Cost-Aware Logical Framework (calf) for Cost and Behavior in Type Theory

Calf (Niu et al., POPL '22) **Decalf** (Grodin et al., POPL '24)

Dependent type theory for *synthetic* cost analysis via call-by-push-value.

- **Cost as an effect:** $\mathsf{charge}^c_X(e)$ prefixes c units of cost
- **Phase distinction:** modalities isolate behavior from cost (erasure): $\bigcirc(\mathsf{charge}^c_X(e) = e)$
  e.g. $\bigcirc(\texttt{mergeSort} = \texttt{insertionSort})$ is provable, but it is not the case in general $\texttt{mergeSort} = \texttt{insertionSort}$ due to cost differences
- **Cost bound:** prove sequential and parallel cost bounds
  e.g. $\mathsf{IsBounded}(\texttt{mergeSort}(l), \lceil \log |l| \rceil \cdot |l|, 2 \cdot |l| + \lceil \log |l| \rceil)$
- **Directed type theory:** program inequality $e \leq e'$ compares cost and behavior simultaneously
  e.g. $\texttt{mergeSort}(l) \leq \mathsf{charge}^{|l|^2}(\mathsf{return}(\texttt{sorted}(l)))$

Metatheory for Calf via Synthetic Tait Computability, Mechanized

Meta-theoretical properties for **Calf**: **Canonicity** (Li & Harper, 2025):

$$e : \mathsf{F}(\mathsf{nat}) \implies e = \mathsf{charge}^c(\mathsf{return}(\mathsf{suc}^n(\mathsf{zero})))$$

Proved via **Synthetic Tait Computability** - a synthetic approach to proof-relevant logical relations:

- Two-world Kripke model: • cost-sensitive, ∘ behavioral
- Each type carries a *cost algebra* (carrier + charge + erasure)

Mechanized in Istari proof assistant (Li, Yao & Harper, CPP '26):

- Extensional TT — equality reflection, strong extensionality principles, no transport boilerplate
- One of the first mechanized, categorical gluing style logical relations proof for type theory

Cost and Behavior in Type Theory: Modularity, Metatheory, and Mechanization

Runming Li j.w.w. Harrison Grodin Yue Yao Lukas Kebuladze Andrew Zhou Robert Harper
Carnegie Mellon University

Synthetic Modular Verification via Hoare's Abstraction Functions

Abstraction functions provide an informal way to specify correctness of data structures modularly:

Analytical abstraction functions:

$$\begin{pmatrix} X_\bullet \\ \downarrow \chi \\ X_\circ \end{pmatrix}$$

concrete implementation

↓ abstraction function

abstract specification

e.g. to verify `enqueue`, we prove:

$$\begin{array}{ccc} \texttt{List}E \times \texttt{List}E & \xrightarrow{\text{enqueue } e} & \texttt{List}E \times \texttt{List}E \\ \downarrow \texttt{revAppend} & & \downarrow \texttt{revAppend} \\ \texttt{List}E & \xrightarrow[\text{enqueue } e]{} & \texttt{List}E \end{array}$$

Synthetic abstraction functions using modalities (Grodin, Li & Harper, POPL '26):

$\texttt{Glue}(X_\bullet, X_\circ, \chi)$ as a type in univalent type theory

- abstract modality $\bigcirc$ isolates specification
  $\bigcirc\, \texttt{Glue}(X_\bullet, X_\circ, \chi) = X_\circ$
- concrete modality $\bullet$ isolates implementation
  $\bullet\, \texttt{Glue}(X_\bullet, X_\circ, \chi) = X_\bullet$

e.g. to specify queues that behave correctly:

$$\Sigma_{q:\texttt{Queue}}\, \bigcirc(q = \texttt{referenceQueue})$$

Retains the possibility for q to have different (e.g. better) cost behavior than `referenceQueue`.

Case Studies: Verified Algorithms and Data Structures in `agda-calf`

Variants of **Calf** have been implemented in the Agda theorem prover with different foundations:

Paper	Agda variant	Reasoning principle
Niu et al., POPL '22; Grodin et al., POPL '24	Extensional Agda	Set-level (UIP)
Grodin, Li & Harper, POPL '26	Cubical Agda	Univalent (HoTT)

Over the years, our group has built up a library of verified cost and behavior analyses in `agda-calf`, including but not limited to the following:

Algorithm	Bound	Technique	Reference
Merge sort	$\lceil \log \lvert l \rvert \rceil \cdot \lvert l \rvert$ work, $2 \cdot \lvert l \rvert + \lceil \log \lvert l \rvert \rceil$ span	Recurrences	Niu et al., POPL '22
Insertion sort	$\lvert l \rvert^2$	Recurrences	Niu et al., POPL '22
Red-black trees	$1 + 2 \cdot \lvert h_1 - h_2 \rvert$ (join)	Intrinsic types	Li, Grodin & Harper, '23
Dynamic arrays	$O(1)$ amortized	Coinduction	Grodin & Harper, MFPS '24
Quicksort (prob.)	$\lvert l \rvert^2$ worst case	Directed type theory	Grodin et al., POPL '24
Random sublist	binomial $\lvert l \rvert$	Directed type theory	Grodin et al., POPL '24
Parallel prefix sum	$14 \cdot \lvert l \rvert$ work, $4 \cdot \lceil \log \lvert l \rvert \rceil$ span	Recurrences	Zhou, '25
Splay trees	$(3\lfloor \log n \rfloor + 1) + \frac{n}{m}\lfloor \log n \rfloor$ amortized	Lax homomorphism	Kebuladze, POPL SRC '26
Batched queues	$O(1)$ amortized	Modular abstr. fns	Grodin, Li & Harper, POPL '26