Efficient Coinductives through State-Machine Corecursors

Optimising the terminal F-coalgebra in Set for computational behaviour in Lean4

2025-02-23

William Sørensen

Polynomials (background)

- A head H,
- child families c_h in $h : H$.
- Morally, a collection of constructors H, and data at each constructor c_h.
- For now we will not consider “recursive” (inductive) data structures, rather I will have this be a variable we take the fix point in.
- Examples:
  - natural numbers `NatF(X) := 1 + X`,
  - lists `ListF(A, X) := 1 + A * X`,
  - streams `StreamF(A, X) := A * X`.

$$(h : H) \times c_h \to \alpha \equiv \sum_{h:H} \alpha^{c_h}$$

See (nLab authors, 2026) and (Neel Krishnaswami, 2026)

Coinductives (background)

- Unbounded tree of data.
- Greatest cofixpoint of a polynomial.
- Morally, the process of putting the word “lazy” in front of a functor,
  - on lists it sends them to lazy lists.
- Have a corecursion principle rather than an recursion principle,
  - instead of `rec : (f a → a) → fix f → a`,
  - we have `corec : (a → f a) → a → cofix f`.
- On our examples and their correspondencens,
  - `cofix NatF`, natural numbers union infinity,
  - `cofix ListF(A)`, lists of `A`s with arbitrary length union streams of `A`,
  - `cofix StreamF(A)`, infinite streams of `A`s.

See (Avigad et al., 2019)

Perfomance of coinductives (core)

- Previous work has demonstrated implementation of coinductives.
- Computational behaviour was not a concern.

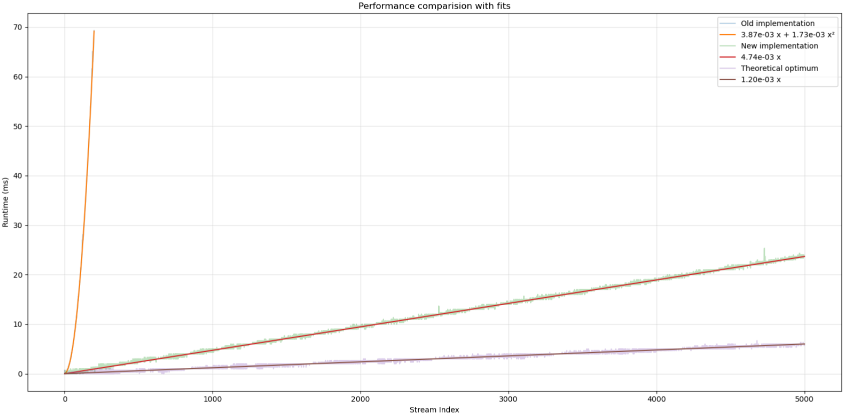

Figure 1: Perfomance of the old, new, and a perfect implementation

ABI Type (extention)

- Given $\alpha : \text{Type}_{\mathcal{U}}$, $\beta : \text{Type}_{\mathcal{V}}$ and $\text{eq} : \alpha \simeq \beta$.
- Univalence gives uniqueness.
- Adds a form of “weak-univalence”.
- Allows for a zero cost universe transliteration.
- Symetric form of `Shrink`.

$$\begin{tikzcd}
\alpha \arrow[rr, "\mathrm{mk}"] \arrow[ddrr, "\mathbb{1}_{\alpha}"'] & & \text{Shrink } \alpha\beta \arrow[dd, "\mathrm{dest}"'] \arrow[dr, "\mathbb{1}_{\text{Shrink } \alpha\beta}"] & \\
& & & \text{Shrink } \alpha\beta \\
& & \alpha \arrow[ur, "\mathrm{mk}"] &
\end{tikzcd}$$

Figure 2: Operations on Shrink

$$\begin{tikzcd}
& & \beta \arrow[dd, "\mathrm{mkB}"'] \arrow[ddrr, "\mathbb{1}_{\beta}"] & & \\
& & & & \\
\alpha \arrow[uurr, equal, "\mathrm{eq}"] \arrow[rr, "\mathrm{mkA}"] \arrow[ddrr, "\mathbb{1}_{\alpha}"'] & & \text{ABI } \alpha\beta \arrow[rr, "\mathrm{destB}"] \arrow[dd, "\mathrm{destA}"'] \arrow[dr, "\mathbb{1}_{\text{ABI } \alpha\beta}"] & & \beta \arrow[dl, "\mathrm{mkB}"'] \arrow[ddll, equal, bend left, "\mathrm{eq}"] \\
& & & \text{ABI } \alpha\beta & \\
& & \alpha \arrow[ur, "\mathrm{mkA}"] & &
\end{tikzcd}$$

Figure 3: Operations on ABI

Interaction Trees (extention)

- Free monad with non-termination as an effect.

```lean
coinductive ITree
    (E : Type u → Type u) (R : Type v) :=
  | tau : ITree E R → ITree E R
  | ret : R → ITree E R
  | vis {A : Type u} (e : E A) (k : A → ITree E R)
```

- ITerm E forms a monad,
  - With a method trigger : $EV \to \text{ITree}\ E\ V$,
  - and iter : $(A \to \text{ITree}\ E(A + B)) \to A \to \text{ITree}\ E\ B$.
- And an equivelence relation $A \approx B$,
  - satisfying $A \approx B \leftrightarrow$ `tau` $A \approx B$.

Project statistics

```
❯ nix-shell -p cloc --run "cloc . --exclude-dir=.lake"
      95 text files.
      87 unique files.
      14 files ignored.

github.com/AlDanial/cloc v 2.06  T=0.09 s (925.5 files/s, 99584.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Lean                            54           1174            232           5869
Typst                           20            304            119            994
SVG                              1              0              0            171
JSON                             3              0              0            125
TeX                              1              4              0             76
YAML                             3              8              7             69
Markdown                         2             27              0             68
Python                           2             22             18             57
Nix                              1              2              0             15
-------------------------------------------------------------------------------
SUM:                            87           1541            376           7444
-------------------------------------------------------------------------------
```

(excluding mathlib PRs)

Bibliography

Avigad, J., Carneiro, M., & Hudon, S. (2019). Data Types as Quotients of Polynomial Functors. In J. Harrison, J. O'Leary, & A. Tolmach (Eds.), *10th International Conference on Interactive Theorem Proving (ITP 2019): Vol. 141. 10th International Conference on Interactive Theorem Proving (ITP 2019).* https://doi.org/10.4230/LIPIcs.ITP.2019.6

Neel Krishnaswami. (2026, February). *Algebraic Techniques for Programming 25-26.*

nLab authors. (2026, January). *polynomial functor.*

Yang, Z., & Wu, N. (2022). Fantastic Morphisms and Where to Find Them. In E. Komendantskaya (Ed.), *Mathematics of Program Construction: Mathematics of Program Construction.*

Further work

Futumorphic productivity (extention)

- A futomorphism is a morphism `(c → f (Free f c)) → c → cofix f`.
-

See (Yang & Wu, 2022)

CTree (extention)

CTree