Coq as a Metatheory for Nuprl with Bar Induction

Vincent Rahli and Mark Bickford *

Cornell University

Abstract

These past few years, we have been experimenting in Nuprl with versions of Brouwer's Bar Induction principle. Until recently we had no formal proof that these rules are valid Nuprl reasoning principles. Thanks to our formalization of Nuprl's metatheory in Coq, we can now rigorously check whether these principles are consistent with Nuprl. In this paper we present a proof, using our Coq framework, of the validity of Brouwer's Bar Induction principle on sequences of natural numbers. To prove this result we added all Coq functions from natural numbers to natural numbers to Nuprl's computation system.

Introduction. Nuprl [9, 3] is a dependent type theory à la Martin-Löf based on an untyped functional programming language. Nuprl has a rich type theory including identity (or equality) types, a hierarchy of universes, W types, quotient types [10], set types, union and (dependent) intersection types [14], image types [15], PER types [4], simulation types [17], and partial types [11]. Type checking is undecidable but in practice this is mitigated by type inference and checking heuristics implemented as tactics. Nuprl types are defined as partial equivalence relations (PERs) on closed terms [2, 1, 11]. We have implemented Nuprl's PER semantics in Coq and verified a large number of its inference rules [5, 6].

Bar Induction. These past few years we have been experimenting with versions of Brouwer's Bar Induction principle [8, 18]. In [7], we showed how to build parametrized families of W types using a general form of Bar Induction. We present here our first attempt at proving that this rule is valid w.r.t. Nuprl PER semantics. The rule we have proved is:

$$
\begin{array}{ll}
H \vdash \downarrow(X\ 0\ \texttt{norm}(c,0)) & \\
\quad \text{BY } \texttt{[BarInduction]} & \\
\quad (\texttt{dec}) & H, n:\mathbb{N}, s:\mathbb{N}^{\mathbb{N}_n} \vdash B\ n\ s\ \vee\ \neg B\ n\ s \\
\quad (\texttt{bar}) & H, s:\mathbb{N}^{\mathbb{N}} \vdash \downarrow\exists n:\mathbb{N}.\ B\ n\ \texttt{norm}(s,n) \\
\quad (\texttt{imp}) & H, n:\mathbb{N}, s:\mathbb{N}^{\mathbb{N}_n}, m:B\ n\ s \vdash X\ n\ s \\
\quad (\texttt{ind}) & H, n:\mathbb{N}, s:\mathbb{N}^{\mathbb{N}_n}, x:(\forall m:\mathbb{N}.\ X\ (n+1)\ \texttt{ext}(s,n,m)) \vdash X\ n\ s
\end{array}
$$

where

$$
\begin{array}{c}
\texttt{norm}(s,n) = \lambda x.\texttt{if}\ x{<}0\ \texttt{then}\ \bot\ \texttt{else if}\ x{<}n\ \texttt{then}\ s(x)\ \texttt{else}\ \bot \\
\texttt{ext}(s,n,m) = \lambda x.\texttt{if}\ x =_{\mathbb{Z}} n\ \texttt{then}\ m\ \texttt{else}\ s(x) \\
\downarrow T = \{\texttt{Unit} \mid T\}
\end{array}
$$

The *normalized* sequence $\texttt{norm}(s,n)$ returns $s(x)$ for inputs x between 0 and n, and otherwise returns $\bot$. Our first attempt to verify this rule did not use `norm` and, to use `[BarInduction]`, required one to, in addition to the above subgoals, prove that X is a well-formed function of type $n{:}\mathbb{N} \to \mathbb{N}^{\mathbb{N}_n} \to \texttt{Type}$. This is undesirable and can be avoided using `norm`.

As mentioned in [7], we can define a *bar recursion* operator and using this *squashed* bar induction rule (using the squashing operator $\downarrow$), we can prove that bar recursion is a realizer for bar induction.

*This work was partially supported by the SnT and by the National Research Fund Luxembourg (FNR), through PEARL grant FNR/P14/8149128.

We have proved that this rule is true in our impredicative metatheory (in Prop) following Dummett's "standard" classical proof [12, pp.55]. Our Coq implementation is available at the following address: `http://www.nuprl.org/html/Nuprl2Coq/`. The `[BarInduction]` rule is often called BID for Bar Induction on Decidable bars. As in Dummett's proof we used the law of excluded middle and the axiom of choice. Note that because the proof is classical, we will not need to use the sequent (`dec`). His proof goes as follows: first we assume the falsity of the conclusion using the law of excluded middle. Then, we contrapose (`ind`), and using the axiom of choice we obtain a function F that, for all $n \in \mathbb{N}$, $s \in \mathbb{N}^{\mathbb{N}_n}$, and proof of $\neg X\ n\ s$, returns a number m such that $\neg X\ (n+1)\ \texttt{ext}(s,n,m)$. Because $\neg(X\ 0\ \texttt{norm}(c,0))$, this gives us a sequence $\alpha \in \mathbb{N}^{\mathbb{N}}$ such that for all $n \in \mathbb{N}$, $\neg X\ \alpha(n)\ \texttt{norm}(\alpha,\alpha(n))$. Finally, we get a contradiction using (`bar`) and then (`imp`).

Coq sequences as Nuprl terms. How did we construct the sequence α? From F we can get a Coq functions from numbers to numbers, but for our proof we need a Nuprl term that computes numbers to numbers. Therefore, to remedy that we added all Coq functions from natural numbers to natural numbers to Nuprl's computation system (this was suggested to us by our colleague Evan Moran). Adding all Coq function from numbers to numbers is similar to adding free choice sequences, which is how Brouwer justified bar induction. In our metatheory, $\texttt{seq}(f)$, where f is a Coq function from numbers to numbers, is now a valid Nuprl expression. It computes as follows: to reduce the application $\texttt{seq}(f)\ t$, we first reduce t down to a value. If t computes to a natural number n, then we return the natural number $f(n)$. Otherwise, the computation gets stuck or diverges.

Current and Future work. Using our Coq framework, we have also recently proved Brouwer's continuity principle for numbers [16]. From BID and continuity we derived, in Nuprl, bar induction on monotone bars [13]. We are now investigating a bar induction rule for sequences of closed terms and not only natural numbers, which is the version we used to build parametrized W types [7]. For that we are adding all Coq functions from numbers to closed terms to our computation system, and we are investigating the implications of this modification.

References

[1] Stuart F. Allen. A non-type-theoretic definition of Martin-Löf's types. In *LICS*, pages 215–221. IEEE Computer Society, 1987.

[2] Stuart F. Allen. *A Non-Type-Theoretic Semantics for Type-Theoretic Language*. PhD thesis, Cornell University, 1987.

[3] Stuart F. Allen, Mark Bickford, Robert L. Constable, Richard Eaton, Christoph Kreitz, Lori Lorigo, and Evan Moran. Innovations in computational type theory using Nuprl. *J. Applied Logic*, 4(4):428–469, 2006. `http://www.nuprl.org/`.

[4] Abhishek Anand, Mark Bickford, Robert L. Constable, and Vincent Rahli. A type theory with partial equivalence relations as types. Presented at TYPES 2014, 2014.

[5] Abhishek Anand and Vincent Rahli. Towards a formally verified proof assistant. In *ITP 2014*, volume 8558 of *LNCS*, pages 27–44. Springer, 2014.

[6] Abhishek Anand and Vincent Rahli. Towards a formally verified proof assistant. Technical report, Cornell University, 2014. `http://www.nuprl.org/html/Nuprl2Coq/`.

[7] Mark Bickford and Robert Constable. Inductive construction in nuprl type theory using bar induction. Presented at the TYPES 2014 Workshop `http://nuprl.org/KB/show.php?ID=723`, 2014.

[8] Douglas Bridges and Fred Richman. *Varieties of Constructive Mathematics*. London Mathematical Society Lecture Notes Series. Cambridge University Press, 1987.

[9] R.L. Constable, S.F. Allen, H.M. Bromley, W.R. Cleaveland, J.F. Cremer, R.W. Harper, D.J. Howe, T.B. Knoblock, N.P. Mendler, P.Panangaden, J.T. Sasaki, and S.F. Smith. *Implementing mathematics with the Nuprl proof development system*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1986.

[10] Robert L. Constable. Constructive mathematics as a programming logic I: some principles of theory. In *Fundamentals of Computation Theory, Proceedings of the 1983 International*, volume 158 of *LNCS*, pages 64–77. Springer, 1983.

[11] Karl Crary. *Type-Theoretic Methodology for Practical Programming Languages*. PhD thesis, Cornell University, Ithaca, NY, August 1998.

[12] Michael A. E. Dummett. *Elements of Intuitionism*. Clarendon Press, second edition edition, 2000.

[13] S.C. Kleene and R.E. Vesley. *The Foundations of Intuitionistic Mathematics, especially in relation to recursive functions*. North-Holland Publishing Company, 1965.

[14] Alexei Kopylov. *Type Theoretical Foundations for Data Structures, Classes, and Objects*. PhD thesis, Cornell University, Ithaca, NY, 2004.

[15] Aleksey Nogin and Alexei Kopylov. Formalizing type operations using the "image" type constructor. *Electr. Notes Theor. Comput. Sci.*, 165:121–132, 2006.

[16] Vincent Rahli and Mark Bickford. A nominal exploration of intuitionism. Extended version avaible at `http://www.nuprl.org/html/Nuprl2Coq/`, 2015.

[17] Vincent Rahli, Mark Bickford, and Abhishek Anand. Formal program optimization in Nuprl using computational equivalence and partial types. In *ITP'13*, volume 7998 of *LNCS*, pages 261–278. Springer, 2013.

[18] Michael Rathjen. A note on bar induction in constructive set theory. *Math. Log. Q.*, 52(3):253–258, 2006.