Dealing with Soft Types in Naproche's Logical Backend

Adrian De Lon, Peter Koepke, and Anton Lorenzen

University of Bonn, Germany, https://www.math.uni-bonn.de/ag/logik/

In formal mathematics there is a demand for readable or natural input languages. The Formal Abstracts project of Thomas Hales [3] proposes to

- give a statement of the main theorem of each published mathematical paper in a language that is both human- and machine-readable,
- link each term in theorem statements to a precise definition of that term (again in both human- and machine-readable form).

Ideally, the language of Formal Abstracts is part of the common natural language for mathematics, including symbolic terms, and – at the same time – is fully formal with respect to an efficiently implementable formal grammar. The language of Formal Abstracts should thus be a controlled natural language (CNL) for mathematics. Even though the Formal Abstracts project does not demand proofs of main theorems, Formal Abstracts have to be well-formed with respect to the CNL grammar and well-posed mathematically. The latter may involve non-trivial or even substantial proving: the simple well-definedness of the Euclidean norm $\sqrt{a^2+b^2}$ on $\mathbb{R}^2$, e.g., involves checking that the term a^2+b^2 is always non-negative, which requires a proof from the axioms for ordered fields.

The Naproche (Natural Proof Checking) system [2] uses the mathematical CNL ForTheL [5]. Before proving a ForTheL statement, the reasoner checks for its well-definedness in what is called an ontological check which corresponds to type-checking. ForTheL as a natural language employs a "soft" type system where types may have non-trivial definitions. So far, Naproche translates soft types into first-order type guards which are dealt with by an external ATP like E [6]. Although ontological checks are typically shallow in comparison to the proof of the statement itself, the number of ontological checks often surpasses the number of logical checks. This may lead to a high proof burden or even failure of overall checking.

A main motivation for our present work is to internally employ adequate and efficient type mechanisms to model ForTheL's soft types. This requires a redesign of Naproche's logical backend which transforms parse trees into formulae of the internal logic. We have made experiments with the many-sorted logic TFF0 [7] together with automatic coercions for subtypes. This covers, e.g., number systems with types for integers, rationals and reals, or set theories with sets and classes where one has obvious direct embeddings between types.

Coercions are often implemented using type-classes and are inferred either implicitly (for example in Coq [1]) or explicitly by a call to a function like `coe` (in Lean). Type-classes can also be used for subtyping and are used to implement algebraic structures like groups or metric spaces in Lean's mathlib. This technique is very powerful, but requires a worst-case exponential time algorithm.

In the case of direct embeddings (which form a transitive partial order) we can instead use a data structure for transitive closures [4], which can infer a coercion in $O(1)$ time when using arrays. Because we expect the graph of coercions to be relatively sparse we use a binary tree instead for storing the coercions. The asymptotic lookup time is then $O(\log n)$, where n is the number of introduced types. This makes it feasible to check for coercions at every function application. Furthermore, the total time consumed by insertions into the data structure is bounded by $O(nm)$, where n is the number of types and m the number of different coercions

between two types. These coercions yield TFF0 formulae that can be processed directly by E or Vampire without introducing type guards. We now have parsing and checking algorithms that are cubic in the worst case and linear for several interesting formalizations. The ontological checking of those formalizations is accelerated by orders of magnitude and provides informative error messages in case of failure.

The new Naproche backend includes a variety of other enhancements like caching of proof results and automatic translations to Lean. The backend also serves as basis for a future web interface to Naproche.

References

[1] Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Yann Coscoy, David Delahaye, Daniel de Rauglaudre, Jean-Christophe Filliâtre, Eduardo Giménez, Hugo Herbelin, et al. The Coq proof assistant reference manual. *INRIA, version*, 6(11), 1999.

[2] Adrian De Lon, Peter Koepke, Anton Lorenzen, Adrian Marti, Marcel Schütz, and Makarius Wenzel. The Isabelle/Naproche natural language proof assistant (to appear). In *CADE-28*, 2021.

[3] Thomas Hales. Formal Abstracts, 2020. URL: https://formalabstracts.github.io/.

[4] G.F. Italiano. Amortized efficiency of a path retrieval data structure. *Theoretical Computer Science*, 48:273–281, 1986.

[5] Andrei Paskevich. The syntax and semantics of the ForTheL language, 2007.

[6] Stephan Schulz. The E theorem prover. URL: https://eprover.org.

[7] Geoff Sutcliffe, Stephan Schulz, Koen Claessen, and Peter Baumgartner. The TPTP typed first-order form with arithmetic. In Nikolaj Bjørner and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 406–419, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.