UNIVERSITY OF CAMBRIDGE
Department of Computer
Science and Technology

May 22, 2025

Efficient coinductives through state-machine corecursors

William Sørensen

Gonville & Caius College

Submitted in partial fulfilment of the requirements for the
Computer Science Tripos, Part III

Declaration

I, William Sørensen of Gonville & Caius College, being a candidate for the course, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose. **Signed**:
Date: May 22, 2025

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit.

Acknowledgements

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit.

Contents

Chapter 1

Introduction

1.1 Dependent type theory

1.2 Polynomial functors

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et.

1.2.1 Lean formalization

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et.

Chapter 2

Preparation

Chapter 3

Implementation

3.1 The ABI Type

3.2 Stream implementation

3.3 Expanding the progressive approximation theory

During the pheasability assesment I noticed that, in the current formalised theory of polynomials, the statement wouldn't even type-check. This stemmed from a problem with the corecursive principle for the M type in the old implementation. $\text{corec} : \{\alpha : \text{TypeVec}.\{\mathcal{U}\}n\} \to \{\beta : \text{Type}\ \mathcal{U}\} \to (g : \beta \to P(\alpha ::: \beta)) \to \beta \to M\alpha$[1]. The problem here is that both α and β have to both reside in $\mathcal{U}$. Solving this is done through the next two sections.

3.3.1 Universe lifting of polynomial functors.

The main problem caused here comes from the fact that lean isnt cummulative. This means it is impossible to express a universe hetrogouns typevector. In other words $\alpha ::: \beta$ is only typable if $\alpha : \text{TypeVec}.\{\mathcal{U}\}n$ and $\beta : \text{Type}\ \mathcal{U}$. The natural way of solving this is using the supremum in universe levels you get from $\text{ULift} : \text{Type}\ \mathcal{U} \to \text{Type}\ (\max \mathcal{U}\mathcal{V})$. This means we can have $\beta : \text{Type}\ \mathcal{U}$ and $\alpha : \text{Type}\ \mathcal{V}$, then ulift both of them to a common universe $\text{ULift}\ \alpha ::: \text{ULift}\ \beta : \text{TypeVec}.\{\max \mathcal{U}\mathcal{V}\}(n+1)$[2].

Noticable the next hurdle we encounter is that PFunctors are restricted to a universe level. Recall the definition from Section 1.2.1. Observe how for a $\text{MvPfunctor}.\{\mathcal{U}\}n$, we require that both the head and child reside in $\mathcal{U}$. This will also cause problems, as looking back at the definition of the corecursor, we will require P to be able to accept $\text{ULift}\ \alpha ::: \text{ULift}\ \beta$. If we do not add the ability to lift P, the unifier will force $\mathcal{U} = \mathcal{V}$, thereby invalidating all the work we did in the previous section. Luckily lifting a PFunctor is relatively easy. We define it as

[1] https://github.com/leanprover-community/mathlib4/blob/7a60b315c7441b56020c4948c4be7b54c222247b/Mathlib/Data/PFunctor/Multivariate/M.lean#L152-L154

[2] Note we overload ULift as a notation to refer to lifting TypeVecs as well

ULift $P \triangleq \langle \text{ULift } P.1, \lambda x \mapsto \text{ULift } (P.2x) \rangle$. This works and now we can move on to our goal[3].

3.3.2 Generalizing the corecursor

Now with all the work in the previous section, by generalizing corec'[4], we can define corecU : $\{\alpha : \text{TypeVec}.\{\mathcal{U}\}n\} \to \{\beta : \text{Type } \mathcal{V}\} \to (g : \beta \to \text{ULift } P(\text{ULift } \alpha ::: \text{ULift } \beta)) \to \beta \to M.\{\mathcal{U}\}\alpha$. Notably we are able to fit the object into $\mathcal{U}$ (as opposed to in the SME).

The expected diagram using corecU and dest commutes.

3.4 State machine encoding

Noting the definition of corecU, one might wonder if you could define M from first principles for this. The problem one encounters is one of universes. As seen in the definition above, if one were to define a type whos construcor is directly the corecU definiton, it would hold a β : Type $\mathcal{V}$. This then forces the object to reside in $\text{Type} \max \mathcal{U} (\mathcal{V} + 1)$. This is a problem as one loses most closure results as you will be lifting more and more. The main beinifit from this is the performance aspect though. This will be seen in Section 4.1. We will henceforth refer to the datatype SME.PreM.

3.5 Proving the equvilence

3.6 Cofix implementation

[3]TODO: Speak with JV / W to see if this might be done in the lit, Ex : Locally presentable and accessable categories Adameck roshiski

[4]Done in PR NUMBER

Chapter 4

Evaluation

4.1 Performance between SME and PA

Chapter 5

Conclusions

Chapter 6

Appendicies