Towards a Parametricity Logic for Gradual Type Theory

March 25, 2025

1 Introduction

It is well known that binary parametricity is a useful principle for reasoning about programs in a polymorphic language[6][8]. don't just state, provide elaboration on why it is useful. examples of parametricity proofs in different domains (ex security/noninterference) Parametricity in the presence of a dynamic type has proven tricky to establish and existing definitions break full abstraction relative to System F[2]. Even so, we claim that the *Type World Logical Relation* method used to establish parametricity in the presence of a dynamic type captures relevant relational reasoning principleslike what?. To demonstrate this, we provide a parametricity logic[5] to reason about data abstraction. This logic is sound relative to our operational semantics? and the model is constructed in Iris.

2 Languages

2.1 Poly G

[3] page 274.

2.2 Poly C

[3] page 282

2.3 Poly G to Poly C

[3] page 283

2.4 Poly C to OSum

[3] page 292 syntax for OSum
contract translation?

3 Parametricity

Parametricity is formulated and proved via a family of proof methods titled *logical relations*. The typical structure of a binary logical relation proof method consists of an inductive assignment of types to binary relations on values. These relations are meant to capture when two values of a type should be considered related. The prototypical example of parametricity is given by a Reynolds-style binary logical relation defined on the types of pure System F.

It is well known that parametricity changes in the presence of effects. To accomidate effects such as higher order store and recursion, the logical relation is enhanced to a step indexed Kripke logical relation. The logical relation used to establish parametricity for a language with a dynamic type is based off of that of higher order store. In order establish parametricity in the presence of a dynamic type....

3.1 Binary Logical Relation for OSum

Untyped, using de Bruijn indicies, based on [7]

$$\begin{aligned}
\mathcal{V}[\![X]\!]\rho &= \rho(X) \\
\mathcal{V}[\![Unit]\!]\rho &= \{(*,*)\} \\
\mathcal{V}[\![Bool]\!]\rho &= \{(true,true),(false,false)\} \\
\mathcal{V}[\![OSum]\!]\rho &= \{(inj_{\sigma_1}v_1, inj_{\sigma_2}v_2) \mid (\sigma_1,\sigma_2)\mapsto R \ \wedge\ (v_1,v_2)\in \triangleright R\} \\
\mathcal{V}[\![Case\ X]\!]\rho &= \{(\sigma_1,\sigma_2) \mid (\sigma_1,\sigma_2)\mapsto \mathcal{V}[\![X]\!]\rho\} \\
\mathcal{V}[\![X\times Y]\!]\rho &= \{((v_1,v_2),(v_3,v_4)) \mid (v_1,v_3)\in\mathcal{V}[\![X]\!]\rho \ \wedge\ (v_2,v_4)\in\mathcal{V}[\![Y]\!]\rho\} \\
\mathcal{V}[\![X\to Y]\!]\rho &= \{(f,g) \mid \Box(\forall xy, (x,y)\in\mathcal{V}[\![X]\!]\rho \implies (fx,gy)\in\mathcal{E}[\![Y]\!]\rho)\} \\
\mathcal{V}[\![\forall X.Y]\!]\rho &= \{(f,g) \mid \Box(\forall R, (f[X],g[X])\in\mathcal{E}[\![Y]\!](\rho, X\mapsto R))\} \\
\mathcal{V}[\![\exists X.Y]\!]\rho &= \{(pack\ v_1, pack\ v_2) \mid \exists R, (v_1,v_2)\in\mathcal{E}[\![Y]\!](\rho, X\mapsto R)\ \}
\end{aligned}$$

3.2 Binary Logical Relation for PolyG

4 Logic

5 Examples

By switching from a RLR to a TWLR[2], we have lost full abstraction. But do we care? Do we loose the ability to reason about contextual equivalences we'd like to show?

5.1 Simple, Pure examples

swap (see theorem 273 of [3])

5.2 Data Abstraction

5.3 Universal Properties

5.4 Local Store

[1] [4] An example of relating programs which use different numbers of allocations.

$$let\ \sigma = new\ \tau; M \cong_{ctx} M$$

where σ is not used in M.

References

[1] N. Benton and B. Leperchey. Relational reasoning in a nominal semantics for storage. In *Proceedings of the 7th International Conference on Typed Lambda Calculi and Applications*, TLCA'05, pages 86–101, Nara, Japan. Springer-Verlag, 2005. ISBN: 3540255931. DOI: 10.1007/11417170_8. URL: https://doi.org/10.1007/11417170_8.

[2] D. Devriese, M. Patrignani, and F. Piessens. Two parametricities versus three universal types. *ACM Trans. Program. Lang. Syst.*, 44(4), Sept. 2022. ISSN: 0164-0925. DOI: 10.1145/3539657. URL: https://doi.org/10.1145/3539657.

[3] M. S. New, M. Felleisen, M. Wand, D. Licata, R. Garcia, and P. Thiemann. *A Semantic Foundation for Sound Gradual Typing*. PhD thesis, USA, 2020. ISBN: 9798557038935. AAI28263083.

[4] P. W. O'Hearn and R. D. Tennent. Parametricity and local variables. *J. ACM*, 42(3):658–709, May 1995. ISSN: 0004-5411. DOI: 10.1145/210346.210425. URL: https://doi.org/10.1145/210346.210425.

[5] G. D. Plotkin and M. Abadi. A logic for parametric polymorphism. In *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, TLCA '93, pages 361–375, Berlin, Heidelberg. Springer-Verlag, 1993. ISBN: 3540565175.

[6] J. C. Reynols. Types, Abstraction and Parametric Polymorphism. In R. Mason, editor, *Information Processing 83*, volume 9 of *IFIP Congress Series*, pages 513–523, Amsterdam, The Netherlands. Elsevier Science Publishers B.V., 1983.

[7] A. Timany, R. Krebbers, D. Dreyer, and L. Birkedal. A logical approach to type soundness. *J. ACM*, 71(6), Nov. 2024. ISSN: 0004-5411. DOI: 10.1145/3676954. URL: https://doi.org/10.1145/3676954.

[8] P. Wadler. Theorems for free! In *Proceedings of the fourth International Ponference on Functional Programming languages and computer architecture - FPCA '89*, volume 112, pages 347–359, New York, New York, USA. ACM Press, 1989. ISBN: 0-89791-328-0. DOI: 10.1145/99370.99404.