Abstraction Logic Is All You Need

Steven Obua[0000−0002−4362−752X]

Practal
https://practal.com
obua@practal.com

Abstract. Abstraction logic is a minimal yet maximally expressive logical system that serves both as a logic and a logical framework. While existing logics often trade simplicity for expressiveness, abstraction logic achieves both, making it an ideal foundation for machine-learning approaches to theorem proving. Its simplicity enables ML systems to manipulate and adapt it easily, while its expressiveness ensures no practical limitations. We present the core principles of abstraction logic and discuss its unique characteristics that make it well suited for AITP (artificial intelligence theorem proving).

Keywords: Logic · Logical Frameworks · Mathematical Foundations · Abstraction · Algebra · Natural Deduction · Sequent Calculus

Theorem proving could very well become one of the biggest applications of modern AI. The idea is that while AI creativity can be put to full use for coming up with new mathematical ideas, theories, theorems, and proofs, the downside of generative AI, hallucination, can be reined in because *logic* has an intrinsic notion of truth through proof.

But which logic should we use for building our AITP systems? The most pragmatic approach may be to just use the logic that comes with the theorem proving system of your choice. For example, first-order logic with set theory is the logic of Mizar [1], simply-typed higher-order logic is the foundation for Isabelle [2], HOL [3] and HOL Light [4], and dependently-typed higher-order logic forms the basis of Lean [5] and Coq [6]. Furthermore, existing systems come with large libraries such as the Archive of Formal Proof [7] and Mathlib [8], providing much needed fuel for data-hungry AI. In the end, the bitter lesson [9] is that it should not matter too much which system and which logic we choose.

Here, we are *not* advocating the above pragmatic approach.

Instead, we propose starting from scratch. Now is the time to rethink from the ground up how things are done and build a modern AITP system free from the shackles of the past but still strongly rooted in the lessons the past taught us. Perhaps the most important such lesson is that neither first-order logic, nor second-order logic, nor simply-typed higher-order logic, nor dependently typed logic is fully satisfactory. To understand why, let us classify a logic along two axes. The first axis is whether the logic allows working with a ***single mathematical universe of mathematical objects***, or whether the logic needs to underapproximate the universe through an infinite hierarchy of typed universes. The

second axis is whether the logic supports the introduction of ***general variable binding constructs (general binders)***. First-order and second-order logic score along the first axis, but don't have general binders. The only binders they allow are universal and existential quantification, and therefore they don't score on the second axis. Higher-order logics on the other hand have general binders via the lambda calculus. Lambdas are mathematical objects, and this introduces the need for types, as otherwise **Cantor's theorem** [10, p. 4] immediately leads to inconsistency. So both simply-typed and dependently typed higher-order logics score along the second axis, but will never be able to score along the first axis.

Abstraction logic is a new logic, and described in detail in its 2025 technical report [10]. *Abstraction logic scores along both axes, providing both a single mathematical universe and general binders.* This is possible because abstraction logic is based on **abstraction algebra**, which is an extension of *abstract algebra*. While abstract algebra talks about a universe together with a collection of *operations*, abstraction algebra talks about a universe together with a collection of *operators*. Here an **n-ary operation** on a universe is a function taking n values[1] from the universe as inputs, yielding a value from the universe. An **operator of shape** $[m_1, \ldots, m_n]$ is a function taking n operations o_i of arity m_i as arguments, resulting again in a value from the universe.[2]

Abstraction algebra is turned into a formal language via *variables* and *abstractions*. An **abstraction** is simply a name meant to denote some operator of fixed shape. A **variable** ranges over values and operations. The resulting language consists of *terms* denoting values and *templates* denoting operations. A **term** is either a **variable application** $x[t_1, \ldots, t_n]$, where x is a variable and the t_i are terms, or it is an **abstraction application** $a\,T_1 \ldots T_n$, where a is an abstraction of shape $[m_1, \ldots, m_n]$ and each T_i is a template of arity m_i. An **n-ary template** has the form $(x_1 \ldots x_n.\, t)$, where the x_i are pairwise distinct variables and t is a term. A template of arity 0 is simply a term. That's it!

From abstraction algebra, we obtain abstraction logic by considering **sequents** (L, R), where L and R are finite collections of templates modulo α-equivalence. A logic is then just a collection of abstractions together with a collection of sequents called **axioms**. If we restrict R to have the form $\{t\}$, where t is a term, we can give a proof system which we call **natural deduction**[3] that is both **sound** and **complete** with respect to a simple semantics, *regardless of which abstractions we choose, and regardless of which axioms we choose*! The key to the semantics is that *any* term t can be viewed as a **formula** $(\emptyset, \{t\})$.

In this way, we obtain a logic that can adapt to any situation by choosing suitable abstractions and axioms. For example, predicate calculus can be expressed like this [10, pp. 94-96]. Higher-order logics can be expressed similarly by treating types as ordinary mathematical objects. Even paraconsistency [11] can be modelled. More generally, *abstraction logic is both a logic and a logical framework*. Now, we are ready to embrace the bitter lesson.

[1] A value is the same as a mathematical object.
[2] Every value is also a nullary operation, and hence every operation is also an operator.
[3] Without this restriction of sequents to **rules**, we obtain **sequent calculus**.

References

1. Adam Grabowski, Artur Kornilowicz and Adam Naumowicz. *Mizar in a Nutshell.* https://doi.org/10.6092/issn.1972-5787/1980, 2010.
2. Lawrence C. Paulson. *The foundation of a generic theorem prover.* https://doi.org/10.1007/BF00248324, 1989.
3. M.J.C. Gordon and A.M. Pitts. *The HOL Logic and System.* https://doi.org/10.1016/B978-0-444-89901-9.50012-4, 1994.
4. John Harrison. *The HOL Light manual (1.1).* https://www.cl.cam.ac.uk/~jrh13/hol-light/manual-1.1.pdf, April 2000.
5. Mario Carneiro. *The Type Theory of Lean.* https://github.com/digama0/lean-type-theory, April 2019.
6. *The Coq Proof Assistant.* https://coq.inria.fr, visited January 2025.
7. *The Archive of Formal Proofs.* https://isa-afp.org, visited January 2025.
8. *The Math Library of Lean 4.* https://github.com/leanprover-community/mathlib4, visited January 2025.
9. Rich Sutton. *The Bitter Lesson.* http://www.incompleteideas.net/IncIdeas/BitterLesson.html, March 2019.
10. Steven Obua. *Abstraction Logic (Founder's Edition): A New Foundation for Reasoning, Computing, and Understanding.* https://doi.org/10.5281/zenodo.14729557, January 2025. Free download for workshop reviewers and participants: https://tinyurl.com/al-euro-wg5.
11. Steven Obua and Claude. *A Conversation with Graham Priest About Abstraction Logic.* https://practal.com/press/cwgpaal/1, January 2025.