Eliminating Unintended Stable Fixpoints in Approximation Fixpoint Theory

Spencer Killen[1,*], Jia-Huai You[1]

[1]University of Alberta, 11011 - 88 Avenue, Edmonton, AB, Canada, T6G 2G5

Abstract

A wide variety of nonmonotonic semantics can be expressed as approximators defined under AFT (Approximation Fixpoint Theory). Using traditional AFT theory, it is difficult to derive falsity from falsity. However, this type of reasoning is essential for systems that incorporate classical negation into nonmonotonic reasoning. In this work, we introduce a methodology that can better leverage falsity in stable revision to capture a more precise semantics. We show that our framework fits within the theoretical confines of AFT and can be utilized without modifying the existing theory.

Keywords

approximation fixpoint theory, answer set programming, bilattices

1. Introduction

At the heart of approximation fixpoint theory (AFT) [1], a stable operator enhances an approximator with the capacity to rule out cyclically justified inferences. Mechanically, this is achieved by eliminating undesired fixpoints from the approximator. The remaining fixpoints, called *stable fixpoints*, characterize a semantics. Semantics that have been treated by AFT include the stable, partial stable, and well-founded semantics of many nonmonotonic reasoning systems (e.g. [2, 3]). Sometimes, the intended semantics is only a subset of the captured stable fixpoints, thus, an AFT characterization of a semantics can be improved by removing additional unintended stable fixpoints. In this work, we enhance stable revision and provide a general framework that fits within the existing theory of AFT.

Currently, stable revision is limited in the following way: The approximator embedded in the stable operator does not have access to the atoms that were false when the stable operator was invoked. This information is required by semantics that infer atoms to be false based on the falsity of other atoms. As a result, stable revision applied to such semantics will have additional unintended stable fixpoints. We concretely demonstrate this point in the following logic program example.

Example 1. *For logic programming, stable revision is an iterative process that first fixes negation, then assigns all atoms to be false, and then computes the minimal set of true atoms. Here, our goal is to capture a program's answer sets as stable fixpoints. Let $\mathcal{P}$ be the program containing the following rules.*

$$\leftarrow a \qquad\qquad b \leftarrow \boldsymbol{not}\ a \qquad\qquad a \leftarrow \boldsymbol{not}\ b$$

Informally, an answer set is a minimal set of atoms such that if a rule's body (the right side) is satisfied, then the rule's head (the left side) is contained in the answer set. Since its head is empty, the constraint $\leftarrow a$ requires that no answer set contains a, and the remaining rules work to set up a "choice" between assigning a or b to be true. This program has one answer set $(\{b\}, \{b\})$, which assigns b to be true and a to be false.

A forward-reasoning approximator, such as the one given by Denecker et al., will ignore the constraint and propagate that both a and b are possibly true (but not necessarily true)[1]. That is, the interpretation

23rd International Workshop on Nonmonotonic Reasoning, November 11-13, 2025, Melbourne, Australia

*Corresponding author.
sjkillen@ualberta.ca (S. Killen); jyou@cs.ualberta.ca (J. You)
https://sjkillen.ca (S. Killen)
0000-0003-3930-5525 (S. Killen); 0000-0001-9372-4371 (J. You)

[1]Note that Denecker et al.'s approximator was not designed with constraints in mind.

$(\emptyset, \{a, b\})$ that assigns both a and b to be unknown, is a stable fixpoint. This overly cautious interpretation approximates both $(\{b\}, \{b\})$ and $(\{a\}, \{a\})$. Because $(\{a\}, \{a\})$ is not an answer set, the approximator need not consider it when answer sets are the intended semantics. If we wish to remove the unintended stable fixpoint $(\emptyset, \{a, b\})$, our approximator must leverage backward-reasoning to conclude that a is false.

Deceptively so, it appears like we can accomplish this with a simple modification: our approximator must not derive atoms that would satisfy the body of a constraint. However, this approach quickly falls apart when we add some layers of indirection. For example, let us change the constraint to use a new atom c and add an additional rule $c \leftarrow a$.

$$\leftarrow c \qquad\qquad b \leftarrow \textbf{\textit{not}}\ a \qquad\qquad c \leftarrow a \qquad\qquad a \leftarrow \textbf{\textit{not}}\ b$$

*This program is similar to $\mathcal{P}$ in that $(\{b\}, \{b\})$ is the only answer set, however, multiple steps of reasoning are required to conclude that a must be false. A reasoning step must conclude that c is false (from $\leftarrow c$) for a subsequent reasoning step to conclude that a is false (from $c \leftarrow a$). Constructing the appropriate approximator is challenging because there is no mechanism to access which atoms were false at the invocation of stable revision. If we invoke the stable operator with an interpretation (T, P), the complement of P contains the atoms that are false. Yet, the computation of **lfp** $o(T, \cdot)_2$, which determines the atoms that are false, has no access to this information.*

When stable revision is used for iterative construction, each iteration more atoms with an unknown truth value are assigned a value of either true or false. However, in each iteration, the underlying approximator "forgets" which atoms it assigned false. Thus, from the approximator's perspective, any atom may become true. In actuality, atoms that are assigned false will remain false on subsequent iterations due to monotonicity.

The primary contribution of this work is a formulation of stable revision that can incorporate false information in its reasoning to conclude that more atoms are false. Surprisingly, our extension does not require a new theory. We simply modify the underlying bilattice on which the approximators operate. As a result, the theory can be easily adopted to extend existing approximators. Our framework can help eliminate unintended stable fixpoints for a semantics if removing these fixpoints requires the propagation of false information.

We are motivated by our approximator defined for hybrid MKNF knowledge bases [4]. In this work, we formalize a general theory for the approximator defined there. For simplicity, and as a testament to the generality of our theory, we limit our focus to non-disjunctive logic programs with constraints. They serve well as a proxy for hybrid systems with falsity propagation. A hybrid MKNF knowledge base [5] pairs an ontology $\mathcal{O}$ with a logic program $\mathcal{P}$. The ontology is a decidable fragment of first-order logic, thus $\mathcal{O} = \neg a$ is semantically similar to a rule $\leftarrow a$ in that it rules out all models with a to be true. With the additional reasoning power, our approximator expanded the class of known knowledge bases with polynomial-computable well-founded models [4].

We organize the paper as follows. Section 2 details lattice theory and the notation adopted throughout this work, Section 2.1 covers approximators and stable revision as used in this work, and Section 2.2 introduces logic programming. In Section 3, we begin by providing an illustrative example and a sketch of a solution. The next subsection expands upon these ideas by formalizing *recurrent approximators*, approximators defined over a *tetralattice*, a bilattice formed from a bilattice. These operators operate on 4-tuples (pairs of pairs) and provide a theoretical backing on AFT with an extra parameter of pairs. In Section 4 we offer an additional use for recurrent approximators by describing how they can be used to make approximators increasing/decreasing. Finally, we wrap up and discuss in Section 5.

2. Preliminaries

We summarize common theory of lattices [6] to establish the notation used throughout this work. A *poset* $\langle S, \preceq_\alpha \rangle$ is a relation $\preceq_\alpha$ over a set of elements S that satisfies: *reflexivity* ($\forall x \in S, x \preceq_\alpha x$), *transitivity* ($\forall x, y, z \in S$, *having both* $x \preceq_\alpha y$ and $y \preceq_\alpha z$ implies $x \preceq_\alpha z$), and *antisymmetry* ($\forall x, y \in S,$ if $x \preceq_\alpha y$

and $y \preceq_\alpha x$ then $x = y$). We refer to a poset $\langle S, \preceq_\alpha \rangle$ simply by S when $\preceq_\alpha$ is clear from context. Given a poset S, we call an element $x \in S$ an *upper bound* (resp. a *lower bound*) of a subset $Q \subseteq S$ if $\forall y \in Q, y \preceq x$ (resp. $\forall y \in Q, x \preceq y$). An upper bound of Q w.r.t. a poset $\langle S, \preceq_\alpha \rangle$ is a *least upper bound*, denoted $lub(Q)$ (resp. *greatest lower bound*, denoted as $glb(Q)$) if it is a lower bound of the set of all upper bounds of Q (resp. an upper bound of the set of all lower bounds of Q). A poset $\langle \mathcal{L}, \preceq_\alpha \rangle$ is a *complete lattice* if every subset $S \subseteq \mathcal{L}$ has a least upper bound and a greatest lower bound. For a complete lattice $\langle \mathcal{L}, \preceq_\alpha \rangle$ we denote $glb(\mathcal{L})$ as $\bot_{\preceq_\alpha}$ and $lub(\mathcal{L})$ as $\top_{\preceq_\alpha}$ when $\mathcal{L}$ is clear from context or simply as $\bot$ and $\top$ when the relation is unambiguous. An *operator* over a complete lattice $\langle \mathcal{L}, \preceq_\alpha \rangle$ is a function $o(x) : \mathcal{L} \to \mathcal{L}$. The operator is $\preceq_\alpha$-monotone (resp. $\preceq_\alpha$-antitone) if $\forall x, y \in \mathcal{L}$ whenever $x \preceq_\alpha y$ we also have $o(x) \preceq_\alpha o(y)$ (resp. $o(y) \preceq_\alpha o(x)$). An operator is $\preceq_\alpha$-monotone increasing (resp. decreasing) if $\forall x, x \preceq_\alpha o(x)$ (resp. $\forall x, o(x) \preceq_\alpha x$). An element of a complete lattice $x \in \mathcal{L}$ is a fixpoint of an operator o if $o(x) = x$. The set of all fixpoints of a $\preceq_{\mathcal{L}}$-monotone operator o on a lattice $\langle \mathcal{L}, \preceq_{\mathcal{L}} \rangle$ forms a complete lattice [7]. We call the greatest lower bound of this lattice *the least fixpoint* and denote it as $\mathbf{lfp}_{\preceq_{\mathcal{L}}} o$. This element can be constructed by iteratively applying o to $\bot_{\preceq_{\mathcal{L}}}$. We denote the cartesian product of two sets S and D with $S \times D$ or S^2 if $S = D$. , that is,

$$\begin{aligned} S \times D &\coloneqq \{(s, d) \mid s \in S, d \in D\} \\ S^2 &\coloneqq \{(s, a) \mid s \in S, a \in S\} \end{aligned}$$

Given a lattice $\langle \mathcal{L}, \preceq_{\mathcal{L}} \rangle$, its induced bilattice [1] consists of the two complete lattices $\langle \mathcal{L}^2, \preceq_p^2 \rangle$ and $\langle \mathcal{L}^2, \preceq_t^2 \rangle$. These are the lattices formed from the two orderings $\preceq_p^2$ and $\preceq_t^2$ such that for each $x, y, z, w \in \mathcal{L}$

- $(x, y) \preceq_p^2 (z, w)$ iff $x \preceq_{\mathcal{L}} z$ and $w \preceq_{\mathcal{L}} y$ (the precision-ordering)
- $(x, y) \preceq_t^2 (z, w)$ iff $x \preceq_{\mathcal{L}} z$ and $y \preceq_{\mathcal{L}} w$ (the truth-ordering)

We denote the powerset of a set S, as $\wp(S)$. We use subscript notation to denote the projection of particular components of a tuple, for example, given an operator $o(T, P) : \mathcal{L}^2 \to \mathcal{L}^2$, we have $o(T, P) = (o(T, P)_1, o(T, P)_2)$ and $o(T, P)_{2,1} = (o(T, P)_2, o(T, P)_1)$. We create partial functions by using a "$\cdot$" in place of arguments to be filled in, that is, for an operator $o(T, P) : \mathcal{L}^2 \to \mathcal{L}^2$, we write $o(\cdot, P)$ (resp. $o(T, \cdot)$) to mean $\lambda x.\ o(x, P)$ (resp. $\lambda x.\ o(T, x)$). Naturally, if a "$\cdot$" is used within a function application that is then projected, the projections are included within the body of the lambda abstraction For example,

$$f(x, \cdot)_1 = \lambda y.\ (f(x, y)_1) \qquad\qquad (\textit{where } f(x, y) : \mathcal{L}^2 \to \mathcal{L}^2)$$

This makes it possible to write $\mathbf{lfp} \preceq_{\mathcal{L}} o(T, \cdot)_1$.

For convenience and ergonomics, we may write 4-tuples as a pair of 2-tuples or as a tuple with four members. As a general rule, we consider two tuples to be equivalent if they are equal when all nested tuples are "flattened". For example, the following equivalences hold.

$$\begin{aligned} \mathcal{L}^4 = \mathcal{L}^2 \times \mathcal{L}^2 &= (\mathcal{L}^2)^2 \\ (T, F, U, P) &= ((T, F), (U, P)) \\ f((T, F), (U, P)) &= f(T, F, U, P) \end{aligned}$$

2.1. Approximation Fixpoint Theory

Approximation fixpoint theory (AFT) was first defined by Denecker et al. [1] to provide a uniform fixpoint characterization of a multitude of nonmonotonic semantics. This work adopts the generalized framework of AFT described by Liu and You [8]. This theory can be viewed as a relaxation of AFT. Rather than ensuring that stable fixpoints are always consistent, inconsistent stable fixpoints are permitted. One can provide an additional condition to deterimine whether a stable fixpoint is intended (e.g. it must be consistent).

We introduce the definitions of approximators and stable revision.

Definition 1. *An* approximator *is a $\preceq_p^2$-monotone operator $o(T,P) : \mathcal{L}^2 \to \mathcal{L}^2$ on the complete lattice $\langle \mathcal{L}^2, \preceq_p^2 \rangle$*

Traditional AFT [1] requires some additional properties on approximators, including exactness, symmetry, or consistency. Following Liu and You [8], we do not impose any of these requirements.

Definition 2. *Given an approximator $o : \mathcal{L}^2 \to \mathcal{L}^2$, we can construct its* stable revision operator $S(o)$.

$$S : \left(\mathcal{L}^2 \to \mathcal{L}^2\right) \to \mathcal{L}^2 \to \mathcal{L}^2$$
$$S(o)(T,P) \coloneqq (\mathbf{lfp}_{\preceq_\mathcal{L}}(o(\cdot,P)_1), \mathbf{lfp}_{\preceq_\mathcal{L}}(o(T,\cdot)_2))$$

For an approximator o, we refer to fixpoints of $S(o)$ as *stable fixpoints.* Since the operator o is $\preceq_p^2$-monotone, it is easy to check that both operators $o(\cdot,P)_1$ and $o(T,\cdot)_2$ are $\preceq_\mathcal{L}$-monotone, so stable revision is well-defined.

To apply AFT to a semantics is to characterize the intended models as stable fixpoints. If there are stable fixpoints that do not correspond to the intended semantics, we say they are unintended stable fixpoints. Such fixpoints are unavoidable in hybrid reasoning systems[8]. They may also appear due to an inability to capture intended semantics (e.g. the intended semantics is super-polynomial but a polynomial approximator is desired).

It is desirable to have fewer unintended stable fixpoints. Ultimate approximation theory [9] embodies this ideal by ordering approximators by their ability to capture the intended stable fixpoints. However, climbing this ordering may introduce additional unintended exact stable fixpoints.

2.2. Logic Programming

A *logic program* is a set of rules formed by if/then constraints. Most logic programming semantics seek a truth-minimal assignment to atoms appearing in the program such that every rule is satisfied. Here, we introduce Przymusinski's [10] extension of answer set semantics, a language which only allows a single atom in the head of each rule and the bodies of rules can leverage negation as failure. Additionally, we allow for integrity constraints.

For a logic program $\mathcal{P}$, each rule $r \in \mathcal{P}$ consists of a head, a set containing at most one atom, denoted $hd(r)$, and a set of possibly negated atoms called its body, denoted $body(r)$. A rule r with the parts $hd(r) = \{h\}$ and $body(r) = \{p_1, \ldots, p_k, \boldsymbol{not}\, b_1, \ldots, \boldsymbol{not}\, b_i\}$ is written as $h \leftarrow p_1, \ldots, p_k, \boldsymbol{not}\, b_1, \ldots, \boldsymbol{not}\, b_i$. Rules with empty heads are called *integrity constraints* and are written as above with h omitted. For a program, $\mathcal{P}$, we use $constraints(\mathcal{P})$ to capture the set of such rules, that is, $constraints(\mathcal{P}) = \{r \mid r \in \mathcal{P}, hd(r) = \emptyset\}$. For a rule r, we also define $body^+(r) \coloneqq \{p_1, \ldots, p_k\}$ and $body^-(r) \coloneqq \{b_1, \ldots, b_i\}$ to extract the positive body and the negative body of the rule. Note that the elements in $body^-(r)$ are atoms rather than negated atoms. Without loss of generality, we do not consider logic programs with variables.

An interpretation of a logic program $\mathcal{P}$ is a true/false/unknown assignment to every atom that appears in $\mathcal{P}$. If the assignment does not use the value unknown, then it is *two-valued* and represented as the set of true atoms. Otherwise, the interpretation is *three-valued* and represented by a pair of sets (T,P) such that $T \subseteq P$, the set T contains the atoms that are true, and P contains the atoms that are not false. Every two-valued interpretation T has an equivalent three-valued interpretation (T,T). The valuation $(T,P)(a)$ for an atom a w.r.t. an interpretation (T,P) is given as follows.

$$(T,P)(a) \coloneqq \begin{cases} \boldsymbol{t}\ (true) & \text{iff } a \in T \cap P \\ \boldsymbol{f}\ (false) & \text{iff } a \notin T \cup P \\ \boldsymbol{u}\ (unknown) & \text{iff } a \notin T, a \in P \end{cases}$$

The orderings $\preceq_t^2$ and $\preceq_p^2$ defined earlier are extended to interpretations using the underlying ordering $\subseteq$. These order the interpretations based on the individual valuations of each atom [11]. That is, with $Atoms(\mathcal{P})$ denoting the set containing all atoms appearing in a logic program $\mathcal{P}$,

$$(T,P) \preceq_t^2 (T',P') \text{ iff } \forall a \in Atoms(\mathcal{P}), (T,P)(a) \preceq_t (T',P')(a)$$

$$(T, P) \preceq_p^2 (T', P') \text{ iff } \forall a \in Atoms(\mathcal{P}), (T, P)(a) \preceq_p (T', P')(a)$$

where $\boldsymbol{f} \preceq_t \boldsymbol{u} \preceq_t \boldsymbol{t}$, $\boldsymbol{u} \preceq_p \boldsymbol{t}$, and $\boldsymbol{u} \preceq_p \boldsymbol{f}$. The ordering $\preceq_t^2$ is used to minimize truth while $\preceq_p^2$ measures the proximity to a two-valued interpretation. Treated as an interval, an interpretation (T, P) contains all two-valued interpretations related by $\preceq_p^2$. That is,

$$T \subseteq X \subseteq P \iff (T, P) \preceq_p^2 (X, X)$$

We use the following to relate rules and interpretations by describing the set of rules whose bodies are satisfied by an interpretation.

$$bodysat_{(T,P)}(\mathcal{P}) \coloneqq \{r \in \mathcal{P} \mid body^+(r) \subseteq T, body^-(r) \cap P = \emptyset\}$$

We use $hd(R)$ to denote $\{hd(r) \mid r \in R\}$ for a set of rules R.

Przymusinski [10] extends Gelfond and Lifschitz's [12] answer set semantics to three-valued interpretations. We adopt our own definition due to its close proximity to fixpoint operators [13].

Definition 3. *We call an interpretation (T, P) a* model *of a logic program $\mathcal{P}$ if*

$$(hd(bodysat_{(T,P)}(\mathcal{P})), hd(bodysat_{(P,T)}(\mathcal{P}))) \preceq_t^2 (T, P)$$

We say an interpretation (T', P') models the reduct of $\mathcal{P}$ w.r.t. (T, P) *if*

$$(hd(bodysat_{(T',P)}(\mathcal{P})), hd(bodysat_{(P',T)}(\mathcal{P}))) \preceq_t^2 (T', P')$$

A model (T, P) of a logic program $\mathcal{P}$ is a partial stable model of $\mathcal{P}$ if there does not exist $(T', P') \prec_t^2 (T, P)$ s.t. (T', P') models of the reduct of $\mathcal{P}$ w.r.t. (T, P).

Przymusinski [14] shows that partial stable models are faithful to Gelfond and Lifschitz's answer set semantics [15].

3. Recurrent Approximators

During the computation of stable revision, the underling approximator cannot access the information passed to invoke the stable operator. That is, $S(o)(T, P)_2$ cannot be based on P. This makes perfect sense given that stable revision removes circular justifications. If $S(o)(T, P)_2$ were to introduce conclusions based on atoms in P, that would undoubtedly introduce circular reasoning. However, this rationale only holds for positive conclusions. Concluding that an atom should not be in $S(o)(T, P)_2$ based on the atoms that are not in P can be perfectly valid. This type of reasonig is not possible in AFT. In this section, we demonstrate this limitation and then we introduce an extension of AFT to remedy it.

3.1. An Illustrative Sketch

First, return to the example presented in the introduction. We focus on constraints in answer set programming due to their simplicity. A constraint $\leftarrow c$ can alternatively be represented as a rule $d \leftarrow c, \boldsymbol{not}\ d$. Adopting this representation is possible and straightforward, however, it would only make the coming examples more difficult.

We define the following set of rules.

$$\leftarrow a \qquad\qquad b \leftarrow \boldsymbol{not}\ a \qquad\qquad a \leftarrow \boldsymbol{not}\ b$$

This program has two partial stable models: $(\emptyset, \{a, b\})$ and $(\{b\}, \{b\})$. The interpretation $(\{a\}, \{a\})$ is not a partial stable model because it violates the integrity constraint $\leftarrow a$.

Suppose our intended semantics is just the *answer sets* (the two-valued partial stable models). Eliminating $(\emptyset, \{a, b\})$ as a stable fixpoint is a welcome improvement because it cannot be extended to an answer set, but we cannot remove every stable fixpoint that is not an answer set [9].

We introduce Denecker et al.'s approximator for capturing partial stable models of a program $\mathcal{P}$.

$$\Gamma_{\mathcal{P}}(T, P) \coloneqq (hd(bodysat_{(T,P)}(\mathcal{P})), hd(bodysat_{(P,T)}(\mathcal{P})))$$

Recall that we use $hd(bodysat_{(T,P)}(\mathcal{P}))$ to denote $\{hd(r) \mid r \in bodysat_{(T,P)}(\mathcal{P})\}$. We have $\mathbf{lfp}\ \Gamma_{\mathcal{P}}(\emptyset, \cdot)_1 = \emptyset$ and $\mathbf{lfp}\ \Gamma_{\mathcal{P}}(\cdot, \{a, b\})_2 = \{a, b\}$. Thus, $(\emptyset, \{a, b\})$ is a stable fixpoint of $\Gamma_{\mathcal{P}}$. Because this approximator was not constructed to support integrity constraints, both $(\{a\}, \{a\})$ and $(\{b\}, \{b\})$ are also stable fixpoints.

Let us define a modified version that prevents the derivation of atoms that would satisfy constraints.

$$\begin{aligned}
\Gamma'_{\mathcal{P}}(T, P)_1 &\coloneqq \Gamma_{\mathcal{P}}(T, P)_1 \\
\Gamma'_{\mathcal{P}}(T, P)_2 &\coloneqq \Gamma_{\mathcal{P}}(T, P)_2 \setminus block'_{\mathcal{P}}(T, P) \\
block'_{\mathcal{P}}(T, P) &\coloneqq \{a \in Atoms(\mathcal{P}) \mid \exists r' \in constraints(\mathcal{P}), r' \in bodysat_{(T \cup \{a\}, P)})\}
\end{aligned}$$

Recall that in $bodysat_{(T,P)}$ the set T is used to evaluate positive atoms in rules and P to evaluate negated atoms (e.g. $\boldsymbol{not}\ a$). Intuitively, the set $block_{\mathcal{P}}(T, P)$ is the set of atoms that, if true, would satisfy a constraint. For example, with the constraint $\leftarrow a, b, \boldsymbol{not}\ c$ we have $a \in block_{\mathcal{P}}(\{b\}, \{a, b\})$. Thus, if there were another rule $a \leftarrow \boldsymbol{not}\ c$, the atom a would not be derived in $\Gamma_{\mathcal{P}}(\{b\}, \{a, b\})_2$. Clearly, as an interpretation (T, P) becomes more precise, the set $block_{\mathcal{P}}(T, P)$ grows. Because these atoms are extracted from P, the function $\Gamma'_{\mathcal{P}}$ is $\preceq_p^2$-monotone (our only requirement for it to be an approximator). Note that if an interpretation (T, P) satisfies some constraint in $\mathcal{P}$, then $block_{\mathcal{P}}(T, P) = Atoms(\mathcal{P})$. Thus, inconsistent stable fixpoints, which are not possible in traditional AFT [1], are possible. Using the generalized AFT of Liu and You [8], we can tolerate inconsistent pairs[2].

While we have succeeded in removing the stable fixpoint $(\emptyset, \{a, b\})$ for the small program, our approximator $\Gamma'_{\mathcal{P}}(T, P)$ does not hold up for programs where falsity propagation must be performed over multiple rules. Take the following program to be $\mathcal{P}$, which modifies the constraint in the other program to use a new atom c and adds an additional rule $c \leftarrow a$.

$$\leftarrow c \qquad\qquad b \leftarrow \boldsymbol{not}\ a \qquad\qquad c \leftarrow a \qquad\qquad a \leftarrow \boldsymbol{not}\ b$$

Because the third rule is not satisfied by $(\emptyset, \{a, b\})$, it is not a model of the program (and thus not a partial stable model). However, it is a stable fixpoint of both Γ and Γ'. While our modified approximator Γ' does block the derivation of c, it only blocks constraints, thus, it does not block a.

Let us modify Γ' so that all rules, and not just constraints, are a part of the backward-chaining.

$$\begin{aligned}
\Omega_{\mathcal{P}}(T, P)_1 &\coloneqq \Gamma_{\mathcal{P}}(T, P)_1 \\
\Omega_{\mathcal{P}}(T, P)_2 &\coloneqq \Gamma_{\mathcal{P}}(T, P)_2 \setminus block^{\Omega}_{\mathcal{P}}(T, P) \\
block^{\Omega}_{\mathcal{P}}(T, P) &\coloneqq \{a \in Atoms(\mathcal{P}) \mid \exists r' \in \mathcal{P}, hd(r') \cap P = \emptyset, r' \in bodysat_{(T \cup \{a\}, P)})\}
\end{aligned}$$

Intuitively, $\Omega_{\mathcal{P}}$ functions identically to Γ', except it also blocks atoms in the bodies of rules whose heads are false (they do not appear in P). As expected, $block_{\mathcal{P}}(\{b\}, \{a, b\}) = \{c, a\}$ and $\Omega_{\mathcal{P}}(\{b\}, \{a, b\}) = (\{b\}, \{b\})$. This is, given that c is false, we conclude that a must also be false. The interpretation $(\{b\}, \{b\})$ is a fixpoint of $\Omega_{\mathcal{P}}$ (a requirement for it to be a stable fixpoint of $\Omega_{\mathcal{P}}$). We show below that $\Omega_{\mathcal{P}}$ can be used to construct the partial stable model $(\{b\}, \{b\})$.

$block_{\mathcal{P}}(\emptyset, Atoms(\mathcal{P})) = \{c\}$	$\Omega_{\mathcal{P}}(\emptyset, Atoms(\mathcal{P})) = (\emptyset, \{a, b\})$	(No rule to derive c)
$block_{\mathcal{P}}(\emptyset, \{a, b\}) = \{c, a\}$	$\Omega_{\mathcal{P}}(\emptyset, \{a, b\}) = (\emptyset, \{b\})$	(a is blocked)
$block_{\mathcal{P}}(\emptyset, \{b\}) = \{c, a\}$	$\Omega_{\mathcal{P}}(\emptyset, \{b\}) = (\{b\}, \{b\})$	(With a false, b can be true)

While approximators can compute some partial stable models, stable revision is required to eliminate cyclic justifications. However, the stable operator of $\Omega_{\mathcal{P}}$ does not behave as expected.

[2]In practice, this requires pairing stable fixpoint checking with an additional condition to verify an intended model. Here, we can verify stable fixpoints as partial stable models by checking that they are consistent.

To compute the least stable fixpoint of $\Omega_{\mathcal{P}}$, we begin with $(\emptyset, Atoms(\mathcal{P}))$, the interpretation that assigns all atoms to be unknown.

$$\begin{aligned}
S(\Omega_{\mathcal{P}})(\emptyset, Atoms(\mathcal{P})) &= (\mathbf{lfp}\ \Omega_{\mathcal{P}}(\cdot, Atoms(\mathcal{P})), \mathbf{lfp}\ \Omega_{\mathcal{P}}(\emptyset, \cdot)) \\
\Omega_{\mathcal{P}}(\emptyset, Atoms(\mathcal{P}))_1 &= \emptyset && \text{(fixpoint)} \\
\Omega_{\mathcal{P}}(\emptyset, \emptyset)_2 &= \Gamma_{\mathcal{P}}(\emptyset, \emptyset)_2 \setminus block^{\Omega}_{\mathcal{P}}(\emptyset, \emptyset) \\
\Gamma_{\mathcal{P}}(\emptyset, \emptyset)_2 &= \{a, b\} \\
block^{\Omega}_{\mathcal{P}}(\emptyset, \emptyset) &= \{a, b\} \\
S(\Omega_{\mathcal{P}})(\emptyset, \emptyset)_2 &= \{a, b\} \setminus \{a, b\} = \emptyset && \text{(fixpoint)}
\end{aligned}$$

Thus, the least stable fixpoint is $(\emptyset, \emptyset)$. Due to $\preceq^2_p$-monotonicity and $(\emptyset, Atoms(\mathcal{P})) \preceq^2_p (\{b\}, \{b\})$, we can conclude that $(\{b\}, \{b\})$ is not a stable fixpoint. This can also be easily worked out by hand. Clearly, this approximator does not capture our intended semantics! If we analyze the approximator, it becomes clear that we cannot trust the atoms that are not in P to be false. In a constructive computation of $S(\Omega_{\mathcal{P}})(T, P)_2$, stable revision first "resets" P to $\emptyset$, thus $block^{\Omega}_{\mathcal{P}}(T, P)$ will treat every atom as false! Note that while P is used in two places in $block^{\Omega}_{\mathcal{P}}$, and we only demonstrated issues with one, both are problematic.

It seems impossible to formulate $block_{\mathcal{P}}$ so that it is powerful enough to perform false backward-chaining. We remedy this issue by introducing a meta operator Φ_P. When used inside an approximator, Φ_P has the same value as P. When used in stable revision, Φ_P has the value of P that was passed to the stable operator.

We modify $\Omega_{\mathcal{P}}$ to formulate a new operator $\Gamma''_{\mathcal{P}}$ by replacing $block^{\Omega}_{\mathcal{P}}$ with the following.

$$block''_{\mathcal{P}}(T, P) \coloneqq \{a \in Atoms(\mathcal{P}) \mid \exists r' \in \mathcal{P}, hd(r') \cap \Phi_P = \emptyset, r' \in bodysat_{(T \cup \{a\}, \Phi_P)})\}$$

Now, when we compute our least stable fixpoint for $\Gamma''_{\mathcal{P}}$, we get our stable model $(\{b\}, \{b\})$.

$$\begin{aligned}
S(\Gamma''_{\mathcal{P}})(\emptyset, Atoms(\mathcal{P})) &= (\mathbf{lfp}\ \Gamma''_{\mathcal{P}}(\cdot, Atoms(\mathcal{P}))_2, \mathbf{lfp}\ \Gamma''_{\mathcal{P}}(\emptyset, \cdot)_2) \\
\Gamma''_{\mathcal{P}}(\emptyset, Atoms(\mathcal{P}))_1 &= \emptyset \\
\Gamma''_{\mathcal{P}}(\emptyset, \emptyset)_2 &= \Gamma_{\mathcal{P}}(\emptyset, \emptyset)_2 \setminus block''_{\mathcal{P}}(\emptyset, \emptyset) \\
\Gamma_{\mathcal{P}}(\emptyset, \emptyset)_2 &= \{a, b\} \\
\Phi_P &= \{a, b, c\} \\
block''_{\mathcal{P}}(\emptyset, \emptyset) &= \{c\} \\
\Gamma''_{\mathcal{P}}(\emptyset, \emptyset)_2 &= \{a, b\} \setminus \{c\} = \{a, b\}
\end{aligned}$$

No longer is $(\emptyset, \emptyset)$ the least stable fixpoint. When we continue the computation of $S(\Gamma''_{\mathcal{P}})(\emptyset, Atoms(\mathcal{P}))$ with $(\emptyset, \{a, b\})$, it is critical that Φ_P remains as $\{a, b, c\}$. With $\Phi_P = \{a, b\}$, that is, with c removed, we have $block''_{\mathcal{P}}(\emptyset, \{a, b\}) = \{c\}$ and $\Gamma''_{\mathcal{P}}(\emptyset, \{a, b\})_2 = \{b\}$. Thus, if we update Φ_P in this way, the function $\Gamma''_{\mathcal{P}}(\emptyset, \cdot)_2$ is not monotone and the stable operator is not well-defined.

After we have computed $S(\Gamma''_{\mathcal{P}})(\emptyset, Atoms(\mathcal{P}))$ to be $(\emptyset, \{a, b\})$ we can apply $S(\Gamma''_{\mathcal{P}})$ again with the more precise interpretation $(\emptyset, \{a, b\})$ to use $\Phi_P = \{a, b\}$.

$$\begin{aligned}
block''_{\mathcal{P}}(\emptyset, \{a, b\}) &= \{c, a\} \\
\Gamma''_{\mathcal{P}}(\emptyset, \{a, b\})_2 &= \{b\} \\
\Gamma''_{\mathcal{P}}(\emptyset, \{b\})_2 &= \{b\} \qquad \text{(fixpoint)}
\end{aligned}$$

While Φ_P seems to have solved the issues with $\Omega_{\mathcal{P}}$, its introduction raises many questions. The introduction of this operator appears to be a significant deviation from AFT. How do we determine if a recurrent approximator is $\preceq^2_p$-monotone? What is the relationship between fixpoints and stable fixpoints? Do stable fixpoints still exist? To address these questions, we show that Φ_P fits within the existing AFT theory. Thus, any recurrent approximator can be expressed as an approximator which does not use Φ_P. Due to this relationship, the properties of AFT continue to hold for recurrent approximators.

3.2. A Formalization of Φ_P

We have shown that making inferences based on false information is limited in stable revision, and we have provided a sketch which extends the stable revision operator with an operator to access false information so that it can be incorporated safely in reasoning. We formalize this operator by establishing a larger complete lattice for approximators to operate on. Intuitively, this lattice is formed by taking the bilattice of the bilattice to form a *tetralattice*. The additional data in each element is used to store Φ_P.

For simplicity and throughout the remainder of this work, we assume that every complete lattice $\langle \mathcal{L}, \preceq_{\mathcal{L}} \rangle$ has a complement operation, denoted as $a^{\mathbf{c}}$, that satisfies the following two properties

$$i.\ \forall a \in \mathcal{L},\ (a^{\mathbf{c}})^{\mathbf{c}} = a \qquad\qquad ii.\ \forall a, b \in \mathcal{L},\ a \preceq_{\mathcal{L}} b \iff b^{\mathbf{c}} \preceq_{\mathcal{L}} a^{\mathbf{c}}$$

Not every lattice has such a complement operation, thus it appears limiting, however, this operation is not necessary to apply our theory. We rely upon it only for simplicity. One can instead define orderings differently so that the criteria of the complement is satisfied.

We intend to isolate a family of approximators defined on a "bilattice formed from a bilattice" that can be used to propagate information from the stable operator to the approximators within. First, we formally describe this lattice.

Definition 4. *Given a complete lattice $\langle \mathcal{L}, \preceq_{\mathcal{L}} \rangle$ we construct its bilattice $\langle \mathcal{L}^2, \preceq_t^2 \rangle$, $\langle \mathcal{L}^2, \preceq_p^2 \rangle$, and then we define the following pair of complete lattices which we refer to collectively and individually as a* tetralattice.

$$\langle \mathcal{L}^4, \preceq_t^4 \rangle, \langle \mathcal{L}^4, \preceq_p^4 \rangle$$

A tetralattice is the bilattice formed from turning $\langle \mathcal{L}^2, \preceq_t^2 \rangle$ into a bilattice. The orderings $\preceq_t^4$ and $\preceq_p^4$ are naturally defined. For the ordering $\preceq_t^4$ and any two 4-tuples $(T, F, U, P), (T', F', U', P') \in \mathcal{L}^4$, the following three expressions are equivalent

- $((T, F), (U, P)) \preceq_t^4 ((T', F'), (U', P'))$,
- $(T, F) \preceq_t^2 (T', F') \wedge (U, P) \preceq_t^2 (U', P')$, *and*
- $T \preceq_{\mathcal{L}} T' \wedge F \preceq_{\mathcal{L}} F' \ \wedge U \preceq_{\mathcal{L}} U' \wedge P \preceq_{\mathcal{L}} P'$

For $\preceq_p^4$, the following are equivalent

- $((T, F), (U, P)) \preceq_p^4 ((T', F'), (U', P'))$,
- $(T, F) \preceq_t^2 (T', F') \wedge (U', P') \preceq_t^2 (U, P)$,
- $(T, P) \preceq_p^2 (T', P') \wedge (F, U) \preceq_p^2 (F', U')$, *and*
- $T \preceq_{\mathcal{L}} T' \wedge F \preceq_{\mathcal{L}} F' \wedge U' \preceq_{\mathcal{L}} U \wedge P' \preceq_{\mathcal{L}} P$

We take the process applied to $\langle L, \preceq_{\mathcal{L}} \rangle$ to obtain $\langle \mathcal{L}^2, \preceq_t^2 \rangle$ and $\langle \mathcal{L}^2, \preceq_p^2 \rangle$, then we apply it to the lattice $\langle \mathcal{L}^2, \preceq_t^2 \rangle$. The result is a pair of complete lattices because bilattices are complete lattices [16]. The element F of a pair will be used to carry false information, that is, the complement of P. We symmetrically define U to carry information that is not true, however, U does not seem as useful as F and our approximators for logic programs do not make use of U[3]. Most of the approximators defined on the full tetralattice lattice are of no interest here. Thus, we restrict our attention to the approximators which have a complementary relationship between P and F (also T and U). We use the notation $tuple_{2,3}$ to project multiple elements from a tuple (e.g. $(a, b, c, d)_{2,3} = (b, c)$).

Definition 5. *A recurrent operator $o(T, F, U, P) : \mathcal{L}^4 \to \mathcal{L}^4$ is an operator on the tetralattice $\mathcal{L}^4$ with the inner components fixed such that the following equivalence is satisfied.*

$$o(T, F, U, P)_{2,3} = \left(P^{\mathbf{c}}, T^{\mathbf{c}}\right)$$

We call $\preceq_p^4$-monotone recurrent operators recurrent approximators.

[3]In Section 4, we make use of U to define meta approximators.

By restricting the class of approximators, we ensure that F is a carrier of false information. We complement the information so that we can adopt the same orderings of AFT.

Note that $o(T,F,U,P)_{2,3}$ is of type $L^4 \to L^2$. To construct a recurrent operator, we only need to define $o(\cdot,\cdot,\cdot,\cdot)_{1,4}$, a traditional approximator that additionally receives an older computation of T and P (in complement form) and returns a new approximation (T', P'). The utility of these additional arguments is not fully apparent until they are embedded in the stable revision operator.

The least element of the lattice $\langle \mathcal{L}^4, \preceq_p^4 \rangle$ is the pair $((\bot_\mathcal{L}, \bot_\mathcal{L}), (\top_\mathcal{L}, \top_\mathcal{L}))$ which is equivalant to $(\bot_{\preceq_t^2}, \top_{\preceq_t^2})$.

Example 2. *All approximators in AFT can easily be converted to a recurrent approximator. Let $o : \mathcal{L}^2 \to \mathcal{L}^2$ be an approximator over $\langle \mathcal{L}^2, \preceq_p^2 \rangle$. We define a recurrent approximator $o^* : \mathcal{L}^4 \to \mathcal{L}^4$*

$$\begin{aligned} o^*(T,F,U,P)_1 &\coloneqq o(T,P)_1 \\ o^*(T,F,U,P)_2 &\coloneqq P^\mathbf{c} \\ o^*(T,F,U,P)_3 &\coloneqq T^\mathbf{c} \\ o^*(T,F,U,P)_4 &\coloneqq o(T,P)_2 \end{aligned}$$

This approximator does not make use of F or U, thus for any $(T,P) \in \mathcal{L}^2$ and $F, U \in \mathcal{L}$

$$o(T,P) = o^*(T,F,U,P)_{1,4}$$

Due to its underlying structure (an approximator formed from a bilattice), a recurrent approximator is required to be $\preceq_p^4$-monotone. Because the inner components of the approximator are fixed, we can simplify the process of checking whether a recurrent approximator is monotone.

Lemma 1. *For a tetralattice $\langle \mathcal{L}^4, \preceq_p^4 \rangle$, a recurrent operator $o(T,F,U,P)$ is $\preceq_p^4$-monotone iff for each $(T,F,U,P), (T',F',U',P') \in L^4$ s.t. $(T,F,U,P) \preceq_p^4 (T',F',U',P')$, we have $o(T,F,U,P)_{1,4} \preceq_p^2 o(T',F',U',P')_{1,4}$*

Proof. ($\Rightarrow$) trivial.

($\Leftarrow$) It's sufficient to show $o(T,F,U,P)_{2,3} \preceq_p^2 o(T',F',U',P')_{2,3}$. We have $o(T,F,U,P)_2 = P^\mathbf{c}$ and $o(T',F',U',P')_2 = P'^\mathbf{c}$. Clearly since $P' \preceq_\mathcal{L} P$, we have $P^\mathbf{c} \preceq_\mathcal{L} P'^\mathbf{c}$. We have $o(T,F,U,P)_3 = T^\mathbf{c}$ and $o(T',F',U',P')_3 = T'^\mathbf{c}$. From $T \preceq_\mathcal{L} T'$, we have $T'^\mathbf{c} \preceq_\mathcal{L} T^\mathbf{c}$. We conclude $(P^\mathbf{c}, T^\mathbf{c}) \preceq_p^2 (P'^\mathbf{c}, T'^\mathbf{c})$. □

When defining a new recurrent approximator, we need only focus on the T/P components in the image when proving monotonicity.

We can now rebuild $\Gamma''_\mathcal{P}$ from Section 3.1 and show that it is $\preceq_p^4$-monotone. First, we need the appropriate lattice for interpretations.

Definition 6. *A powerset tetralattice $\langle \wp(\mathcal{L})^4, \preceq_p^4 \rangle$ is the tetralattice formed from a powerset lattice $\langle \wp(\mathcal{L}), \subseteq \rangle$ using $\alpha^\mathbf{c} = \mathcal{L} \setminus \alpha$ as the complement operation.*

Example 3. *Define $\Gamma'''_\mathcal{P}$ as follows.*

$$\begin{aligned} \Gamma'''_\mathcal{P}(T,F,U,P)_1 &\coloneqq \Gamma_\mathcal{P}(T,P)_1 \\ \Gamma'''_\mathcal{P}(T,F,U,P)_4 &\coloneqq \Gamma_\mathcal{P}(T,P)_2 \setminus \mathit{block}'''_\mathcal{P}(T,F) \\ \mathit{block}'''_\mathcal{P}(T,F) &\coloneqq \{a \in \mathit{Atoms}(\mathcal{P}) \mid \exists r' \in \mathcal{P}, \mathit{hd}(r') \subseteq F, r' \in \mathit{bodysat}_{(T \cup \{a\}, F^\mathbf{c})}\} \end{aligned}$$

As $\Gamma'''_\mathcal{P}$ is a recurrent approximator, $\Gamma'''_\mathcal{P}(T,F,U,P)_{2,3}$ is fixed. As (T,F,U,P) becomes more precise w.r.t. $\preceq_p^4$, the sets T and F grow. With a larger T and F, there is more opportunity for a rule r s.t. $\mathit{hd}(r') \subseteq F$ and $r' \in \mathit{bodysat}_{(T \cup \{a\}, F^\mathbf{c})}$. Thus, $\mathit{block}'''_\mathcal{P}(T,F)$ also grows. It follows that $\Gamma'''_\mathcal{P}(T,F,U,P)_4$ shrinks and then that $\Gamma'''_\mathcal{P}(T,F,U,P)_{1,4}$ is monotone from $\preceq_p^4$ to $\preceq_p^2$. With Lemma 1, we can conclude $\Gamma'''_\mathcal{P}$ is $\preceq_p^4$-monotone.

Now that we've established approximators that can capture Φ_P, it remains to show that stable revision works as expected. Because recurrent approximators are approximators over the lattice $\langle \mathcal{L}^2, \preceq_t^2 \rangle$, stable revision is already defined. For convenience, we repeat the definition of the stable revision operator using the tetralattice.

$$\begin{gathered} S : \left(\mathcal{L}^4 \to \mathcal{L}^4\right) \to \mathcal{L}^4 \to \mathcal{L}^4 \\ S(o)(T,F,U,P) \coloneqq (\mathbf{lfp}_{\preceq_t^2}(o(\cdot,(U,P))_{1,2}), \mathbf{lfp}_{\preceq_t^2}(o((T,F),\cdot)_{3,4})) \end{gathered}$$

The stable operator is itself a recurrent approximator.

Proposition 1. *For a recurrent approximator o, we have that $S(o)$ is $\preceq_p^2$-monotone and $S(o)(T,F,U,P)_{2,3} = (P^{\mathbf{c}}, T^{\mathbf{c}})$.*

Proof. We show (i) that $S(o)$ is a recurrent operator and then (ii) that $S(o)$ is $\preceq_p^4$-monotone. (i) Let $(T,F,U,P) \in \mathcal{L}^4$. The functions $o(\cdot,(U,P))_2$ and $o((T,F),\cdot)_3$ are constant, therefore

$$S(o)(T,F,U,P)_{2,3} = (\mathcal{L} \setminus P, \mathcal{L} \setminus T)$$

(ii) By [7] and [1] the $S(o)$ operator is well-defined, that is, o has fixpoints that exist when o is monotone. Let $(T,F,U,P),(T',F',U',P') \in L^4$ such that $(T,F,U,P) \preceq_p^4 (T',F',U',P')$. It is sufficient to show

(a) $\mathbf{lfp}_{\preceq_t^2}(o(\cdot,(U,P))_{1,2}) \preceq_t^2 \mathbf{lfp}_{\preceq_t^2}(o(\cdot,(U',P'))_{1,2})$
(b) $\mathbf{lfp}_{\preceq_t^2}(o((T',F'),\cdot)_{3,4}) \preceq_t^2 \mathbf{lfp}_{\preceq_t^2}(o((T,F),\cdot)_{3,4})$

(a) Let $x = \mathbf{lfp}_{\preceq_t^2}(o(\cdot,(U',P'))_{1,2})$. By the $\preceq_p^4$-monotonicity of o, we have

$$o(x,(U,P)) \preceq_t^2 o(x,(U',P'))$$

Here, x is a prefixpoint of $o(\cdot,(U,P))$. $\mathbf{lfp}_{\preceq_t^2}(o(\cdot,(U,P))_{1,2})$ corresponds to the least prefixpoint of o [7], thus $\mathbf{lfp}_{\preceq_t^2}(o(\cdot,(U,P))_{1,2}) \preceq_t^2 x$. A nearly identical procedure can be used to show that $\mathbf{lfp}_{\preceq_t^2}(o((T',F'),\cdot)_{3,4}) \preceq_t^2 \mathbf{lfp}_{\preceq_t^2}(o((T,F),\cdot)_{3,4})$. We conclude that $S(o)$ is $\preceq_p^4$ monotone and with (i) it is a recurrent approximator.

$$\begin{gathered} (\mathbf{lfp}_{\preceq_t^2}(o(\cdot,(U,P))_{1,2}), \mathbf{lfp}_{\preceq_t^2}(o((T,F),\cdot)_{3,4})) \preceq_p^4 \\ (\mathbf{lfp}_{\preceq_t^2}(o(\cdot,(U',P'))_{1,2}), \mathbf{lfp}_{\preceq_t^2}(o((T',F'),\cdot)_{3,4})) \end{gathered}$$

Least fixpoints can be computed by repeated application of an operator on the least element. We have the following:

$$\begin{gathered} (\bot_{\preceq_\mathcal{L}}, \bot_{\preceq_\mathcal{L}}, U, P) \preceq_p^4 (\bot_{\preceq_\mathcal{L}}, \bot_{\preceq_\mathcal{L}}, U', P') \\ (T, F, \bot_{\preceq_\mathcal{L}}, \bot_{\preceq_\mathcal{L}}) \preceq_p^4 (T', F', \bot_{\preceq_\mathcal{L}}, \bot_{\preceq_\mathcal{L}}) \end{gathered}$$

and with the $\preceq_p^4$-monotonicity of o, we have

$$\begin{gathered} o(\bot_{\preceq_\mathcal{L}}, \bot_{\preceq_\mathcal{L}}, U, P)_{1,2} \preceq_t^2 o(\bot_{\preceq_\mathcal{L}}, \bot_{\preceq_\mathcal{L}}, U', P')_{1,2} \\ o(T', F', \bot_{\preceq_\mathcal{L}}, \bot_{\preceq_\mathcal{L}})_{2,3} \preceq_t^2 o(T, F, \bot_{\preceq_\mathcal{L}}, \bot_{\preceq_\mathcal{L}})_{2,3} \end{gathered}$$

With the $\preceq_p^4$-monotonicity of o, this relation continues to hold when we reapply the operator inductively:

$$\begin{gathered} o\big(o(\bot_{\preceq_\mathcal{L}}, \bot_{\preceq_\mathcal{L}}, U, P)_{1,2}, U, P\big)_{1,2} \preceq_t^2 o\big(o(\bot_{\preceq_\mathcal{L}}, \bot_{\preceq_\mathcal{L}}, U', P')_{1,2}, U', P'\big)_{1,2} \\ o\big(T', F', o(T', F', \bot_{\preceq_\mathcal{L}}, \bot_{\preceq_\mathcal{L}})_{3,4}\big)_{3,4} \preceq_t^2 o\big(T, F, o(T, F, \bot_{\preceq_\mathcal{L}}, \bot_{\preceq_\mathcal{L}})_{3,4}\big)_{3,4} \end{gathered}$$

We conclude the following

$$\mathbf{lfp}_{\preceq_t^2}(o(\cdot,(U,P))_{1,2}) \preceq_t^2 \mathbf{lfp}_{\preceq_t^2}(o(\cdot,(U',P'))_{1,2})$$
$$\mathbf{lfp}_{\preceq_t^2}(o((T',F'),\cdot)_{3,4}) \preceq_t^2 \mathbf{lfp}_{\preceq_t^2}(o((T,F),\cdot)_{3,4})$$

□

We now return to our example this time to compute the least stable fixpoint.

Example 4. *Let $\mathcal{P}$ be our program from before.*

$$\leftarrow c \qquad\qquad b \leftarrow \textit{\textbf{not}}\ a \qquad\qquad c \leftarrow a \qquad\qquad a \leftarrow \textit{\textbf{not}}\ b$$

Using $\Gamma'''_{\mathcal{P}}$ from Example 3, we begin to compute its least stable fixpoint, beginning with the least element $(\emptyset, \emptyset, Atoms(\mathcal{P}), Atoms(\mathcal{P}))$. Note that for this element $F = P^{\mathbf{c}}$ and $U = T^{\mathbf{c}}$.

$$S(\Gamma'''_{\mathcal{P}})(\emptyset,\emptyset,Atoms(\mathcal{P}),Atoms(\mathcal{P})) = \big((\textit{\textbf{lfp}}\ \Gamma'''_{\mathcal{P}}(\cdot,\cdot,Atoms(\mathcal{P}),Atoms(\mathcal{P})))_{1,2}, \\ (\textit{\textbf{lfp}}\ \Gamma'''_{\mathcal{P}}(\emptyset,\emptyset,\cdot,\cdot)_{3,4}\big)$$

Because the first component is computed by Γ and the second component is $P^{\mathbf{c}}$, we have

$$\textit{\textbf{lfp}}\ \Gamma'''_{\mathcal{P}}(\cdot,\cdot,Atoms(\mathcal{P}),Atoms(\mathcal{P})))_{1,2} = (\emptyset, Atoms(\mathcal{P})^{\mathbf{c}})$$

Now for the other side,

$$\textit{\textbf{lfp}}\ \Gamma'''_{\mathcal{P}}(\emptyset,\emptyset,\cdot,\cdot)_{3,4} = (\emptyset^{\mathbf{c}}, \{a,b\})$$

Because Φ_P is empty (F here), we derive a and b to be unknown as normal. Our resulting four pair is $(\emptyset,\emptyset,Atoms(\mathcal{P}),\{a,b\})$. However, on the next iteration of stable revision, F will be updated. That is,

$$S(\Gamma'''_{\mathcal{P}})((\emptyset,\emptyset,Atoms(\mathcal{P}),\{a,b\}))_2 = \{a,b\}^{\mathbf{c}} = \{c\}$$

The other components remain unchanged. Now that $F = \{c\}$, our approximator can block the derivation of a.

$$S(\Gamma'''_{\mathcal{P}})((\emptyset,\{c\},Atoms(\mathcal{P}),\{a,b\}))_4 = \{b\}$$

After several more iterations, we obtain the least stable fixpoint.

$$S(\Gamma'''_{\mathcal{P}})((\{b\},\{c,a\},\{b\}^{\mathbf{c}},\{b\}))_{1,4} = (\{b\},\{b\})$$

We have defined approximators with enhanced stable revision that can leverage false information computed on previous iterations. Because our developments exist within the confines of AFT, that is, every recurrent approximator is an approximator, it has the properties of an approximator.

4. Increasing/Decreasing Approximators

Despite both utilizing fixpoint operators, AFT semantics and traditional fixpoint semantics are not always compatible. For example, the solver we define for hybrid MKNF knowledge bases [17] performs all of its propagation using a fixpoint operator. In order for the solver to function correctly, this operator o must be increasing, that is, Γ^* can be defined such that the following holds.

$$o(T,P) \coloneqq (\Gamma^*(T,P)_1 \cup T, \Gamma^*(T,P)_2 \cup P)$$

While this holds for approximators when applied to pairs that are postfixpoints, a solver must occasionally select some unknown atoms and assign them a value of true or false. The stable revision operator may “revert” these assignments which is undesirable, thus a stable revision operator is not increasing.

We say that an approximator o is *increasing* (resp. *decreasing*) if every pair (T, P) is a postfixpoint (resp. a prefixpoint), that is, $(T, P) \preceq_p^2 o(T, P)$ (resp. $o(T, P) \preceq_p^2 (T, P)$). Another use for recurrent approximators is a method to turn any approximator into an increasing/descreasing approximator. Given an approximator $o : \wp(\mathcal{L}^2) \to \wp(\mathcal{L}^2)$, we define variants which are increasing and decreasing, which we denote as o^+ and o^- respectively.

$$o^+(T, F, U, P)_{1,4} \coloneqq \Big((o(T, P)_1 \cup U^{\mathbf{c}}), (o(T, P)_2 \setminus F)\Big)$$
$$o^-(T, F, U, P)_{1,4} \coloneqq \Big((o(T, P)_1 \cap U), (o(T, P)_2 \cup F^{\mathbf{c}})\Big)$$

Both o^+ and o^- are recurrent approximators. As an example, if we apply $o^+(T, P^{\mathbf{c}}, T^{\mathbf{c}}, P)_{1,4}$, we get $(o(T, P)_1 \cup T, o(T, P) \cap P)$ which is $\preceq_p^2$-increasing.

Their stable operators are also $\preceq_p^2$-increasing.

Lemma 2. *For an approximator $o(T, P) : \wp(\mathcal{L}^2) \to \wp(\mathcal{L}^2)$, we have for any pair (T, P)*

$$(T, P) \preceq_p^2 S(o^+)(T, P^{\mathbf{c}}, T^{\mathbf{c}}, P)_{1,4}$$
$$S(o^-)(T, P^{\mathbf{c}}, T^{\mathbf{c}}, P)_{1,4} \preceq_p^2 (T, P)$$

With this, we can repeatedly invoke a stable operator to obtain more precise pairs. We've limited the theory above to approximators over the powerset lattice, however, its straightforward to extend it to recurrent approximators over any tetralattice.

The approximator o^+ can be interleaved with any other propagation method within a solver without the worry that a fixpoint will not be reached. In Algorithm 1 below, we briefly sketch a solver that uses $\Gamma'''_{\mathcal{P}}$ to find two-valued answer sets of a program $\mathcal{P}$. Given an interpretation (T, P), the algorithm will

Algorithm 1 A solver for normal programs with constraints

```
Require: $T, P \in \mathcal{L}, T \subseteq P$
  while $S(\Gamma'''_{\mathcal{P}})(T, P^{\mathbf{c}}, T^{\mathbf{c}}, P)_{1,4} \neq (T, P)$ do
      while $\exists a \in Atoms(\mathcal{P}), (T, P)(a) = \boldsymbol{u}$ do
          $(T', P') \leftarrow (Atoms(\mathcal{P}), \emptyset)$
          while $(T', P') \neq (T, P)$ do
              $(T', P') \leftarrow (T, P)$
              $(T, P) \leftarrow S(\Gamma'''^{+})(T, P^{\mathbf{c}}, T^{\mathbf{c}}, P)_{1,4}$        ▷ Use increasing stable revision
          end while
          if $\exists r \in bodysat_{(T,P)}(\mathcal{P}) \cap constraints(\mathcal{P})$ then
              backtrack choice or report no answer set.
          else
              choose some $a$ where $(T, P)(a) = \boldsymbol{u}$ to be $\boldsymbol{f}$ or $\boldsymbol{t}$
          end if
      end while
      backtrack choice or report no answer set.
  end while
  Terminate: $(T, P)$ is an answer set
```

attempt to extend it to a two-valued answer set of $\mathcal{P}$. It uses the increasing stable revision operator to reduce the search space.

5. Discussion

We have introduced recurrent approximators, a new type of approximator that can leverage additional false information during stable revision. This formalizes the approximators used in our prior work for hybrid MKNF knowledge bases [4]. The simplicity of this approach is surprising, as approximation fixpoint theory does not need to be modified to support the technique. As a result, other semantics defined with approximators could be extended to define more precise variations using our method. More uses of recurrent approximators remain to be discovered.

Given that the propagations are well-founded, the technique could allow AFT treatment of grounding procedures. This would be interesting for hybrid MKNF knowledge bases in particular, which have not received a thorough investigation of grounding techniques. For hybrid MKNF knowledge bases, computing the well-founded model is intractable [18]. In our prior work, we applied recurrent approximators to identify a larger class of polynomial computable well-founded models [4]. Such has implications for grounding. The ability to perform further propagation with false information can likely be applied to other hybrid reasoning systems such as DL-programs [19] or HEX [20], both of which have received treatment by AFT.

We've shown that recurrent approximators can be made $\preceq_p^2$-monotone. The o^+ operator bears some resemblance to stable revision in consistent AFT, which limits the image of $S(o)(T, P)$ s.t. $S(o)(T, P) \preceq_p^2 (P, T)$. The relationship between consistent and traditional AFT is not yet fully understood. It has been shown that, for nonsymmetric approximators, the stable fixpoints are not the same for stable revision and consistent stable revision [21]. We believe that recurrent approximators could play a role in further understanding this relationship because of their ability to limit a stable operator s.t. $S(o)(T, P) \preceq_p^2 (P, T)$.

Various techniques in answer set solving involve backward-chaining. The recent s(CASP) system [22] blends forward- and backward-reasoning for answer set programming. Additionally, SAT-solver style answer set solvers, such as Clingo [23], use a variety of backward-chaining techniques to perform constraint propagation. It would be interesting to see if these approaches could be characterized using recurrent AFT.

Acknowledgments

Spencer Killen was partially supported by Alberta Innovates and Alberta Advanced Education.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] M. Denecker, V. Marek, M. Truszczyński, Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning, in: Logic-Based Artificial Intelligence, Springer, 2000, pp. 127–144. doi:10.1007/978-1-4615-1567-8_6.

[2] L. Vanbesien, M. Bruynooghe, M. Denecker, Analyzing semantics of aggregate answer set programming using approximation fixpoint theory, Theory Pract. Log. Program. 22 (2022) 523–537. doi:10.1017/S1471068422000126.

[3] S. Marynissen, B. Bogaerts, M. Denecker, On the relation between approximation fixpoint theory and justification theory, in: Z. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021, ijcai.org, 2021, pp. 1973–1980. doi:10.24963/ijcai.2021/272.

[4] S. Killen, W. Gao, J. You, Expanding the class of polynomial time computable well-founded semantics for hybrid MKNF, in: J. Arias, S. Batsakis, W. Faber, G. Gupta, F. Pacenza, E. Papadakis, L. Robaldo, K. Rückschloß, E. Salazar, Z. G. Saribatur, I. Tachmazidis, F. Weitkämper, A. Z. Wyner (Eds.), Proceedings of the International Conference on Logic Programming 2023 Workshops co-located with the 39th International Conference on Logic Programming (ICLP 2023), London, United

Kingdom, July 9th and 10th, 2023, volume 3437 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3437/paper8ASPOCP.pdf.

[5] B. Motik, R. Rosati, Reconciling description logics and rules, J. ACM 57 (2010) 30:1–30:62. doi:10.1145/1754399.1754403.

[6] S. Roman, Lattices and Ordered Sets, Springer New York, 2008. doi:10.1007/978-0-387-78901-9.

[7] A. Tarski, A lattice-theoretical fixpoint theorem and its applications., Pacific Journal of Mathematics 5 (1955) 285 – 309. doi:10.2140/pjm.1955.5.285.

[8] F. Liu, J. You, Alternating fixpoint operator for hybrid MKNF knowledge bases as an approximator of AFT, Theory Pract. Log. Program. 22 (2022) 305–334. doi:10.1017/S1471068421000168.

[9] M. Denecker, V. W. Marek, M. Truszczynski, Ultimate approximation and its application in nonmonotonic knowledge representation systems, Inf. Comput. 192 (2004) 84–121. doi:10.1016/J.IC.2004.02.004.

[10] T. C. Przymusinski, The well-founded semantics coincides with the three-valued stable semantics, Fundam. Inform. 13 (1990) 445–463.

[11] N. D. Belnap, A Useful Four-Valued Logic, Springer Netherlands, Dordrecht, 1977, pp. 5–37. doi:10.1007/978-94-010-1161-7_2.

[12] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R. Kowalski, Bowen, Kenneth (Eds.), Proceedings of International Logic Programming Conference and Symposium, MIT Press, 1988, pp. 1070–1080. URL: http://www.cs.utexas.edu/users/ai-lab?gel88.

[13] S. Killen, J.-H. You, J. Heyninck, An alternative theory of stable revision for nondeterministic approximation fixpoint theory and the relationships, in: Thirty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2025, AAAI Press, 2025.

[14] T. C. Przymusinski, Stable semantics for disjunctive programs, New Gener. Comput. 9 (1991) 401–424. doi:10.1007/BF03037171.

[15] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Gener. Comput. 9 (1991) 365–386. doi:10.1007/BF03037169.

[16] M. Fitting, Fixpoint semantics for logic programming a survey, Theor. Comput. Sci. 278 (2002) 25–51. doi:10.1016/S0304-3975(00)00330-3.

[17] S. Killen, J. You, Unfounded sets for disjunctive hybrid MKNF knowledge bases, in: M. Bienvenu, G. Lakemeyer, E. Erdem (Eds.), Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021, 2021, pp. 432–441. doi:10.24963/kr.2021/41.

[18] F. Liu, J. You, Three-valued semantics for hybrid MKNF knowledge bases revisited, Artif. Intell. 252 (2017) 123–138. doi:10.1016/j.artint.2017.08.003.

[19] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, H. Tompits, Combining answer set programming with description logics for the semantic web, Artif. Intell. 172 (2008) 1495–1539. URL: https://doi.org/10.1016/j.artint.2008.04.002. doi:10.1016/J.ARTINT.2008.04.002.

[20] T. Eiter, G. Ianni, R. Schindlauer, H. Tompits, A uniform integration of higher-order reasoning and external evaluations in answer-set programming, in: L. P. Kaelbling, A. Saffiotti (Eds.), IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005, Professional Book Center, 2005, pp. 90–96. URL: http://ijcai.org/Proceedings/05/Papers/1353.pdf.

[21] Y. Bi, J.-H. You, Z. Feng, A generalization of approximation fixpoint theory and application, in: R. Kontchakov, M.-L. Mugnier (Eds.), Web Reasoning and Rule Systems, Springer International Publishing, Cham, 2014, pp. 45–59.

[22] J. Arias, M. Carro, E. Salazar, K. Marple, G. Gupta, Constraint answer set programming without grounding, Theory Pract. Log. Program. 18 (2018) 337–354. URL: https://doi.org/10.1017/S1471068418000285. doi:10.1017/S1471068418000285.

[23] M. Gebser, B. Kaufmann, T. Schaub, Conflict-driven answer set solving: From theory to practice, Artif. Intell. 187 (2012) 52–89. doi:10.1016/j.artint.2012.04.001.