Extending Cubical Agda with Internal Parametricity*

Antoine Van Muylder[1], Andrea Vezzosi, Andreas Nuyts[1], and Dominique Devriese[1]

[1]KU Leuven, Belgium

Abstract. Internally parametric type theories are type systems augmented with additional primitives and typing rules allowing the user to prove parametricity statements within the system, without resorting to axioms. We implement such a type system by extending the cubical type theory of Cubical Agda [17] with parametricity primitives proposed by Cavallo and Harper [9]. To assess the implementation, we formalize a general parametricity theorem within the system, which entails a large spectrum of free theorems including a Church encoding for the circle and a straightforward parametric model for System F[1].

A type-polymorphic function is *parametric* if its type argument is merely used for typing, not for computing purposes. Such a parametric function necessarily applies the same algorithm irrespective of the type it is being used at. Reynolds' relational parametricity [15, 13] is a semantic account of this property for, e.g., terms of System F (a.k.a. the second-order polymorphic lambda calculus). Useful information can systematically be extracted by only looking at the type of a parametric function. These facts commonly known as "free theorems" [18] provide, for instance, a formal explanation as to why there are only two functions with type $\forall\alpha.\, \alpha \to \alpha \to \alpha$. Indeed, in a parametric model (where all denotations are parametric), only the first and second polymorphic projections qualify as valid, as they do not inspect the implementation of α.

Enforcing free theorems. Dependent type theory (DTT) has been proven to admit parametric models [16, 6, 12, 3]. Therefore, whenever a free theorem is needed, it can soundly be added as an axiom. In fact, evidence that a given closed term is parametric (w.r.t. a syntactic notion of relation) can even be obtained constructively: this is what *parametricity translations* [11, 1] achieve. However, parametricity is known to be logically independent from plain DTT [7] and this prevents the above meta-theoretical translations to be internalized. Hence, to obtain internal parametricity, novel principles must be added to plain DTT.

Internal parametricity for cubical type theory. Cavallo and Harper (CH) [9] extend cubical type theory [2] with parametricity primitives [5], in a style reminiscent of cubical type theory itself. Proofs of *relatedness* (versus equality) between $a_0, a_1 : A$ are built using functions $p : \mathsf{BI} \to A$ from an abstract *bridge interval* BI, satisfying $p(0) = a_0$, $p(1) = a_1$ definitionally. Such proofs are called *bridges* and written $\lambda^{\mathsf{BI}}\, r.\, p(r) : \mathsf{Bridge}_A\; a_0\; a_1$ (versus paths $\lambda^{\mathsf{I}}\, i.\, p(i) : \mathsf{Path}_A\, a_0\, a_1$). Contrary to path variables, the logic of bridge variables is sub-structural (*affine*): weakening and exchange hold, but not contraction. Concretely, one can eliminate a bridge $\Gamma_1, r : \mathsf{BI}, \Gamma_2 \vdash b : \mathsf{Bridge}_A\; a_0\; a_1$ at a bridge variable r only if r is *fresh* for b, meaning that every free variable appearing in b is in Γ_1 or is a bridge/path variable in Γ_2. This sub-structurality is crucial to formulate the inference rules of the extent and Gel primitives. The purpose of those primitives is to guarantee several *bridge commutation principles*: theorems explaining how the Bridge type former commutes with other type formers. The extent primitive and its rules provide commutation with Π. For non-dependent functions this reads

*Van Muylder/Nuyts hold a PhD/Postdoctoral fellowship of the Research Foundation – Flanders (FWO).

[1]Files, instructions, comparison with other systems: https://github.com/antoinevanmuylder/bridgy-lib.

$\left(\Pi_{(a_0,a_1:A)(\bar{a}:\mathsf{Bridge}\,a_0\,a_1)}\mathsf{Bridge}_B(f_0a_0)(f_1a_1)\right) \simeq \mathsf{Bridge}_{A\to B}\ f_0\ f_1$. The principle is analogous to function extensionality and asserts that functions are related if they map related inputs to related outputs. The Gel primitive and its rules, together with univalence, prove commutation with the universe: $(A_0 \to A_1 \to \mathsf{Type}) \simeq \mathsf{Bridge}_{\mathsf{Type}}\,A_0\,A_1$. This is analogous to univalence and called relativity by CH. Commutation principles make bridges (and paths) behave as structured relations (and isomorphisms, resp.). This is most blatant for types of algebraic structures. Consider the type of "magmas" $\mathsf{Mag} = \Sigma_{M:\mathsf{Type}}\,M \times M \to M$. Commutation with $\Sigma, \mathsf{Type}, \times, \to$ grants a characterization of $\mathsf{Bridge}_{\mathsf{Mag}}\,M_0\,M_1$ (and Path, resp.) as the type of relations (and isomorphisms, resp.) compatible with the binary functions from M_0, M_1. Such structured relations are also known as *logical relations* [10].

Contributions. We implement CH's internally parametric type theory [9] on top of the cubical type theory [8] underlying the Cubical Agda [17] proof assistant. As discussed above, we must be able to generate freshness constraints for bridge variables during typechecking. Our implementation hence reuses the existing affine variable infrastructure of Guarded Cubical Agda [14]. Interestingly and unprecedented in Agda, the extent_β computational rule fires only if a certain argument M satisfies a specific freshness condition. As this condition is not reflected by β-reduction, the rule has to operate on the normal form of M in the worst case scenario. Our current implementation of extent_β is sound but not complete because of this peculiar behaviour. Implementation of Kan operations for $\mathsf{Bridge}, \mathsf{Gel}$ is work in progress as well.

Our long term goals include assessing the precise expressivity of CH's internal parametricity, connecting it to existing alternate formulations (unary, Kripke, etc.) and evaluating its usefulness in practical applications. For now, we have already formalised a (binary) parametricity statement from which a wide range of free theorems ensue. A *native reflexive graph* is by definition a type of vertices G equipped, for any $g_0, g_1 : G$ with a type of edges $G\{g_0, g_1\}$ and an equivalence $\eta^G : G\{g_0, g_1\} \simeq \mathsf{Bridge}_G\,g_0\,g_1$. The type of native reflexive graphs is of course equivalent to Type, but η^G can contain non-trivial information. For instance, Type equipped with relations $A_0 \to A_1 \to \mathsf{Type}$ as edges is native exactly thanks to relativity, and formalizing the relativity theorem was non-trivial. Similarly, a *native relator* F between native reflexive graphs $G, H : \mathsf{Type}$ acts both on vertices $F_{\text{vrt}} : G \to H$ and on edges $F^{g_0,g_1}_{\text{edge}} : G\{g_0, g_1\} \to H\{Fg_0, Fg_1\}$ and the latter action must satisfy $\mathsf{Path}_{\ldots}(F_{\text{bdg}} \circ \eta^G)(\eta^H \circ F_{\text{edge}})$ where $F_{\text{bdg}} = (\lambda q.\ \lambda^{\mathsf{BI}}x.\ F_{\text{vrt}}(qx))$. Parametricity now reads as follows: for any native relator $F : G \to \mathsf{Type}$ and any function $f : \Pi_{x:G}\,Fx$, inputs related by an edge $e : G\{g_0, g_1\}$ result in a proof $(\mathsf{param} : F_{\text{edge}}\,e\,(fg_0)\,(fg_1))$. Bridge commutation principles ensure that native relators abound. For instance the arrow relator $\mathsf{Type} \times \mathsf{Type} \to \mathsf{Type} : A, B \mapsto A \to B$ is native, and using the above param constant it is easy to show that $\left(\Pi_{(X:\mathsf{Type})}X \to X \to X\right) \simeq \mathsf{Bool}$

Less standard is the following Church encoding for the circle [4]: $\left(\Pi_{X_*:\mathsf{Type}_*}\Omega(X_*) \to X\right) \simeq S^1$, where $\mathsf{Type}_* = \Sigma_{X:\mathsf{Type}}X$ and $X_* = (X, x_0)$ and $\Omega(X_*) = \mathsf{Path}_X\,x_0\,x_0$. The above param constant provides a direct proof, granted that $\lambda X_*.\ \Omega(X_*) \to X$ is a native relator. The formal proof of Ω nativeness is ongoing.

Finally we connect Reynolds' statement to ours and describe a shallow embedding of System F that can serve as a parametric model. We wrongly assume $\mathsf{Type} : \mathsf{Type}$ to comply with System F impredicativity and simplify matters. Semantic open types $\alpha_1, ..., \alpha_n : * \models \tau : *$ are defined as native relators $\mathsf{Type}\times\ldots\times\mathsf{Type} \to \mathsf{Type}$. There is a semantic arrow type given by the above arrow relator, and a semantic $\forall$ type as well. Semantic open terms $(\alpha_1, ..., \alpha_n : *)\,|\,(x_1 : \tau_1, ..., x_m : \tau_m) \models t : \tau$ are functions $\Pi_{\theta:\mathsf{Type}^n}\,\Pi_j\tau_j(\theta) \to \tau(\theta)$. Once again, as $\lambda\theta.\ \Pi_j\tau_j(\theta) \to \tau(\theta)$ is a native relator, semantic terms are proven parametric thanks to param.

References

[1] Abhishek Anand and Greg Morrisett. Revisiting Parametricity: Inductives and Uniformity of Propositions. *arXiv:1705.01163 [cs]*, July 2017.

[2] Carlo Angiuli, Robert Harper, et al. Cartesian cubical computational type theory: Constructive reasoning with paths and equalities. In *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[3] Robert Atkey, Neil Ghani, and Patricia Johann. A Relationally Parametric Model of Dependent Type Theory. In *Principles of Programming Languages*, pages 503–515. ACM, 2014.

[4] Steve Awodey, Jonas Frey, and Sam Speight. Impredicative encodings of (higher) inductive types. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 76–85, 2018.

[5] Jean-Philippe Bernardy, Thierry Coquand, and Guilhem Moulin. A presheaf model of parametric type theory. *Electronic Notes in Theoretical Computer Science*, 319:67–82, 2015.

[6] Jean-Philippe Bernardy, Patrik Jansson, and Ross Paterson. Parametricity and dependent types. In *Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*, pages 345–356, 2010.

[7] Auke B Booij, Martín H Escardó, Peter LeFanu Lumsdaine, and Michael Shulman. Parametricity, automorphisms of the universe, and excluded middle. In *22nd International Conference on Types for Proofs and Programs*, 2018.

[8] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: A constructive interpretation of the univalence axiom. In *21st International Conference on Types for Proofs and Programs (TYPES 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[9] Robert Harper and Evan Cavallo. Internal parametricity for cubical type theory. *Logical Methods in Computer Science*, 17, 2021.

[10] Claudio Hermida, Uday S Reddy, and Edmund P Robinson. Logical relations and parametricity–a Reynolds programme for category theory and programming languages. *Electronic Notes in Theoretical Computer Science*, 303:149–180, 2014.

[11] Chantal Keller and Marc Lasson. Parametricity in an impredicative sort. In *Computer Science Logic (CSL'12)-26th International Workshop/21st Annual Conference of the EACSL*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[12] Neelakantan R. Krishnaswami and Derek Dreyer. Internalizing relational parametricity in the extensional calculus of constructions. In *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, pages 432–451, 2013.

[13] QingMing Ma and John C Reynolds. Types, abstraction, and parametric polymorphism, part 2. In *International Conference on Mathematical Foundations of Programming Semantics*, pages 1–40. Springer, 1991.

[14] Rasmus Ejlers Møgelberg and Niccolò Veltri. Bisimulation as path type for guarded recursive types. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–29, 2019.

[15] John C Reynolds. Types, abstraction and parametric polymorphism. In *IFIP congress*, volume 83, 1983.

[16] Izumi Takeuti. The theory of parametricity in lambda cube. Technical report 1217, Kyoto University, 2001.

[17] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical agda: A dependently typed programming language with univalence and higher inductive types. *Journal of Functional Programming*, 31, 2021.

[18] Philip Wadler. Theorems for free! In *Proceedings of the fourth international conference on Functional programming languages and computer architecture*, pages 347–359, 1989.