Safe Inductions: An Algebraic Study*

Bart Bogaerts and Joost Vennekens and Marc Denecker
KU Leuven, Department of Computer Science
Celestijnenlaan 200A, Leuven, Belgium

Abstract

In many knowledge representation formalisms, a constructive semantics is defined based on sequential applications of rules or of a semantic operator. These constructions often share the property that rule applications must be delayed until it is safe to do so: until it is known that the condition that triggers the rule will remain to hold. This intuition occurs for instance in the well-founded semantics of logic programs and in autoepistemic logic. In this paper, we formally define the safety criterion algebraically. We study properties of so-called safe inductions and apply our theory to logic programming and autoepistemic logic. For the latter, we show that safe inductions manage to capture the intended meaning of a class of theories on which all classical constructive semantics fail.

1 Introduction

In many fields of computational logic, natural forms of induction show up. Such an induction can be seen as a sequence of semantic structures obtained by iterative applications of rules or a semantic operator. For instance, in logic programming, it is natural to think of sequences of interpretations where at each stage a number of rules whose bodies are satisfied are triggered (i.e., their head is added to the current interpretation). For positive logic programs, all such sequences converge to the minimal model. For non-positive programs, this strategy may yield meaningless results. For instance, for the program $P = \{a, \ b \leftarrow \neg a\}$, one such sequence is $\emptyset, \{b\}, \{b, a\}$, the limit of which is not even a supported model of the logic program. Intuitively, what is wrong here is that the rule $b \leftarrow \neg a$ is applied before the value of $a$ is established. For stratified programs, like $P$, this problem has been resolved [Apt et al., 1988]. For the general case, the well-founded semantics [Van Gelder et al., 1991] offers a solution that uses three-valued interpretations instead of two-valued interpretations.

In recent work, the notions of natural and safe inductions for inductive definitions were introduced [Denecker and Vennekens, 2014; Denecker et al., 2017]. It was argued that this kind of processes forms the essence of our understanding of inductive definitions.

In this paper, we lift those ideas to a more general setting: we provide a principled study of inductions in approximation fixpoint theory (AFT) (Denecker, Marek and Truszczyński (DMT) 2000), an algebraic theory that provides a unifying framework of semantics of nonmonotonic logics. We show convergence of safe inductions in general and study the relationship between safe inductions and various fixpoints defined in approximation fixpoint theory.

By presenting our theory in AFT, our results are broadly applicable. DMT [2000] originally developed AFT to unify semantics of logic programs, autoepistemic logic and default logic. Later, it was also used to define semantics of extensions of logic programs, such as HEX logic programs [Antic et al., 2013] and an integration of logic programs with description logics [Liu et al., 2016]. Strass [2013] showed that many semantics from Dung’s argumentation frameworks (AFs) [Dung, 1995] and abstract dialectical frameworks (ADFs) [Brewka et al., 2013] can be obtained by direct applications of AFT. Cruz-Filipe [2016] and Bogaerts and Cruz-Filipe [2017] showed that AFT has applications in database theory, for defining semantics of active integrity constraints [Flesca et al., 2004].

The theory we present in this paper induces for each of the above logics notions of (safe) inductions and a safe semantics. Our complexity results are obtained for general operators and hence can also be transferred to various logics of interest. Throughout the paper, we give examples from logic programming: in Section 6, we apply our theory to autoepistemic logic. There, we show that safe inductions induce a constructive semantics that captures the intended semantics of a class of theories for which classical constructive semantics fail. This failure was recently exposed and solved using a notion of set-inductions which is based on sets of lattice elements instead of intervals (which are standard in AFT) [Bogaerts et al., 2016]. We show that safe inductions provide an alternative solution to this problem. Our solution is more direct: in contrast to set-inductions or well-founded inductions [Denecker and Vennekens, 2007], safe inductions do not require any form of approximation; they are sequences in the original lattice. For logic programming, this means that they are sequences of interpretations such that some atoms are derived in each step. For AEL, this means that they are

* Bart Bogaerts is a postdoctoral fellow of the Research Foundation – Flanders (FWO).
sequences of possible world structures such that additional knowledge is derived in each step.

2 Preliminaries: Lattices and Operators

A partially ordered set (poset) \((L, \leq)\) is a set \(L\) equipped with a partial order \(\leq\), i.e., a reflexive, antisymmetric, transitive relation. We write \(x < y\) for \(x \leq y \land x \neq y\). We call \((L, \leq)\) a complete lattice if every subset \(S\) of \(L\) has a least upper bound \(\bigvee S\) and a greatest lower bound \(\bigwedge S\). A complete lattice has a least element \(\bot\) and a greatest element \(\top\). We use notations \(x \lor y = \bigvee\{x, y\}\) and \(x \land y = \bigwedge\{x, y\}\).

An operator \(O : L \to L\) is monotone if \(x \leq y\) implies that \(O(x) \leq O(y)\). An element \(x \in L\) is a prefixpoint, a fixpoint, a postfixpoint of \(O\) if \(O(x) \leq x\), respectively \(O(x) = x\), \(x \leq O(x)\). Every monotone operator \(O\) in a complete lattice has a least fixpoint [Tarski, 1955], denoted \(\text{lfp}(O)\), which is also \(O\)'s least prefixpoint and the limit of any monotone induction, i.e., of any increasing sequence \((x_i)_{i \geq 0}\) satisfying

- \(x_0 = \bot\),
- \(x_i \leq x_{i+1} \leq O(x_i)\), for successor ordinals \(i + 1\),
- \(x_\lambda = \bigvee\{x_i \mid i < \lambda\}\), for limit ordinals \(\lambda\).

Logic Programming Let \(\Sigma\) be an alphabet, i.e., a collection of symbols which are called atoms. A logic program \(P\) is a set of rules \(r\) of the form \(h \leftarrow \varphi\), where \(h\) is an atom called the head of \(r\), denoted \(\text{head}(r)\), and \(\varphi\) is a conjunction of literals called the body of \(r\), denoted \(\text{body}(r)\). An interpretation \(I\) of \(\Sigma\) is a subset of \(\Sigma\). The set of interpretations \(2^\Sigma\) forms a lattice equipped with the order \(\subseteq\). The truth value (\(t\) or \(f\)) of a propositional formula \(\varphi\) in a structure \(I\), denoted \(\varphi^I\), is defined as usual. With a logic program \(P\), we associate an immediate consequence operator \(T_P\) [van Emden and Kowalski, 1976] that maps a structure \(I\) to the structure

\[
\{p \in \Sigma \mid \exists r \in P : \text{head}(r) = p \land \text{body}(r)^I = t\}.
\]

This is an operator on the lattice \((2^\Sigma, \subseteq)\).

3 Safe Inductions

Let \(L\) be a lattice and \(O\) an operator on \(L\), fixed throughout the rest of this paper.

Definition 3.1. We call \(y \in L\) derivable from \(x \in L\) if \(x \leq y \leq O(x)\).

Definition 3.2. Let \(x\) be an element of \(L\). An \(O\)-induction in \(x\) is a sequence \((x_i)_{i \leq \beta}\) such that

- \(x_0 = x\),
- \(x_{i+1}\) is derivable from \(x_i\) for each \(i < \beta\),
- \(x_\lambda = \bigvee\{x_i \mid i < \lambda\}\), for limit ordinals \(\lambda \leq \beta\).

Intuitively, we view \(O\) as an operator that constructs certain lattice points. An \(O\)-induction is the associated construction process. Intuitively, if we are at a stage \(x_i\), \(O(x_i)\) represents what can be concluded from this given stage. Therefore, the next step \(x_{i+1}\) in the induction is at least \(x_i \leq x_{i+1} \geq x_i\) and at most the combination of \(x_i\) and what can be concluded from it \((x_i \leq x_{i+1} \leq O(x_i))\). In the context of a powerset lattice (a lattice of the form \((2^S, \subseteq))\), this means that \(x_{i+1} \subseteq x_i \cup O(x_i)\), i.e., \(x_{i+1}\) only contains elements that were already in \(x_i\) or such that \(O\) concludes them from \(x_i\).

Definition 3.3. Let \(N = (x_i)_{i \leq \beta}\) and \(N' = (y_i)_{i \leq \alpha}\) be two \(O\)-inductions. We say that \(N'\) extends \(N\) if \(\alpha \geq \beta\) and \(x_i = y_i\) for all \(i \leq \beta\). The extension is strict if \(y_\alpha \neq x_\beta\).

Definition 3.4. An \(O\)-induction is terminal if there exists no \(O\)-induction that strictly extends it.

Proposition 3.5. An \(O\)-induction \((x_i)_{i \leq \beta}\) is terminal if and only if \(x_\beta\) is a prefixpoint of \(O\).

Proposition 3.6. If \(O\) is monotone, all monotone inductions are \(O\)-inductions in \(\bot\) and vice versa.

Corollary 3.7. If \(O\) is monotone, all terminal \(O\)-inductions in \(\bot\) converge to \(\text{lfp}(O)\).

There is a high degree of non-determinism in \(O\)-inductions. For monotone operators \(O\), despite this non-determinism, all \(O\)-inductions converge to the same point. As such, \(O\)-inductions provide (if \(O\) is monotone) a way to construct an intended lattice point (\(\text{lfp}(O)\)). For non-monotone operators, the situation is quite different: \(O\)-inductions might not converge to a single point.

Example 3.8. Let \(P\) be the logic program

\[
\{ \ p , \ q \leftarrow \neg p \ \}
\]

This is a simple, stratified logic program [Apt et al., 1988; Przymusinski, 1988]. Its intended fixpoint (its so-called perfect model) is \(\{p\}\). Let \(T_P\) denote its immediate consequence operator. The following are the three terminal strict \(T_P\)-inductions in \(\bot = \emptyset\).

- \(N_1 = (\emptyset, \{q\}, \{p, q\})\)
- \(N_2 = (\emptyset, \{p, q\})\)
- \(N_3 = (\emptyset, \{p\})\)

In Example 3.8 it can be seen that certain derivations in an \(O\)-induction happen prematurely. For instance, in \(N_1\) and \(N_2\), \(q\) is derived by the non-monotonic rule \(q \leftarrow \neg p\). As soon as \(p\) is derived, this rule no longer applies: \(q \notin T_P(\{p, q\}) = \{p\}\). In this sequence, the rule was applied when it was not safe to do so. Below, we define a notion of safety to avoid such premature derivations, i.e., to only derive facts that remain derivable, regardless of what other derivations are made further on in the induction process.

Definition 3.9. Let \(x'\) be derivable from \(x\). We say that \(x'\) is safely derivable from \(x\) if for each \(O\)-induction \((x_i)_{i \leq \beta}\) in \(x\), it holds that \(x' \leq x \lor O(x_i)\).

An \(O\)-induction \((x_i)_{i \leq \beta}\) is safe if \(x_{i+1}\) is safely derivable from \(x_i\) for each \(i < \beta\).

In words, \(x'\) is safely derivable from \(x\) if no matter what other derivations we make (ending up in \(x_\beta\)), \(x'\) consists at most of what we have in \(x\) combined with what \(O\) concludes from \(x_\beta\), i.e., \(x' \leq x \lor O(x_\beta)\).

An induction is terminal if it cannot be extended into a strictly larger induction. We define a similar concept for safe inductions.

Definition 3.10. A safe \(O\)-induction \(N\) is safe-terminal if there exists no strict extension \(N'\) of \(N\) that is safe.

In Example 3.8, we showed that not all terminal \(O\)-inductions converge to the same lattice point. Luckily, the safety criterion warrants a better situation.

Theorem 3.11. For each \(x \in L\), all safe-terminal \(O\)-inductions in \(x\) converge to the same lattice point.
In order to prove the following result.

**Lemma 3.12.** Let \( N = (x_i)_{i \leq \beta} \) and \( N' = (y_i)_{i \leq \gamma} \) be two safe \( O \)-inductions with \( x_0 = y_0 \). For every \( i \leq \beta, j \leq \gamma \) it holds that if \( i + 1 \leq \beta \), then \( x_{i+1} \lor y_j \) is safely derivable from \( x_i \lor y_j \) and if \( j + 1 \leq \gamma \), then \( x_i \lor y_{j+1} \) is safely derivable from \( x_i \lor y_j \).

**Proof of Theorem 3.11.** Let \( N = (x_i)_{i \leq \beta} \) and \( N' = (y_i)_{i \leq \gamma} \) be two safe-terminal \( O \)-inductions. Consider the sequence \((z_i)_{i \leq \beta + \gamma}\) where \( z_i = x_i \) if \( i \leq \beta \) and \( z_{\beta + j} = x_{\beta} \lor y_j \) if \( i > \beta \). By Lemma 3.12, this sequence is a safe \( O \)-induction.

Since \( N \) is safe-terminal, this sequence cannot be an strict extension of \( N \) and hence \( x_\beta \lor y_\gamma = z_{\beta + \gamma} = x_\beta \lor y_\gamma \). A symmetric argument shows that \( x_\beta \leq y_\gamma \), hence \( x_\beta = y_\gamma \), as desired. \( \square \)

**Definition 3.13.** The safely defined point by \( O \), denoted \( safe(O) \), is the limit of all safe-terminal \( O \)-inductions in \( L \).

By Theorem 3.11, the safely defined point is well-defined. We now study some properties of the safely defined point.

**Proposition 3.14.** For any operator \( O \), \( safe(O) \) is a postfixed-point of \( O \), i.e., \( safe(O) \subseteq O(safe(O)) \).

**Example 3.15.** Consider a lattice \( \{ \bot, \top \} \) with two elements and an operator \( O \) that maps \( \bot \) to \( \top \) and \( \top \) to \( \bot \). The safely defined point by \( O \) is \( \bot \), since \( O(\bot) = \top \), and \( \top \) is not safely derivable \( \top \notin \bot \lor O(\top) \).

Here, the \( O \)-induction \( N = (\bot) \) is safe-terminal, but not terminal. \( \blacksquare \)

**Definition 3.16.** We call an operator \( O \) complete if the safely defined point by \( O \) is a fixpoint of \( O \), i.e., if \( O(safe(O)) = safe(O) \).

We will be mostly interested in complete operators \( O \), as they uniquely determine a fixpoint of interest of \( O \).

**Proposition 3.17.** An operator \( O \) is complete if and only if every safe-terminal \( O \)-induction in \( L \) is terminal.

**Proposition 3.18.** If \( O \) is a monotone operator, then \( O \) is complete and \( safe(O) = \text{lfp}(O) \).

Theorem 3.11 shows that safe \( O \)-inductions, despite their non-determinism, uniquely determine a lattice point of interest. Furthermore, if \( O \) is monotone, this point is the least fixpoint of \( O \). The question now arises: what if \( O \) is non-monotone? How does the safely defined point by \( O \) relate to other points of interest? In particular, how does it relate to fixpoints defined in approximation fixpoint theory? We study this in Section 5. First, we study complexity.

*Complexity.* The **height** of a finite lattice \( L \) is the length \( n \) of the longest sequence \( \bot = x_0 < x_1 < \cdots < \top = x_n \) in \( L \). We call \( y \in L \) a direct successor of \( x \in L \) if \( x < y \) and there is no \( z \) such that \( x < z < y \). The **branching width** of a finite \( L \) is the maximum over \( x \in L \) of the number of direct successors of \( x \). All complexity results presented below are in terms of the sum of the branching width and the height of the input lattice. This means that we use the sum of the branching width and the height of the measure of our input.

In this section, we assume that a class \( C = \{(L, O)\} \) of pairs of a finite lattice \( L \) and an operator \( O : L \to L \) is given. Let \( F_C \) denote the function problem: given one of the \( \{L, O\} \in C \) and \( p, p' \) in \( L \), compute (1) \( O(p) \), (2) \( p \lor p' \), and (3) \( \{x \mid x \text{ is a direct successor of } p\} \). We assume that \( F_C \) can be solved in polynomial time.

The kind of setting used here is not so unusual: it is an algebraic variant of data complexity. For instance, in logic programming, each non-ground program \( P \) determines a class of lattices and associated operators (immediate consequence operators of the groundings of \( P \) with respect to a given domain). The height and branching width of the lattice are then polynomial in terms of the domain size. In this setting, the problem \( F_{C_P} \) is indeed polynomially solvable.

**Theorem 3.19.** Let \( C \) be a class as above. The decision problem given \( L_i \in C, x, y \in L_i \), is \( y \text{ safely derivable from } x \text{ by } O \)? is in \( \text{co-NP} \).

**Sketch of the proof.** To prove this, we build a program that nondeterministically traverses an \( O \)-induction from \( x \). To determine that \( y \) is not safely derivable from \( x \), it suffices to find one run of the algorithm such that \( y \nleq x \lor O(s) \) with \( s \) a state in the \( O \)-induction. Such algorithm runs in polynomial time in the height of the lattice. \( \square \)

**Theorem 3.20.** Let \( C \) be a class as above. The decision problem given \( L \in C, x, y \in L \), is \( safe(O) \) \( \geq \) \( s \) is in \( \left( \Delta_2^P \right) \).

For some classes \( C \), this problem is \( \text{co-NP-hard} \).

**Sketch of the proof.** Containment follows from Theorem 3.19. Hardness follows from Theorem 6.10 of Denecker et al. [2017]. \( \square \)

4 Preliminaries: AFT

Given a lattice \( L \), approximation fixpoint theory makes use of the lattice \( L^2 \). We define **projections** for pairs as usual: \( (x, y)_1 = x \) and \( (x, y)_2 = y \). Pairs \( (x, y) \in L^2 \) are used to approximate all elements in the interval \( [x, y] = \{z \mid x \leq z \land z \leq y\} \). We call \( (x, y) \in L^2 \) **consistent** if \( x \leq y \). We use \( L^c \) to denote the set of consistent elements. Elements \( (x, y) \in L^c \) are called **exact**. We sometimes use the tuple \( (x, y) \) and the interval \( [x, y] \) interchangeably. The **precision ordering** on \( L^2 \) is defined as \( (x, y) \leq_{p} (u, v) \) if \( x \leq u \) and \( v \leq y \). In case \( (u, v) \) is consistent, this means that \( (x, y) \) approximates all elements approximated by \( (u, v) \), or in other words that \( [u, v] \subseteq [x, y] \). If \( L \) is a complete lattice, then \( L^2, \leq_{p} \) is also a complete lattice.

AFT studies fixpoints of lattice operators \( O : L \to L \) through operators approximating \( O \). An operator \( A : L^2 \to L^2 \) is an **approximator** of \( O \) if it is \( \leq_{p} \)-monotone, and \( O(x) \in A(x, x) \) for all \( x \in L \). Approximators map \( L^c \) into \( L^c \). As usual, we restrict our attention to **symmetric** approximators: approximators \( A \) such that for all \( x \) and \( y \), \( A(x, y)_1 = A(y, x)_2 \). DMT [2004] showed that the consistent fixpoints of interest (defined below) are uniquely determined by an approximator’s restriction to \( L^c \), hence, sometimes we only define approximators on \( L^c \).

AFT studies fixpoints of \( O \) using fixpoints of \( A \). The A-Kripke-Kleene fixpoint is the \( \leq_p \)-least fixpoint of \( A \) and has the property that it approximates all fixpoints of \( O \). A partial \( A \)-stable fixpoint is a pair \((x, y)\) such that \( x = \text{lfp}(A(\cdot, y)_1) \)
and \( y = \text{lfp}(A(x, \cdot)_2) \), where \( A(\cdot, y)_1 \) denotes the operator \( L \mapsto x \mapsto A(x, y)_1 \) and analogously for \( A(x, \cdot)_2 \). The \( A \)-well-founded fixpoint is the least precise partial \( A \)-stable fixpoint. An \( A \)-stable fixpoint of \( O \) is a fixpoint \( x \) of \( O \) such that \( (x, x) \) is a partial \( A \)-stable fixpoint. This is equivalent to the condition that \( x = \text{lfp}(A(\cdot, x)_1) \). \( A \)-stable fixpoints are minimal fixpoints of \( O \). The \( A \)-Kripke-Kleene fixpoint of \( O \) can be constructed by iterative applications of \( A \), starting from \( (\bot, \top) \). For the \( A \)-well-founded fixpoint, a similar constructive characterization has been worked out.

**Definition 4.1.** An \( A \)-refinement of \( (x, y) \) is a pair \( (x', y') \in L^2 \) satisfying one of the following two conditions:
- \( (x, y) \leq_p (x', y') \leq_p A(x, y) \), or
- \( x' = x \) and \( A(x, y)_2 \leq y' \leq y \).

An \( A \)-refinement is strict if \( (x, y) \neq (x', y') \).

**Definition 4.2.** A well-founded induction of \( A \) is a sequence \((x_i, y_i)_{i \in \beta}\) with \( \beta \) an ordinal such that:
- \((x_0, y_0) = (\bot, \top)\);
- \((x_{i+1}, y_{i+1}) \) is an \( A \)-refinement of \((x_i, y_i)\), for all \( i < \beta \);
- \((x_i, y_i) = \bigvee_{\lambda \leq \beta} \{(x_{i}, y_{i}) | i < \lambda \}\) for limit ordinals \( \lambda \leq \beta \).

A well-founded induction is terminal if its limit \((x_\beta, y_\beta)\) has no strict \( A \)-refinements.

Denecker and Vennekens [2007] showed that all terminal \( A \)-inductions converge to the \( A \)-well-founded fixpoint of \( O \).

**Logic Programming** For logic programming, DMT [Fitting, 2002] is an approximator of \( O \)-fixpoint. The \( \Psi \)-well-founded fixpoint is the well-founded model of \( P \) [Van Gelder et al., 1991] and that \( \Psi \)-stable fixpoints are the stable models of \( P \) [Gelfond and Lifschitz, 1988].

5 Safe Inductions and AFT

In this section, we study how (safe) \( O \)-inductions relate to the fixpoints studied in AFT.

**Theorem 5.1.** Let \( O \) be an operator and \( A \) an approximator of \( O \). The \( A \)-well-founded fixpoint approximates the safely defined point of \( O \).

The proof makes use of the following proposition.

**Proposition 5.2.** Let \( O \) be an operator and \( A \) an approximator of \( O \). Let \((x_i, y_i)_{i \leq \beta}\) be an \( A \)-well-founded induction. The following claims hold:
- (1) \((x_i)_{i \leq \beta}\) is a safe \( O \)-induction, and
- (2) for each \( i \leq \beta \) and each \( O \)-induction \( N = (z_j)_{j \leq \alpha} \) with \( z_0 = x_i \), it holds that \( z_\alpha \leq y_\beta \).

**Proof of Theorem 5.1.** Let \( z \) denote the safely defined point of \( O \) and \((x_\beta, y_\beta)\) denote the \( A \)-well-founded fixpoint of \( O \). For any terminal \( A \)-well-founded induction \((x_i, y_i)_{i \leq \beta}\), it holds that \( x_\beta \leq z \) by the first point of Proposition 5.2. Furthermore, by the second point of Proposition 5.2 it holds that any \( O \)-induction stays under \( y_\beta \); hence \( z \leq y_\beta \).

**Corollary 5.3.** If the \( A \)-well-founded fixpoint of \( O \) is exact, i.e., equal to \((x, x)\) for some \( x \in L \), then \( O \) is complete and safe(\( O \)) = \( x \).

**Corollary 5.4.** Let \( O \) be an operator and \( A \) an approximator of \( O \). The \( A \)-Kripke-Kleene fixpoint of \( O \) approximates the safely defined point of \( O \).

**Corollary 5.5.** If the \( A \)-Kripke-Kleene fixpoint of \( O \) is exact, i.e., equal to \((x, x)\) for some \( x \in L \), then \( O \) is complete and safe(\( O \)) = \( x \).

Safe \( O \)-inductions identify a unique lattice point of interest. Since an operator can have multiple stable fixpoints, we cannot expect a strong link between the safely defined point and stable fixpoints. However, we do find the following relation between stable fixpoints and \( O \)-inductions.

**Theorem 5.6.** Let \( A \) be an approximator of \( O \). If \( x \) is an \( A \)-stable fixpoint of \( O \), then \( x \) is the limit of a terminal \( O \)-induction.

**Sketch of the proof.** If \( x \) is an \( A \)-stable fixpoint of \( O \), then \( x = \text{lfp}(A(\cdot, x)_1) \). The clue to proving this proposition is to show that monotone inductions of \( A(\cdot, x)_1 \) are \( O \)-inductions. The result then easily follows.

**Example 5.7.** Consider the logic program
\[
\mathcal{P} = \{ \ p \leftarrow \neg q, \quad q \leftarrow \neg p \ \}
\]
It holds that \( \{p\} \) is a stable model of \( \mathcal{P} \) (a \( \Psi \)-stable fixpoint of \( T_\mathcal{P} \)). Also, \( \{p\} \) is the limit of the \( T_\mathcal{P} \)-induction \( (\emptyset, \{p\}) \). This induction is not safe since \( (\emptyset, \{q\}) \) is also a \( T_\mathcal{P} \)-induction and \( \{p\} \not\subseteq T_\mathcal{P}(\{q\}) \lor \emptyset = \{q\} \).

The limit of a terminal \( O \)-induction is not always a stable fixpoint of \( O \) (for some approximator \( A \)), as we show below.

**Example 5.8.** Consider the logic program
\[
\mathcal{P} = \{ \ p \leftarrow p, \quad p \leftarrow q, \quad q \leftarrow q \ \}
\]
In this case \((\emptyset, \{q\}, \{q, p\})\) is the unique terminal \( T_\mathcal{P} \)-induction. It can be verified that this is a safe induction and that \( T_\mathcal{P} \) is complete. The safely defined point is a non-minimal fixpoint of \( T_\mathcal{P} \), hence it is also non-grounded (see [Bogaerts et al., 2015]) and not an \( A \)-stable fixpoint for any approximator \( A \) of \( T_\mathcal{P} \). In the well-founded model of \( \mathcal{P} \), all atoms are unknown.

6 Safe Inductions and Autoepistemic Logic

Recently Bogaerts et al. [2016] exposed a problem in several semantics of autoepistemic logic (AEL). They showed that for very simple, stratified theories, the well-founded and other semantics fail to identify the intended model. They solved this problem by defining, algebraically, a new constructive semantics that is based on a refined notion of approximations of a lattice point (more refined than intervals, i.e., elements of \( L^2 \)). In this section, we show that safe inductions provide a direct solution to the aforementioned problem without the need for any approximation. First, we recall some background on AEL.
6.1 AFT and Autoepistemic Logic

AEL is a non-monotonic logic for modeling the beliefs or knowledge of a rational agent with perfect introspection capabilities [Moore, 1985].

Let $\mathcal{L}$ be the language of propositional logic based on a set of atoms $\Sigma$. Extending this language with a modal operator $K_r$, which is read "I (the agent) know", yields a language $\mathcal{L}_K$ of modal propositional logic. An autoepistemic theory is a set of formulas in $\mathcal{L}_K$. A crucial assumption about such theories that distinguishes this logic from the standard modal logic S5 is that all of the agent’s knowledge is encoded in the theory: it either belongs to the theory, or can be derived from it. Levesque [1990] called this the “all I know assumption”.

A modal formula is a formula of the form $K \psi$; an objective formula is a formula without modal subformulas. If $\phi$ is a formula, $At(\phi)$ denotes the set of all atoms that occur in $\phi$ and $At(\phi)$ the set of all atoms that occur objectively in $\phi$, i.e., outside of the scope of an operator $K_r$.

An interpretation is a subset of $\Sigma$. A possible world structure is a set of interpretations. A possible world structure can be seen as a Kripke structure in which the accessibility relation is total. The set of all possible world structures is denoted $\mathcal{W}_\Sigma$; it forms a lattice with the knowledge order $\leq_k$ such that $Q \leq_k Q'$ if $Q \supseteq Q'$. A possible world structure $Q$ is a mathematical object to represent all situations that are possible according to the agent: interpretations $q \in Q$ represent possible states of affairs, i.e., states of affairs consistent with the agent’s knowledge, and interpretations $\neg q \notin Q$ represent impossible states of affairs, i.e., states of affairs that violate the agent’s knowledge.

If $\phi$ is a formula in $\mathcal{L}_K$, $Q$ is a possible world structure and $I$ is an interpretation, satisfaction of $\phi$ with respect to $Q$ and $I$ (denoted $Q, I \models \phi$) is defined as in the modal logic S5 by the standard recursive rules of propositional satisfaction augmented with one additional rule:

$$Q, I \models K \psi \text{ if } Q, I' \models \phi \text{ for every } I' \subseteq Q.$$  

In this formula, $Q$ represents the belief of the agent and $I$ represents the actual state of the world. Modal formulas are evaluated with respect to the agent’s belief, while objective formulas are evaluated with respect to the state of the actual world. We furthermore define $Q \models K \phi$ ($\phi$ is known in $Q$) if $Q, I \models \phi$ for every $I \subseteq Q$. Moore [1985] associated with every theory $T$ an operator $D_T$ on $\mathcal{W}_\Sigma$ as follows:

$$D_T(Q) = \{ I \in \mathcal{W}_\Sigma \mid Q, I \models T \}.$$  

The intuition behind this operator is that $D_T(Q)$ is a revision of $Q$ consisting of all worlds that are consistent with the agent’s current beliefs ($Q$) and the constraints in $T$.

DMT [2003] defined approximators for $D_T$ and showed that AFT induces all main and some new semantics for AFT.

**Monotonically Stratified AEL Theories** Following Vennekens et al. [2006], we call an autoepistemic theory $T$ stratifiable$^2$ w.r.t. a partition $(\Sigma_i)_{0 \leq i \leq n}$ of its alphabet if there exists a partition $(T_i)_{0 \leq i \leq n}$ of $T$ such that for each $i$, $At_0(T_i) \subseteq \Sigma_i$ and $At(T_i) \subseteq \bigcup_{0 \leq j \leq i} \Sigma_j$. This notion of stratification significantly extends the notion from Marek and Truszczyński [1991]. A stratification is modally separated if for every modal subformula $K \psi$ of $T_i$, either $At(\psi) \subseteq \Sigma_i$ or $At(\psi) \subseteq \bigcup_{0 \leq j \leq i} \Sigma_j$.

Let $\Sigma_1$ and $\Sigma_2$ be two disjoint vocabularies. If $Q_1$ and $Q_2$ are possible world structures over $\Sigma_1$ and $\Sigma_2$ respectively, then the extension of $Q_1$ by $Q_2$ is the possible world structure over $\Sigma_1 \cup \Sigma_2$ defined as $Q_1 \oplus Q_2 = \{ I_1 \cup I_2 \mid I_1 \in Q_1 \land I_2 \in Q_2 \}$. If $Q$ is a possible world structure over $\Sigma_1 \cup \Sigma_2$, the restriction of $Q$ to $\Sigma_1$ is $Q|_{\Sigma_1} = \{ I \cap \Sigma_1 \mid I \in Q \}$.

DMT [2011] have made strong arguments in favor of a constructive semantics for AEL. Bogaerts et al. [2016], however, showed that the two constructive semantics induced by AFT (well-founded and Kripke-Kleene semantics) are too weak for AEL. They gave the following example.

**Example 6.1.** Consider the autoepistemic theory

$$T = \{ q \iff \neg K p, r \iff \neg K q \}.$$ 

The informal reading of this theory is as follows: I (an introspective autoepistemic agent) only know the following: $q$ holds iff I do not know $p$ and $r$ holds iff I do not know $q$.

Since $p$ does not occur objectively in $T$, an agent who only knows $T$ does not have any information about $p$. Thus, in the intended model, it knows neither $p$ nor $\neg p$, i.e., $\neg K p$ and $\neg K \neg p$ must hold in the intended model. The first sentence then entails $q$, hence $K q$ must hold. Now, the last sentence implies $\neg r$; the intended model is thus $\{ \{ p, q \}, \{ q \} \}$, the unique possible world structure in which $\neg K p$, $\neg K \neg p$, and $K \neg r$ hold.

Bogaerts et al. [2016] showed that the well-founded semantics (for any approximator) fails to identify the intended model in the above example. They generalized this example to the class of monotonically stratified theories and defined a notion of perfect model for them.

**Definition 6.2.** We say that $T$ is monotonically stratified with respect to a partition $(\Sigma_i)_{0 \leq i \leq n}$ of its alphabet if there is a monotonically separated stratification $(T_i)_{0 \leq i \leq n}$ of $T$ such that all subformulas $K \psi$ of $T_i$ with $At(\psi) \subseteq \Sigma_i$ occur negatively (in the scope of an odd number of negations) in $T_i$.

The construction of the perfect model of an autoepistemic theory is as follows. In a monotonically stratified theory, each theory $T_i$ defines knowledge of the symbols in $\Sigma_i$ in terms of knowledge of symbols in lower strata $\Sigma_j$ with $j < i$. The last condition guarantees that for a fixed interpretation of the knowledge of lower strata, $D_T$ is a monotone operator and hence its intended fixpoint is clear. The perfect model of $T$ is then constructed by iterated monotone inductions, each of them computing the knowledge of symbols in $\Sigma_i$ based on the knowledge of symbols in lower strata. In the example above, first ignorance of $p$ is established; next, knowledge of $q$ is established and in the final stage, knowledge of $\neg r$ is concluded. This construction was formalized as follows.

**Proposition 6.3** (Proposition 3.3 from Bogaerts et al. [2016]). Let $(T_i)_{0 \leq i \leq n}$ be a monotonically stratification of $T$ w.r.t.
**Theorem 6.9.** The safely defined semantics respects stratification. That is: for each monotonically stratified theory $T$: if $T$ is weakly permaconsistent, then $D_T$ is complete and safe($D_T$) is the perfect model of $T$.

The proof of this theorem makes use of the following two results.

**Lemma 6.10.** Suppose $(T_i)_{0 \leq i \leq n}$ is a monotonic stratification of $T$ w.r.t. $(\Sigma_i)_{0 \leq i \leq n}$. Let $\Sigma'_i$ denote $\bigcup_{j \leq i} \Sigma_j$ for each $i$. For every possible world structure $Q$ it holds that

$$D_T(Q) = \bigoplus_{0 \leq i \leq n} D_{T_i}(Q|_{\Sigma'_i})|_{\Sigma'_i}.$$

**Lemma 6.11.** Suppose $T$ is monotonically stratified w.r.t. $(\Sigma_i)_{0 \leq i \leq n}$. Furthermore suppose $T$ is weakly permaconsistent. Let $\Sigma'_i$ denote $\bigcup_{j \leq i} \Sigma_j$ for each $i$. If $Q_1$ and $Q_2$ are two possible world structures such that $Q_1|_{\Sigma'_i} = Q_2|_{\Sigma'_i}$, then also $D_T(Q_1)|_{\Sigma'_i} = D_T(Q_2)|_{\Sigma'_i}$.

Lemma 6.10 shows how $D_T$ is composed from the various $D_{T_i}$. Lemma 6.11 states that if two possible world structures agree on the lower strata, then so does their image under $D_T$ for any weakly permaconsistent theory $T$. In other words: the knowledge of symbols in a given stratum in $D_T(Q)$ only depends on the knowledge of symbols of smaller (or equal) strata in $Q$.

**Sketch of the proof of Theorem 6.9.** The central idea in this proof is to turn the construction of the perfect model into a safe $D_T$-induction. To show that it is a $D_T$-induction and to show that it is safe, we repeatedly exploit the fact that if $Q$ agrees with $pm(T)$ on all strata below $i$, so does $D_T(Q)$. Completeness follows from the fact that we find a safe $D_T$ induction whose limit is the perfect model of $T$. Since this model is a fixpoint of $D_T$, this induction is terminal and $D_T$ is indeed complete.

**7 Conclusion**

In this paper, we presented the notions of $O$-inductions and safe $O$-inductions for a lattice operator $O$. We studied how they relate to various fixpoints of $O$ studied in AFT. We studied the semantics induced by these inductions in the context of autoepistemic logic, where we find that the safely defined point has interesting properties for a class of operators. It is a topic of future work to study the semantics induced by safe inductions for other application domains of AFT, such as abstract argumentation [Dung, 1995] and active integrity constraints [Flesca et al., 2004; Cruz-Filipe, 2016], where we conjecture that safe inductions will prove helpful to tackle the problems with the well-founded semantics such as Example 18 of Cruz-Filipe [2016].
References


