ICFP 2020

# Denotational Recurrence Extraction for Amortized Analysis


**Joe Cutler** Dan Licata Norman Danner

Wesleyan University

# Informal Recurrence Extraction

$$\begin{aligned}
&\texttt{mergeSort} : \texttt{int list} \to \texttt{int list}\\
&\texttt{mergeSort}\ [\,] = [\,]\\
&\texttt{mergeSort}\ \texttt{xs} = \texttt{let}\ (\texttt{l}, \texttt{r}) = \texttt{split}\ \texttt{xs}\ \texttt{in}\\
&\qquad\qquad\qquad \texttt{merge}\ (\texttt{mergeSort}\ \texttt{l}, \texttt{mergeSort}\ \texttt{r})
\end{aligned}$$

$$T_{\texttt{mergeSort}}(n) = T_{\texttt{split}}(n) + T_{\texttt{merge}}(n) + 2T_{\texttt{mergeSort}}\left(\frac{n}{2}\right)$$

**How do we make this informal process formal?**

# Formally Extracting Recurrences

**Source Language**

$$\Gamma \vdash M : A$$

→

**Recurrence Language** $\lambda^{\mathbb{C}}$

$$\langle\!\langle \Gamma \rangle\!\rangle \vdash \|M\| : \mathbb{C} \times \langle\!\langle A \rangle\!\rangle$$

Cost to evaluate M

Result of running M
(size or use-cost)

Functions get translated to
recurrences in the traditional sense

$$\langle\!\langle A \to B \rangle\!\rangle = \langle\!\langle A \rangle\!\rangle \to \mathbb{C} \times \langle\!\langle B \rangle\!\rangle$$

$$\langle\!\langle A \times B \rangle\!\rangle = \langle\!\langle A \rangle\!\rangle \times \langle\!\langle B \rangle\!\rangle$$

$$\dots$$

$$\|\lambda x.M\| = (0, \lambda x.\, \|M\|)$$

$$\|(M, N)\| = (\pi_1 \|M\| + \pi_1 \|N\|, (\pi_2 \|M\|, \pi_2 \|N\|))$$

Monadic translation
to recurrence language
into writer monad

# Proving Extraction Correctness

**Cost-Indexed Big-Step Operational Semantics**

$$M \downarrow^{n} v$$

**Bounding Theorem**

For $M : A$ if $M \downarrow^{n} v$,

then $n \leq \pi_1 \, \|M\|$

and $v \sqsubseteq^{A}_{\texttt{val}} \pi_2 \, \|M\|$

**Size-Abstraction Semantics**

$$\lambda^{\mathbb{C}} \xrightarrow{[\![\cdot]\!]} \mathbf{Poset}$$

# Prior Work & Limitations

## This technique works for:

| STLC | Inductive Types | PCF | Let-Polymorphism |
| --- | --- | --- | --- |
| [PLPV '13] | [ICFP '15] | [POPL '20] | [arxiv:2002.07262] |

## but it can't handle…

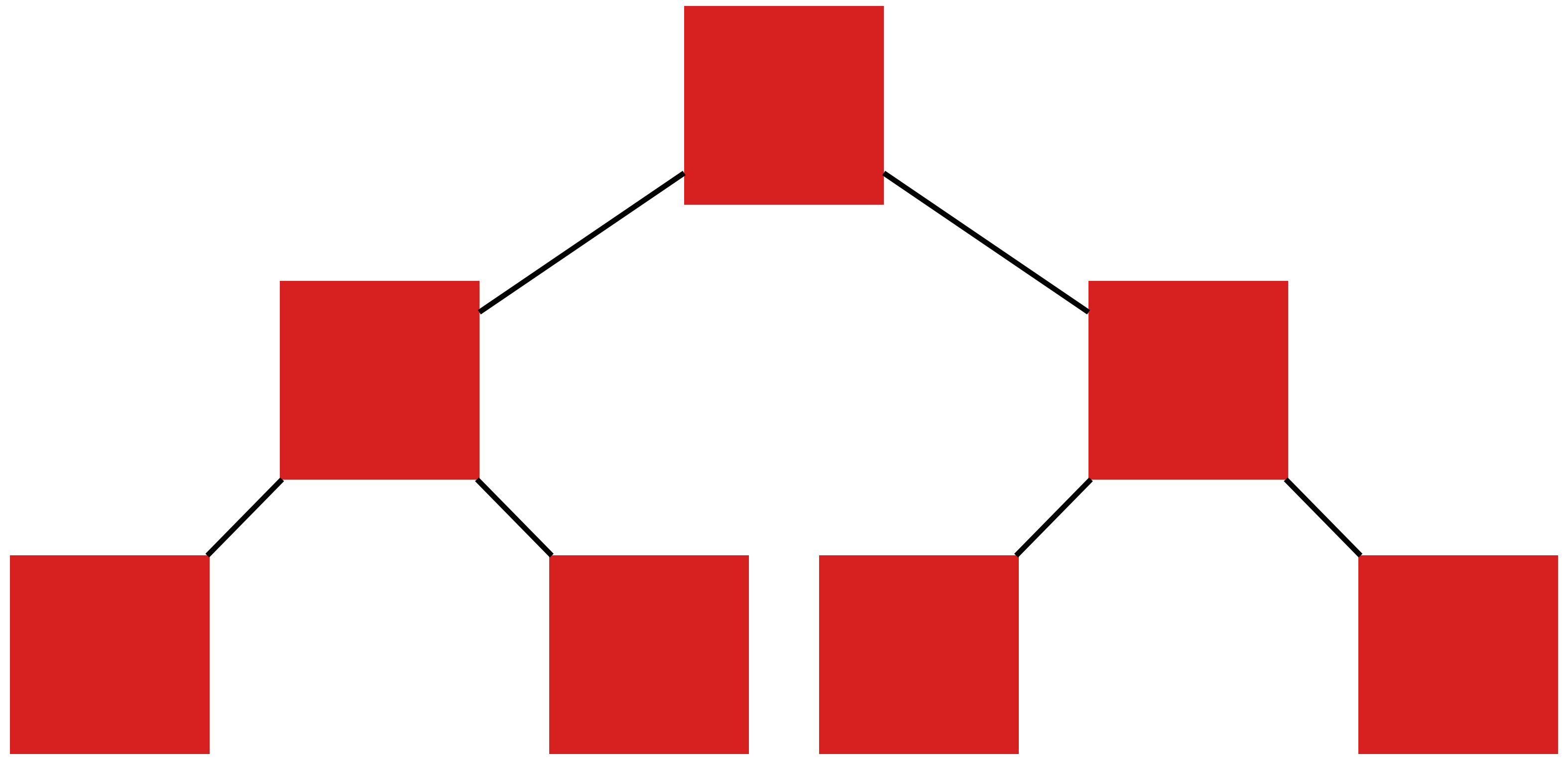
**Amortized Analysis**

# This Work:
# Amortized Analysis by
# Formal Recurrence Extraction

## Examples in the Paper:

### Binary Counter


### Splay Trees



# Binary Counter

```
type bit = 0 | 1
```

Cost model:
Cons operations incur 1 cost

$\texttt{inc} : \texttt{bit list} \to \texttt{bit list}$

$\texttt{inc}\ [\,] = [1]$

$\texttt{inc}\ (0 :: \texttt{bs}) = 1 :: \texttt{bs}$

$\texttt{inc}\ (1 :: \texttt{bs}) = 0 :: (\texttt{inc bs})$

$$T_{\texttt{inc}}(0) = 1$$
$$T_{\texttt{inc}}(n) = \max(1, 1 + T_{\texttt{inc}}(n-1))$$

$\texttt{set} : \texttt{nat} \to \texttt{bit list}$

$\texttt{set}\ 0 = [\,]$

$\texttt{set}\ (\texttt{S n}) = \texttt{inc}\ (\texttt{set n})$

$$T_{\texttt{set}}(0) = 0$$
$$T_{\texttt{set}}(n) = T_{\texttt{inc}}(\log n) + T_{\texttt{set}}(n-1)$$

$$T_{\texttt{inc}}(n) \in O(n)$$
$$T_{\texttt{set}}(n) \in O(n \log_2 n)$$

**But we can do better!**

$$T_{\texttt{set}}(n) \in O(n)$$

# Binary Counter, Formally

Source

$$\texttt{inc} : \texttt{bit list} \to \texttt{bit list}$$
$$\texttt{inc}\ [\,] = [1]$$
$$\texttt{inc}\ (0 :: \texttt{bs}) = 1 :: \texttt{bs}$$
$$\texttt{inc}\ (1 :: \texttt{bs}) = 0 :: (\texttt{inc}\ \texttt{bs})$$

$\|\cdot\|$

$\lambda^{\mathbb{C}}$

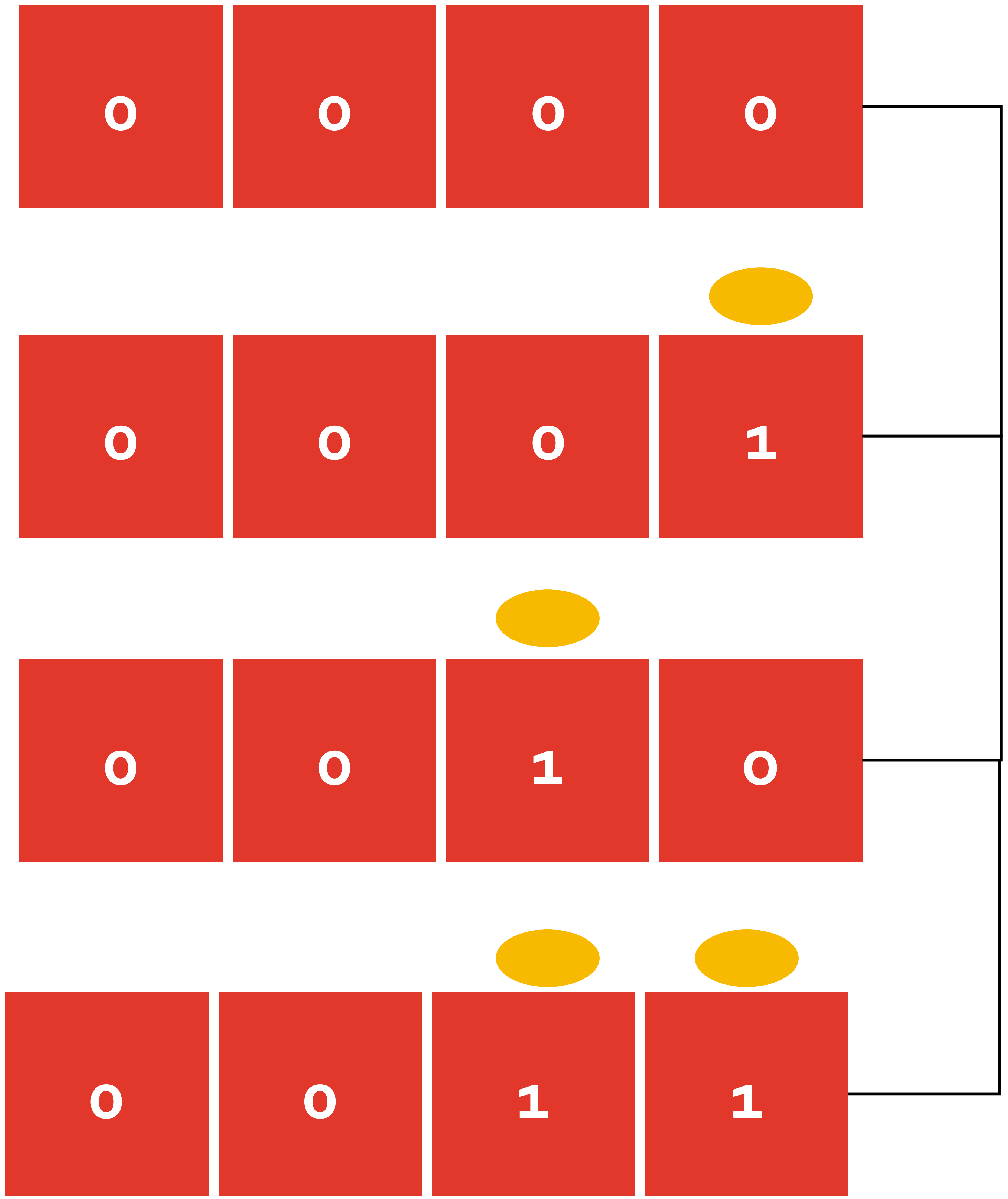
$$\|\texttt{inc}\|_c : \texttt{bit list} \to \mathbb{C}$$
$$\|\texttt{inc}\|_c\ [\,] = 1$$
$$\|\texttt{inc}\|_c\ (0 :: \texttt{bs}) = 1$$
$$\|\texttt{inc}\|_c\ (1 :: \texttt{bs}) = 1 + (\|\texttt{inc}\|_c\ \texttt{bs})$$

$[\![\texttt{bit list}]\!] = \mathbb{N}$ $\quad$ $[\![\texttt{b} :: \texttt{bs}]\!] = 1 + [\![\texttt{bs}]\!]$

Posets

$$[\![\|\texttt{inc}\|_c]\!](n) \in O(n) \implies [\![\|\texttt{set}\|_c]\!](n) \in O(n \log_2 n)$$

# Amortized Analysis




| Cons Ops | + | Credits Created | - | Credits Spent | = | Amortized Cost |
|---|---|---|---|---|---|---|
| 1 | | 1 | | 0 | | 2 |
| 2 | | 1 | | 1 | | 2 |
| 1 | | 1 | | 0 | | 2 |

# New Source Language $\lambda^A$

## Credits in Context

$$\Gamma \vdash_c M : A$$

(Affine type system!)

## Credit Modality

$$!_c A$$

(Graded modal types!)

## Attaching Credits

$$\frac{\Gamma \vdash_a M : A}{\Gamma \vdash_{a+c} \texttt{save}_c(M) : !_c A}$$

## Transferring Credits

$$\frac{\Gamma \vdash_a M : !_c A \quad \Gamma, x : A \vdash_{b+c} N : C}{\Gamma \vdash_{a+b} \texttt{transfer}\ !_c\, x = M \ \texttt{to}\ N : C}$$

## Creating Credits

$$\frac{\Gamma \vdash_{a+c} M : A}{\Gamma \vdash_a \texttt{create}_c(M) : A}$$

## Spending Credits

$$\frac{\Gamma \vdash_a M : A}{\Gamma \vdash_{a+c} \texttt{spend}_c(M) : A}$$

# Binary Counter in $\lambda^A$

$$\texttt{type bit} = \texttt{unit} \oplus !_1\texttt{unit}$$

$$\texttt{inc} : \texttt{bit list} \to \texttt{bit list}$$

$$\texttt{inc}\ [\,] = [\texttt{create}_1(\texttt{inr}\,(\texttt{save}_1()))]$$

$$\texttt{inc}\ ((\texttt{inl}\ \_) :: \texttt{bs}) = (\texttt{create}_1(\texttt{inr}\,(\texttt{save}_1()))) :: \texttt{bs}$$

$$\texttt{inc}\ ((\texttt{inr}\,x) :: \texttt{bs}) = \texttt{transfer}\ \_ = x\ \texttt{to}\ \texttt{spend}_1\ ((\texttt{inl}\,()) :: (\texttt{inc}\,\texttt{bs}))$$

# Extracting Amortized Recurrences

**Creating Credits Incurs a Cost**

$$\left\|\texttt{create}_a(M)\right\| = (a + \pi_1 \left\|M\right\|, \pi_2 \left\|M\right\|)$$

**Spending Credits Frees up Cost**

$$\left\|\texttt{spend}_a(M)\right\| = (-a + \pi_1 \left\|M\right\|, \pi_2 \left\|M\right\|)$$

**Extraction Erases the Modality**

$$\langle\!\langle !_c A \rangle\!\rangle = \langle\!\langle A \rangle\!\rangle$$

$$\left\|\texttt{save}_c(M)\right\| = \left\|M\right\|$$

$$\left\|\texttt{transfer}\ x = M\ \texttt{to}\ N\right\| = \texttt{let}\ (c, x) = \left\|M\right\|\ \texttt{in}\ (c + \pi_1 \left\|N\right\|, \pi_2 \left\|N\right\|)$$

# Binary Counter... Again

$$\texttt{type bit} = \texttt{unit} \oplus !_1\texttt{unit}$$

$\lambda^A$

$$\begin{aligned}
&\texttt{inc} : \texttt{bit list} \to \texttt{bit list} \\
&\texttt{inc}\ [\,] = [\texttt{create}_1(\texttt{inr}\,(\texttt{save}_1()))] \\
&\texttt{inc}\ ((\texttt{inl}\ _) :: \texttt{bs}) = (\texttt{create}_1(\texttt{inr}\,(\texttt{save}_1()))) :: \texttt{bs} \\
&\texttt{inc}\ ((\texttt{inr}\ x) :: \texttt{bs}) = \texttt{transfer}\ _ = x\ \texttt{to spend}_1\ ((\texttt{inl}\ ()) :: (\texttt{inc bs}))
\end{aligned}$$

$\Downarrow \ \|\cdot\|$

$\lambda^{\mathbb{C}}$

$$\begin{aligned}
&\|\texttt{inc}\|_c : (\texttt{unit} + \texttt{unit})\ \texttt{list} \to (\texttt{unit} + \texttt{unit})\ \texttt{list} \\
&\|\texttt{inc}\|_c\ [\,] = 2 \\
&\|\texttt{inc}\|_c\ (\texttt{inl}\ _ :: \texttt{bs}) = 2 \\
&\|\texttt{inc}\|_c\ (\texttt{inr}\ _ :: \texttt{bs}) = \|\texttt{inc}\|_c\ \texttt{bs}
\end{aligned}$$

$\Downarrow \ [\![\cdot]\!]$

**Posets**

$$[\![\|\texttt{inc}\|_c]\!](n) = 2 \implies [\![\|\texttt{set}\|_c]\!](n) = 2n \in O(n)$$

# Proving Extraction Correctness

**Amortized Cost Indexed Big-Step Operational Semantics**

$$M \downarrow^{n} v$$

**Bounding Theorem**

For $M : A$ if $M \downarrow^{n} v$,
then $n \leq \pi_1 \|M\|$
and $v \sqsubseteq^{A}_{\mathtt{val}} \pi_2 \|M\|$

**Key Corollary**

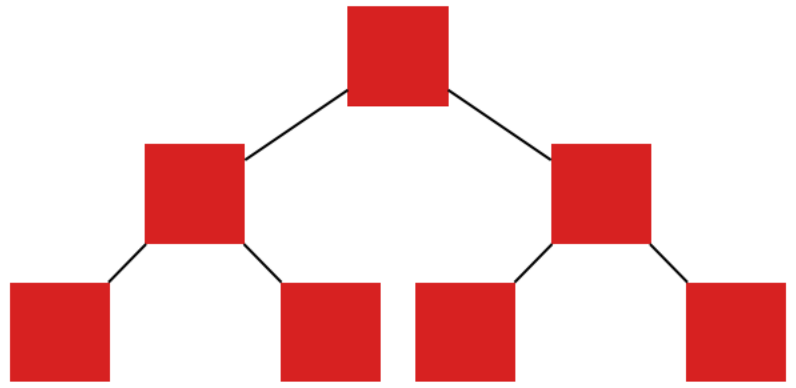
For closed terms typed in a context with no credits,
recurrence-predicted amortized cost
is a bound on real evaluation cost

# Thank you!

$!_c A$ - Affine type system & modality for tracking credits

$\|M\|$ - Automatic recurrence extraction translation

$n \leq \pi_1 \|M\|$ - Correctness proof relative to operational semantics by logical relations



- Expressive enough to handle non-trivial analyses like splay trees (not handled by existing techniques)