Efficent coinductives through state-machine corecursors

Supervisor: Alex Keizer

Marking supervisor: Tobias Grosser OR Jamie Vicary

Word count: 934

1 Work to be undertaken

Previous work has shown how coinductives can be encoded in lean as quotients of polynomial functors (QPFs). These are highly expressive but accessing cofixed points to a depth n takes $\mathcal{O}(n^2)$ time, which compounds when you map data m times to $\mathcal{O}(n^{2^m})$, this is a problem when using lean as a general purpose programming language. Previous work has focused on trying to achieve this performance within the same universe. An alternative approach is using the state-machine implementation of a corecursor. This is a higher universe object which uses the corecursor directly. Showing an equivalence between these two representations layes the groundwork for performance gains through future lean features. With these we get nice computational behaviour, meaning the entire expression collapses to $\mathcal{O}(n)$. A demonstration of the possible gains can be seen in Figure 1, where the x-axis is the index into the array and the y-axis is a ns (scaled to 1e8) duration

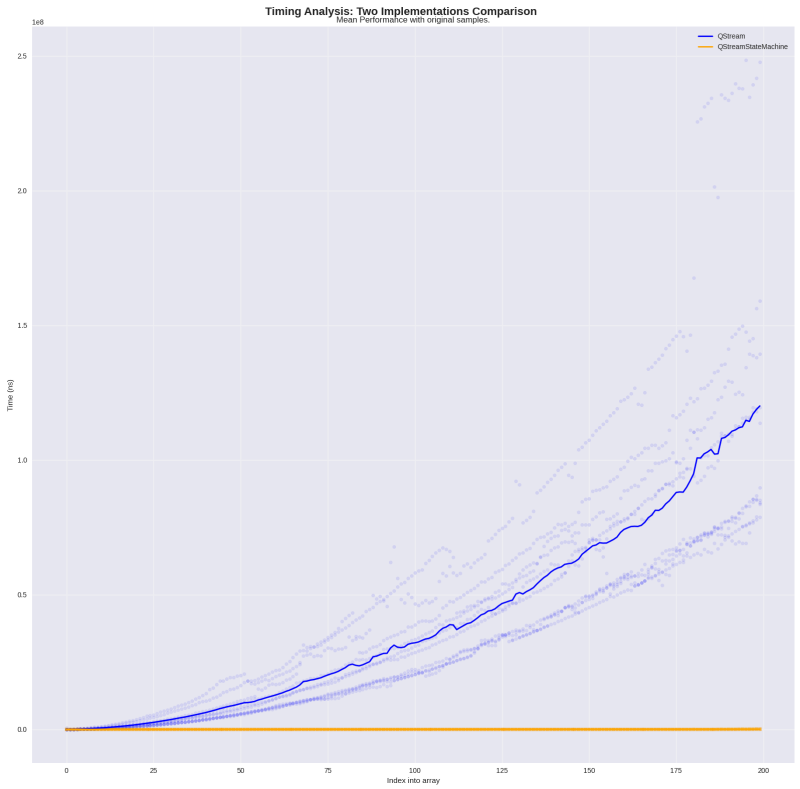


Figure 1: Graph plotting current performance v possible gains

2 Starting point

I have worked with QPFs on the meta-programming side for an internship between my Part Ia and Part Ib. During this I learnt of the basics of polynomial functors, and TypeVecs. This means I am aware of what the underlying structures are when it comes to the raw implementation. I have also done a pheasability assessment of the project by seeing how the current polynomials respond to universe levels. This lead to me making 2 pull requests (#28095, #28279) to mathlib on TypeVec in preparation for my project. I also tried to merge (#28112) an implementation of computable Shrink to mathlib that later got reverted when it was found to be absurd.

There are more minor refactoring pull requests towards the mathlib reposity which don't change any behaviour but in general all of these can be found on this link. I Include these for completeness and transparency.

3 Core project

This project will relate to implementing the state-machine representation of coinductives along with their theory. The goal will be showing the equivalence between this and the progressive approximation representation. Initially this will showing the equivalence between a `Stream` the two representations, then building up the the general cofix structure.

4 Substance

There are a few structures that will be worked with during this project.

An attempt was made to make `Shrink` computable by using `unsafeCasts` but these had to be reverted. For now the exact type is undecided as there is a RFC by the Lean FRO adding repr types for this purpose.

4.1 `M`

The `M` type is the name given to the terminal coalgebra of polynomial functors; the possibly infinitely deep trees. These are generated by progressive approximation where earlier trees must "agree" with the later ones. Agreement is given by them being the same up to the previous depth. A visual example is given in Figure 2. We can have approximations for any n, thereby letting the trees take any depth including infinite depth. Trees can be terminated by having no children as one might expect.

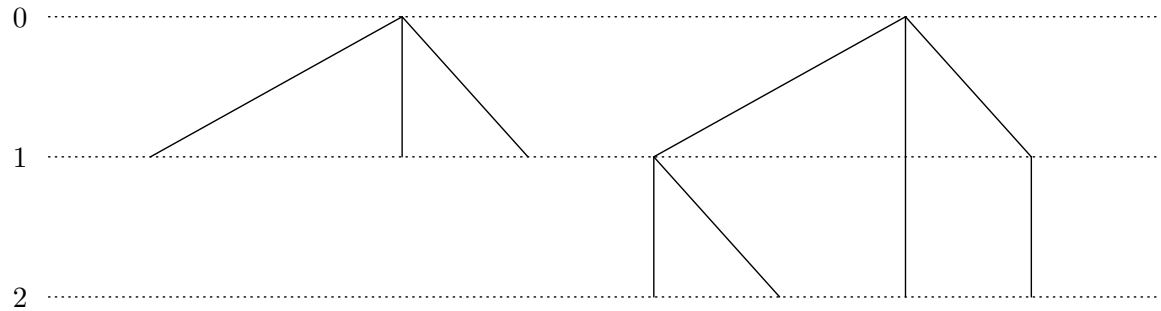


Figure 2: Agreement of two trees of height 1 and 2

The current QPF implementation has 2 `M` types; univariate and multivariate implementations. These both have the undesired computational behaviour. `M` types are polynomial

4.2 `Cofix`

`Cofix` is the terminal coalgebra in QPFs, (possibly) infinitely big quotiented trees. This is the slightly concerning part of the project and by far the highest risk section as working with `Quot` in Lean is a painful experience.

4.3 `Shrink`

`Shrink` is temporarily the choice used for doing the ABI translation between the two implementations. Given that the two types are equivalent then we can non-computably extract a model in the desired universe. Given two types $\alpha : \text{Type } u$ and $\beta : \text{Type } v$ for which an isomorphism exists, we can construct the type $\text{Shrink } \alpha\beta : \text{Type } v$ for which both diagrams in Figure 3 commute.

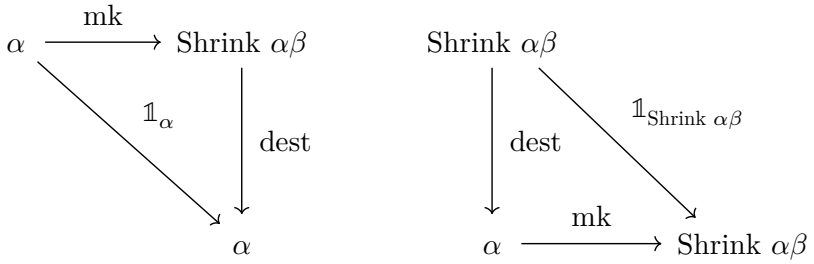
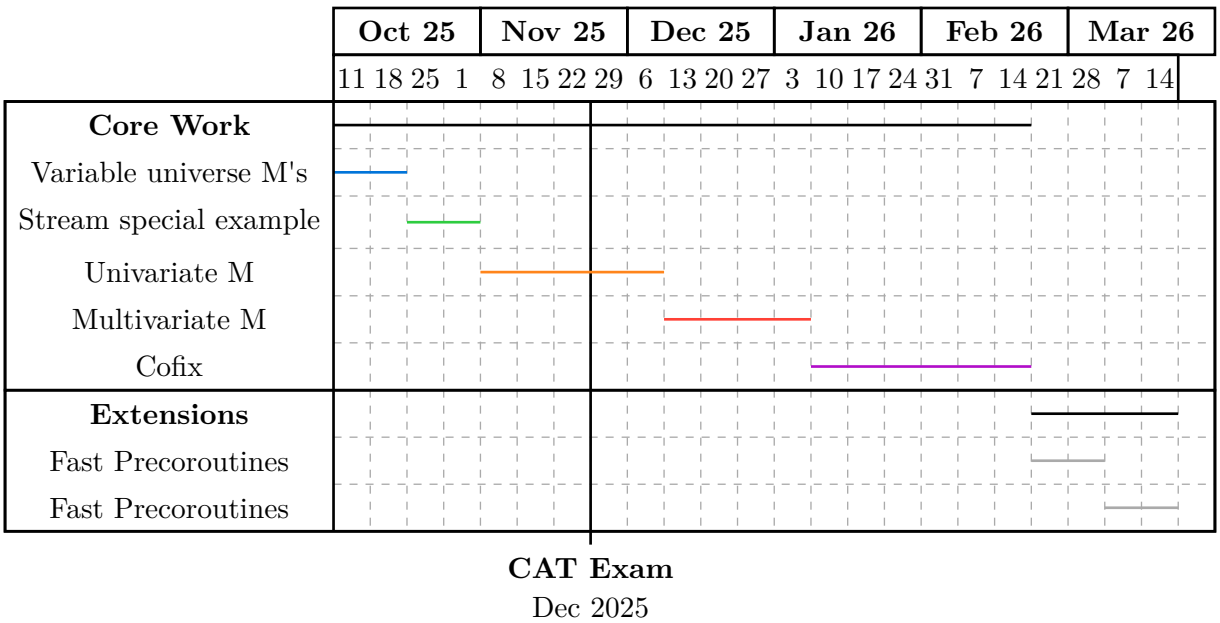


Figure 3: Operations on Shrink

5 Evaluation

The success of this project can be given by how close to the theorised performance we can get to. The goal would be getting to the same order or magnitude.

6 Core timeline



The plan for work would be divided into a few different stages.

6.1 Variable universe `Ms` (2025-10-11 2w 2025-10-24)

To begin, the pull requests created from before the start of the project will have to be completed and merged into `mathlib`.

6.2 `Stream` special example (2025-10-25 2w 2025-11-7)

An early step would be to familiarise myself with using the bisimilarity features given by the `M` type. This asses pheasability to prove equivalences of two `M` types with a simple functor.

6.3 Univariate `M` (2025-11-8 2w 2025-11-21 + (CAT exam) + 2025-12-05 3w 2025-12-23)

After a special example I would move over to the univariate example. This will be much easier than the multivariate case as I don't have to suffer with `Typevec`s.

6.4 Multivariate `M` (2025-12-26 4w 2026-01-23)

This will be the next natural step. Will me much harder than the univariate.

6.5 `Cofix` (2026-01-24 6w 2026-03-06)

Finally, it has to be proven for `Cofix`. This will be hard as I will have to suffer with Quot which is really concerning to work with.

7 Extensions

7.1 `Shrink`ing the representations (2026-03-07 2w 2026-03-20)

This is a research heavy component of the project. This involves finding a sound way of implementing Section 4.3. This will be novel work.

7.2 A fast implementation of `Precoroutines` (2026-03-21 2w 2026-04-04)

Precoroutines are a type Alex Keizer is interested in. They generalise interaction trees and other similar data-types useful for denotational purposes. These leverage some of the powers of QPFs. This should come for free from the prior the core.

8 Resources

Access to the restricted side of the lab is needed for working with the further team.