A Formalisation of Approximation Fixpoint Theory

Bart Bogaerts[1*] and Luís Cruz-Filipe[2]

[1] Vrije Universiteit Brussel (VUB), Dept. Computer Science, Brussels, Belgium
bart.bogaerts@vub.be

[2] Univ. Southern Denmark, Dept. Mathematics and Computer Science, Odense, Denmark
lcfilipe@gmail.com

Introduction. *Approximation Fixpoint Theory* (AFT) is an abstract lattice-theoretic framework originally designed to unify semantics of non-monotonic logics [9]. Its first applications were on unifying all major semantics of logic programming [25], autoepistemic logic (AEL) [18], and default logic (DL) [20], thereby resolving a long-standing issue about the relationship between AEL and DL [14, 10, 11]. AFT builds on Tarski's fixpoint theory of monotone operators on a complete lattice [23], starting from the key realisation that, by moving from the original lattice L to the bilattice L^2, Tarski's theory can be generalized into a fixpoint theory for arbitrary (i.e., also non-monotone) operators. Crucially, all that is required to apply AFT to a formalism and obtain several semantics is to define an appropriate approximating operator $L^2 \to L^2$ on this bilattice; the algebraic theory of AFT then directly defines different types of fixpoints that correspond to different types of semantics of the application domain.

In the last decade, AFT has seen several new application domains, including abstract argumentation [22], extensions of logic programming [19, 1, 8, 15], extensions of autoepistemic logic [26], and active integrity constraints [4]. Around the same time, also the theory of AFT has been extended signitificantly with new types of fixpoints [6, 7], and results on *stratification*, [27, 5], *predicate introduction* [28], and *strong equivalence* [24]. All of these results were developed in the highly general setting of lattice theory, making them directly applicable to all application domains, and such ensuring that researchers do not "reinvent the wheel".

Given the success and wide range of applicability of AFT, it sounded natural to formalise this theory in the Coq theorem prover. In this work we report on the first steps of this endeavour.

AFT in a nutshell. AFT studies fixpoints of operators $O\colon L \to L$, where $\langle L, \leq\rangle$ is a lattice, through operators approximating O. These operators work in the *bilattice* $L^2 = \langle L \times L, \leq_p\rangle$, where the *precision order* $\leq_p$ is defined as $(x, y) \leq_p (u, v)$ if $x \leq u$ and $y \geq v$.

Intuitively, a pair $(x, y) \in L^2$ approximates elements in the interval $[x, y] = \{z \in L \mid x \leq z \leq y\}$. We call $(x, y) \in L^2$ *consistent* if $x \leq y$, i.e., if $[x, y]$ is non-empty. The set of consistent elements is denoted by L^c. Pairs (x, x) are called *exact*, since they only approximate x. If (u, v) is consistent and $(x, y) \leq_p (u, v)$, then $[u, v] \subseteq [x, y]$, i.e., (x, y) approximates all elements that (u, v) approximates. We say that (u, v) is *more precise* than (x, y).

An operator $A\colon L^2 \to L^2$ is an *approximator* of O if it is $\leq_p$-monotone and has the property that $A(x, x) = (O(x), O(x))$ for all $x \in L$. As usual in AFT, we often restrict our attention to *symmetric* approximators: approximators A such that, for all x and y, $A(x, y)_1 = A(y, x)_2$. AFT defines the following fixpoints of A in order to study fixpoints of O.

- A *partial supported fixpoint* of A is a fixpoint of A.
- The *Kripke-Kleene fixpoint* of A is the $\leq_p$-least fixpoint of A.
- A *partial stable fixpoint* of A is a pair (x, y) where $x = \mathrm{lfp}(A(\cdot, y)_1)$ and $y = \mathrm{lfp}(A(x, \cdot)_2)$. $A(\cdot, y)_1$ denotes the function $L \to L\colon z \mapsto A(z, y)_1$, and analogously for $A(x, \cdot)_2$.
- The *well-founded fixpoint* of A is the least precise partial stable fixpoint of A.

*This work was partially supported by Fonds Wetenschappelijk Onderzoek – Vlaanderen (project G0B2221N).

Our formalisation. Our aim is to formalise AFT in Coq without using any axioms – in particular, by following a constructive development of AFT. This is a natural choice, since an important motivations for studying fixpoints in computer science is their computability.

Most AFT proofs are by transfinite induction, and we chose to follow the published results closely. We define a type `Ordinal` of (unbounded) ordinals as a record type containing a `Type`, an equivalence relation `eq` (defined equality), a distinguished element `zero`, a successor function `succ` and a strict total order `lt` that is well-founded and compatible with `eq`. We require `succ x` to be the least element strictly greater than `x`, and that we can decide whether an element is of the form `succ x` for some `x`; the elements for which this does not hold are called limits.

Intuitively, the elements of `o:Ordinal` are the ordinals smaller than `o`. For sanity check, we show that the natural numbers form an ordinal, that we can add ω to an ordinal, and that we can build the type of all polynomials in ω (with support `list nat`). If `o:Ordinal`, then we can prove properties of all elements of `o` by transfinite induction. Since we work with defined equality, we need to include a case showing that the property being proved is stable under equality. We do not deal with arithmetic on ordinals, since this is immaterial for our development.

(Complete) lattices are similarly defined as a record type consisting of a carrier type `C` with an equivalence relation, a partial order, and an operator `lub:(C → Prop) → C` computing least upper bounds. Requiring least upper bounds to be computable restricts the kind of lattices that we can define; still we show that we capture e.g. powerset lattices (which appear in all applications of AFT so far). We also define an operator `BiLattice : Lattice → Lattice`. Given an operator $O : L \to L$, we inductively define a (O-)chain as a predicate over L that is closed under applications of O and lubs. We prove that, if O is monotonic, then the lub of any chain is an element of the chain, and it is the least fixpoint (lfp) of O (Knaster–Tarski theorem).

AFT provides an alternative characterisation of lfps using so-called O-inductions. Given an operator $O : L \to L$, $y \in L$ is an O-refinement of x if $x \leq y$ and $y \leq x \vee O(x)$. An O-induction is a transfinite sequence i such that $i_{\eta+1}$ is an O-refinement of i_η and $i_\eta = \mathrm{lub}\{i_{\eta'} \mid \eta' < \eta\}$ for every limit ordinal η. An O-induction is terminal if there is an ordinal η such that the only refinement of i_η is i_η. We prove that every terminal O-induction converges to the least fixpoint of O, and, conversely, that an O-induction that reaches a fixpoint of O is terminal.

Finally, we formalise the notions of approximator and the four types of fixed points defined earlier, and prove their main properties. The whole development can be found at https://doi.org/10.5281/zenodo.4893264.

Discussion. The main challenges encountered so far have to do with our choice to work constructively (without adding any axioms to Coq), which required adapting some proofs in AFT that assume decidability of equality on the lattice.

Several results in AFT require the existence of a "large enough" ordinal for a given lattice. Since we have not been able to construct this ordinal from the lattice, we have defined a notion of "large enough" ordinal, and explicitly add it as a hypothesis when needed. In this way, we choose to accept its existence as a postulate, or prove it using classical logic.

Related work. Transfinite induction has been formalised previously [21, 2, 12, 13, 17], in some cases with a proof of the Knaster–Tarski theorem. Some of these works are based on classical set theory; others formalise ordinals in a way that we found cumbersome to use, which led us to defining our own. The CoLoR library [3, 16] includes a formalisation of the Knaster–Tarski theorem similar to ours. To the best of our knowledge, AFT has not been formalised before.

References

[1] Christian Antic, Thomas Eiter, and Michael Fink. Hex semantics via approximation fixpoint theory. In Pedro Cabalar and Tran Cao Son, editors, *Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings*, volume 8148 of *LNCS*, pages 102–115. Springer, 2013.

[2] Bruno Barras. Sets in Coq, Coq in Sets. *J. Formaliz. Reason.*, 3(1):29–48, 2010.

[3] Frédéric Blanqui and Adam Koprowski. Color: a coq library on well-founded rewrite relations and its application to the automated verification of termination certificates. *Math. Struct. Comput. Sci.*, 21(4):827–859, 2011.

[4] Bart Bogaerts and Luís Cruz-Filipe. Fixpoint semantics for active integrity constraints. *Artif. Intell.*, 255:43–70, 2018.

[5] Bart Bogaerts and Luís Cruz-Filipe. Stratification in approximation fixpoint theory and its application to active integrity constraints. *ACM Trans. Comput. Log.*, 22(1):6:1–6:19, 2021.

[6] Bart Bogaerts, Joost Vennekens, and Marc Denecker. Grounded fixpoints and their applications in knowledge representation. *Artif. Intell.*, 224:51–71, 2015.

[7] Bart Bogaerts, Joost Vennekens, and Marc Denecker. Safe inductions and their applications in knowledge representation. *Artificial Intelligence*, 259:167 – 185, 2018.

[8] Angelos Charalambidis, Panos Rondogiannis, and Ioanna Symeonidou. Approximation fixpoint theory and the well-founded semantics of higher-order logic programs. *CoRR*, abs/1804.08335, 2018. to appear in TPLP.

[9] Marc Denecker, Victor Marek, and Mirosław Truszczyński. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, volume 597 of *The Springer International Series in Engineering and Computer Science*, pages 127–144. Springer US, 2000.

[10] Marc Denecker, Victor Marek, and Mirosław Truszczyński. Uniform semantic treatment of default and autoepistemic logics. *Artif. Intell.*, 143(1):79–122, 2003.

[11] Marc Denecker, Victor Marek, and Mirosław Truszczyński. Reiter's default logic is a logic of autoepistemic reasoning and a good one, too. In Gerd Brewka, Victor Marek, and Mirosław Truszczyński, editors, *Nonmonotonic Reasoning – Essays Celebrating Its 30th Anniversary*, pages 111–144. College Publications, 2011.

[12] Hervé Grall. Proving fixed points. Technical Report hal-00507775, HAL archives ouvertes, 2010.

[13] José Grimm. Implementation of three types of ordinals in Coq. Technical Report RR-8407, INRIA, 2013.

[14] Kurt Konolige. On the relation between default and autoepistemic logic. *Artif. Intell.*, 35(3):343–382, 1988.

[15] Fangfang Liu, Yi Bi, Md. Solimul Chowdhury, Jia-Huai You, and Zhiyong Feng. Flexible approximators for approximating fixpoint theory. In Richard Khoury and Christopher Drummond, editors, *Advances in Artificial Intelligence - 29th Canadian Conference on Artificial Intelligence, Canadian AI 2016, Victoria, BC, Canada, May 31 - June 3, 2016. Proceedings*, volume 9673 of *Lecture Notes in Computer Science*, pages 224–236. Springer, 2016.

[16] Frédéric Blanqui (maintainer). `https://github.com/fblanqui/color/blob/master/Util/Relation/Tarski.v`. Accessed: 2021-06-02.

[17] Pierre Castéran (maintainer). `https://github.com/coq-community/hydra-battles/tree/master/theories/ordinals`. Accessed: 2021-06-02.

[18] Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artif. Intell.*, 25(1):75–94, 1985.

[19] Nikolay Pelov, Marc Denecker, and Maurice Bruynooghe. Well-founded and stable semantics of logic programs with aggregates. *TPLP*, 7(3):301–353, 2007.

[20] Raymond Reiter. A logic for default reasoning. *Artif. Intell.*, 13(1-2):81–132, 1980.

[21] Carlos Simpson. Set-theoretical mathematics in Coq. *CoRR*, abs/math/0402336, 2004.

[22] Hannes Strass. Approximating operators and semantics for abstract dialectical frameworks. *Artif. Intell.*, 205:39–70, 2013.

[23] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 1955.

[24] Mirosław Truszczyński. Strong and uniform equivalence of nonmonotonic theories - an algebraic approach. *Ann. Math. Artif. Intell.*, 48(3-4):245–265, 2006.

[25] Maarten H. van Emden and Robert A. Kowalski. The semantics of predicate logic as a programming language. *J. ACM*, 23(4):733–742, 1976.

[26] Pieter Van Hertum, Marcos Cramer, Bart Bogaerts, and Marc Denecker. Distributed autoepistemic logic and its application to access control. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1286–1292. IJCAI/AAAI Press, 2016.

[27] Joost Vennekens, David Gilis, and Marc Denecker. Splitting an operator: Algebraic modularity results for logics with fixpoint semantics. *ACM Trans. Comput. Log.*, 7(4):765–797, 2006.

[28] Joost Vennekens, Maarten Mariën, Johan Wittocx, and Marc Denecker. Predicate introduction for logics with a fixpoint semantics. Parts I and II. *Fundamenta Informaticae*, 79(1-2):187–227, 2007.