Cubical type theory with several universes in Nuprl

Mark Bickford, Cornell University, Computer Science

May 20, 2020

1 Abstract

In 2018 we formalized in Nuprl a semantics for cubical type theory [1] (without higher inductive types) that had been developed by Coquand, et. al. [2]. This formalization shows that cubical type theory indeed had a completely constructive meaning, but it does not give us a user friendly proof assistant for cubical type theory implemented inside of Nuprl, because it uses a variable-free, higher order abstract syntax, and because it does not include a verified type-checking algorithm for cubical type theory.

We have now made a data type for the concrete syntax (with variables) of cubical type theory, and a *definition of truth* for the judgments of cubical type theory expressed in this syntax. Although Nuprl's types are not the same as the types of cubical type theory the univalence does not hold in Nuprl, we can still include all the Nuprl types (in some universe) as *discrete* cubical types where all paths are trivial (refl) paths.

We have defined a category of *weakly complete enough* (WCE) metric spaces and showed that there is a π_1 functor from the category of WCE spaces to the category of groups, so at least the fundamental group is constructively definable for WCE spaces. We conjecture that every WCE space X can give rise to a corresponding cubical type, but we have not yet verified this conjecture. If so, we would have a way to relate the constructive analysis on metric spaces to the synthetic methods of homotopy type theory by using univalence to prove that the WCE types are equal to their synthetic versions.

Years ago, N.G. DeBruijn (who created the AutoMath system) remarked to Robert Constable that type theory "needs only three universes. Think about it, and you will see that I am right!". So, we decided to put three cubical universes U_1, U_2, and U_3 into our interpretation of the syntax of cubical type theory. We discovered that while contexts (*cubical sets*) and cubical types in a context are both defined as families of Nuprl types, these families can be in different universes (in particular, the level of the type may be lower than the level of the context). To handle three universes in our definition of truth, we had to generalize about 950 lemmas that we had proved previously to allow the levels of context and type to be instantiated independently. This took about one month, because some of the proofs in this theory are the longest ever created in Nuprl.

The meaning of expressions of cubical type theory are *provisional.* They exist provided that the term is indeed meaningful. We defined a *provisional* monad where for $t \in$ Provisional(T), allowed(t) is a proposition, and allowed(t) $\Rightarrow$

allow$(t) \in T$. The meaning of an expression that should be a cubical type (in context G) is, provisionally, a triple of a level, a cubical type at that level, and a uniform composition operator for that cubical type, so it has Nuprl type Provisional(cttType(G)) where

$$\mathrm{cttType}(G) == \mathit{lvl}\!:\!\mathbb{N}_4 \times T\!:\!\mathit{CubicalType}(G, \mathit{lvl}) \times \mathrm{Comp}(G, \mathit{lvl}, T)$$

The meaning of an expression that should be a cubical term is, provisionally, a triple of a level, a cubical type at that level, and a cubical term of that cubical type, so it has Nuprl type type Provisional(cttTerm(G)) where

$$\mathrm{cttTerm}(G) == \mathit{lvl}\!:\!\mathbb{N}_4 \times T\!:\!\mathit{CubicalType}(G, \mathit{lvl}) \times \mathrm{CubicalTerm}(G, T)$$

Since the syntax uses variables, a semantic context X is not merely a cubical set G. It is a triple $\langle G, \mathit{vars}, f\rangle$ where G is a cubical set, vars is a list of variables bound in the context, and f is a function from vars to cttTerm(G). So f assigns to each bound variable a term meaning. We give an inductive definition of $\mathrm{Mng}(X; t)$ to return Provisional(cttTerm(G)) for terms and Provisional(cttType(G)) for types, using our Provisional monad and its bind and return operations.

This gives a semantic definition of the provisos for meaningfulness, but if we can prove that these provisos are always decidable, then we can extract a verified type-checking algorithm from that proof. In many cases, the proviso can be decided by checking α-equality of type expressions, but cubical type theory must also check that certain equations hold in restricted contexts, and that certain restrictions imply other restrictions. These restrictions are formulae of the *face lattice* so a decision procedure for that theory will be needed as well as some other bookkeeping. Because we have added all the Nuprl types and Nuprl terms into the cubical type theory (as discrete types and constant terms), and type checking in Nuprl is not decidable, we will get decidable type checking for only a fragment of the syntax.

Much work remains to done. We need to formalize the semantics of higher inductive types. This can be done using W-types, but will take work. To build the proof assistant for cubical type theory in Nuprl, we need to either derive the verified type-checking algorithm, or, alternatively, use our definition of truth to verify sound rules of inference and tactics to use these rules (this is how Nuprl proofs are constructed).

Many, many details can be found at
`http://www.nuprl.org/wip/Mathematics/cubical!type!theory/`

References

[1] Bickford, M. Formalizing category theory and presheaf models of type theory in nuprl, 2018.

[2] Cohen, C., Coquand, T., Huber, S., and Mörtberg, A. Cubical type theory: A constructive interpretation of the univalence axiom. In *TYPES* (2015).

trm= $\langle 0, 0\rangle$	typ=$\langle 0, 1\rangle$
bdtrm= $\langle 1, 0\rangle$	typfam=$\langle 1, 1\rangle$

type operator	arity
Glue	[typ; trm; typ; trm]
case	[trm; trm; typ; typ]
Pi	[typ; typfam]
Sigma	[typ; typfam]
Path	[typ; trm; trm]
F	[ ]
I	[ ]
decode(i)	[trm]
universe(i)	[ ]
nuprlType(T)	[ ]

term operator	arity
encode(i)	[typ]
pathabs	[typ; bdtrm]
pathapp	[typ; trm; trm]
lambda	[typ; bdtrm]
apply	[trm; typfam; trm]
pair	[trm; typfam; trm]
fst	[typ; typfam; trm]
snd	[typ; typfam; trm]
glue	[trm; trm; trm; trm]
unglue	[typ; trm; typ; trm; trm]
comp	[trm; typfam; bdtrm; trm]
0	[ ]
1	[ ]
max	[trm; trm]
min	[trm; trm]
inv	[trm]
meet	[trm; trm]
join	[trm; trm]
eq0	[trm]
eq1	[trm]
nuprlTerm(T,t)	[ ]

Figure 1: Arities for operators of cubical type theory

2 Appendix

We include Figure 1 to show the operators of cubical type theory that are included in our definition of truth.