

A Logic of Non-Monotone Inductive Definitions and its Modularity Properties

Marc Denecker¹ and Eugenia Ternovska²

¹ Department of Computer Science, K.U.Leuven, Belgium
marcd@cs.kuleuven.ac.be

² School of Computing Science, Simon Fraser University, Vancouver, Canada
ter@cs.sfu.ca

Abstract. Well-known principles of induction include monotone induction and different sorts of non-monotone induction such as inflationary induction, induction over well-ordered sets and iterated induction. In this work, we define a logic formalizing induction over well-ordered sets and monotone and iterated induction. Just as the principle of positive induction has been formalized in FO(LFP), and the principle of inflationary induction has been formalized in FO(IFP), this paper formalizes the principle of iterated induction in a new logic for Non-Monotone Inductive Definitions (NMID-logic). The semantics of the logic is strongly influenced by the well-founded semantics of logic programming.

Our main result concerns the modularity properties of inductive definitions in NMID-logic. Specifically, we formulate conditions under which a simultaneous definition Δ of several relations is logically equivalent to a conjunction of smaller definitions $\Delta_1 \wedge \dots \wedge \Delta_n$ with disjoint sets of defined predicates. The difficulty of the result comes from the fact that predicates P_i and P_j defined in Δ_i and Δ_j , respectively, may be mutually connected by simultaneous induction. Since logic programming and abductive logic programming under well-founded semantics are proper fragments of our logic, our modularity results are applicable there as well. As an example of application of our logic and theorems presented in this paper, we describe a temporal formalism, the inductive situation calculus, where causal dependencies are naturally represented as rules of inductive definitions.

1 Introduction

This paper fits into a broad project aiming at studying general forms of inductive definitions and their role in diverse fields of mathematics and computer science. Monotone inductive definitions and inductive definability have been studied extensively in mathematical logic [Mos74, Acz77]. The algebraic foundations for monotone induction are laid by Tarski's fixed point theory of monotone lattice operators [Tar55]. The notion of an inductive definition is the underlying concept in fixed point logics [DG02], which are closely connected to database query languages. Fixed point constructs for monotone inductive and co-inductive definitions play a central role in the specification languages for formal verification

of dynamic systems such as the μ -calculus. Induction axioms have been used successfully in proving properties of protocols using automated reasoning tools, and the concept of a definition is fundamental in the area of description logics. Thus, it appears that the notion of a definition and its inductive generalizations emerges as a unifying theme in many areas of mathematics and computational science. Hence, the study of inductive definitions could provide further insight in the interrelations between these areas and lead to synergy between them.

Despite an abundance of logics that capture some aspects of inductive definitions, there is no sufficiently general logic that represents monotone induction, and, at the same time, captures non-monotone forms of induction, such as iterated induction and induction over well-ordered sets. Recently, [Den98,DBM01] have argued that semantical studies of logic programming may contribute to a better understanding of such generalized forms of induction. In particular, it was argued that the well-founded semantics of logic programming [VRS91] formalizes and generalizes the non-monotone forms of induction mentioned above. In [DMT00], a fixed point theory of general non-monotone lattice operators, called *approximation theory*, was developed. This theory generalizes Tarski's theory of fixed points of monotone lattice operators, and provides the algebraic foundation of this generalized principle of iterated induction modeled by the well-founded semantics.

Our main objective here is to study modularity properties of non-monotone induction. Modularity is widely recognized as an important property in formal verification and knowledge representation. For example, in the context of specification and verification of complex dynamic systems, it is crucial that we can specify a complex system by describing its components in independent modules which can then be conjoined to form a correct description of the complete system. Thus, the operation of joining modules should preserve the correctness of the component modules. The dual operation of *splitting* a complex theory into an equivalent set of smaller modules is equally important. It allows to investigate complex theories by studying its modules independently, and reduces the analysis of the correctness of the complex theory to the much simpler problem of analyzing the correctness of its modules.

Our main result is a set of techniques that allow one to break up a definition into a conjunction of smaller and simpler definitions. More precisely, we investigate general formal conditions that guarantee that a simultaneous definition of several predicates can be split into the conjunction of a set of components of this definition, each component defining a subset of the defined predicates. Our theorems also provide conditions under which joining a set of definitions for distinct sets of predicates into one simultaneous definition is equivalence preserving. The problem we study is similar to that studied in [Sch95,VDD00], but our results are uniformly stronger in that they are proven for a more expressive logic and under more general conditions.

The paper is structured as follows. In section 2, we extend classical logic with generalized non-monotone definitions. In section 3, the modularity is investigated. Then we demonstrate an application of our logic on the example of

a non-monotonic reasoning formalism of the inductive situation calculus. We conclude with section 5, where we summarize our results, and outline future directions.

2 NMID-Logic

In this section, we present an extension of classical logic with non-monotone inductive definitions. This work extends previous work of [Den00].

Syntax First, we introduce the notion of a definition. We introduce a new binary connective $\leftarrow$, called the *definitional implication*. A *definition* Δ is a set of rules of the form

$$\forall \bar{x} (X(\bar{t}) \leftarrow \varphi) \text{ where} \quad (1)$$

- $\bar{x}$ 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 .

The definitional implication $\leftarrow$ must be distinguished from material implication. A rule $\forall \bar{x} (X(\bar{t}) \leftarrow \varphi)$ in a definition does not correspond to the disjunction $\forall \bar{x} (X(\bar{t}) \vee \neg \varphi)$. Note that in front of rules, we allow only universal quantifiers. In the rule (1), $X(\bar{t})$ is called the *head* and φ is the *body* of the rule.

Example 1. The following expression is a simultaneous definition of even and odd numbers on the structure of the natural numbers with zero and the successor function: $\{\forall x(E(x) \leftarrow x = 0), \forall x(E(s(x)) \leftarrow O(x)), \forall x(O(s(x)) \leftarrow E(x))\}$. An example of a non-monotone definition in the well-ordered set of natural numbers is the expression $\{\forall x(E(x) \leftarrow x = 0), \forall x(E(s(x)) \leftarrow \neg E(x))\}$.

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*. In the Example 1 above, E and O are defined predicate symbols, and s is an open function symbol.

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 can be 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.

Let $free(\phi)$ denotes the set of free symbols of ϕ . We shall use notation $\phi(x_1, \dots, x_n)$ to emphasize that symbols $x_1, \dots, x_n$ are distinct and are free in ϕ . Let τ be a vocabulary interpreting 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., $\tau_\Delta^d \cup \tau_\Delta^o = \tau$, and $\tau_\Delta^d \cap \tau_\Delta^o = \emptyset$. A *well-formed formula* of the Logic for Non-Monotone Inductive Definitions, briefly a NMID-formula, is defined by the following induction:

- If X is an n -ary predicate symbol, and $t_1, \dots, t_n$ are terms then $X(t_1, \dots, t_n)$ is a formula.

- If Δ is a definition then Δ is a formula.
- If ϕ, ψ are formulas, then so are $(\neg\phi)$ and $(\phi \wedge \psi)$.
- If ϕ is a formula, then $\exists\sigma \phi$ is a formula.

A formula ϕ is an NMID-formula over a vocabulary τ if $\text{free}(\phi) \subseteq \tau$. We use $\text{SO}(\text{NMID})[\tau]$ to denote the set of all formulas of our logic over fixed vocabulary τ . The first order fragment $\text{FO}(\text{NMID})[\tau]$ is defined in the same way, except that quantification over set and function symbols is not allowed.

Example 2. In the language of the natural numbers, the following formula 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 with the standard induction axiom:

$$\exists N[\{\forall x(N(x) \leftarrow x = 0), \forall x(N(s(x)) \leftarrow N(x))\} \wedge \forall x N(x)].$$

Semantics. The exposition below is a synthesis of different approaches to the well-founded semantics, in particular those presented in [Van93, Fit03, DBM01]. We assume familiarity with semantics of classical logic. A structure I of a vocabulary τ consists of a domain $\text{dom}(I)$ and for each symbol $\sigma \in \tau$ a *value* σ^I in $\text{dom}(I)$, i.e. a function for a function symbol and a relation for a predicate symbol of corresponding arity. The value t^I of a term t in I is defined by the standard recursion. The structure obtained from I by assigning value v to symbol σ is denoted $I[\sigma : v]$.

Let Δ be an arbitrary definition. For each defined symbol X of Δ , we define:

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

where $\bar{x}$ is a tuple of new variables, and $\forall \bar{y}_1 (X(\bar{t}_1) \leftarrow \varphi_1), \dots, \forall \bar{y}_m (X(\bar{t}_m) \leftarrow \varphi_m)$ are the rules of Δ with X in the head.

The basis of the construction of the well-founded model is an operator T_Δ mapping pairs of τ -structures to τ -structures. Given such a pair (I, J) , the operator T_Δ evaluates the bodies of the rules in a particular way. In particular, it evaluates positive occurrences of defined symbols in rule bodies by I , and negative occurrences of defined symbols by J . To formally define this operator, we simply rename the negative occurrences in rule bodies of Δ . We extend the vocabulary τ with, for each defined symbol X , a new relation symbol X' of the same arity. The extended vocabulary $\tau \cup \bar{X}'$ will be denoted τ' . Then in each rule body in Δ , we substitute the symbol X' for each negative occurrence of a defined symbol X , thus obtaining a new definition Δ' .

Now all defined predicates of Δ' occur positively in the bodies of the rules, so Δ' is a positive definition based on vocabulary τ' . The new definition Δ' has the same defined symbols as Δ , and the symbols with primes are its new open symbols.

Let Δ' be obtained from Δ as described. As in the case of T_Δ , we obtain formula (2), but now we use Δ' instead of Δ . For any pair of τ -structures I, J which share the same domain, define I_J as the τ' -structure which satisfies the following:

- its domain is the same as the domain of I and J
- each open symbol of Δ is interpreted by J
- each defined symbol of Δ is interpreted by I
- the value of each new symbol X' is X^J , the value of X in J .

It is clear that for some defined symbol X , evaluating φ'_X under I_J simulates the non-standard structure of φ_X where J is “responsible” for the open and the negative occurrences of the defined predicates, while I is “responsible” for the positive ones.

Definition 1 (operator T_Δ). *We introduce a partially defined binary operator $T_\Delta : \mathcal{I} \times \mathcal{I} \mapsto \mathcal{I}$, where $\mathcal{I}$ is the class of all τ -structures. The operator is defined on pairs of structures which share the same domain, and is undefined otherwise. We have $I' = T_\Delta(I, J)$ if and only if*

- $\text{dom}(I') = \text{dom}(J) = \text{dom}(I)$,
- for each open symbol σ , $\sigma^{I'} := \sigma^J$ and
- for each defined symbol $X \in \tau_\Delta^d$, $X^{I'} := \{\bar{a} \mid I_J \models \varphi'_{X_i}[\bar{a}]\}$ where formula φ'_X is defined by equation (2) applied to Δ' .

Let J be a τ -structure, and $I_o := J|_{\tau_\Delta^o}$, the restriction of J to τ_Δ^o . The unary operator $\lambda J \ T_\Delta(I, J)$, often denoted by $T_\Delta(\cdot, J)$, is a monotone operator; and its least fixpoint in this lattice is computed by $\text{lfp}(T_\Delta(\cdot, J)) := \bigsqcup_\xi E^\xi$, where $E^\xi := T_\Delta(E^{<\xi}, J)$, and $E^{<\xi} := \bigsqcup_{\eta < \xi} E^\eta$.

Definition 2 (stable operator). *Define the stable³ operator $ST_\Delta : \mathcal{I} \mapsto \mathcal{I}$ as follows: $ST_\Delta(J) := \text{lfp}(T_\Delta(\cdot, J))$.*

Operator ST_Δ is anti-monotone. As is standard for anti-monotone operators, the operator ST_Δ gives rise to a sequence $(I^\xi, J^\xi)_{\xi \geq 0}$ defined by

$$\begin{aligned} I^\xi &:= ST_\Delta(J^{<\xi}), \quad \text{where } J^{<\xi} := \sqcap_{\eta < \xi} J^\eta, \\ J^\xi &:= ST_\Delta(I^{<\xi}), \quad \text{where } I^{<\xi} := \sqcup_{\eta < \xi} I^\eta. \end{aligned}$$

Notice that $I^{<0}$ is, by definition, the bottom element $\perp_{I_o}$ of the lattice of τ -structures extending I_o , i.e., the structure which assigns $\emptyset$ to every defined symbol; and $J^{<0}$ is the top element $\top_{I_o}$ which assigns Cartesian product A^r to each r -ary defined symbol X of Δ .

The anti-monotonicity of ST_Δ implies that the sequence $(I^\xi)_{\xi \geq 0}$ is increasing and $(J^\xi)_{\xi \geq 0}$ is decreasing. Moreover, for each ξ , $I^\xi \sqsubseteq J^\xi$. Thus, it holds that the sequence $(I^\xi, J^\xi)_{\xi \geq 0}$ is indeed a sequence of increasingly precise approximations. This sequence has a limit (I, J) , which is the maximal oscillating pair of ST_Δ . Equivalently, I and J are fixpoints of the square ST_Δ^2 , $\text{lfp}(ST_\Delta^2)$ and $\text{gfp}(ST_\Delta^2)$, respectively.

We define $I_o^{\Delta\downarrow} := \text{lfp}(ST_\Delta^2)$, and $I_o^{\Delta\uparrow} := \text{gfp}(ST_\Delta^2)$. We extend this notation to any structure K which interprets at least τ_Δ^o and define $K^{\Delta\downarrow} := (K|_{\tau_\Delta^o})^{\Delta\downarrow}$,

³ This operator is often called the Gelfond-Lifschitz operator. The corresponding transformation was introduced in [GL91].

and $K^{\Delta\uparrow} := (K|_{\tau_{\Delta}^o})^{\Delta\uparrow}$. Note that $K^{\Delta\downarrow}$ and $K^{\Delta\uparrow}$ agree with K on the open symbols but not necessarily on the defined symbols.

Definition 3. *Definition Δ is total in τ_{Δ}^o -structure I_o if $I_o^{\Delta\downarrow} = I_o^{\Delta\uparrow}$.*

The aim of an inductive definition is to *define* its defined symbols. Therefore, a natural quality requirement for a definition is that it is total. We will define that τ -structure I satisfies Δ , or equivalently, that Δ is true in I (denoted $I \models \Delta$) if Δ is total in I and $I^{\Delta\downarrow}$ is identical to I .

Definition 4 (Δ -extension of I). *Let Δ be a definition. Let I be a structure of vocabulary τ_I such that $\tau_{\Delta}^o \subseteq \tau_I \subseteq \tau$. If Δ is total in I , define the Δ -extension of I , denoted I^{Δ} , as $I^{\Delta} := I^{\Delta\downarrow}$ (or, equivalently, $I^{\Delta} := I^{\Delta\uparrow}$). If Δ is not total in I , then I has no Δ -extension.*

Note that for any τ_{Δ}^o -structure I_o , there is at most one Δ -extension extending I_o .

Definition 5 (ϕ true in structure I). *Let ϕ be a NMID-formula and I any structure such that $\text{free}(\phi) \subseteq \tau_I$. 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:*

- $I \models X(t_1, \dots, t_n)$ 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 $\text{dom}(I)$ of I , $I[\sigma : v] \models \psi$;
- $I \models \Delta$ if $I = I^{\Delta\downarrow} = I^{\Delta\uparrow}$.

Given an NMID-theory 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 $I \models T$.

3 Modularity

In this section, we split a definition Δ into subdefinitions $\{\Delta_1, \Delta_2, \dots, \Delta_n\}$. We study under what conditions we can guarantee that for structure I , $I \models \Delta$ iff $I \models \Delta_1 \wedge \Delta_2 \wedge \dots \wedge \Delta_n$. This is the subject of the Modularity theorem.

Definition 6 (partition of definitions). *A partition of definition Δ is a set $\{\Delta_1, \dots, \Delta_n\}$, $1 \leq n$, such that $\Delta = \Delta_1 \cup \dots \cup \Delta_n$, and if defined symbol P appears in the head of a rule of Δ_i , $1 \leq i \leq n$, then all rules of Δ with P in the head belong 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 $\tau = \tau_{\Delta}^o \cup \tau_{\Delta}^d = \tau_{\Delta_i}^o \cup \tau_{\Delta_i}^d$, $1 \leq i \leq n$. Also, $\cup_i \tau_{\Delta_i}^d = \tau_{\Delta}^d$ and $\tau_{\Delta_i}^d \cap \tau_{\Delta_j}^d = \emptyset$ whenever $i \neq j$.

The following theorem demonstrates that a model of a definition, is, at the same time, a model of each its sub-definitions. As a side effect, we demonstrate that the totality of the large definition implies the totality of its sub-definitions.

Theorem 1 (decomposition). *Let Δ be a definition over τ with partition $(\Delta_1, \dots, \Delta_n)$. Let I be a τ -structure. Then $I \models \Delta$ implies $I \models \Delta_1 \wedge \dots \wedge \Delta_n$.*

Let I be a model of Δ . The theorem says that for each i , $1 \leq i \leq n$, Δ_i is total in the restriction $I|_{\tau_{\Delta_i}^\circ}$ of I to the open symbols of Δ_i and moreover that I is the Δ_i -extension of $I|_{\tau_{\Delta_i}^\circ}$. Using the notations of Definition 4, this means that $I = I^{\Delta_i \downarrow} = I^{\Delta_i \uparrow}$. We obtain the following corollary.

Corollary 1. *If $I \models \Delta$, then for each i , $1 \leq i \leq n$, Δ_i is total in $I|_{\tau_{\Delta_i}^\circ}$.*

The following example shows that the inverse direction of theorem 1 does not hold in general.

Example 3. Let Δ , Δ_1 , Δ_2 be the following definitions.

$$\Delta := \left\{ \begin{array}{l} P \leftarrow Q, \\ Q \leftarrow P \end{array} \right\} \quad \Delta_1 := \{ P \leftarrow Q \} \quad \Delta_2 := \{ Q \leftarrow P \}.$$

Definition Δ is total, and its unique model is $\emptyset$ in which both P and Q are false. According to theorem 1, $\emptyset$ satisfies Δ_1 and Δ_2 . Note that $\{P, Q\}$ is not a model of Δ and yet, it satisfies Δ_1 and Δ_2 . Indeed, $\{P, Q\}$ is the Δ_1 -extension of the $\tau_{\Delta_1}^\circ$ -structure $\{Q\}$ and the Δ_2 -extension of the $\tau_{\Delta_2}^\circ$ -structure $\{P\}$. The reason why the equivalence between $\Delta_1 \wedge \Delta_2$ and Δ breaks down is that in Δ there is a circular dependency between P and Q . After splitting, this circular dependency is broken up into two non-circular dependencies.

The example suggests that splitting a definition will be equivalence preserving if the splitting does not break circular dependencies between atoms. To this end, we introduce the notion of a *reduction relation*. A reduction relation will be defined as a binary relation on ground atoms, which expresses possible dependencies between atoms in the definition.

A domain atom over vocabulary τ in domain A is any atom $P(x_1, \dots, x_n)$ where P is relation symbol of τ and $x_1, \dots, x_n$ are elements of A . Let At_A^τ be the set of domain atoms over τ in A . Let $\prec$ be any binary relation on At_A^τ . If $Q[\bar{b}] \prec P[\bar{a}]$, we will say that $P[\bar{a}]$ depends on $Q[\bar{b}]$ (according to $\prec$). We use $Q[\bar{b}] \prec P[\bar{a}]$ as an abbreviation for $Q[\bar{b}] \prec P[\bar{a}] \wedge P[\bar{a}] \not\prec Q[\bar{b}]$. 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 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 K_o be a structure with domain A interpreting at least all object and function symbols of τ and no defined predicates of Δ .

Definition 7 (reduction relation). *A binary relation $\prec$ on At_A^τ is a reduction relation (or briefly, a reduction) of Δ in K_o if for each domain atom $P[\bar{a}]$ with P a defined symbol, for all τ -structures I, J, I', J' extending K_o , if $(I, J) \cong_{\prec P[\bar{a}]} (I', J')$ then $I_J \models \varphi_P[\bar{a}]$ iff $I'_{J'} \models \varphi_P[\bar{a}]$.*

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 $\prec$.

Recall that a pre-well-order is a reflexive and transitive relation such that every non-empty subset contains a minimal element. Note that every definition Δ has a trivial reduction pre-well-order in every τ_Δ^0 -structure. Indeed, the total binary relation on At_A^τ trivially satisfies the conditions of a reduction pre-well-order. The following definition is crucial for the right-to-left direction of the Modularity theorem.

Definition 8 (reduction partition). *Call partition $\{\Delta_1, \dots, \Delta_n\}$ of definition Δ a reduction partition of Δ in τ_Δ^0 -structure I_o if there is a reduction pre-well-order $\prec$ of Δ in I_o and if $Q[\bar{b}] \prec P[\bar{a}]$ and $P[\bar{a}]$, $Q[\bar{b}]$ are not defined in the same Δ_i then $Q[\bar{b}] \ll P[\bar{a}]$.*

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 is strictly less in the reduction ordering and hence does not depend on the first atom.

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

Theorem 2 (modularity). *If $\{\Delta_1, \dots, \Delta_n\}$ is a total reduction partition of Δ in τ_Δ^0 -structure I_o , then for any τ -structure M extending I_o , $M \models \Delta_1 \wedge \dots \wedge \Delta_n$ iff $M \models \Delta$.*

Corollary 2. *Let T_o a theory over τ_Δ^0 such that for any τ_Δ^0 -model M_o of T_o , $\{\Delta_1, \dots, \Delta_n\}$ is a total reduction partition of Δ in M_o .*

Then $T_o \wedge \Delta$ and $T_o \wedge \Delta_1 \wedge \dots \wedge \Delta_n$ are logically equivalent.

Now we consider for two special cases of definitions how to translate them in 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 $PID(\Delta) := \bigwedge \Delta \wedge \forall \bar{X} (\bigwedge \Delta[\bar{P}/\bar{X}] \rightarrow \bar{P} \subseteq \bar{X})$. 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 $(\forall \bar{x} P_1(\bar{x}) \rightarrow X_1(\bar{x})) \wedge \dots \wedge (\forall \bar{x} P_n(\bar{x}) \rightarrow X_n(\bar{x}))$. The formula $PID(\Delta)$ is the standard second-order formula to express that predicates $\bar{P}$ satisfy the positive inductive definition Δ .

Theorem 3. *Δ is total in each τ_Δ^0 -structure. For any structure I , $I \models \Delta$ iff $I \models PID(\Delta)$.*

Define the *completion* of Δ , denoted $comp(\Delta)$, as the conjunction, for each defined symbol X of Δ , of formulas $\forall \bar{x} (X(\bar{x}) \leftrightarrow \varphi_X[\bar{x}])$. 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 $P \prec Q$ holds, then the bodies of rules defining Q may depend on the truth value of P , but not vice versa.

Theorem 4 (completion). *Suppose $\prec$ is a strict reduction relation of Δ in τ_{Δ}^{α} -structure I_o . Then (a) definition Δ is total in I_o , and (b) the model I_o^{Δ} of Δ is the unique model of $\text{comp}(\Delta)$ extending I_o .*

4 Example: Inductive Situation Calculus

The vocabulary τ_{sc} of the situation calculus (see, e.g. [Rei01]) is a many-sorted vocabulary with equality with sorts for actions (*Act*), situations (*Sit*), and possibly a finite number of domain specific sorts called object sorts ($Ob_1, \dots, Ob_k$). The vocabulary contains a potentially infinite set of domain-dependent function symbols of the sort *Act*. Such functions take only objects of object sorts as their arguments. For example, in the block world domain, we may have actions *pick_up*(x) of arity 1 and *put_on*(x, y) of arity 2. 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). They 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 .

Here 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-ordered set of situations. Ramifications describing propagation of effects of actions are modeled as 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 *foundational axioms of the inductive situation calculus*, $\mathcal{D}_f$, are the set of axioms consisting of the unique name axioms for situations

$$\begin{aligned} & \forall a \forall s \neg (S_0 = do(a, s)), \\ & \forall a_1 \forall a_2 \forall s_1 \forall s_2 (do(a_1, s_1) = do(a_2, s_2) \rightarrow a_1 = a_2 \wedge s_1 = s_2), \end{aligned} \quad (3)$$

and the domain closure axiom for situations

$$\exists X \left\{ \begin{array}{l} \forall s (X(s) \leftarrow s = S_0), \\ \forall s' \forall a (X(do(a, s')) \leftarrow X(s')) \end{array} \right\} \wedge \forall s X(s). \quad (4)$$

The role of axiom (4) 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. It is equivalent to Reiter's induction axiom for situations.

The second and main part of the inductive situation calculus is the definition Δ_{sc} defining the fluents. To define it, the vocabulary τ_{ind} of the inductive situation calculus extends τ_{sc} by three types of symbols. For each fluent F , the

symbol I_F is used to describe the fluent in the initial situation and has the same object arguments as F but misses the situation argument of F . The other two types of symbols are symbols denoting causality predicates. For each fluent F , we introduce predicates C_F and $C_{\neg F}$ with the same arguments as F plus one argument of sort *Act*. The atom $C_F(\bar{x}, a, s)$ means that action a has the direct or indirect effect of initiating F for objects $\bar{x}$ in situation s . Likewise, $C_{\neg F}(\bar{x}, a, s)$ represents that a has the effect of terminating F for $\bar{x}$ in situation s .

The definition Δ_{sc} simultaneously defines all fluents and causality predicates in terms of the initial situation predicates. This definition can be split in the partition $\{\Delta_{\text{fluent}}, \Delta_{\text{effect}}\}$. The subdefinition Δ_{fluent} consists of for each fluent F , the following three rules

$$\begin{aligned} \forall a \forall s \forall \bar{x} (F(\bar{x}, S_0) &\leftarrow I_F(\bar{x})), \\ \forall a \forall s \forall \bar{x} (F(\bar{x}, do(a, s)) &\leftarrow C_F(\bar{x}, a, s)), \\ \forall a \forall s \forall \bar{x} (F(\bar{x}, do(a, s)) &\leftarrow F(\bar{x}, s) \wedge \neg C_{\neg F}(\bar{x}, a, s)) \end{aligned}$$

The first rule of this subdefinition simply associates the fluent in the initial situation with its initial situation predicate. The second rule says that if a fluent is caused to hold in a situation, then it holds in that situation. The third rule models the inertia law.

The set Δ_{effect} defines all the causality predicates and contains rules of the form:

$$\begin{aligned} \forall a \forall s \forall \bar{x} (C_F(\bar{x}, a, s) &\leftarrow \gamma_F^+(\bar{x}, a, s)) \\ \forall a \forall s \forall \bar{x} (C_{\neg F}(\bar{x}, a, s) &\leftarrow \gamma_F^-(\bar{x}, a, s)) \end{aligned}$$

where $\gamma_F^+(\bar{x}, a, s)$ and $\gamma_F^-(\bar{x}, a, s)$ are formulas with only positive occurrences of causality predicates and in which the free variable s is the situation argument of all fluent and causality predicates. Notice that the definitional implications, $\leftarrow$, in these definitions represent causal dependencies. The semantic correspondence between causality rules and rules in an inductive definition was first pointed out in [DTV98].

A third and last part of the inductive situation calculus is the *description of the initial situation*, $\mathcal{D}_{S_0}$. It is a domain dependent theory containing neither fluent nor causality symbols. For example, in case of complete knowledge of the initial situations, $\mathcal{D}_{S_0}$ may consist of a set of definitions of the initial predicates I_F .

An inductive situation calculus consists then of the elements $\mathcal{D}_f \cup \{\Delta_{sc}\} \cup \mathcal{D}_{S_0}$.

Example 4 (gear wheels). Let us describe a simple idealized mechanical system of two connected gear wheels. Suppose fluents T_1, T_2 represent the fact that the first (respectively, the second) wheel is turning. If a wheel starts or stops turning, we denote this change by action $Start_1, Start_2, Stop_1$, or $Stop_2$, respectively. The following definitions specify direct and indirect effects of actions:

$$\Delta_{\text{effect}}^{\text{wheels}} := \left\{ \begin{array}{ll} \forall a \forall s (C_{T_1}(a, s) \leftarrow a = Start_1), & \forall a \forall s (C_{T_2}(a, s) \leftarrow a = Start_2), \\ \forall a \forall s (C_{\neg T_1}(a, s) \leftarrow a = Stop_1), & \forall a \forall s (C_{\neg T_2}(a, s) \leftarrow a = Stop_2) \\ \forall a \forall s (C_{T_1}(a, s) \leftarrow C_{T_2}(a, s)), & \forall a \forall s (C_{T_2}(a, s) \leftarrow C_{T_1}(a, s)) \\ \forall a \forall s (C_{\neg T_1}(a, s) \leftarrow C_{\neg T_2}(a, s)), & \forall a \forall s (C_{\neg T_2}(a, s) \leftarrow C_{\neg T_1}(a, s)) \end{array} \right\}$$

Each rule represents a local property of the system, namely an individual causal effect. By combining them we obtain a modular description of the entire system. Notice that the causal dependencies form a positive cycle.

Define $\Delta_{\text{fluent}}^{\text{wheels}}$ as the set of rules defining fluents T_1 and T_2 and $\Delta_{\text{sc}}^{\text{wheels}} := \Delta_{\text{fluent}}^{\text{wheels}} \cup \Delta_{\text{effect}}^{\text{wheels}}$. Let in the initial situation neither wheel be turning:

$$\mathcal{D}_{\text{init}}^{\text{wheels}} := \{\neg I_{T_1} \wedge \neg I_{T_2}\}.$$

Finally, let $\mathcal{D}_{\text{una}(\text{Act})}^{\text{wheels}}$ be the unique name axioms for the sort *Act*. Thus, the gear wheels example is axiomatized by

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

Our goal now is to simplify our axiomatization of this example. Using the properties proven in the previous sections, we are able to simplify our axiomatization of the Gear Wheels example by reducing definitions to formulas in classical logic.

Proposition 1. *The theory $\mathcal{D}^{\text{wheels}}$ is logically equivalent to*

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

Proof. We give a sketch of the proof to illustrate how the theorems of Section 3 can be used.

First, we show that $\Delta_{\text{fluent}}^{\text{wheels}}$ is total in each structure satisfying the foundational axioms $\mathcal{D}_{\text{f}}$ and moreover, that

$$\mathcal{D}_{\text{f}} \models \Delta_{\text{fluent}}^{\text{wheels}} \leftrightarrow \text{comp}(\Delta_{\text{fluent}}).$$

To this aim, we show that in each structure I satisfying $\mathcal{D}_{\text{f}}$, there is a strict reduction relation of $\Delta_{\text{fluent}}^{\text{wheels}}$. Consider the following binary relation $\prec_1$:

$$\left\{ \begin{array}{l} (I_{T_i}, T_i(S_0^I)) \\ (T_i(s), T_i(\text{do}^I(a, s))) \\ (C_{T_i}(a, s), T_i(\text{do}^I(a, s))) \\ (C_{\neg T_i}(a, s), T_i(\text{do}^I(a, s))) \end{array} \middle| i = 1, 2 \wedge a \in \text{Act}^I \wedge s \in \text{Sit}^I \right\}$$

It expresses that $T_i(S_0)$ depends on its initial predicate, and $T_i(\text{do}^I(a, s))$ depends on $T_i(s)$, $C_{T_i}(a, s)$ and $C_{\neg T_i}(a, s)$. It is easy to verify that $\prec_1$ is a reduction relation in I and that its transitive closure is a strict reduction relation of $\Delta_{\text{fluent}}^{\text{wheels}}$. Then Theorem 4 guarantees that Δ_{fluent} is total in I and that Δ_{fluent} and $\text{comp}(\Delta_{\text{fluent}})$ are equivalent in I .

Next, consider that $\Delta_{\text{effect}}^{\text{wheels}}$ is a positive inductive definition, so by Theorem 3, it is total in each structure and it is equivalent to $\text{PID}(\Delta_{\text{effect}}^{\text{wheels}})$.

We have shown now that given $\mathcal{D}_{\text{f}}$ the component definitions $\Delta_{\text{fluent}}^{\text{wheels}}$ and $\Delta_{\text{effect}}^{\text{wheels}}$ can be translated to the appropriate classical logic axioms. It remains to be shown that $\Delta_{\text{sc}}^{\text{wheels}}$ can be split in the conjunction $\Delta_{\text{fluent}}^{\text{wheels}} \wedge \Delta_{\text{effect}}^{\text{wheels}}$. Consider the following binary relation:

$$\prec_2 := \prec_1 \cup \left\{ \begin{array}{l} (C_{T_1}(a, s), C_{T_2}(a, s)) \\ (C_{T_2}(a, s), C_{T_1}(a, s)) \\ (C_{\neg T_1}(a, s), C_{\neg T_2}(a, s)) \\ (C_{\neg T_2}(a, s), C_{\neg T_1}(a, s)) \end{array} \middle| a \in \text{Act}^I \wedge s \in \text{Sit}^I \right\}$$

It is easy to verify that $\prec_2$ is a reduction relation of Δ_{sc}^{wheels} and, since each superset of a reduction relation is obviously also a reduction relation, its transitive reflexive closure $\prec_2^*$ is also a reduction relation. Also, one can verify that $\prec_2^*$ is a pre-well-order on At_A^r . Moreover, no causality atom depends on a fluent atom according to $\prec_2^*$. Consequently, if $P[\bar{a}]$ is a fluent atom and $Q[\bar{b}]$ a causation atom, then $P[\bar{a}] \not\prec_2^* Q[\bar{b}]$ and $Q[\bar{b}] \prec_2^* P[\bar{a}]$ implies $Q[\bar{b}] \prec_2^* P[\bar{a}]$. We already showed that Δ_{fluent}^{wheels} and Δ_{effect}^{wheels} are total in each structure satisfying $\mathcal{D}_f$. It follows that $\{\Delta_{fluent}^{wheels}, \Delta_{effects}^{wheels}\}$ is a total reduction partition of Δ_{sc}^{wheels} in each structure satisfying $\mathcal{D}_f$. We can apply Theorem 2 and the proposition follows.

5 Conclusions

Recently, we argued [Den00,DBM01] that non-monotone forms of inductive definitions such as iterated inductive definitions and definitions over well-orders, can play a unifying role in logic, AI and knowledge representation, connecting remote areas such as non-monotonic reasoning, logic programming, description logics, deductive databases and fixpoint logics. In this paper, we further substantiated this claim by defining a more general logic integrating classical logic and monotone and non-monotone inductive definitions and applying it to what has always been the most important test domain of knowledge representation — temporal reasoning.

The main technical theorems here are the Modularity theorem and the theorems translating certain classes of NMID-definitions into classical logic formalisations. Problem-free composition is crucial while axiomatizing a complex system. Because definitions in our logic are non-monotone, composing or decomposing definitions is in general not equivalence preserving. However, the conditions we have presented allow one to separate problem-free (de)compositions from those causing change in meaning. We have shown that the Modularity theorem is useful also for analyzing complex definitions — some properties of large definitions are implied by properties of sub-definition. The Modularity theorem is also an important tool for simplifying logical formulas with definitions by translating them into formulas of classical logic.

As an application of our logic, we presented the inductive situation calculus. In this formalisation, the situation calculus is modeled as an inductive definition, defining fluents and causality predicates by simultaneous induction in the well-ordered set of situations. An important aspect of our formalisation is that causation rules can be represented in a modular way by rules in an inductive definition. We used the Modularity theorem to demonstrate its equivalence with a situation calculus axiomatization based on completion and circumscription. In a forthcoming paper, we present a general solution to the ramification problem for the situation calculus within the NMID-logic.

Future Work Given the connection between the μ -calculus and inductive definitions, it would be interesting to study NMID-logic on Kripke structures, and to understand what form the splitting conditions take on these structures. This may lead to new results on compositional verification. Imagine a complex system

consisting of numerous inter-related components (e.g. gear wheels). A formal description of the behavior of such a system over time requires mutual recursion between its modules. If the specification can be decomposed into specifications of individual parts, then the complexity of verification of the entire system will be reduced since one has to deal with a reduced state space each time. Other directions include the study of the connections of our logic with the work on iterated induction by Feferman and others, and also applications of our modularity techniques for simplifying formulas with definitions through their translation to first- and second-order logic.

References

- [Acz77] P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 739–782. Elsevier, 1977.
- [DBM01] M. Denecker, M. Bruynooghe, and V. Marek. Logic programming revisited: Logic programs as inductive definitions. *ASM Transactions on Computational Logic (TOCL)*, 4(2), 2001.
- [Den98] M. Denecker. The well-founded semantics is the principle of inductive definitions. In J. Dix, L. Farinas del Cerro, and U. Furbach, editors, *Logics in Artificial Intelligence*, volume 1489 of *Lecture Notes in Artificial Intelligence*, pages 1–16. Springer-Verlag, 1998.
- [Den00] M. Denecker. Extending classical logic with inductive definitions. In *Proc. CL'2000*, 2000.
- [DG02] A. Dawar and Y. Gurevich. Fixed point logics. *The Bulletin of Symbolic Logic*, 8(1):65–88, 2002.
- [DMT00] M. Denecker, V. Marek, and M. Truszczyński. Approximating operators, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In J. Minker, editor, *Logic-Based AI*, pages 127–144. Kluwer Academic Publishers, Boston/Dordrecht/London, 2000.
- [DTV98] M. Denecker, D. Theseider Duprè, and K. Van Belleghem. An inductive definition approach to ramifications. *Linköping Electronic Articles in Computer and Information Science*, 3(1998): nr 7, 1998. URL: <http://www.ep.liu.se/ea/cis/1998/007/>.
- [Fit03] M. Fitting. Fixpoint semantics for logic programming - a survey. *Theoretical Computer Science*, 2003. To appear.
- [GL91] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
- [Mos74] Y.N. Moschovakis. *Elementary Induction on Abstract Structures*. Amsterdam, North Holland, 1974.
- [Rei01] R. Reiter. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press, 2001.
- [Sch95] J. Schlipf. The expressive powers of the logic programming semantics. *Journal of Computer and System Sciences*, 51:64–86, 1995.
- [Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5:285–309, 1955.
- [Van93] A. Van Gelder. An alternating fixpoint of logic programs with negation. *Journal of computer and system sciences*, 47:185–221, 1993.
- [VDD00] S. Verbaeten, M. Denecker, and D. De Scheye. Compositionality of normal open logic programs. *journal of Logic Programming*, 1(3):151–183, 2000.
- [VRS91] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. *Journal of Assoc. Comput. Mach.*, 38(3):620–650, 1991.