δ^{o}. The sets τ_Δ^{d} and τ_Δ^{o} form a partition of τ, i.e., $\tau_\Delta^{\mathrm{d}} \cup \tau_\Delta^{\mathrm{o}} = \tau$, and $\tau_\Delta^{\mathrm{d}} \cap \tau_\Delta^{\mathrm{o}} = \emptyset$.

A *well-formed formula* of ID-logic over vocabulary τ is defined by the following (monotone) induction:

(1) If X is an n-ary predicate symbol, and $t_1, \ldots, t_n$ are terms then $X(t_1, \ldots, t_n)$ is a formula.
(2) If Δ is a definition then Δ is a formula.
(3) If ϕ, ψ are formulas, then so is $(\phi \wedge \psi)$.
(4) If ϕ is a formula, then so is $(\neg\phi)$.
(5) If ϕ is a formula, then $\exists\sigma\phi$ is a formula (σ can be either a first- or second-order symbol).

A formula ϕ is an *ID-logic-formula over vocabulary* τ if the set of free symbols of ϕ is a subset of τ. It is a FO(ID)[τ]-formula if it does not contain any second-order quantifiers, and it is a SO(ID)[τ]-formula otherwise.

As an example, in the language of the natural numbers, the following SO(ID)[τ] formula, where $\tau = \{0, s/1\}$, expresses that there is a set which is the least set containing 0 and closed under taking successor numbers, and which contains all domain elements. It is equivalent to the standard induction axiom and to the domain closure axiom:

$$\exists N \left[\left\{ \begin{array}{l} \forall x\ (N(x) \leftarrow x = 0), \\ \forall x\ (N(s(x)) \leftarrow N(x)) \end{array} \right\} \wedge \forall x N(x) \right]. \tag{1}$$

Note that this formula contains an existential quantification over the second-order variable N. The second-order quantification can be avoided by skolemizing N and using a predicate constant instead. In fact, all examples of second-order quantification that appear in this paper, are of the same kind as in this example and can be eliminated in the same way, by skolemization of the existentially quantified second-order variable.

The semantics of the ID-logic is an extension of classical logic semantics with the well-founded semantics from logic programming [6,12,40]. We now define the well-founded model of a definition Δ extending a τ_Δ^{o}-structure I_{o}. For each defined symbol X of Δ, we define

$$\varphi_X(\bar{x}) := \exists \bar{y}_1(\bar{x} = \bar{t}_1 \wedge \varphi_1) \vee \cdots \vee \exists \bar{y}_m(\bar{x} = \bar{t}_m \wedge \varphi_m), \tag{2}$$

where $\bar{x}$ is a tuple of new variables, and $\forall \bar{y}_1(X(\bar{t}_1) \leftarrow \varphi_1), \ldots, \forall \bar{y}_m(X(\bar{t}_m) \leftarrow \varphi_m)$ are the rules of Δ with X in the head. For every defined symbol Y, we introduce a new relation symbol Y' of the same arity. We obtain φ'_X from $\varphi_X(\bar{x})$ by substituting Y' for each negative occurrence of each defined symbol Y.

For any pair of τ-structures I, J extending I_o, define I_J as the extension of I_o which interprets each defined symbol X of Δ as X^I, the value of X in I, and each new symbol X' as X^J, the value of X in J. The basis of the construction of the well-founded model extending I_o is the operator T_Δ which maps pairs I, J of extensions of I_o to a structure I', also extending I_o, such that for each defined symbol X, $X^{I'} := \{\bar{a} \mid I_J \models \varphi'_X[\bar{a}]\}$. Thus, the operator T_Δ evaluates positive occurrences of defined symbols in rule bodies by I, and negative occurrences of defined symbols by J.

In the lattice of τ-structures extending I_o, the operator T_Δ is monotone in its first argument and anti-monotone in its second argument. Define the *stable*[3] *operator* ST_Δ as follows: $ST_\Delta(J) := \mathit{lfp}(\mathsf{T}_\Delta(\cdot, J))$. This stable operator is anti-monotone, hence its square is monotone and has a least and largest fixpoint. We define ${I_o}^{\Delta\downarrow} := \mathit{lfp}(ST_\Delta^2)$, and ${I_o}^{\Delta\uparrow} := \mathit{gfp}(ST_\Delta^2)$.

For an intuitive explanation of the well-founded semantics and an argument why it formalizes different forms of inductive definitions, we refer to [6].

Definition 1. Definition Δ is *total* in τ_Δ^o-structure I_o if ${I_o}^{\Delta\downarrow} = {I_o}^{\Delta\uparrow}$. When this is the case, ${I_o}^{\Delta\downarrow}$ (or ${I_o}^{\Delta\uparrow}$) is called the *Δ-extension of* I_o and is abbreviated as ${I_o}^{\Delta}$. More generally, Δ is *total in a structure* K_o interpreting a subset of τ_Δ^o if Δ is total in each τ_Δ^o-structure extending K_o. Δ is *total in a theory T over* τ_Δ^o if Δ is total in each τ_Δ^o-model of T.

The aim of an inductive definition is to *define* its defined symbols. This is the case only when ${I_o}^{\Delta\downarrow} = {I_o}^{\Delta\uparrow}$. Therefore, a natural quality requirement for a definition is that it is total.

Definition 2 *(ϕ true in structure I).* Let ϕ be a ID-logic-formula and I any structure interpreting all free symbols of ϕ. We define $I \models \phi$ (in words, *ϕ is true in I*, or *I satisfies ϕ*, or I is a *model* of ϕ) by the following induction:

(1) $I \models X(t_1, \ldots, t_n)$ if $(t_1^I, \ldots, t_n^I) \in X^I$;
(2) $I \models \Delta$ if $I = {I_o}^{\Delta\downarrow} = {I_o}^{\Delta\uparrow}$, where I_o is the restriction of I to τ_Δ^o;
(3) $I \models \psi_1 \wedge \psi_2$ if $I \models \psi_1$ and $I \models \psi_2$;
(4) $I \models \neg\psi$ if $I \not\models \psi$;
(5) $I \models \exists\sigma\psi$ if for some value v of σ in the domain $\mathit{dom}(I)$ of I, $I[\sigma : v] \models \psi$.

Given an ID-logic theory[4] T over τ, a τ-structure *I satisfies T* (*is a model of T*) if I satisfies each $\phi \in T$. This is denoted by $I \models T$.

Definition 2 is a prototypical example of a non-monotone inductive definition, more specifically a definition over a well-founded poset, namely the set of ID-logic formulas ordered by the sub-formula relation. It contains non-monotone recursion in rule 4. This is an example of the sort of induction formalized in ID-logic.

As mentioned before, the definitional implication should be distinguished from material implication. Rule $\forall\bar{x}(X(\bar{t}) \leftarrow \varphi)$ in a definition does not correspond to the disjunction $\forall\bar{x}(X(\bar{t}) \vee \neg\varphi)$, although it implies it. Intuitively, the definitional implication should be understood as the "if" found in rules in inductive definitions (e.g. Definition 2 consists of 5 such rules).

A definitional implication always contains an atom in the head, never a negative literal. This reflects a general principle of inductive definitions, which is that one defines a concept by enumerating *positive cases*, i.e., cases in which a defined predicate is true. Given such an enumeration, the closure mechanism underlying inductive definitions yields the negative cases.

2.3. Analysing definitions

In this section, we recall the modularity, totality and translation theorems from [8].

Definitions in ID-logic are non-monotone, in the sense that adding a rule to a definition in general does not preserve logical consequence. As a consequence, splitting a definition into a conjunction of two or more parts is, in general,

[3] This operator is often called the Gelfond–Lifschitz operator and was introduced in [13].
[4] By a theory, we mean a finite set of axioms.

not equivalence preserving. This is quite obviously the case when we split up rules defining the same predicate. For example, the definition

$$\left\{\begin{array}{l}\forall x\ (P(x) \leftarrow x = a),\\ \forall x\ (P(x) \leftarrow x = b)\end{array}\right\}$$

and the conjunction of its partition

$$\left\{\forall x\ (P(x) \leftarrow x = a)\right\} \wedge \left\{\forall x\ (P(x) \leftarrow x = b)\right\}$$

are not equivalent. Indeed, in a Herbrand model I of the definition, $P^I = \{a, b\}$, and such a model satisfies neither the first component definition (since $P^I \neq \{a\}$) nor the second (since $P^I \neq \{b\}$).

This motivates the following definition.

Definition 3 *(partition of definitions).* A *partition of definition* Δ is a set $\{\Delta_1, \ldots, \Delta_n\}$, $1 < n$, such that $\Delta = \Delta_1 \cup \cdots \cup \Delta_n$, and if defined symbol P appears in the head of a rule of Δ_i, $1 \leqslant i \leqslant n$, then all rules of Δ with P in the head belong to Δ_i and only to Δ_i.

Notice that Δ_i has some "new" open symbols. For instance, if P is defined in Δ, but not in Δ_i, then it is a new open symbol of Δ_i. Of course, it holds that $\tau = \tau^{\mathrm{o}}_{\Delta} \cup \tau^{\mathrm{d}}_{\Delta} = \tau^{\mathrm{o}}_{\Delta_i} \cup \tau^{\mathrm{d}}_{\Delta_i}$, $1 \leqslant i \leqslant n$. Also, $\bigcup_i \tau^{\mathrm{d}}_{\Delta_i} = \tau^{\mathrm{d}}_{\Delta}$ and $\tau^{\mathrm{d}}_{\Delta_i} \cap \tau^{\mathrm{d}}_{\Delta_j} = \emptyset$ whenever $i \neq j$.

Even if we put all rules defining the same predicate in the same module, splitting may not be equivalence preserving. For example, in the unique model of the definition

$$\{P \leftarrow Q, Q \leftarrow P\}, \tag{3}$$

P and Q are false, whereas the conjunction

$$\{P \leftarrow Q\} \wedge \{Q \leftarrow P\}$$

has two models, one in which P and Q are both false and another in which they are both true. A non-trivial example of a splittable definition is the following simultaneous inductive definition of even and odd numbers:

$$\left\{\begin{array}{l}\forall x\ (E(x) \leftarrow x = 0),\\ \forall x\ (E(S(x)) \leftarrow O(x)),\\ \forall x\ (O(S(x)) \leftarrow E(x))\end{array}\right\}. \tag{4}$$

In the domain of the natural numbers (interpreting 0 by 0 and S by the successor function *Succ*), this definition can be shown to be equivalent to the conjunction

$$\left\{\begin{array}{l}\forall x\ (E(x) \leftarrow x = 0),\\ \forall x\ (E(S(x)) \leftarrow O(x))\end{array}\right\} \wedge \left\{\forall x\ (O(S(x)) \leftarrow E(x))\right\}. \tag{5}$$

By splitting this definition, we obtain two non-inductive definitions, one of even numbers, the other of odd numbers. Such non-inductive definitions can be translated to classical logic using our translation results:

$$\begin{array}{l}\forall x\ (E(x) \equiv x = 0 \vee \exists y\ (x = S(y) \wedge E(y))) \wedge\\ \forall x\ (O(x) \equiv \exists y\ (x = S(y) \wedge E(y))).\end{array} \tag{6}$$

This example illustrates the potential use of modularity and translation results for analysis of inductive definitions.

Whether a partition of a definition is equivalent to the original definition depends on whether the split breaks up circular dependencies between defined facts. For example, in the definition (3), P and Q mutually *depend* on each other. Splitting the definition breaks up this cycle, hence the equivalence is lost. In the definition (4), although E and O are defined in terms of each other, there are no cyclic dependencies at the level of atoms. I.e., an atom $E(n)$ depends only on $O(n-1)$, which in turn depends on $E(n-2)$, etc. So, by splitting the definition, no dependency cycles are broken and the equivalence is preserved. Below, the intuitive notion of a dependency relation between atoms is formalized by the concept of a *reduction relation*.

Assume a definition Δ over τ and a structure K_{o} with domain A such that $\tau_{K_{\mathrm{o}}} \subseteq \tau^{\mathrm{o}}_{\Delta}$. Let At^{τ}_{A} be the set of domain atoms over τ in domain A, i.e., the set of atoms $P[a_1, \ldots, a_n]$ (or $P[\bar{a}]$), where P is relation symbol of τ and

$a_1, \ldots, a_n$ are elements of A. Let $\prec$ be any binary relation on At^τ_A. If $Q[\bar{b}] \prec P[\bar{a}]$, we will say that $P[\bar{a}]$ depends on $Q[\bar{b}]$ (according to $\prec$). For any domain atom $P[\bar{a}]$ and any pair I, J of τ-structures with domain A, we define $I \cong_{\prec P[\bar{a}]} J$ if $f^I = f^J$ for every constant and function symbol f appearing in Δ, and for each atom $Q[\bar{b}] \prec P[\bar{a}]$, $I \models Q[\bar{b}]$ iff $J \models Q[\bar{b}]$. We extend this to pairs by defining $(I, J) \cong_{\prec P[\bar{a}]} (I', J')$ if $I \cong_{\prec P[\bar{a}]} I'$ and $J \cong_{\prec P[\bar{a}]} J'$.

Let $\varphi_P[\bar{a}]$ be as in (2).

Definition 4 *(reduction relation).* A binary relation $\prec$ on At^τ_A is a *reduction relation* (or briefly, a reduction) of a rule $\forall \bar{x}\ (P(\bar{t}[\bar{x}]) \leftarrow \varphi(\bar{x})) \in \Delta$ if for all τ-structures I, J, I', J' extending K_o, for all tuples $\bar{a}$ and $\bar{d}$ such that $\bar{a} = \bar{t}^{J[\bar{x}:\bar{d}]}$, if $(I, J) \cong_{\prec P[\bar{a}]} (I', J')$ then it holds that $I_J \models \varphi'[\bar{d}]$ iff $I'_{J'} \models \varphi'[\bar{d}]$.

The relation $\prec$ is a reduction relation of Δ in K_o if for each defined predicate P of Δ, $\prec$ is a reduction relation of the rule $\forall \bar{x}(P(\bar{x}) \leftarrow \varphi_P)$ in K_o.

Intuitively, the definition expresses that $\prec$ is a reduction relation if the truth of the formulas $\varphi_P[\bar{a}]$ depends only on the truth of the atoms on which $P[\bar{a}]$ depends according to $\prec$.

Proposition 1. *If $\prec$ is a reduction relation of a rule or of a definition Δ in K_o, then any superset of $\prec$ is also a reduction relation of that rule, resp. of Δ in K_o.*

Proposition 2. *Let $\prec$ be a reduction relation of each rule in Δ in K_o. Then $\prec$ is a reduction relation of Δ in K_o.*

The above propositions suggest a methodology for constructing reductions of a definition: define reductions for each of its rules and take the union. We illustrate this for definition (4) in the context of the natural numbers. We obtain a reduction for this definition as the union of reductions for each of its rules:

$$\begin{aligned}
&\emptyset \cup \\
&\{(E(n), O(n+1)), \mid n \in \mathbb{N}\} \cup \\
&\{(O(n+1), E(n+2)) \mid n \in \mathbb{N}\}.
\end{aligned} \tag{7}$$

Recall that a pre-well-founded order is a reflexive and transitive relation such that every non-empty subset contains a minimal element. The following definition is crucial for two decomposition theorems.

Definition 5 *(reduction partition).* Call partition $\{\Delta_1, \ldots, \Delta_n\}$ of definition Δ a *reduction partition* of Δ in τ^o_Δ-structure I_o if there is a reduction pre-well-founded order $\prec$ of Δ in I_o and if for each pair of defined domain atoms $P[\bar{a}]$, $Q[\bar{b}]$ such that $Q[\bar{b}] \prec P[\bar{a}]$ and $P[\bar{a}] \prec Q[\bar{b}]$, P and Q are defined in the same Δ_i.

The intuition underlying this definition is that in a reduction partition, if an atom defined in one module depends on an atom defined in another module, then the latter atom does not depend on the first atom and hence is strictly less in the reduction ordering.

A partition $\{\Delta_1, \ldots, \Delta_n\}$ of Δ is called total in K_o if each Δ_i is total in K_o.

Theorem 1 *(modularity, [10, Theorem 5.20]). If $\{\Delta_1, \ldots, \Delta_n\}$ is a reduction partition of Δ in K_o, then for any τ-structure M extending K_o, $M \models \Delta_1 \wedge \cdots \wedge \Delta_n$ iff $M \models \Delta$.*

Theorem 2 *(totality, [10, Theorem 5.24]). If $\{\Delta_1, \ldots, \Delta_n\}$ is a total reduction partition of Δ in K_o, then Δ is total in K_o.*

It is easy to see that the reduction relation (7) turns the partition in (5) into a reduction partition. Hence, in the context of the natural numbers the definition is equivalent to its partition. Moreover, since the two component definitions are non-inductive and hence, total, it follows that definition (4) is total. This illustrates the role of these theorems: they tell us when we can understand a large definition as a conjunction of smaller ones, and they allow us to prove totality of a large definition given the totality of smaller ones.

Suppose T_o is a theory over τ^o_Δ such that for any τ^o_Δ-model M_o of T_o, $\{\Delta_1, \ldots, \Delta_n\}$ is a reduction partition of Δ in M_o. Under this assumption, two important corollaries hold:

Corollary 1. *$T_o \wedge \Delta$ and $T_o \wedge \Delta_1 \wedge \cdots \wedge \Delta_n$ are logically equivalent.*

Corollary 2. *If in addition, $\{\Delta_1, \ldots, \Delta_n\}$ is a total partition in T_o, then Δ is total in T_o.*

Now we consider two special cases of definitions and explain how to translate them into classical logic. Let Δ be a positive definition, i.e., with only positive occurrences of defined symbols in rule bodies, defining the symbols $\bar{P}$. Let X_i and P_i have the same arity. Define

$$\mathit{PID}(\Delta) := \bigwedge \Delta \wedge \forall \bar{X}\Big(\bigwedge \Delta[\bar{P}/\bar{X}] \rightarrow \bar{P} \subseteq \bar{X}\Big)$$

and

$$\mathit{CIRC}(\Delta) := \bigwedge \Delta \wedge \forall \bar{X}\Big(\bigwedge \Delta[\bar{P}/\bar{X}] \rightarrow (\bar{X} \subseteq \bar{P} \rightarrow \bar{P} \subseteq \bar{X})\Big).$$

Here, $\bigwedge \Delta$ is the conjunction of formulas obtained by replacing definitional with material implications in Δ, $\Delta[\bar{P}/\bar{X}]$ is the definition obtained by substituting X_i for each defined symbol P_i and $\bar{P} \subseteq \bar{X}$ is a shorthand for the formula

$$\forall \bar{x}\ (P_1(\bar{x}) \supset X_1(\bar{x})) \wedge \cdots \wedge \forall \bar{x}\ (P_n(\bar{x}) \supset X_n(\bar{x})).$$

The formula $\mathit{PID}(\Delta)$ is the standard second-order formula to express that predicates $\bar{P}$ satisfy the positive inductive definition Δ. The theory $\mathit{PID}(\Delta)$ expresses that the defined relations are the least relations closed under the rules of Δ in a τ_Δ^{o}-structure. Because the rules of Δ are positive, such least relations are guaranteed to exist. The theory $\mathit{CIRC}(\Delta)$ is a circumscription axiom [23] and expresses that the defined relations are minimal relations closed under the rules of Δ in a τ_Δ^{o}-structure. Since the least relations closed under the rules of Δ are the unique minimal relations closed under the rules of Δ, both formulas are equivalent.

Theorem 3. *(See [10, Theorem 6.3].) If Δ is positive (i.e., contains no negative occurrences of defined symbols in rule bodies) then it is total in each τ_Δ^{o}-structure, and for any τ-structure I, $I \models \Delta$ iff $I \models \mathit{PID}(\Delta)$ iff $I \models \mathit{CIRC}(\Delta)$.*

These results can be applied, for example, in the context of definition (4) of even and odd numbers. Another result is concerned with (possibly non-monotone) definitions over well-founded posets. First, we propose a formalization for this informal concept in ID-logic.

Definition 6 *(strict reduction relation).* A reduction relation $\prec$ of Δ on At_A^τ is *strict* in K_{o} if it is a strict well-founded order (i.e., an antisymmetric, transitive binary relation without infinite descending chains).

Thus, if $P[\bar{a}] \prec Q[\bar{b}]$ holds, then the bodies of rules defining $Q[\bar{b}]$ may depend on the truth value of $P[\bar{a}]$, but not vice versa.

Definition 7 *(definition by well-founded induction).* Let Δ be a definition with a strict reduction relation $\prec$ in K_{o}. We call Δ a definition by well-founded induction (over $\prec$) in K_{o}.

This type of definitions can be formalized in first-order logic using the well-known concept of completion [3]. Define the *completion of* Δ, denoted $comp(\Delta)$, as the conjunction, for each defined symbol X of Δ, of formulas $\forall \bar{x}(X(\bar{x}) \equiv \varphi_X(\bar{x}))$.

In general, a definition Δ entails its completion $comp(\Delta)$ but not vice versa. However, in case of a definition by well-founded induction, the inverse is true as well.

Theorem 4 *(completion, [10, Theorem 6.9]). Suppose Δ is a definition by well-founded induction in τ_Δ^{o}-structure I_{o}. Then* (a) *definition Δ is total in I_{o}, and* (b) *the model $I_{\mathrm{o}}{}^\Delta$ of Δ is the unique model of $comp(\Delta)$ extending I_{o}.*

An example of a definition by well-founded induction is the definition (4) in the structure $\langle \mathbb{N}, 0, \mathit{Succ}\rangle$. Indeed, the transitive closure of the reduction relation in (7) is a strict well-founded order and a reduction relation of definition (4). As a consequence, this definition and its completion, formula (6), are equivalent in this structure: each model of the definition extending $\langle \mathbb{N}, 0, \mathit{Succ}\rangle$ is a model of its completion and vice versa.

One observes that the above theorem does not prove logical equivalence, i.e., equivalence in all structures, but only equivalence in the context of a given structure. In fact, a definition may have a strict reduction relation in one structure and not in other structures. Consider for example the structure K_o with domain $\{a,b\}$ and $0^{K_o} = a$, and $S^{K_o} = \{(a,a),(b,b)\}$. In this structure, the least reduction relation of definition (4) is

$$\{(E(a),O(a)),(O(a),E(a)),(E(b),O(b)),(O(b),E(b))\}.$$

Each other reduction is a superset of this relation. Since each transitive relation extending this relation contains $(E(a),E(a))$, the definition has no strict reduction relation in this structure. In fact, the definition (4) and its completion (6) are not equivalent in this structure. The unique model I of the definition interprets $E^I = \{a\}$ and $O^I = \{a\}$. On the other hand, the completion has an additional model in which $E^I = \{a,b\}$ and $O^I = \{a,b\}$.

Suppose T_o is a theory over τ^o_Δ such that for any τ^o_Δ-model M_o of T_o, Δ is a definition by well-founded induction in M_o. Under this assumption, two useful corollaries hold:

Corollary 3. *$T_o \wedge \Delta$ and $T_o \wedge comp(\Delta)$ are logically equivalent.*

Corollary 4. *Δ is total in T_o.*

Notice that positive inductive definitions and definitions by well-founded induction have different (and, in general, non-equivalent) formalizations in classical logic.

Some of the techniques that were introduced here are similar to methods found in logic programming. For example, Theorem 4 shows similarity to Fages theorem [11]. However, Fages notion of tight program is not equivalent to our notion of definition over a well-founded order; Fages theorem is only for Herbrand interpretations, and relates completion with stable semantics while ours is for general interpretations and relates completion with well-founded semantics.

3. Reiter-style situation calculus

From now on, we are dealing with many-sorted logic. All results and definitions introduced so far easily extend to this case. The vocabulary τ_{sc} of the situation calculus is a many-sorted vocabulary with equality and with sorts for actions (*Act*), situations (*Sit*), and possibly a finite number of domain-specific sorts called object sorts ($Ob_1,\dots,Ob_k$), where each Ob_i is an arbitrary name. The vocabulary contains a potentially infinite set of domain-dependent function symbols of the sort *Act*. The sort of each argument of such a function is an object sort. For example, in the block world domain, we may have actions $pick\_up(x)$ and $put\_on(x,y)$ ranging over the sort *Block*.

The vocabulary contains a binary relation $\sqsubseteq$ with arguments of sort *Sit* and denoting precedence of situations. The constant S_0 of sort *Sit* denotes the initial situation. Function *do* of sort *Sit* maps actions and situations to situations, i.e., given a and s, term $do(a,s)$ denotes the successor situation which is obtained from situation s by performing action a. The predicate constants $F_1, F_2, \dots$ are called *fluents* and denote properties of the world (both in the initial situation and in other situations). Fluents always have exactly one argument of sort *Sit*, while the sort of each other argument is an object sort. For example, $On(x,y,s)$ of arity 3 denotes that object x is on object y in situation s.

Definition 8 *($\mathcal{D}_{\mathrm{una}(S)}$).* The theory of *unique name axioms for sort S*, $\mathcal{D}_{\mathrm{una}(S)}$, is the set of axioms in the following axiom schema: for distinct function symbols f and g of sort S

$$\forall\bar{x}\forall\bar{y}\ \neg(f(\bar{x}) = g(\bar{y})) \tag{8}$$

$$\forall\bar{x}\forall\bar{y}\ (f(x_1,\dots,x_n) = f(y_1,\dots,y_n) \supset x_1 = y_1 \wedge \cdots \wedge x_n = y_n). \tag{9}$$

The axioms (9) hold for every function symbol f with arity greater than zero.

Definition 9 *($\mathcal{D}_f$).* The *foundational axioms of the situation calculus*, $\mathcal{D}_f$, are the set of axioms consisting of the unique name axioms for situations $\mathcal{D}_{\mathrm{una}(Sit)}$, the domain closure axiom for situations

$$\forall P\ (P(S_0) \wedge \forall s'\forall a\ (P(s') \supset P(do(a,s'))) \supset \forall s P(s)) \tag{10}$$

and the precedence axioms for situations

$$\forall s\, \neg(s \sqsubset S_0), \tag{11}$$

$$\forall s \forall s' \forall a\ (s \sqsubset do(a, s') \equiv s \sqsubseteq s') \tag{12}$$

where $s \sqsubset s'$ is an abbreviation for $s \sqsubseteq s' \wedge \neg(s' \sqsubseteq s)$.

The role of axiom (10) is to guarantee that the domain of situations *Sit* is the smallest set closed under applications of the function symbol *do*, which satisfies the unique name axioms for situations. Every two models of $\mathcal{D}_{\mathrm{f}}$ with identical domains of sort *Act* will have identical domains of sort *Sit* (modulo isomorphism).

Definition 10 *($\mathcal{D}_{\mathrm{ss}}$).* The *successor state axioms*, $\mathcal{D}_{\mathrm{ss}}$, are of the form:

$$\forall \bar{x} \forall a \forall s\ (F(\bar{x}, do(a, s)) \equiv (\gamma_F^+(\bar{x}, a, s) \vee F(\bar{x}, s) \wedge \neg\gamma_F^-(\bar{x}, a, s))). \tag{13}$$

Formula $\gamma_F^+(\bar{x}, a, s)$ (respectively, $\gamma_F^-(\bar{x}, a, s)$) denotes a first-order formula specifying the conditions under which action a causes fluent F to become true (respectively, false) in the situation s [33]. The only free variables of these formulas are among $\bar{x}$, a, s and the only symbol of sort *Sit* is the free variable s. An example of a successor axiom is

$$\begin{aligned}&\forall sw \forall a \forall s\ (On(sw, do(a, s)) \equiv\\ &\quad a = toggle(sw) \wedge \neg On(sw, s) \vee\\ &\quad On(sw, s) \wedge a \neq toggle(sw)).\end{aligned}$$

This axiom says that a switch is on in situation $do(a, s)$ if and only if this situation was obtained by performing action $toggle(sw)$ in situation s where this switch was off, or the switch was already on and an action other than $toggle(sw)$ was performed.

Successor state axioms are obtained from a set of *effect rules* of the form:

$$\forall \bar{x} \forall a \forall s\ (\delta_F^i(\bar{x}, a, s) \supset F(\bar{x}, do(a, s))) \tag{14}$$

for $i \in \{1, \dots, k\}$, and

$$\forall \bar{x} \forall a \forall s\ (\nu_F^j(\bar{x}, a, s) \supset \neg F(\bar{x}, do(a, s))), \tag{15}$$

for $j \in \{1, \dots, m\}$, where formulas δ_F^i and ν_F^j satisfy the same conditions as γ_F^+ and γ_F^-. Each of these rules specifies an initiating or terminating effect of a particular action in one particular condition. Together these rules exhaustively describe all effects. Effect rules are transformed into the successor state axioms using the following equations:

$$\gamma_F^+(\bar{x}, a, s) := \bigvee_{i=1}^{k} \delta_F^i(\bar{x}, a, s) \quad \text{and} \quad \gamma_F^-(\bar{x}, a, s) := \bigvee_{j=1}^{m} \nu_F^j(\bar{x}, a, s).$$

This transformation is not equivalence preserving but transforms an incomplete specification of the action domain into a final axiomatization.

Definition 11 *($\mathcal{D}_{S_0}$).* A *description of the initial situation*, $\mathcal{D}_{S_0}$, is a set of first-order sentences that are uniform in S_0, that is, it contains no situation term other than S_0.

A *basic action theory* consists of $\mathcal{D}_{\mathrm{f}} \cup \mathcal{D}_{\mathrm{una}(Act)} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{\mathrm{ss}}$.

4. Inductive situation calculus

In this section, we define a variant of Reiter-style situation calculus, which we call the *inductive situation calculus*. All fluents will be defined by simultaneous induction on the well-founded set of situations. Ramifications describing propagation of effects of actions are modeled as monotone or non-monotone inductions at the level of situations. The result is an iterated inductive definition with alternating phases of monotone and non-monotone induction. Below we describe the components of the inductive situation calculus.

The vocabulary τ_{isc} of the inductive situation calculus extends τ_{sc} by two types of symbols. Symbols $I_{F_1}, I_{F_2}, \ldots$ are used to describe the initial situation and correspond to the fluents $F_1, F_2, \ldots, F_n$, but have no situation argument. They are open symbols of the inductive situation calculus. The other type of symbols denotes causality relations. These symbols will be introduced a bit later. The *initial state vocabulary* τ_{init} consists of all symbols of τ_{isc} not involving sorts *Act* or *Sit*. The *open vocabulary* $\tau_{\text{isc}}^{\text{o}}$ of the inductive situation calculus extends τ_{init} with S_o, *do*, $\sqsubseteq$, and the action symbols. This vocabulary consists of all symbols of τ_{isc} except for all fluents and causality predicates.

The ID-logic induction axiom (1) of the natural numbers can be extended to arbitrary sorts in the following way.

Definition 12 *($\mathcal{D}_{\text{DCA}(S)}$).* Given a vocabulary τ, the *domain closure axiom for sort* S, $\mathcal{D}_{\text{DCA}(S)}$, is the axiom:

$$\exists P\left[\left\{\begin{array}{l}\forall x\ (P(x) \leftarrow x = c),\\ \ldots\\ \forall x_1 \ldots \forall x_n\ (P(f(x_1, \ldots, x_n)) \leftarrow P(x_{i_1}) \wedge \cdots \wedge P(x_{i_m}))\\ \ldots\end{array}\right\} \wedge \forall x P(x)\right]$$

where the definition contains one rule for every constant c of sort S and for every function symbol $f \in \tau$ of sort S with arguments $i_1, \ldots, i_m$ of sort S.

For example, the domain closure axiom $\mathcal{D}_{\text{DCA}(Sit)}$ for situations is:

$$\exists P\left[\left\{\begin{array}{l}\forall s\ (P(s) \leftarrow s = S_0),\\ \forall a \forall s\ (P(do(a, s)) \leftarrow P(s))\end{array}\right\} \wedge \forall s P(s)\right]. \tag{16}$$

The role of axiom (16) is to guarantee that the domain of situations *Sit* is the smallest set containing S_0 and closed under applications of the function symbol *do*. It is equivalent to Reiter's induction axiom for situations. Recall that the second order variable can be eliminated by skolemization.

A general property of the combination of unique names axioms and domain closure axiom is that they are consistent and fix the domain in a unique way.

Proposition 3. *Let τ_o be a sorted vocabulary, and τ extends τ_o with a new sort S and a set of constant and function symbols of sort S. For every τ_o-structure I_o, there is a non-empty class of τ-structures extending I_o and satisfying $\mathcal{D}_{\text{una}(S)}$, and there is a unique (modulo isomorphism) extension of I_o satisfying $\mathcal{D}_{\text{una}(S)} \wedge \mathcal{D}_{\text{DCA}(S)}$.*

The *foundational axioms of the inductive situation calculus*, $\mathcal{D}_{\text{if}}$, are the unique name axioms $\mathcal{D}_{\text{una}(Sit)}$ for situations, the domain closure axiom $\mathcal{D}_{\text{DCA}(Sit)}$ for situations and the following definition of the precedence relation

$$\Delta_{\sqsubseteq} := \left\{\begin{array}{l}\forall s \forall s'\ (s \sqsubseteq s' \leftarrow s = s'),\\ \forall s \forall s' \forall a\ (s \sqsubseteq do(a, s') \leftarrow s \sqsubseteq s')\end{array}\right\}. \tag{17}$$

An *initial structure* $\mathcal{A}$ of the inductive situation calculus is a multi-sorted structure with a non-empty domain for each sort of the language, which interprets all symbols of $\tau_{\text{isc}}^{\text{o}}$ and which satisfies the foundational axioms $\mathcal{D}_{\text{if}}$.

Proposition 4. *For any τ_{init}-structure I_o and arbitrary extension I of I_o to (the symbols of) sort Act, there is a unique (modulo isomorphism) initial structure $\mathcal{A}$ extending I. I_o has a unique extension satisfying $\mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{DCA}(Act)}$.*

Proof. By application of Proposition 3 for sort *Sit*, we can prove that an arbitrary extension I of I_o to sort *Act*, has a unique (modulo isomorphism) $(\tau_{\text{isc}}^{\text{o}} \setminus \{\sqsubseteq\})$-structure satisfying $\mathcal{D}_{\text{una}(Sit)} \cup \mathcal{D}_{\text{DCA}(Sit)}$. By Theorem 3, this structure can be extended in a unique way to a definition of $\Delta_{\sqsubseteq}$. By, again, Proposition 3, only one of these extensions satisfies $\mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{DCA}(Act)}$. □

Proposition 5. *Let $\mathcal{A}$ be an initial structure. In every such structure, the substructure $\langle Sit^{\mathcal{A}}, \sqsubseteq^{\mathcal{A}}\rangle$ is a well-founded poset (and, thus a pre-well-founded set).*

Proof. In an initial structure $\mathcal{A}$, the collection of situations is isomorphic to the set of finite sequences of elements of sort *Act*, where $S_0^{\mathcal{A}}$ corresponds to the empty sequence, and $do^{\mathcal{A}}$ to the constructor appending an action object to a finite sequence. Since $\mathcal{A}$ satisfies the definition (17), it holds for two situations s, s' that $s \sqsubseteq^{\mathcal{A}} s'$ iff the sequence of s is an initial segment of that of s'. This is a well-founded order. □

The following proposition demonstrates that the remaining two foundational axioms of the situation calculus as presented in [34], are implied by the definition above.

Proposition 6. *The theories $\mathcal{D}_{\mathrm{f}}$ and $\mathcal{D}_{\mathrm{if}}$ are logically equivalent.*

Proof. Both theories contain $\mathcal{D}_{\mathrm{una}(Sit)}$. The induction axiom of $\mathcal{D}_{\mathrm{f}}$ is equivalent to $\mathcal{D}_{\mathrm{DCA}(Sit)}$ in $\mathcal{D}_{\mathrm{if}}$. It is not difficult to prove that in a structure $\mathcal{A}$ satisfying $\mathcal{D}_{\mathrm{una}(Sit)} \cup \mathcal{D}_{\mathrm{DCA}(Sit)}$, where situations correspond to finite sequences of actions, the unique relation $\sqsubseteq^{\mathcal{A}}$ that satisfies sentences (11) and (12) is the same relation defined by definition (17). □

In place of $\mathcal{D}_{S_0}$, the description of the initial situation in terms of fluents which hold in S_0, in the inductive situation calculus we describe the initial situation in terms of symbols I_{F_i}. The corresponding collection of axioms is $\mathcal{D}_{\mathrm{init}}$. This is any theory in the vocabulary τ_{init}.

A *basic action theory of the inductive situation calculus* will be a collection of axioms of the form:

$$\mathcal{D}_{\mathrm{if}} \cup \mathcal{D}_{\mathrm{una}(Act)} \cup \mathcal{D}_{\mathrm{init}} \cup \{\Delta_{\mathrm{sc}}\}, \tag{18}$$

where Δ_{sc} is an inductive definition of the fluents. We will often include also the domain closure $\mathcal{D}_{\mathrm{DCA}(Act)}$ for actions in an inductive situation calculus theory. In the next sections, we present three variants of Δ_{sc}.

4.1. Specifying direct effects of actions

For each fluent F_i, we introduce two additional auxiliary relations, C_{F_i} and $C_{\neg F_i}$. These relations represent initiating and terminating causes for F_i, respectively. Both C_{F_i} and $C_{\neg F_i}$ have the same sort of arguments as F_i plus one action argument. Let $\mathcal{D}_{\mathrm{init}}$ axiomatize the initial situation using $I_{F_1}, \ldots, I_{F_m}$.

We augment $\mathcal{D}_{\mathrm{if}} \cup \mathcal{D}_{\mathrm{una}(Act)} \cup \mathcal{D}_{\mathrm{init}}$ with the following definition

$$\Delta_{\mathrm{sc}} = \bigcup_{i=1}^{n} \Delta^i_{\mathrm{fluent}} \cup \bigcup_{i=1}^{n} \Delta^i_{\mathrm{effect}}$$

where

$$\Delta^i_{\mathrm{fluent}} := \left\{\begin{array}{l} \forall \bar{x}_i\ (F(\bar{x}_i, S_0) \leftarrow I_F(\bar{x}_i)), \\ \forall \bar{x}_i \forall a \forall s\ (F_i(\bar{x}_i, do(a,s)) \leftarrow C_{F_i}(\bar{x}_i, a, s)), \\ \forall \bar{x}_i \forall a \forall s\ (F_i(\bar{x}_i, do(a,s)) \leftarrow F_i(\bar{x}_i, s) \wedge \neg C_{\neg F_i}(\bar{x}_i, a, s)) \end{array}\right\},$$

$$\Delta^i_{\mathrm{effect}} := \left\{\begin{array}{l} \forall \bar{x}_i \forall a \forall s\ (C_{F_i}(\bar{x}_i, a, s) \leftarrow \gamma^+_{F_i}(\bar{x}_i, a, s)), \\ \forall \bar{x}_i \forall a \forall s\ (C_{\neg F_i}(\bar{x}_i, a, s) \leftarrow \gamma^-_{F_i}(\bar{x}_i, a, s)) \end{array}\right\}.$$

Here, formulas $\gamma^+_{F_i}(\bar{x}_i, a, s)$, $\gamma^-_{F_i}(\bar{x}_i, a, s)$ are analogous to those found in Reiter-style situation calculus. They are formulas without causality predicates, with free variables among $\bar{x}_i, a, s$ and the only symbol of sort *Sit* is the free variable s. All fluent atoms in such formulas are of the form $F_j(\bar{t}, s)$.

The intuitive meaning of this definition is as follows. The first rule of $\Delta^i_{\mathrm{fluent}}$ defines the fluent F_i in situation S_0. The second rule says that if an action causes a fluent in some situation, then the fluent holds in the successor situation. The third rule deals with the case where a fluent is not affected by an action and will be referred to as the *law of inertia*. The rules in $\Delta^i_{\mathrm{effect}}$ describe direct effects of actions on the fluent F_i.

Note that any fluent may appear in the rules for a causality predicate. Hence, the definition Δ_{sc} is one large simultaneous inductive definition. Moreover, since the inertia law contains a negative occurrence of $C_{\neg F_i}$, and this predicate may be defined in terms of fluents, Δ_{sc} is, in general, a non-monotone inductive definition. In what follows, we use the results of Section 2.3 in order to obtain the standard successor state axioms of the situation calculus.

Proposition 7. *The definition Δ_{sc} is a definition by well-founded induction in each $\tau^{\mathrm{o}}_{\mathrm{isc}}$-model of the foundational axioms $\mathcal{D}_{\mathrm{if}}$.*

Proof. Let $\mathcal{A}$ be an initial structure with domain A. We shall construct a reduction relation $\prec$ of Δ_{sc} in $\mathcal{A}$. This binary relation on $At_A^{\tau_{\text{isc}}}$, the set of all domain atoms, should represent (at least) all potential dependencies between domain atoms that are not interpreted by $\mathcal{A}$, i.e., between the fluent and causality atoms. We construct $\prec$ in a rule-by-rule way, suggested by Proposition 2, as the set of all tuples:

$$(C_{F_i}[\bar{u}, a, s], F_i[\bar{u}, do^{\mathcal{A}}(a, s)]),$$
$$(F_i[\bar{u}, s], F_i[\bar{u}, do^{\mathcal{A}}(a, s)]), (C_{\neg F_i}[\bar{u}, a, s], F_i[\bar{u}, do^{\mathcal{A}}(a, s)]),$$
$$(F_i[\bar{u}, s], C_{(\neg)F_j}[\bar{v}, a, s]),$$

for arbitrary tuples of objects $\bar{u}$ and $\bar{v}$, for arbitrary elements a of the action sort and s of the situation sort, for each i, j.

It is easy to see that the tuples in the first two lines provide a reduction relation of the two inductive rules of fluents, while the tuples in the third line represent all possible dependencies in direct effect rules. It follows from Proposition 2 that $\prec$ is a reduction relation of Δ_{sc} in $\mathcal{A}$.

Since, by Proposition 1, any superset of a reduction relation is also a reduction relation, the transitive closure $\prec^*$ of $\prec$ is a reduction relation. Moreover, it follows from the fact that $\langle Sit^{\mathcal{A}}, \sqsubseteq^{\mathcal{A}}\rangle$ is a well-founded set (Proposition 5), that $\prec^*$ is a strict well-founded order on $At_A^{\tau_{\text{isc}}}$. □

This proposition, together with Corollary 3, has an interesting consequence.

Proposition 8. *The theories $\mathcal{D}_{\text{if}} \wedge \Delta_{\text{sc}}$ and $\mathcal{D}_{\text{if}} \wedge comp(\Delta_{\text{sc}})$ are logically equivalent.*

We now formulate the property mentioned in the introduction of this paper.

Definition 13. We say that a basic action theory $\mathcal{D}_{\text{isc}} := \mathcal{D}_{\text{if}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{init}} \cup \{\Delta_{\text{sc}}\}$ of the inductive situation calculus satisfies the *Initial State Expansion property* if for every τ_{init}-model I_o of $\mathcal{D}_{\text{init}}$, for arbitrary extension I of I_o to (symbols of) sort Act, if I satisfies $\mathcal{D}_{\text{una}(Act)}$, then there is a unique τ-model (modulo isomorphism) of $\mathcal{D}_{\text{isc}}$ which extends I.

When $\mathcal{D}_{\text{isc}}$ satisfies the Initial State Expansion property, then, in particular, every τ_{init}-model I_o of $\mathcal{D}_{\text{init}}$ has a unique τ-extension (modulo isomorphism) satisfying $\mathcal{D}_{\text{isc}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{DCA}(Act)}$.

As argued in Section 1, the initial state expansion property shows that the inductive situation calculus satisfies the property of relative satisfiability and correctly solves the frame problem in the sense that a basic action theory has no unintended models with spontaneous generation of effects (or "deus ex machina" effects, as they were called in [9]) of the kind that occurred in early solutions to the frame problem. Another of its implications is that the subtheory modeling the state transitions is *logically independent* of the theory of the initial state and does not interfere with it in determining what are the initial states. Clearly, it would be most unpleasant if describing the effects of actions would somehow impose constraints on the initial state.

Proposition 9. *A basic action theory of the inductive situation calculus* (18) *satisfies the Initial State Expansion property.*

Proof. By Proposition 4, each extension of a τ_{init}-structure to sort Act can be extended in a unique initial structure $\mathcal{A}$. By Corollary 4, the extension of $\mathcal{A}$ that satisfies Δ_{sc} is unique. □

We now investigate the relationship to Reiter's situation calculus.

For a given vocabulary τ, let $M|_\tau$ denote the restriction of the structure M to the symbols of τ.

Definition 14. Suppose that τ_1, τ_2 are vocabularies extending τ, and let T_1, T_2 be theories in respectively τ_1, τ_2. We call T_1 *equivalent in τ to T_2* if for each τ_1-model M_1 of T_1, there exists a τ_2-model M_2 of T_2 such that $M_1|_\tau = M_2|_\tau$ and vice versa.

Recall that τ_{isc} extends τ_{sc} with the new symbols I_{F_i}, C_{F_i} and $C_{\neg F_i}$.

Theorem 5. *A basic action theory of the inductive situation calculus*

$$\mathcal{D}_{\text{if}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{init}} \cup \{\Delta_{\text{sc}}\}$$

is equivalent in τ_{sc} to the Reiter-style basic action theory

$$\mathcal{D}_{\text{f}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{\text{ss}},$$

where $\mathcal{D}_{S_0}$ is the theory obtained from $\mathcal{D}_{\text{init}}$ by substituting $F_i(\bar{t}, S_0)$ for each atom $I_{F_i}(\bar{t})$ and $\mathcal{D}_{\text{ss}}$ is the set of the successor state axioms corresponding to Δ_{sc}.

Proof. $\mathcal{D}_{\text{if}}$ and $\mathcal{D}_{\text{f}}$ are logically equivalent. By Proposition 8, $\mathcal{D}_{\text{f}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{init}} \cup \{\Delta_{\text{sc}}\}$ is logically equivalent to $\mathcal{D}_{\text{f}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{init}} \cup comp(\Delta_{\text{sc}})$, where $comp(\Delta_{\text{sc}})$ is

$$\begin{aligned}
&\bigwedge_{i=1}^{n} \forall \bar{x}_i \forall s\ F_i(\bar{x}_i, s) \equiv (s = S_0 \wedge I_{F_i}(\bar{x}_i) \vee \\
&\quad \exists a \exists s'\ s = do(a, s') \wedge (C_{F_i}(\bar{x}_i, a, s') \vee F_i(\bar{x}_i, s') \wedge \neg C_{\neg F_i}(\bar{x}_i, a, s'))) \\
&\quad \wedge \bigwedge_{i=1}^{n} \forall \bar{x}_i \forall a \forall s\ C_{F_i}(\bar{x}_i, a, s) \equiv \gamma^{+}_{F_i}(\bar{x}_i, a, s) \\
&\quad \wedge \bigwedge_{i=1}^{n} \forall \bar{x}_i \forall a \forall s\ C_{\neg F_i}(\bar{x}_i, a, s) \equiv \gamma^{-}_{F_i}(\bar{x}_i, a, s).
\end{aligned} \tag{19}$$

Since, by the domain closure axiom for situations,

$$\forall s (s = S_0 \vee \exists a \exists s' s = do(a, s')),$$

$\mathcal{D}_{\text{f}} \cup \{(19)\}$ is logically equivalent to $\mathcal{D}_{\text{f}} \cup \{(20), (21)\}$, where

$$\bigwedge_{i=1}^{n} \forall \bar{x}_i \forall s \forall a\ F_i(\bar{x}_i, do(a, s)) \equiv C_{F_i}(\bar{x}_i, do(a, s)) \vee F_i(\bar{x}_i, s) \wedge \neg C_{\neg F_i}(\bar{x}_i, do(a, s)) \tag{20}$$

and

$$\begin{aligned}
&\bigwedge_{i=1}^{n} \forall \bar{x}_i\ I_{F_i}(\bar{x}_i) \equiv F_i(\bar{x}_i, S_0) \\
&\quad \wedge \bigwedge_{i=1}^{n} \forall \bar{x}_i \forall s \forall a\ C_{F_i}(\bar{x}_i, a, s) \equiv \gamma^{+}_{F_i}(\bar{x}_i, a, s) \\
&\quad \wedge \bigwedge_{i=1}^{n} \forall \bar{x}_i \forall s \forall a\ C_{\neg F_i}(\bar{x}_i, a, s) \equiv \gamma^{-}_{F_i}(\bar{x}_i, a, s).
\end{aligned} \tag{21}$$

Given the equivalences in (21), it is clear that $\mathcal{D}_{\text{if}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{init}} \cup \{(20), (21)\}$ is logically equivalent to $\mathcal{D}_{\text{f}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{\text{ss}} \cup \{(21)\}$.

Finally, observe that in the latter theory, the predicate symbols I_{F_i}, C_{F_i} and $C_{\neg F_i}$ occur only at the left-hand side of the explicit definitions in (21). It follows that $\mathcal{D}_{\text{f}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{\text{ss}} \cup \{(21)\}$ is equivalent in τ_{sc} to $\mathcal{D}_{\text{f}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{\text{ss}}$. □

Note that our definition $\Delta^{i}_{\text{fluent}}$ does not contain rules of the form

$$\forall \bar{x}_i \forall a \forall s\ (\neg F_i(\bar{x}_i, a, s) \leftarrow C_{\neg F_i}(\bar{x}_i, a, s)). \tag{22}$$

However, under a natural requirement, we can derive negative effect axioms of actions, as we demonstrate below. The requirement is that a fluent and its negation are not caused to hold in the same situation. Formally, the requirement is that the basic action theory should entail the following sentence:

$$\bigwedge_{i=1}^{n} \forall \bar{x}_i \forall a \forall s\ \neg(\gamma^{+}_{F_i}(\bar{x}_i, a, s) \wedge \gamma^{-}_{F_i}(\bar{x}_i, a, s)).$$

It is easy to show now that if this requirement is satisfied, then the negative effect axiom is implied. Observe that each successor state axiom entails

$$\forall \bar{x}_i \forall a \forall s\ (\neg\gamma^{+}_{F_i}(\bar{x}_i, do(a, s)) \wedge \gamma^{-}_{F_i}(\bar{x}_i, do(a, s)) \supset \neg F_i(\bar{x}_i, do(a, s))).$$

Under the requirement, the first literal in the condition is entailed by the second, so we can drop it and we obtain the negative effect rule

$$\forall \bar{x}_i \forall a \forall s\ (\gamma^{-}_{F_i}(\bar{x}_i, do(a, s)) \supset \neg F_i(\bar{x}_i, do(a, s))).$$

Therefore, in the context of Inductive Situation Calculus, rule (22) is not necessary. This observation illustrates a principle of inductive definitions which was mentioned in Section 2.2. In an inductive definition, one defines a concept by enumerating positive cases. Given such an enumeration, the closure mechanism underlying inductive definitions yields the negative cases.

Recall that in Reiter's situation calculus, the successor state axioms, in particular the formulas $\gamma^{+}_{F}(\bar{x}, a, s)$ and $\gamma^{-}_{F}(\bar{x}, a, s)$ are obtained by compiling a set of effect rules of the form $\forall \bar{x} \forall a \forall s\ (\delta^{i}_{F}(\bar{x}, a, s) \supset F(\bar{x}, do(a, s)))$ and $\forall \bar{x} \forall a \forall s\ (\nu^{j}_{F}(\bar{x}, a, s) \supset \neg F(\bar{x}, do(a, s)))$. In ID-logic, the unique rule defining C_F can be replaced by a set of definitional implications:

$$\forall \bar{x} \forall a \forall s\ (C_F(\bar{x}, a, s) \leftarrow \delta^{i}_{F}(\bar{x}, a, s)).$$

Likewise, the predicate $C_{\neg F}$ can be defined directly by a set of effect rules of the form:

$$\forall \bar{x} \forall a \forall s\ (C_{\neg F}(\bar{x}, a, s) \leftarrow \nu^{j}_{F}(\bar{x}, a, s)).$$

Note that in Reiter's situation calculus, the compilation of effect rules into successor state axioms is crucial to obtain a correct axiomatization of the action domain. While each effect axiom is correct independently, together they are too weak to axiomatize the action domain, i.e., there are many unintended models. The compilation into successor state axioms modifies the meaning of the theory and eliminates all unintended models. The transformation turns an incorrect (in the sense of too weak) theory into a correct theory (provided the set of effect axioms is correct and complete).

In the inductive situation calculus, the situation is very different. Indeed, substituting all definitional effect rules defining C_F by the unique rule

$$\forall \bar{x} \forall a \forall s\ \left(C_F(\bar{x}, a, s) \leftarrow \bigvee_{i=1}^{k} \delta^{i}_{F}(\bar{x}, a, s)\right)$$

is equivalence preserving, and likewise for $C_{\neg F}$. Compilation is not necessary anymore to obtain a correct representation! It is this phenomenon that we had in mind in the introduction when we claimed that *Reiter-style situation calculus contains hidden forms of definitions*.

4.2. Indirect effects

The ramification problem arises when one wants to capture indirect effects of actions in a logic-based formalism. It has been shown (e.g., [19]) that state constraints are generally inadequate for deriving indirect effects of actions, and that some notion of causation is needed. Unlike material implication, causal implications are not contrapositive which makes them similar to the rules of inductive definitions. This correspondence between inductive definition rules and causal rules is the foundation of our solution to the ramification problem. The semantic correspondence between causality rules and rules in an inductive definition was independently pointed out in [36,37] and in [9]. The resulting definition Δ_{sc} is not a definition by well-founded induction but, in general, an iterated inductive definition.

Let, as before, C_{F_i} and $C_{\neg F_i}$ represent initiating and terminating causes for F_i, respectively. We extend the use of the causality predicates to specify indirect effects of actions. For example, according to the causal rule $\forall a \forall s(C_{F_2}(a,s) \leftarrow C_{\neg F_1}(a,s))$, when an action a causes termination of F_1, then the same action, indirectly, causes the initiation of F_2. We relax the conditions on Δ^i_{effect}, so that any number of rules of the following form can appear in it:

$$\begin{aligned} &\forall \bar{x} \forall a \forall s\ (C_{F_i}(\bar{x}_i, a, s) \leftarrow \Psi^+_{F_i}(\bar{x}_i, a, s)),\\ &\forall \bar{x} \forall a \forall s\ (C_{\neg F_i}(\bar{x}_i, a, s) \leftarrow \Psi^-_{F_i}(\bar{x}_i, a, s)) \end{aligned} \tag{23}$$

where Ψ^+ and Ψ^- are formulas with free variables among $\bar{x}_i, a, s$, with only free occurrences of a and s and no other symbols of sort *Act* or *Sit*. In such a formula, a fluent atom is of the form $F_j(\bar{t}_j, s)$ and a causality atom is of the form $C_{(\neg)F_k}(\bar{t}_k, a, s)$. Note that in the direct effect case, causality predicates were excluded from bodies of rules of Δ^i_{effect}.

The basic action theory (18) in which now we allow ramification rules of the form (23) in Δ_{sc}, encodes our solution to the ramification problem in the inductive situation calculus.

Let us define $\Delta_{\text{effect}} = \Delta^1_{\text{effect}} \cup \cdots \cup \Delta^n_{\text{effect}}$, the collection of direct and indirect effect rules for all fluents. Consider the following partition of Δ_{sc}:

$$\{\Delta^1_{\text{fluent}}, \ldots, \Delta^n_{\text{fluent}}, \Delta_{\text{effect}}\}. \tag{24}$$

Proposition 10. *Partition* (24) *is a reduction partition of* Δ_{sc} *in each initial structure* $\mathcal{A}$.

Proof. Let $\mathcal{A}$ be an initial structure with domain A. We construct a reduction $\prec$ of Δ_{sc} on $At^{\tau_{\text{isc}}}_A$ in the rule-by-rule way. This produces the following tuples:

$$\begin{aligned} &(C_{F_i}[\bar{u}, a, s], F_i[\bar{u}, do^{\mathcal{A}}(a, s)]),\\ &(F_i[\bar{u}, s], F_i[\bar{u}, do^{\mathcal{A}}(a, s)]), (C_{\neg F_i}[\bar{u}, a, s], F_i[\bar{u}, do^{\mathcal{A}}(a, s)]),\\ &(F_j[\bar{u}, s], C_{(\neg)F_i}[\bar{v}, a, s]),\\ &(C_{(\neg)F_j}[\bar{u}, a, s], C_{(\neg)F_i}[\bar{v}, a, s]), \end{aligned}$$

for arbitrary tuples of objects $\bar{u}$ and $\bar{v}$, for arbitrary elements a of the action sort and s of the situation sort and for each i, j. Notice that a causality domain atom $C_{(\neg)F_j}[\bar{v}, a, s]$ depends on each fluent atom in s and each other causality atom in a and s. This is a reduction of rules of the form (23).

Since any superset of a reduction relation is also a reduction relation, the reflexive, transitive closure $\prec^*$ is a reduction relation. Moreover, it follows from the fact that $\langle Sit^{\mathcal{A}}, \sqsubseteq^{\mathcal{A}}\rangle$ is a (pre-)well-founded set (Proposition 5), that $\prec^*$ is a pre-well-founded order on $At^{\mathcal{A}}_A$. It is easy to see that for atoms $P[\bar{a}]$, $Q[\bar{b}]$ from $At^{\mathcal{A}}_A$, if $Q[\bar{b}] \prec^* P[\bar{a}]$ and $P[\bar{a}] \prec^* Q[\bar{b}]$, then P and Q are defined in the same sub-definition. Therefore, partition (24) is a reduction partition of Δ_{sc}. $\Box$

As a corollary of this proposition, we obtain the following property.

Proposition 11. *A basic action theory* (18) *with indirect effects is equivalent to*

$$\mathcal{D}_{\text{if}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{init}} \cup \left\{ \bigwedge_{i=1}^{n} \Delta^i_{\text{fluent}} \wedge \Delta_{\text{effect}} \right\} \tag{25}$$

and to

$$\mathcal{D}_{\text{f}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{S_0} \cup \left\{ \bigwedge_{i=1}^{n} comp(\Delta^i_{\text{fluent}}) \wedge \Delta_{\text{effect}} \right\}. \tag{26}$$

Proof. The theory (25) is obtained from the basic action theory by splitting Δ_{sc}. Since partition (24) is a reduction partition in $\mathcal{D}_{\text{if}}$, it follows from Corollary 1 that this is equivalence preserving. The proof of the equivalence with (26) is similar as the proof of Theorem 5. The main step is to show that Δ^i_{fluent} and $comp(\Delta^i_{\text{fluent}})$ are equivalent in initial structures. This follows from the fact that each Δ^i_{fluent} is a definition by well-founded induction over $\prec^*$, the strict well-founded order constructed in the proof of Proposition 7. □

Proposition 12. *If Δ_{effect} is total in (any subtheory of) $\mathcal{D}_{\text{if}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{init}}$, the basic action theory* (18) *with indirect effects satisfies the Initial State Expansion property.*

Proof. Let I be an extension of a τ_{init}-model of $\mathcal{D}_{\text{init}}$ to sort Act which satisfies $\mathcal{D}_{\text{una}(Act)}$ and let $\mathcal{A}$ be its unique $\tau^{\text{o}}_{\text{isc}}$-extension satisfying $\mathcal{D}_{\text{if}}$. Since Δ_{effect} is total in $\mathcal{A}$, partition (24) is a total reduction partition of Δ_{sc} in $\mathcal{A}$, and by Theorem 2, Δ_{sc} is total in $\mathcal{A}$ and has a unique model extending $\mathcal{A}$. □

In general, there is no simple uniform way in which a basic action theory with ramification rules can be translated into classical logic. However, the corollary provides a basis for proving several translation results, depending on the properties of Δ_{effect}. The theorem below considers the case that Δ_{effect} is a positive inductive definition.

Theorem 6. *If Δ_{effect} is a positive definition then the basic action theory* (18) *satisfies the Initial State Expansion property and is equivalent to the theory*

$$\mathcal{D}_{\text{f}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{S_0} \cup \left\{ \bigwedge_{i=1}^{n} comp(\Delta^i_{\text{fluent}}) \wedge PID(\Delta_{\text{effect}}) \right\}$$

and to the theory

$$\mathcal{D}_{\text{f}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{S_0} \cup \left\{ \bigwedge_{i=1}^{n} comp(\Delta^i_{\text{fluent}}) \wedge CIRC(\Delta_{\text{effect}}) \right\}.$$

4.2.1. Example: N Gear wheels

Let us describe a simple idealized mechanical system consisting of a number of gear wheels $w_1, \ldots, w_n$, each pair of which may or may not be mechanically connected. For each of these wheels, we consider two states: *turning* or *stopped*. For each of these wheels, we consider two actions $start(w_i)$ and $stop(w_i)$. The first action gives an impulse to the wheel which propagates over the system to all connected gear wheels; the second action brakes the wheel and all connected wheels. We assume that once a wheel turns, it continues to turn (there is no friction; this system behaves as a perpetuum mobile) until there is a stop action.

We are faced here with a ramification problem—the problem of how to describe the propagation of effects through the system of connected gear wheels. The goal is to develop a *modular* temporal theory describing the effects of the basic actions and the propagation of effects. As a correctness criterion, we should be able to prove the state constraint that in all situations, a gear wheel w is turning if and only if all reachable wheels (those connected to w in the transitive closure of the connection graph) are turning as well.

We could represent this example in Reiter's basic situation calculus [34]. To do this we could pre-compute for each wheel the set of reachable wheels in the connection graph; it suffices then to express that the action of starting (respectively, stopping) a wheel w has the immediate effect to initiate (respectively, terminate) the turning state of wheel w and each wheel reachable from w. This representation would have some important drawbacks. First, notice that this pre-compilation would be impossible if the physical connection relation between gear wheels would be a dynamic relation and gear wheels could be connected or disconnected. Such an example is worked out in Section 4.3.1. Second, the transitive closure of the physical connections between gear wheels is an example of a *global property* of the system which emerges as an interaction of *local properties*, namely the physical connections between gear wheels. If we explicitly represent such global properties then a small change of a local property (e.g. adding a new connection or deleting an existing connection between two gear wheels) may have a strong impact on the global properties and hence on the theory (e.g. disconnecting one pair of gear wheels may split a large interconnected set of connected wheels and would affect the representation of the effect of all actions on all wheels in this set). In a *modular* representation, only local properties of the components should be represented explicitly; global properties should be derivable from a generic part of the theory which does not explicitly depend on the actual configuration of the system. This is an aspect of *elaboration tolerance* [24].

To obtain a modular representation in the gear wheel example, we need to be able to express the reachability from a specific wheel in an arbitrary graph. It is well-known that this concept cannot be expressed in first-order logic. Below we present a formalization through an iterated inductive definition.

In the gear wheel example, there is one domain-dependent sort, denoted *Gear_wheel*. Action symbols are *start* and *stop* and have an argument of sort *Gear_wheel*. The unique fluent *Turns* has arguments of sort *Gear_wheel* and *Sit*.

Basic components of the inductive situation calculus for the Gear wheel example are the foundational axioms $\mathcal{D}_{\text{if}}$ of situations and the unique name axioms $\mathcal{D}_{\text{una}(Act)}$ for actions. The main axiom of our theory is the simultaneous iterated inductive definition Δ_{sc} of the fluent *Turns* and its causality predicates C_{Turns} and $C_{\neg Turns}$. The effect propagation process caused by start or stop actions in one situation will be modeled by a monotone induction. To define the fluent *Turns* for all states, the monotone induction is then iterated over the well-founded structure of situations.

The definition Δ_{sc} can be split up in two sub-definitions. The first part of the definition consists of the standard rules for the fluent *Turns*:

$$\Delta^{Turns}_{\text{fluent}} := \left\{\begin{array}{l} \forall g\ (Turns(g, S_0) \leftarrow I_{Turns}(g)) \\ \forall g \forall a \forall s\ (Turns(g, do(a, s)) \leftarrow C_{Turns}(g, a, s)) \\ \forall g \forall a \forall s\ (Turns(g, do(a, s)) \leftarrow Turns(g, s) \wedge \neg C_{\neg Turns}(g, a, s)). \end{array}\right\}.$$

The second part is the definition Δ_{effect} which describes the causation predicates C_{Turns} and $C_{\neg Turns}$. The following set of rules Δ_{effect} specify direct and indirect effects of actions:

$$\Delta_{\text{effect}} := \left\{\begin{array}{l} \forall g \forall a \forall s\ (C_{Turns}(g, a, s) \leftarrow a = start(g)), \\ \forall g \forall a \forall s\ (C_{\neg Turns}(g, a, s) \leftarrow a = stop(g)), \\ \forall g \forall a \forall s\ (C_{Turns}(g, a, s) \leftarrow \exists g' Connected(g, g') \wedge C_{Turns}(g', a, s)), \\ \forall g \forall a \forall s\ (C_{\neg Turns}(g, a, s) \leftarrow \exists g'(Connected(g, g') \wedge C_{\neg Turns}(g', a, s))) \end{array}\right\}.$$

These rules contain positive recursion. Define $\Delta_{\text{sc}} := \Delta^{Turns}_{\text{fluent}} \cup \Delta_{\text{effect}}$. This definition defines the predicates *Turns*, C_{Turns} and $C_{\neg Turns}$ by simultaneous non-monotone induction in terms of the open predicates I_{Turns} and *Connected*.

Notice that the statement of the problem does not specify what gearwheels exist, how they are connected and whether they are initially turning. Consequently, the theory $\mathcal{D}_{\text{init}}$ consists only of two axioms which express general laws of connected gearwheels. The first axiom expresses that the relation symbol *Connected*, which describes the physical connections between the gear wheels, is a symmetric relation:

$$\forall g \forall g'\ (Connected(g, g') \supset Connected(g', g)).$$

The second axiom of $\mathcal{D}_{\text{init}}$ is related to the state constraint of this system which is that interconnected gear wheels are in the same state: either turning or in rest. The ramification rules guarantee that this state constraint is preserved, but not that it holds initially. Therefore, we have to add the constraint for the initial state. This is described by the axiom

$$\forall g \forall g'\ (Connected(g, g') \supset I_{Turns}(g) \equiv I_{Turns}(g')). \tag{27}$$

The full axiomatization of the domain consists of

$$\mathcal{D}_{wheels} := \mathcal{D}_{\text{if}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{init}} \cup \{\Delta_{\text{sc}}\}.$$

Below we analyze the theory $\mathcal{D}_{wheels}$. Since Δ_{effect} is a positive definition, the basic action theory $\mathcal{D}_{wheels}$ satisfies the conditions of Theorem 6. Consequently, we have the following proposition.

Proposition 13. *The theory $\mathcal{D}_{wheels}$ satisfies the Initial State Expansion property and is equivalent to*

$$\mathcal{D}_{\text{f}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{init}} \cup \{comp(\Delta^{Turns}_{\text{Fluent}}) \wedge PID(\Delta_{\text{Effect}})\}.$$

Proposition 14. *The theory $\mathcal{D}_{wheels}$ logically entails the state constraint*:

$$\forall g \forall g' \forall s\ (Connected(g, g') \supset Turns(g, s) \equiv Turns(g', s)).$$

Proof. The proof is model-theoretic. Let I be a model of $\mathcal{D}_{wheels}$. We use induction on the length of the situations. It follows from axiom (27) on the initial state, that

$$\forall g \forall g'\ (Connected(g, g') \supset Turns(g, S_0) \equiv Turns(g', S_0)).$$

Assume that the property is satisfied for situation s. We prove that it holds for the successor situation $do^I(a,s)$, for arbitrary action a.

Select any pair $(g,g') \in Connected^I$. By definition Δ_{effect}, if $C_{Turn}(g',a,s)$ is true then so is $C_{Turn}(g,a,s)$. Because the graph $Connected^I$ is symmetric, it follows that $C_{Turn}(g,a,s)$ and $C_{Turn}(g',a,s)$ have the same truth value. The same holds for $C_{\neg Turn}$. The induction hypothesis states that in situation s all connected wheels are in the same state. By the above observation, the action a has the same effects on all connected wheels. Consequently, the induction hypothesis is preserved in situation $do^I(a,s)$. □

4.2.2. Non-monotone ramification rules

In [9], it was argued that to model certain forms of ramifications, also non-monotone ramification rules are useful. We illustrate this with a variant of the suitcase example from [19].

Example 1 *(suitcase).* Several versions of this example (e.g. [9,19,38]) have been used to demonstrate that domain constraints are not strong enough to solve the ramification problem and that an explicit notion of causality is necessary. Suppose we have a suitcase which is opened by a spring mechanism on the moment both its locks are being open. To model this example, the sort *lock* is used. Fluent O represents the fact that the suitcase is open; fluent *OpenL* with an argument of sort *lock* means that the lock is open. Action symbols *open* and *close* with one argument of sort *lock* represent actions of opening and closing the respective lock. Two constants l_1, l_2 of sort *lock* represent the two locks. Let Δ_{effect} consist of the following rules:

$$\forall l \forall a \forall s\ (C_{OpenL}(l,a,s) \leftarrow a = open(l)),$$
$$\forall l \forall a \forall s\ (C_{\neg OpenL}(l,a,s) \leftarrow a = close(l))$$

and

$$\forall a \forall s \left(C_O(a,s) \leftarrow \forall l \begin{pmatrix} C_{OpenL}(l,a,s) \vee \\ (OpenL(l,s) \wedge \neg C_{\neg OpenL}(l,a,s)) \end{pmatrix} \right). \tag{28}$$

For every fluent F from the set $\{OpenL, O\}$, we have the standard fluent definition Δ^F_{fluent}. Finally, we have:

$$\mathcal{D}_{\text{init}} := \mathcal{D}_{\text{DCA(lock)}} \cup \mathcal{D}_{\text{una(lock)}} \cup \{\forall l(I_{OpenL}(l) \supset I_O\}.$$

Interestingly, the definition Δ_{effect} is not recursive and hence is total in each structure. It can be translated into classical logic by taking its completion. Consequently, the entire definition Δ_{sc} can be translated into the classical logic theory $comp(\Delta_{\text{sc}})$.

The example illustrates several points of interest. First, ramifications may sometimes rely on the absence of certain causes. In such cases, the use of non-monotone causation rules such as the rule defining C_O is appropriate. Second, ramification rules are useful for modeling simultaneous effects and concurrency. Such simultaneous effects cannot occur in the above situation calculus but could occur in extensions with concurrent actions or with additional actions and/or ramifications which could affect both locks simultaneously. For the sake of illustrating this, consider the following extension.

Example 2. Consider an extension of Example 1 in which the suitcase has a central button which switches the state of the two locks of the suitcase. The effects of pushing this button are expressed by the additional effect rules:

$$\forall l \forall a \forall s\ (C_{OpenL}(l, pushbutton, s) \leftarrow \neg OpenL(l,s)),$$
$$\forall l \forall a \forall s\ (C_{\neg OpenL}(l, pushbutton, s) \leftarrow OpenL(l,s)).$$

Note that executing the action *pushbutton* in a state where one lock is open and the other is closed, produces simultaneous effects of opening one lock and closing the other lock, and this would not open the suitcase. In fact, the ramification rule describes the intended behaviour of the suitcase for every combination of effects on both locks.

As a last point, we illustrate an alternative style of representing complex forms of ramifications.

Example 3. In the suitcase example, the suitcase is caused to open on the instant that both locks are open. Let us replace the rule (28) by the following more direct representation of this:

$$\forall a \forall s\ (C_O(a,s) \leftarrow \forall l\, OpenL(l, do(a,s))). \tag{29}$$

According to this rule, when all locks are open in the successor state $do(a,s)$, then a causes the suitcase to open in situation s. Adding this axiom changes the structure of the definition. So far, causality predicates at state s were defined in terms of fluents and causality predicates at state s. Here, C_O in state s is defined in terms of *OpenL* in the future state $do(a,s)$. This is no longer induction on the well-founded set of situations. However, the resulting definition is still a definition by well-founded induction in each initial structure $\mathcal{A}$, although the underlying order does not entirely follow the order of situations. Indeed, the reduction relation $\prec$ constructed in the standard way, consists of the following tuples, for arbitrary lock l, situation s and action a:

$$\begin{aligned}
&(C_{OpenL}[l,a,s], OpenL[l, do^{\mathcal{A}}(a,s)]),\\
&(OpenL[l,s], OpenL[l, do^{\mathcal{A}}(a,s)]), (C_{\neg OpenL}[l,a,s], OpenL[l, do^{\mathcal{A}}(a,s)]),\\
&(C_O[a,s], O[do^{\mathcal{A}}(a,s)]),\\
&(O[s], O[do^{\mathcal{A}}(a,s)]), (C_{\neg O}[a,s], O[do^{\mathcal{A}}(a,s)]),\\
&(OpenL[l, do^{\mathcal{A}}(a,s)], C_O[a,s]).
\end{aligned}$$

It is easy to verify that the transitive closure $\prec^*$ is a strict well-founded order. Hence, Δ_{sc} is a definition by well-founded induction in each initial structure $\mathcal{A}$. It follows that this definition can be translated in FO using completion and the Initial State Expansion property is still satisfied.

4.2.3. Limits of the approach

As pointed out in [9], there is a subtle modeling issue involved in the use of non-monotone ramification rules. In the inductive situation calculus, the ramification rules are used to model an effect propagation process. In the physical reality, the effect propagations in this process are not instantaneous, but take a small lapse of time. The intermediate states during this process are not modeled explicitly in the situation calculus. In each of these intermediate states, certain causes may have occurred already, and other causes did not yet occur. Therefore, it is possible that the condition of an effect rule, if it contains a negative cause literal $\neg C_F(\bar{t}, a, s)$, is satisfied during such an intermediate state, when the cause of $F(\bar{t})$ was not yet produced, but not in the final state. In such a case, according to the inductive situation calculus, the effect described by the rule will not occur. Stated differently, if, during the propagation process, the conditions of an effect rule are satisfied in some intermediate state, and one of the conditions is the absence of a certain cause, then the effect will not be inferred if there is a possibility that this cause is still produced later during the propagation process.

Example 4. Reconsider the effect rule for opening the suitcase of Example 1 and Example 2:

$$\forall a \forall s \left(C_O(a,s) \leftarrow \forall l \left(\begin{array}{c} C_{OpenL}(l,a,s) \vee \\ (OpenL(l,s) \wedge \neg C_{\neg OpenL}(l,a,s)) \end{array} \right) \right).$$

Suppose we push the central button to switch the states of the locks. The mechanism might be just a tiny bit faster to switch the lock l_1 than the lock l_2. If the first one was already closed and the second open, then there will be a brief intermediate state during which the two locks will be open and the conditions of the above effect rule are satisfied. This state will last only a fraction of a second, after which the second lock is closed. For an old-fashioned suitcase with mechanical spring mechanism, this is not enough time to open the suitcase. So, the suitcase does not open. The above rule correctly models this situation.

Now, consider a high-tech suitcase in which a microprocessor monitors the state of the two locks, and when both are open, it sends a signal to an electric motor to open the suitcase. Compared to the old-fashioned suitcase, the effect propagations are the same. However, there is a difference on the level of the reaction time of the opening effect. The microprocessor reacts in microseconds and will detect the state change of the first lock before that of the second. So, in this case, the suitcase will be opened. The ramification rules of the inductive situation calculus cannot be used to model the high-tech suitcase.

4.3. Defined fluents

In this section, we propose a second extension of the inductive situation calculus by allowing *defined* fluents. Such fluents are not governed by effect laws and the law of inertia, but by a definition in terms of existing fluents.

Example 5. In the standard ontology of the blocks world problem, there are basic actions $pick(b)$ and $put(b,l)$, and fluents

- $On(b,l,s)$: the block b is on location l, which is the table or another block;
- $Clear(b,s)$: nothing is on block b;
- $Clasped(b,s)$: the robot clasps block b;
- $Free(s)$: the robot hand is free.

We could represent the effects of the actions on every of these fluents. It is perhaps more natural to represent the effects on *On* and *Clasped* and to define the fluent *Clear* in terms of *On* and the fluent *Free* in terms of *Clasped*. The definitions are:

$$\big\{\forall b \forall s\ (Clear(b,s) \leftarrow \neg\exists b'\ On(b',b,s))\big\},$$
$$\big\{\forall s\ (Free(s) \leftarrow \neg\exists b\ Clasped(b,s))\big\}.$$

We could also define the fluent $Above(b,b',s)$ expressing that in state s, block b is above b'. It is defined as the transitive closure of the fluent $On(b,b',s)$:

$$\left\{\begin{array}{l}\forall b \forall b_1 \forall s\ (Above(b,b_1,s) \leftarrow On(b,b_1,s)),\\ \forall b \forall b_1 \forall s\ (Above(b,b_1,s) \leftarrow \exists b_2 (Above(b,b_2,s) \wedge Above(b_2,b_1,s)))\end{array}\right\}.$$

In a sense, defining fluents is also a way of representing ramifications. Indeed, one could view the fact that $Free(s)$ is initiated or terminated as a ramified effect of terminating or initiating $Clasped(b,s)$. Yet, there is a definite difference with the sort of ramification considered in Section 4.2. In that section, the causality rules model the propagation of effects through the system. Here, a defined fluent is (sometimes) only a new name denoting some, potentially complex, configuration of the primitive fluents.

Below we distinguish between *defined fluents* and *primitive fluents*. Primitive fluents are defined in the standard way. To represent defined fluents, we extend the inductive situation calculus by allowing the definition Δ_{sc} of Section 4.2 to be extended with a set of rules Δ_{def} of the form:

$$\forall \bar{x} \forall s\ (F_d(\bar{x},s) \leftarrow \Psi(\bar{x},s)), \tag{30}$$

where F_d is a defined fluent and Ψ is a formula where s has only free occurrences, contains no causality predicates and every fluent atom is of the form $F_i(\bar{t}_i,s)$, where F_i may be a defined or a primitive fluent. We do not introduce initial state or causation predicates for defined fluents in τ_{isc}.

Consider the following partition of Δ_{sc}:

$$\{\Delta^1_{\text{fluent}}, \ldots, \Delta^n_{\text{fluent}}, \Delta_{\text{effect}}, \Delta_{\text{def}}\} \tag{31}$$

where Δ^i_{fluent} is the standard definition of a primitive fluent F_i.

Proposition 15. *Partition* (31) *is a reduction partition of* Δ_{sc} *in each initial structure* $\mathcal{A}$.

Proof. Let $\mathcal{A}$ be an initial structure with domain A. We construct a reduction relation $\prec$ in $At_A^{\tau_{\text{isc}}}$ in the normal rule-by-rule way. It consists of all tuples:

$$(C_{F_i}[\bar{u},a,s], F_i[\bar{u}, do^{\mathcal{A}}(a,s)]),$$
$$(F_i[\bar{u},s], F_i[\bar{u}, do^{\mathcal{A}}(a,s)]), (C_{\neg F_i}[\bar{u},a,s], F_i[\bar{u}, do^{\mathcal{A}}(a,s)]),$$
$$(F_j[\bar{u},s], C_{(\neg)F_i}[\bar{v},a,s]),$$

$(C_{(\neg)F_j}[\bar{u}, a, s], C_{(\neg)F_i}[\bar{v}, a, s]),$

$(F_i[\bar{u}, s], F_k[\bar{v}, s]),$

for arbitrary i, for arbitrary j and j' such that F_j and $F_{j'}$ are primitive fluents, for arbitrary k such that F_k is a defined fluent, for arbitrary tuples of objects $\bar{u}$ and $\bar{v}$, for arbitrary elements a of the action sort and s of the situation sort. Notice that according to the last line, each defined fluent domain atom $F_k[\bar{v}, s]$ depends on each fluent atom $F_i[\bar{u}, s]$ and hence, $\prec$ is a reduction relation of each rule in Δ_{def}.

From here on, the proof is very similar to that of Proposition 10. Again, it is easy to show that $\prec$ is a reduction relation. Its reflexive, transitive closure $\prec^*$ is a reduction relation and a pre-well-founded order on $At_A^{\tau_{\text{isc}}}$. It is easy to see that for atoms $P[\bar{a}]$, $Q[\bar{b}]$ from $At_A^{\tau_{\text{isc}}}$ such that $Q[\bar{b}] \prec^* P[\bar{a}]$ and $P[\bar{a}] \prec^* Q[\bar{b}]$, P and Q are defined in the same sub-definition. Therefore, partition (31) is a reduction partition of Δ_{sc} in $\mathcal{A}$. □

As a corollary of this proposition and the decomposition theorems, Theorem 1 and Theorem 2, we obtain the following property.

Corollary 5. *If Δ_{effect} and Δ_{def} are total in (each subtheory of) $\mathcal{D}_{\text{if}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{init}}$, then the basic action theory (18) satisfies the Initial State Expansion property, and is equivalent to the theory $\mathcal{D}_{\text{f}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{S_0} \cup \{\bigwedge_{i=1}^{n} \Delta_{\text{fluent}}^{i} \wedge \Delta_{\text{effect}} \wedge \Delta_{\text{def}}\}$.*

This modularity result can be used to prove the correctness of several translations from inductive situation calculus with defined fluents to classical logic.

4.3.1. Example: Gear wheels with friction

In this example, we consider another version of the gear wheel problem in which the connections between gear wheels can be dynamically changed (as in the gears of a car) and in which friction is taken into account. We assume that some of the gear wheels have a fixed connection to an engine. This engine can be started or stopped. A gear wheel is in rest unless it is connected directly or indirectly to a running engine. There are actions to start or stop (the engine of) a gear wheel and to connect or disconnect gear wheels.

We use the following vocabulary with sort *Gear_wheel*:

- $Turn(g, s)$: gear wheel g is turning;
- $Emp(g, s)$: gear wheel g is empowered;
- $DirEmp(g, s)$: gear wheel g is attached to an operating motor (i.e., is directly empowered);
- $Con(g, g', s)$: gear wheels g and g' are directly connected;
- $start(g)$, $stop(g)$: the actions of starting and halting the engine attached to g;
- $connect(g, g')$, $disconnect(g, g')$: the actions of connecting and disconnecting a pair of gear wheels.

Defined fluents are axiomatized by the following set Δ_{def} of rules. We represent that a gear wheel g is turning iff it is empowered. It is empowered if it is attached to a running engine (i.e., $DirEmp(g, s)$ is true), or there is a path of gear wheel connections to such a directly empowered gear wheel:

$$\Delta_{\text{def}} = \left\{\begin{array}{l} \forall g \forall s\ (Turn(g, s) \leftarrow Emp(g, s)), \\ \forall g \forall s\ (Emp(g, s) \leftarrow DirEmp(g, s)), \\ \forall g \forall s\ (Emp(g, s) \leftarrow \exists g_1 (Con(g, g_1, s) \wedge Emp(g_1, s))) \end{array}\right\},$$

$$\Delta_{\text{fluent}}^{DirEmp} = \left\{\begin{array}{l} \forall g\ (DirEmp(g, S_0) \leftarrow I_{DiREmp}(g)), \\ \forall g \forall a \forall s\ (DirEmp(g, do(a, s)) \leftarrow C_{DirEmp}(g, a, s)), \\ \forall g \forall a \forall s\ (DirEmp(g, do(a, s)) \leftarrow DirEmp(g, s) \wedge \neg C_{\neg DirEmp}(g, a, s)), \end{array}\right\},$$

$$\Delta_{\text{fluent}}^{Con} = \left\{\begin{array}{l} \forall g \forall g'\ (Con(g, g', S_0) \leftarrow I_{Con}(g, g')), \\ \forall g \forall g' \forall a \forall s\ (Con(g, g', do(a, s)) \leftarrow C_{Con}(g, g', a, s)), \\ \forall g \forall g' \forall a \forall s\ (Con(g, g', do(a, s)) \leftarrow Con(g, g', s) \wedge \neg C_{\neg Con}(g, g', a, s)) \end{array}\right\},$$

$$\Delta_{\text{effect}} = \left\{ \begin{array}{l} \forall g \forall a \forall s\ (C_{DirEmp}(g, a, s) \leftarrow a = start(g)), \\ \forall g \forall a \forall s\ (C_{\neg DirEmp}(g, a, s) \leftarrow a = stop(g)), \\ \forall g \forall g' \forall a \forall s\ (C_{Con}(g, g', a, s) \leftarrow a = connect(g, g')), \\ \forall g \forall g' \forall a \forall s\ (C_{Con}(g', g, a, s) \leftarrow C_{Con}(g, g', a, s)), \\ \forall g \forall g' \forall a \forall s\ (C_{\neg Con}(g, g', a, s) \leftarrow a = disconnect(g, g')), \\ \forall g \forall g' \forall a \forall s\ (C_{\neg Con}(g', g, a, s) \leftarrow C_{\neg Con}(g, g', a, s)) \end{array} \right\}.$$

In this definition, the fluents *Turn* and *Emp* are defined. The fluent *Emp* is defined inductively. The theory $\mathcal{D}_{\text{init}}$ consists of a single axiom expressing that the initial connection relation between gear wheels is symmetric:

$$\forall g \forall g'\ (I_{Con}(g, g') \supset I_{Con}(g', g)).$$

Let $\mathcal{D}_{\text{Friction}} := \mathcal{D}_{\text{if}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{init}} \cup \{\Delta_{\text{sc}}\}$ where Δ_{sc} consists of $\Delta_{\text{fluent}} \cup \Delta_{\text{effect}} \cup \Delta_{\text{def}}$ as defined above. The conditions of Corollary 5 apply. Because Δ_{fluent} is a definition by well-founded induction and Δ_{effect} and Δ_{def} are positive inductive definitions, this theory can be translated into classical logic.

Proposition 16. *The basic action theory with definitions $\mathcal{D}_{\text{Friction}}$ satisfies the Initial State Expansion property and is equivalent to the SO theory*

$$\mathcal{D}_{\text{f}} \cup \mathcal{D}_{\text{una}(Act)} \cup \mathcal{D}_{\text{init}} \cup \big\{comp(\Delta_{\text{fluent}}) \wedge PID(\Delta_{\text{effect}}) \wedge PID(\Delta_{\text{def}})\big\}.$$

4.4. More extensions

Many other extensions of situation calculus have been proposed, for example natural actions, concurrency and continuous time [34]. Most of these extensions can be integrated seamlessly in the inductive situation calculus. One extension that we briefly discuss here is the one with non-deterministic actions. Obviously, in this case, successor states cannot be defined in terms of predecessor states and actions, since the latter do not determine the first in a unique way. A simple technique to model non-determinism is by introducing new open predicates. For example, consider the non-deterministic effect of rolling a dice, which causes the fluent *Dice* to take a value between 1 and 6. This could be represented in the inductive situation calculus by introducing a new binary open predicate *Thrown* with a first argument of sort *int* and a second of sort *sit*. The effect of rolling a dice on the fluent *Dice* of sort *int* is then represented by the effect rule:

$$\forall n \forall s\ (C_{Dice}(n, throw, s) \leftarrow Thrown(n, s))$$

together with axioms stating that for all situations s, there is a unique number n between 1 and 6 such that $Thrown(n, s)$ holds. This technique can be generalized into a general methodology to represent non-deterministic actions.

5. Related work

The prime goal of this paper is to clarify inductive definitions and their role in common sense knowledge representation. The application domain is temporal reasoning, situation calculus in particular. In this respect, this paper is only preceded by earlier work of the authors. The semantic correspondence between inductive definitions and causality was pointed out independently in [36,37] and [9]. In both cases, the motivation for using inductive definitions was the similarity between the process of effect propagation in a dynamic system and inductive definitions. Inductive definition formalizations of situation calculus were first presented by Ternovska in [36,37] and, a bit later, by Denecker in [5]. The backgrounds of these studies were quite different, in one case the classical logic approach developed at the Cognitive Robotics Group at the University of Toronto, and in the other case, logic programming formalizations of temporal reasoning. Ternovska [36,37] observed that the construction of a model of Reiter-style situation calculus is an induction over the well-founded order on situations, and therefore, proposed to model situation calculus with a logic of inductive definitions. Her approach used Aczel-style abstract monotone induction definitions [1]. To handle the inherent non-monotonicity of the situation calculus, the monotone definitions were constructed by simultaneous non-monotone induction. She also extended her solution to acyclic ramification rules and demonstrated that an inductive definition of the set of situations implies the general induction principle on situations [33]. In [9], Denecker et al.

focussed on the ramification problem in general, and pointed to the similarity with inductive definitions. They studied the ramification problem with cyclic effect rules and negative conditions, and proposed to use the well-founded semantics to model such complex ramifications. The inductive situation calculus integrates this work in the context of the situation calculus.

For an overview of the many different approaches for temporal reasoning and/or the ramification problem, we refer to [9,15,26,34,35,38,39]. Here we limit our discussion to approaches with an explicit representation of causal laws in situation calculus or non-monotonic logic.

It has been observed that the process of effect propagation is a *constructive* process: basic actions cause changes and effects which propagate through the dynamic system; changes do not appear without an external cause (i.e., no *spontaneous generation* of effects, or no *deus ex machina* effects). The same constructive intuition is found in inductive definitions. This explains why inductive definitions can correctly model recursive effect propagations. In this respect, the inductive situation calculus is more general than two other well-known classical logic formalizations of the situation calculus with ramifications, namely Lin's approach [19] and McIlraith's solitary stratified theories [26]. Both approaches impose constraints on ramification rules which preclude recursive ramifications. A strong constraint in solitary stratified theories is that no fluent symbol is allowed to appear both as an effect and in the precondition of the same action. On the other hand, McIlraith addresses the qualification problem, which we don't.

While the above approaches are based on classical logic, causality has also been investigated from a non-monotonic reasoning perspective. Perhaps the best known non-monotonic logic that takes causality as its basic principle is the logic of non-monotonic causal theories [15,22]. This language extends propositional logic with causal implications $\varphi \Leftarrow \psi$. Interestingly, this approach takes the opposite point of view with respect to spontaneous generation of effects than in the inductive situation calculus. Spontaneous generation is not seen as a flaw but as a feature, used to model *exogeneous fluents*, fluents that can change state spontaneously. This is modeled by pairs of causal rules

$$P \Leftarrow P, \qquad \neg P \Leftarrow \neg P.$$

As a consequence, cyclic effect rules as found in the gear wheel example, cannot correctly be modeled by recursive causal rules in this formalism because such a theory would accept unintended models in which two connected gear wheels cause each other to turn, without external cause. While this seems a weakness of this formalism, it is clear that the possibility of representing exogeneous events or fluents is a useful feature. The challenge here is to integrate exogeneous fluents with a correct treatment of cyclic effect rules.

6. Conclusion

This paper explains the inductive nature of the situation calculus. We have shown that—unsuspected by its creators—the original Reiter-style situation calculus makes hidden use of inductive definitions. We made these definitions explicit and found monotone and non-monotone induction. We formalized these definitions in a logic of inductive definitions (ID-logic) thus obtaining a variant of the situation calculus which we call the *inductive situation calculus*. We presented a translation to classical logic to show that the inductive situation calculus is indeed equivalent to the standard formalization in the case without ramifications.

Our ID-logic formalization offers a number of advantages compared to classical logic. First, in the Reiter-style situation calculus, different forms of induction are formalized in different ways. By using a logic of inductive definitions, we obtained a uniform and modular representation. Second, the use of inductive definitions allowed us to extend the situation calculus to cope with complex temporal phenomena such as (recursive) ramifications and (inductively) defined fluents. We also proved that the inductive situation calculus (and Reiter's situation calculus) satisfies the Initial State Expansion property.

Our work contributes also on other levels. First, it presents the situation calculus as an application of the principle of iterated induction in the context of commonsense reasoning thus giving insight into this complex and little known form of non-monotone induction. Second, our analysis showed that the modularity, totality and transformation theorems are powerful tools for analyzing and transforming large definitions, by allowing to break them up using the modularity theorem and translating them piecewise to classical logic.

Finally, our experiment demonstrates that the use of different forms of inductive definitions is not limited to mathematics, but is applicable in a much wider area of knowledge representation and commonsense reasoning. In particular, it seems that inductive definitions are well-suited for reasoning about causality. Perhaps this is not so surprising. After all, mathematical induction is about *construction* of complex mathematical objects. In this respect, mathematical induction may be viewed as an application of causal reasoning in the idealized abstract context of mathematics, rather than in the much more complex realm of commonsense reasoning.

References

[1] P. Aczel, An introduction to inductive definitions, in: J. Barwise (Ed.), Handbook of Mathematical Logic, North-Holland Publishing Company, Amsterdam, 1977, pp. 739–782.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (Eds.), The Description Logic Handbook. Theory, Implementation and Applications, Cambridge University Press, 2002.

[3] K.L. Clark, Negation as failure, in: H. Gallaire, J. Minker (Eds.), Logic and Databases, Plenum Press, 1978, pp. 293–322.

[4] M. Denecker, The well-founded semantics is the principle of inductive definition, in: J. Dix, L. Fariñas del Cerro, U. Furbach (Eds.), Logics in Artificial Intelligence, Schloss Daghstull, in: Lecture Notes in Artificial Intelligence, vol. 1489, Springer, 1998, pp. 1–16.

[5] M. Denecker, Extending classical logic with inductive definitions, in: J. Lloyd, et al. (Eds.), First International Conference on Computational Logic (CL2000), London, in: Lecture Notes in Artificial Intelligence, vol. 1861, Springer, 2000, pp. 703–717.

[6] M. Denecker, M. Bruynooghe, V. Marek, Logic programming revisited: logic programs as inductive definitions, ACM Transactions on Computational Logic 2 (4) (2001) 623–654.

[7] M. Denecker, E. Ternovska, Inductive situation calculus, in: Proceedings of Ninth International Conference on Principles of Knowledge Representation and Reasoning, Delta Whistler Resort, Canada, 2004, pp. 545–553, http://www.cs.kuleuven.ac.be/cgi-bin-dtai/publ_info.pl?id=41085.

[8] M. Denecker, E. Ternovska, A logic of non-monotone inductive definitions and its modularity properties, in: V. Lifschitz, I. Niemelä (Eds.), 7th International Conference on Logic Programming and Nonmonotonic Reasoning, 2004.

[9] M. Denecker, D. Theseider Duprè, K. Van Belleghem, An inductive definition approach to ramifications, Linköping Electronic Articles in Computer and Information Science 3 (7) (1998) 1–43, http://www.ep.liu.se/ea/cis/1998/007/.

[10] M. Denecker, E. Ternovska, A logic of non-monotone inductive definitions, ACM Transactions on Computational Logic (TOCL), 2007.

[11] F. Fages, Consistency of Clark completion and existence of stable models, Journal of Methods of Logic in Computer Science 1 (1994) 51–60.

[12] M. Fitting, Fixpoint semantics for logic programming a survey, Theoretical Computer Science 278 (1–2) (2002) 25–51.

[13] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Generation Computing 9 (1991) 365–385.

[14] M. Gelfond, V. Lifschitz, Describing action and change by logic programs, in: Ninth Joint International Conference and Symposium on Logic Programming JICSLP'92, MIT Press, 1992.

[15] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, H. Turner, Nonmonotonic causal theories, Artificial Intelligence (AIJ) 153 (2004) 49–104.

[16] Y. Gurevich, S. Shelah, Fixed-point extensions of first-order logic, Annals of Pure and Applied Logic 32 (1986) 265–280.

[17] S. Hanks, D. McDermott, Nonmonotonic logic and temporal projection, Artificial Intelligence 33 (1987) 379–412.

[18] H.J. Levesque, F. Pirri, R. Reiter, Foundations for the situation calculus, Electronic Transactions on Artificial Intelligence 2 (1998) 159–178.

[19] F. Lin, Embracing causality in specifying the indirect effects of actions, in: Proc. of IJCAI 95, 1995, pp. 1985–1991.

[20] V.W. Marek, M. Truszczyński, Stable models and an alternative logic programming paradigm, in: K.R. Apt, V. Marek, M. Truszczyński, D.S. Warren (Eds.), The Logic Programming Paradigm: A 25 Years Perspective, Springer, 1999, pp. 375–398.

[21] M. Mariën, R. Mitra, M. Denecker, M. Bruynooghe, Satisfiability checking for PC(ID), in: G. Sutcliffe, A. Voronkov (Eds.), Proc. LPAR'05, in: Lecture Notes in Artificial Intelligence, vol. 3835, Springer, 2005, pp. 565–579.

[22] N. McCain, H. Turner, Causal theories of action and change, in: Proc. of AAAI 97, 2007, pp. 460–465.

[23] J. McCarthy, Circumscription—a form of nonmonotonic reasoning, Artificial Intelligence 13 (1980) 27–39.

[24] J. McCarthy, Elaboration tolerance, in: COMMON SENSE 98, Symposium on Logical Formalizations of Commonsense Reasoning, January 1998.

[25] J. McCarthy, P.J. Hayes, Some philosophical problems from the standpoint of artificial intelligence, in: B. Meltzer, D. Michie (Eds.), Machine Intelligence 4, Edinburgh University Press, 1969, pp. 463–502.

[26] S. McIlraith, An axiomatic solution to the ramification problem (sometimes), Artificial Intelligence 116 (1–2) (2000) 87–121.

[27] D. Mitchell, E. Ternovska, A framework for representing and solving NP search problems, in: Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05), 2005, pp. 430–435.

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

[29] I. Niemelä, Logic programs with stable model semantics as a constraint programming paradigm, Annals of Mathematics and Artificial Intelligence 25 (3,4) (1999) 241–273.

[30] N. Pelov, E. Ternovska, Reducing ID-logic to propositional satisfiability, in: Proceedings of the Twenty First International Conference on Logic Programming (ICLP 2005), 2005.

[31] F. Pirri, R. Reiter, Some contributions to the metatheory of the situation calculus, ACM 46 (3) (1999) 325–361.

[32] R. Reiter, A logic for default reasoning, Artificial Intelligence 13 (1980) 81–132.

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

[34] R. Reiter, Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems, MIT Press, 2001.

[35] M. Shanahan, Solving the Frame Problem, MIT Press, 1997.

[36] E. Ternovskaia, Causality via inductive definitions, in: Working Notes of "Prospects for a Commonsense Theory of Causation", AAAI Spring Symposium Series, March 23–28, 1998.
[37] E. Ternovskaia, Inductive definability and the situation calculus, in: Transaction and Change in Logic Databases, in: Lecture Notes in Computer Science, vol. 1472, Springer, 1998.
[38] M. Thielscher, Ramification and causality, Journal of Artificial Intelligence 89 (1997) 317–364.
[39] M. Thielscher, Introduction to the fluent calculus, Electronic Transactions on Artificial Intelligence 3–4 (1998) 179–192, http://www.ep.liu.se/ej/etai/1998/006/.
[40] A. Van Gelder, An alternating fixpoint of logic programs with negation, Journal of Computer and System Sciences 47 (1993) 185–221.
[41] A. Van Gelder, K.A. Ross, J.S. Schlipf, The well-founded semantics for general logic programs, Journal of the ACM 38 (3) (1991) 620–650.
[42] J. Vennekens, D. Gilis, M. Denecker, Splitting an operator: Algebraic modularity results for logics with fixpoint semantics, ACM Transactions on Computational Logic (TOCL) 2006, submitted for publication.