Sharing-Aware Improvements in a Call-by-Need Functional Core Language

Manfred Schmidt-Schauß and David Sabel

Goethe University Frankfurt am Main, Germany

IFL 2015, Koblenz, Germany

Motivation

Reasoning on program transformations, like

$$\texttt{map}\ f\ (\texttt{map}\ g\ xs) \rightarrow \texttt{map}\ (\lambda x.f\ (g\ x))\ xs$$

- Are transformations **optimizations** / **improvements**?
  - w.r.t. the resource consumption (time / space)
  - we consider the **time consumption**, i.e. the number computation steps
- In a core language of Haskell:
  - extended polymorphically typed lambda calculus
  - with **call-by-need** evaluation

Some Previous and Related Work

- [Moran & Sands, POPL'99]:
  Improvement theory in a call-by-need lambda calculus
  - Counting based on an abstract machine semantics
  - Untyped calculus with restricted syntax (arguments are variables)
  - Tick-algebra for modular reasoning on improvements
- [Schmidt-Schauß &. S., PPDP'15]:
  Improvement in the call-by-need lambda calculus LR
  - Counting essential reduction steps of a small-step semantics
  - Untyped core language with arbitrary arguments, seq-operator
  - Common subexpression elimination is an improvement

Our Contribution

- Improvements in a polymorphically-typed calculus (called LRP)
- Modular reasoning using **sharing-decorations**
  (extend Moran & Sands' tick algebra)
- Particular focus: Improvements for **list expressions** and functions
- **Proof techniques**: Induction schemes and simulation

The Calculus LRP: Syntax

Types:

$$\begin{array}{lcl} \tau \in \mathit{Typ} & ::= & a \mid (\tau_1 \to \tau_2) \mid K\ \tau_1 \ldots \tau_{\mathrm{ar}(K)} \\ \rho \in \mathit{PTyp} & ::= & \tau \mid \boldsymbol{\lambda} a.\rho \end{array}$$

Expressions:

$$\begin{array}{lcl} u \in \mathit{PExpr}_F & ::= & \Lambda a_1.\ldots.\Lambda a_k.\lambda x.s \\ s,t \in \mathit{Expr}_F & ::= & u \\ & \mid & x :: \rho \\ & \mid & (s\ \tau) \\ & \mid & (s\ t) \\ & \mid & (\texttt{seq}\ s\ t) \\ & \mid & (\texttt{letrec}\ x_1 :: \rho_1 = s_1, \ldots, x_n :: \rho_n = s_n\ \texttt{in}\ t) \\ & \mid & (c_{K,i} :: \tau\ s_1\ \ldots\ s_{\mathrm{ar}(c_{K,i})}) \\ & \mid & (\texttt{case}_K\ s\ \texttt{of}\ (\mathit{pat}_{K,1}\ \texttt{->}\ t_1)\ldots(\mathit{pat}_{K,|D_K|}\ \texttt{->}\ t_{|D_K|})) \\ \mathit{pat}_{K,i} & ::= & (c_{K,i} :: \tau\ x_1 :: \tau_1 \ldots x_{\mathrm{ar}(c_{K,i})} :: \tau_{\mathrm{ar}(c_{K,i})}) \end{array}$$

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Normal Order Reduction $\xrightarrow{\text{LRP}}$

- Small-step reduction relation
- Call-by-need strategy using reduction contexts R
- Several reduction rules, e.g.

(lbeta) $((\lambda x.s)\ t) \to \texttt{letrec}\ x = t\ \texttt{in}\ s$

(cp-in) $\texttt{letrec}\ x_1 = (\lambda y.t), \{x_i = x_{i-1}\}_{i=2}^{m}, Env\ \texttt{in}\ C[x_m]$
$\to \texttt{letrec}\ x_1 = (\lambda y.t), \{x_i = x_{i-1}\}_{i=2}^{m}, Env\ \texttt{in}\ C[(\lambda y.t)]$

(seq-c) $(\text{seq}\ v\ t) \to t$ if v is a value

(case-c) $\text{case}_K\ (c\ t_1\ \ldots\ t_n) \ldots ((c\ y_1\ \ldots\ y_n) \to s) \ldots$
$\to \texttt{letrec}\ \{y_i = t_i\}_{i=1}^{n}\ \texttt{in}\ s$

(llet-in) $\texttt{letrec}\ Env_1\ \texttt{in}\ (\texttt{letrec}\ Env_2\ \texttt{in}\ r)$
$\to \texttt{letrec}\ Env_1, Env_2\ \texttt{in}\ r$

...

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Convergence

A weak head normal form (WHNF) is

- a value: $\lambda x.s$, $\Lambda a.u$, or $c\,\overrightarrow{s}$.
- `letrec` Env `in` v, where v is a value
- `letrec` $x_1 = c\,\overrightarrow{s}, \{x_i = x_{i-1}\}_{i=2}^{m}, Env$ `in` x_m

Convergence:

- $s \downarrow t$ iff $s \xrightarrow{\mathrm{LRP},*} t \wedge t$ is a WHNF
- $s \downarrow$ iff $\exists t : s \downarrow t$.

Contextual Equivalence

For $s, t :: \rho$, $s \sim_c t$ iff for all contexts $C[\cdot :: \rho]$: $C[s]{\downarrow} \iff C[t]{\downarrow}$

Program transformation P is correct iff $(s \xrightarrow{P} t \implies s \sim_c t)$

Improvement

Counting Essential Reductions

$$\mathtt{rln}(t) := \begin{cases} \text{number of (lbeta),(case),(seq)-reductions} & \\ \text{in } t \xrightarrow{\text{LRP},*} t', & \text{if } t \downarrow t' \\ & \\ \infty, & \text{otherwise} \end{cases}$$

Improvement Relation

For $s, t :: \rho$, s **improves** t (written $s \preceq t$) iff

- $s \sim_c t$, and
- for all $C[\cdot :: \rho]$ s.t. $C[s], C[t]$ are closed: $\mathtt{rln}(C[s]) \leq \mathtt{rln}(C[t])$.

We write $s \approx t \iff s \preceq t \wedge t \preceq s$ (**improvement equivalence**)

Program transformation P is an **improvement** iff $s \xrightarrow{P} t \implies t \preceq s$

- for reasoning we want to express equations like

"s and t are improvement equivalent upto adding 3 steps of work to t"

Work Decorations: rln-Decorations

- for reasoning we want to express equations like

"s and t are improvement equivalent upto adding 3 steps of work to t"

- adding work: **rln-decorations** $s^{[n]}$ where n is nonnegative integer

  thus we can write $s \approx t^{[3]}$

Work Decorations: rln-Decorations

- for reasoning we want to express equations like

"s and t are improvement equivalent upto adding 3 steps of work to t"

- adding work: **rln-decorations** $s^{[n]}$ where n is nonnegative integer

  thus we can write $s \approx t^{[3]}$

- semantics of rln-decorations: they can be encoded

$$\begin{array}{lcl} s^{[0]} & = & s \\ s^{[n]} & = & (id^n\ s) \text{ where } id^n = \underbrace{(id\ \ldots id)}_{n \text{ times}},\ id = \lambda x.x \end{array}$$

- for reasoning we want to express equations like

"s and t are improvement equivalent upto adding 3 steps of work to t"

- adding work: **rln-decorations** $s^{[n]}$ where n is nonnegative integer

  thus we can write $s \approx t^{[3]}$

- semantics of rln-decorations: they can be encoded

$$\begin{array}{lcl} s^{[0]} & = & s \\ s^{[n]} & = & (id^n\ s) \ \text{ where } id^n = \underbrace{(id\ \ldots id)}_{n \text{ times}},\ id = \lambda x.x \end{array}$$

- allows to locally evaluate and remove environments, e.g.

$$\begin{array}{l} \texttt{letrec}\ \ x = (\lambda y.y)\ \texttt{True}, \\ \qquad\qquad y = (\texttt{seq True (seq False Nil)}) \quad \approx \quad (\texttt{True}^{[1]}, \texttt{Nil}^{[2]}) \\ \texttt{in}\ (x, y) \end{array}$$

- work can also be shared:

$$\begin{array}{l}\texttt{letrec}\ \ x = (\lambda y.y)\ \texttt{True},\\ \qquad\qquad y = (\texttt{seq}\ x\ (\texttt{seq}\ \texttt{False}\ \texttt{Nil})) \quad \approx \quad (\texttt{True}^{[a\mapsto 1]}, \texttt{Nil}^{[2,a\mapsto 1]})\\ \texttt{in}\ (x,y)\end{array}$$

- **shared work-decorations:** $s^{[a\mapsto n]}$
  $=$ **shared work** of n steps between all $^{[a\mapsto n]}$-labelled subexpressions

Work Decorations: Shared Work-Decorations

- work can also be shared:

$$\begin{array}{l} \texttt{letrec}\ \ {\color{red}x} = (\lambda y.y)\ \texttt{True}, \\ \qquad\qquad y = (\texttt{seq}\ {\color{red}x}\ (\texttt{seq}\ \texttt{False}\ \texttt{Nil})) \\ \texttt{in}\ ({\color{red}x}, y) \end{array} \quad \approx \quad (\texttt{True}^{[{\color{red}a\mapsto 1}]}, \texttt{Nil}^{[2,{\color{red}a\mapsto 1}]})$$

- **shared work-decorations:** $s^{[{\color{red}a\mapsto n}]}$
  $=$ **shared work** of n steps between all $^{[a\mapsto n]}$-labelled subexpressions
- semantics: not all cases can be encoded, e.g. $(\texttt{True}^{[a\mapsto 1]}, \texttt{Nil}^{[a\mapsto 1]}) \approx ?$

- work can also be shared:

$$\begin{array}{l}\texttt{letrec}\ \ x = (\lambda y.y)\ \texttt{True},\\ \qquad\qquad y = (\texttt{seq}\ x\ (\texttt{seq}\ \texttt{False}\ \texttt{Nil}))\\ \texttt{in}\ (x,y)\end{array} \quad\approx\quad (\texttt{True}^{[a\mapsto 1]}, \texttt{Nil}^{[2,a\mapsto 1]})$$

- **shared work-decorations:** $s^{[a\mapsto n]}$
  = **shared work** of n steps between all ${}^{[a\mapsto n]}$-labelled subexpressions
- semantics: not all cases can be encoded, e.g. $(\texttt{True}^{[a\mapsto 1]}, \texttt{Nil}^{[a\mapsto 1]}) \approx ?$
- we use additional reduction rules, and counting:

$$R[s^{[a\mapsto 0]}] \xrightarrow{\text{LRP}} R[s] \qquad\qquad \texttt{rln}(R[s^{[a\mapsto 0]}]) = \texttt{rln}(R[s])$$

$$R[s^{[a\mapsto n]}] \xrightarrow{\text{LRP}} R'[s'^{[a\mapsto n-1]}] \qquad\qquad \texttt{rln}(R[s^{[a\mapsto n]}]) = 1 + \texttt{rln}(R'[s'^{[a\mapsto n-1]}])$$

R', s' are R, s where all ${}^{[a\mapsto n]}$-decorations are replaced by ${}^{[a\mapsto n-1]}$

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- work can also be shared:

$$\begin{array}{l} \texttt{letrec}\ \ {\color{red}x} = (\lambda y.y)\ \texttt{True}, \\ \qquad\qquad y = (\texttt{seq}\ {\color{red}x}\ (\texttt{seq False Nil})) \quad \approx \quad (\texttt{True}^{{\color{red}[a\mapsto 1]}}, \texttt{Nil}^{[2,{\color{red}a\mapsto 1}]}) \\ \texttt{in}\ ({\color{red}x}, y) \end{array}$$

- **shared work-decorations:** $s^{[a \mapsto n]}$
  $=$ **shared work** of n steps between all $^{[a\mapsto n]}$-labelled subexpressions
- semantics: not all cases can be encoded, e.g. $(\texttt{True}^{[a\mapsto 1]}, \texttt{Nil}^{[a\mapsto 1]}) \approx ?$
- we use additional reduction rules, and counting:

$$R[s^{[a\mapsto 0]}] \xrightarrow{\text{LRP}} R[s] \qquad\qquad \texttt{rln}(R[s^{[a\mapsto 0]}]) = \texttt{rln}(R[s])$$

$$R[s^{[a\mapsto n]}] \xrightarrow{\text{LRP}} R'[s'^{[a\mapsto n-1]}] \qquad\qquad \texttt{rln}(R[s^{[a\mapsto n]}]) = 1 + \texttt{rln}(R'[s'^{[a\mapsto n-1]}])$$

R', s' are R, s where all $^{[a\mapsto n]}$-decorations are replaced by $^{[a\mapsto n-1]}$

- we use shared work-decorations only in **surface contexts** ($=$ hole not below a λ)

Context Lemma for Improvement

If for all reduction contexts R, s.t. $R[s], R[t]$ are closed: $\mathtt{rln}(R[s]) \leq \mathtt{rln}(R[t])$. Then $s \preceq t$

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Context Lemma for Improvement

If for all reduction contexts R, s.t. $R[s], R[t]$ are closed: $\mathtt{rln}(R[s]) \leq \mathtt{rln}(R[t])$. Then $s \preceq t$

Theorem [Schmidt-Schauß, S., Schütz, 2008]

1. If $s \xrightarrow{\text{LRP},lbeta\vee case\vee seq} t$, then $s \approx t^{[1]}$
2. If $s \xrightarrow{C,a} t$ then
   - if $a \in \{case, seq, lbeta\}$: $s \succeq t$
   - if $a \in \{llet, lapp, lcase, lseq, cp\}$: $s \approx t$
   - if $a \in \{gc, cpx, cpax, xch, cpcx, abs, lwas, ucp\}$: $s \approx t$.

$\{gc, cpx, cpax, xch, cpcx, abs, lwas, ucp\}$ are small optimizations e.g.

(gc) $\texttt{letrec}\ \{x_i = s_i\}_{i=1}^n, Env\ \texttt{in}\ t \to \texttt{letrec}\ Env\ \texttt{in}\ t$, if for all $i : x_i \notin FV(t, Env)$

(gc) $\texttt{letrec}\ x_1 = s_1, \ldots, x_n = s_n\ \texttt{in}\ t \to t$, if for all $i : x_i \notin FV(t)$

(ucp1) $\texttt{letrec}\ Env, x = t\ \texttt{in}\ S[x] \to \texttt{letrec}\ Env\ \texttt{in}\ S[t]$
where $x \notin FV(S, Env, t, r)$ and S is a surface context

...

Proposition

Let S be a surface context, then for all expressions s

- For rln-decorations:
  - $(s^{[k_1]})^{[k_2]} \approx s^{[k_1+k_2]}$
  - $s^{[0]} \approx s$
  - $S[s^{[k]}] \approx S[s]^{[k]}$, if S is strict ($S[\bot] \sim_c \bot$)
  - $S[s^{[k]}] \preceq S[s]^{[k]}$
- For sharing-decorations:
  - $(s^{[a \mapsto n]})^{[a \mapsto n]} \approx s^{[a \mapsto n]}$
  - $S[s^{[a \mapsto n]}] \approx S[s]^{[a \mapsto n]}$, if S is strict ($S[\bot] \sim_c \bot$)
  - $S[s^{[a \mapsto n]}] \preceq S[s]^{[a \mapsto n]}$
  - $S[s_1^{[a \mapsto m]}, \dots, s_n^{[a \mapsto m]}] \approx S[s_1, \dots, s_n]^{[m]}$ if some hole is strict.
  - $S[s_1^{[a \mapsto m]}, \dots, s_n^{[a \mapsto m]}] \preceq S[s_1, \dots, s_n]^{[m]}$

Theorem (An Induction Scheme)

Let S_1, S_2 be surface contexts and

- $S_1[x] \sim_c S_2[x]$ for a fresh variable x
- $S_1[\bot] \preceq S_2[\bot]$
- $S_1[\texttt{Nil}] \approx r^{[m]}$ and $S_2[\texttt{Nil}] \approx r^{[m']}$ with $m \leq m'$
- For fresh variables x and xs:
  - $(S_1[x : xs]) \approx (x : S_1[xs])^{[m]}$
  - $(S_2[x : xs]) \approx (x : S_2[xs])^{[m']}$

  with $m \leq m'$.

Then for all expressions s:

$$\texttt{letrec}\ x = s\ \texttt{in}\ S_1[x] \preceq \texttt{letrec}\ x = s\ \texttt{in}\ S_2[x].$$

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

$$\text{Let } \mathbb{L} := \begin{array}{ll} \texttt{letrec } (\texttt{++}) & = \lambda xs, ys.(\mathtt{case}_{List}\ xs\ \mathtt{of} \\ & \qquad (\texttt{Nil} \to ys) \\ & \qquad ((z : zs) \to z : ((\texttt{++})\ zs\ ys))) \\ \texttt{in } [\cdot] & \end{array}$$

Proposition

$\mathbb{L}[(xs \mathbin{\texttt{++}} (ys \mathbin{\texttt{++}} zs))] \preceq \mathbb{L}[((xs \mathbin{\texttt{++}} ys) \mathbin{\texttt{++}} zs)]$

Use $S_1 := \mathbb{L}[([\cdot] \mathbin{\texttt{++}} (ys \mathbin{\texttt{++}} zs))]$ and $S_2 = \mathbb{L}[(([\cdot] \mathbin{\texttt{++}} ys) \mathbin{\texttt{++}} zs)]$, and apply the induction scheme:

- $S_1[x] \sim_c S_2[x]$ (by standard inductive reasoning on $\sim_c$.)
- $S_1[\bot] \sim_c \bot \sim_c S_2[\bot]$
- $S_1[\texttt{Nil}] \approx (ys \mathbin{\texttt{++}} zs)^{[3]}$ and $S_2[\texttt{Nil}] \approx (ys \mathbin{\texttt{++}} zs)^{[3]}$
- $S_1[x : xs] \approx (x : S_1[xs])^{[3]}$ and $S_2[x : xs] \approx (x : S_2[xs])^{[6]}$.

Simulation $\sqsubseteq_h \subseteq \{(s,t) \mid s,t \text{ are closed}, s,t :: \mathtt{List}(\tau), s \sim_c t\}$
defined coinductively:

1. If $s \sim_c \bot \sim_c t$, then $s \ \sqsubseteq_h \ t$.

List-Simulation

Simulation $\sqsubseteq_h \subseteq \{(s,t) \mid s,t$ are closed$, s,t :: \texttt{List}(\tau), s \sim_c t\}$ defined coinductively:

1. If $s \sim_c \bot \sim_c t$, then $s \ \sqsubseteq_h \ t$.
2. If $s \approx \texttt{Nil}^{[m]}$, $t \approx \texttt{Nil}^{[m']}$ and $m \leq m'$, then $s \ \sqsubseteq_h \ t$.

List-Simulation

Simulation $\sqsubseteq_h \subseteq \{(s,t) \mid s,t \text{ are closed}, s,t :: \mathtt{List}(\tau), s \sim_c t\}$ defined coinductively:

1. If $s \sim_c \bot \sim_c t$, then $s \ \sqsubseteq_h \ t$.
2. If $s \approx \mathtt{Nil}^{[m]}$, $t \approx \mathtt{Nil}^{[m']}$ and $m \leq m'$, then $s \ \sqsubseteq_h \ t$.
3. If $s \preceq (s_1^{[m_1, a \mapsto m_2]} : s_2^{[m_3]})^{[m_4]}$ and $(t_1^{[m_1', a \mapsto m_2']} : t_2^{[m_3']})^{[m_4']} \preceq t$, where
   - $m_i \leq m_i'$
   - $s_1 \ \preceq \ t_1$ where s_1, t_1 are decoration-free, and
   - $s_2 \ \sqsubseteq_h \ t_2$
   - s_2, t_2 may contain further sharing decorations.

   Then $s \ \sqsubseteq_h \ t$.

Simulation $\sqsubseteq_h \subseteq \{(s,t) \mid s,t \text{ are closed}, s,t :: \mathtt{List}(\tau), s \sim_c t\}$ defined coinductively:

1. If $s \sim_c \bot \sim_c t$, then $s \ \sqsubseteq_h \ t$.
2. If $s \approx \mathtt{Nil}^{[m]}$, $t \approx \mathtt{Nil}^{[m']}$ and $m \leq m'$, then $s \ \sqsubseteq_h \ t$.
3. If $s \preceq (s_1^{[m_1, a \mapsto m_2]} : s_2^{[m_3]})^{[m_4]}$ and $(t_1^{[m'_1, a \mapsto m'_2]} : t_2^{[m'_3]})^{[m'_4]} \preceq t$, where
   - $m_i \leq m'_i$
   - $s_1 \ \preceq \ t_1$ where s_1, t_1 are decoration-free, and
   - $s_2 \ \sqsubseteq_h \ t_2$
   - s_2, t_2 may contain further sharing decorations.

   Then $s \ \sqsubseteq_h \ t$.

Theorem

If $s \sqsubseteq_h t$, then also $s \preceq t$.

An Example

$$\mathbb{L} := \texttt{letrec}\ \texttt{from1} = \lambda x.(x : (\texttt{from1}\ (x\texttt{+}1)))$$
$$\texttt{from2} = \lambda x.(\texttt{letrec}\ y = (x\texttt{+}1)\ \texttt{in}\ y : (\texttt{from2}\ y))$$
$$\texttt{in}\ [\cdot]$$

Let n_i denote the i^{th} number and + be strict addition s.t. $\texttt{rln}(n_i\texttt{+}n_j) = 4$

Proposition

For all numbers n_i, n_{i+1}: $\mathbb{L}[\texttt{from1}\ n_{i+1}] \preceq \mathbb{L}[\texttt{from2}\ n_i]$

An Example

$$\mathbb{L} := \texttt{letrec}\ \texttt{from1} = \lambda x.(x : (\texttt{from1}\ (x\texttt{+}1)))$$
$$\texttt{from2} = \lambda x.(\texttt{letrec}\ y = (x\texttt{+}1)\ \texttt{in}\ y : (\texttt{from2}\ y))$$
$$\texttt{in}\ [\cdot]$$

Let n_i denote the i^{th} number and + be strict addition s.t. $\texttt{rln}(n_i\texttt{+}n_j) = 4$

Proposition

For all numbers n_i, n_{i+1}: $\mathbb{L}[\texttt{from1}\ n_{i+1}] \preceq \mathbb{L}[\texttt{from2}\ n_i]$

We show $\mathbb{L}[\texttt{from1}\ n_{i+1}]\ \sqsubseteq_h\ \mathbb{L}[\texttt{from2}\ n_i]$

$$\mathbb{L} := \texttt{letrec}\ \texttt{from1} = \lambda x.(x : (\texttt{from1}\ (x{+}1)))$$
$$\texttt{from2} = \lambda x.(\texttt{letrec}\ y = (x{+}1)\ \texttt{in}\ y : (\texttt{from2}\ y))$$
$$\texttt{in}\ [\cdot]$$

Let n_i denote the i^{th} number and + be strict addition s.t. $\texttt{rln}(n_i{+}n_j) = 4$

Proposition

For all numbers n_i, n_{i+1}: $\mathbb{L}[\texttt{from1}\ n_{i+1}] \preceq \mathbb{L}[\texttt{from2}\ n_i]$

We show $\mathbb{L}[\texttt{from1}\ n_{i+1}] \ \sqsubseteq_h\ \mathbb{L}[\texttt{from2}\ n_i]$

- $\mathbb{L}[\texttt{from1}\ n_{i+1}] \approx ({n_{i+1}}^{[a\mapsto 0]} : \mathbb{L}[\texttt{from1}\ {n_{i+2}}^{[4,a\mapsto 0]}])^{[1]}$
- $\mathbb{L}[\texttt{from2}\ n_i] \quad \approx ({n_{i+1}}^{[a\mapsto 4]} : \mathbb{L}[\texttt{from2}\ {n_{i+1}}^{[a\mapsto 4]}])^{[1]}$

An Example

$\mathbb{L} := \texttt{letrec from1} = \lambda x.(x : (\texttt{from1}\ (x\texttt{+}1)))$
$\qquad\qquad \texttt{from2} = \lambda x.(\texttt{letrec}\ y = (x\texttt{+}1)\ \texttt{in}\ y : (\texttt{from2}\ y))$
$\quad \texttt{in}\ [\cdot]$

Let n_i denote the i^{th} number and + be strict addition s.t. $\texttt{rln}(n_i\texttt{+}n_j) = 4$

Proposition

For all numbers n_i, n_{i+1}: $\mathbb{L}[\texttt{from1}\ n_{i+1}] \preceq \mathbb{L}[\texttt{from2}\ n_i]$

We show $\mathbb{L}[\texttt{from1}\ n_{i+1}] \sqsubseteq_h \mathbb{L}[\texttt{from2}\ n_i]$

- $\mathbb{L}[\texttt{from1}\ n_{i+1}] \approx ({n_{i+1}}^{[a \mapsto 0]} : \mathbb{L}[\texttt{from1}\ {n_{i+2}}^{[4, a \mapsto 0]}])^{[1]}$
- $\mathbb{L}[\texttt{from2}\ n_i] \quad \approx ({n_{i+1}}^{[a \mapsto 4]} : \mathbb{L}[\texttt{from2}\ {n_{i+1}}^{[a \mapsto 4]}])^{[1]}$

$$\mathbb{L} := \texttt{letrec from1} = \lambda x.(x : (\texttt{from1}\ (x\texttt{+}1)))$$
$$\texttt{from2} = \lambda x.(\texttt{letrec}\ y = (x\texttt{+}1)\ \texttt{in}\ y : (\texttt{from2}\ y))$$
$$\texttt{in}\ [\cdot]$$

Let n_i denote the i^{th} number and + be strict addition s.t. $\texttt{rln}(n_i\texttt{+}n_j) = 4$

Proposition

For all numbers n_i, n_{i+1}: $\mathbb{L}[\texttt{from1}\ n_{i+1}] \preceq \mathbb{L}[\texttt{from2}\ n_i]$

We show $\mathbb{L}[\texttt{from1}\ n_{i+1}] \sqsubseteq_h \mathbb{L}[\texttt{from2}\ n_i]$

- $\mathbb{L}[\texttt{from1}\ n_{i+1}] \approx (n_{i+1}{}^{[a\mapsto 0]} : \mathbb{L}[\texttt{from1}\ n_{i+2}{}^{[4,a\mapsto 0]}])^{[1]}$
- $\mathbb{L}[\texttt{from2}\ n_i] \approx (n_{i+1}{}^{[a\mapsto 4]} : \mathbb{L}[\texttt{from2}\ n_{i+1}{}^{[a\mapsto 4]}])^{[1]}$

An Example

$$\begin{aligned}\mathbb{L} := \texttt{letrec}\ &\texttt{from1} = \lambda x.(x : (\texttt{from1}\ (x{+}1)))\\ &\texttt{from2} = \lambda x.(\texttt{letrec}\ y = (x{+}1)\ \texttt{in}\ y : (\texttt{from2}\ y))\\ \texttt{in}\ &[\cdot]\end{aligned}$$

Let n_i denote the i^{th} number and + be strict addition s.t. $\texttt{rln}(n_i{+}n_j) = 4$

Proposition

For all numbers n_i, n_{i+1}: $\mathbb{L}[\texttt{from1}\ n_{i+1}] \preceq \mathbb{L}[\texttt{from2}\ n_i]$

We show $\mathbb{L}[\texttt{from1}\ n_{i+1}] \ \sqsubseteq_h\ \mathbb{L}[\texttt{from2}\ n_i]$

- $\mathbb{L}[\texttt{from1}\ n_{i+1}] \approx ({n_{i+1}}^{[a\mapsto 0]} : \mathbb{L}[\texttt{from1}\ {n_{i+2}}^{[4,a\mapsto 0]}])^{[1]}$
- $\mathbb{L}[\texttt{from2}\ n_i] \quad \approx ({n_{i+1}}^{[a\mapsto 4]} : \mathbb{L}[\texttt{from2}\ {n_{i+1}}^{[a\mapsto 4]}])^{[1]}$

An Example

$$\mathbb{L} := \texttt{letrec}\ \texttt{from1} = \lambda x.(x : (\texttt{from1}\ (x{+}1)))$$
$$\texttt{from2} = \lambda x.(\texttt{letrec}\ y = (x{+}1)\ \texttt{in}\ y : (\texttt{from2}\ y))$$
$$\texttt{in}\ [\cdot]$$

Let n_i denote the i^{th} number and + be strict addition s.t. $\texttt{rln}(n_i{+}n_j) = 4$

Proposition

For all numbers n_i, n_{i+1}: $\mathbb{L}[\texttt{from1}\ n_{i+1}] \preceq \mathbb{L}[\texttt{from2}\ n_i]$

We show $\mathbb{L}[\texttt{from1}\ n_{i+1}] \sqsubseteq_h \mathbb{L}[\texttt{from2}\ n_i]$

- $\mathbb{L}[\texttt{from1}\ n_{i+1}] \approx ({n_{i+1}}^{[a\mapsto 0]} : \mathbb{L}[\texttt{from1}\ {n_{i+2}}^{[4,a\mapsto 0]}])^{[1]}$
- $\mathbb{L}[\texttt{from2}\ n_i] \approx ({n_{i+1}}^{[a\mapsto 4]} : \mathbb{L}[\texttt{from2}\ {n_{i+1}}^{[a\mapsto 4]}])^{[1]}$

It remains to show $\mathbb{L}[\texttt{from1}\ {n_{i+2}}^{[4,a\mapsto 0]}] \sqsubseteq_h \mathbb{L}[\texttt{from2}\ {n_{i+1}}^{[a\mapsto 4]}]$

An Example

$$\begin{array}{ll}\mathbb{L} := \texttt{letrec} & \texttt{from1} = \lambda x.(x : (\texttt{from1}\ (x{+}1))) \\ & \texttt{from2} = \lambda x.(\texttt{letrec}\ y = (x{+}1)\ \texttt{in}\ y : (\texttt{from2}\ y)) \\ \quad \texttt{in}\ [\cdot] & \end{array}$$

Let n_i denote the i^{th} number and + be strict addition s.t. $\texttt{rln}(n_i{+}n_j) = 4$

Proposition

For all numbers n_i, n_{i+1}: $\mathbb{L}[\texttt{from1}\ n_{i+1}] \preceq \mathbb{L}[\texttt{from2}\ n_i]$

We show $\mathbb{L}[\texttt{from1}\ n_{i+1}] \sqsubseteq_h \mathbb{L}[\texttt{from2}\ n_i]$

- $\mathbb{L}[\texttt{from1}\ n_{i+1}] \approx ({n_{i+1}}^{[a\mapsto 0]} : \mathbb{L}[\texttt{from1}\ {n_{i+2}}^{[4,a\mapsto 0]}])^{[1]}$
- $\mathbb{L}[\texttt{from2}\ n_i] \approx ({n_{i+1}}^{[a\mapsto 4]} : \mathbb{L}[\texttt{from2}\ {n_{i+1}}^{[a\mapsto 4]}])^{[1]}$

It remains to show $\mathbb{L}[\texttt{from1}\ {n_{i+2}}^{[4,a\mapsto 0]}] \sqsubseteq_h \mathbb{L}[\texttt{from2}\ {n_{i+1}}^{[a\mapsto 4]}]$

In general for $j \geq 2$:

- $\mathbb{L}[\texttt{from1}\ {n_{i+j}}^{[4,a\mapsto 4*(j-2)]}] \approx ({n_{i+j}}^{[a\mapsto 4*(j-1)]} : \mathbb{L}[\texttt{from1}\ {n_{i+j+1}}^{[4,a\mapsto 4*(j-1)]}])^{[1]}$
- $\mathbb{L}[\texttt{from2}\ {n_{i+j-1}}^{[a\mapsto 4*(j-1)]}] \approx ({n_{i+j}}^{[a\mapsto 4*j]} : \mathbb{L}[\texttt{from2}\ {n_{i+j}}^{[a\mapsto 4*j]}])^{[1]}$

Thus the claim holds.

In the paper are more examples:

- $\texttt{map}\ (\lambda x.f\ (g\ x))\ xs \ \preceq\ \texttt{map}\ f\ (\texttt{map}\ g\ xs)$
- $\texttt{repeat1}\ r \preceq \texttt{repeat2}\ r$ where

  $$\begin{aligned}\texttt{repeat1}\ x &= \texttt{letrec}\ xs = x : xs\ \texttt{in}\ xs\\ \texttt{repeat2}\ x &= x : \texttt{repeat2}\ xs\end{aligned}$$

- $\texttt{iterate1}\ g\ x \preceq \texttt{iterate2}\ g\ x$ where

  $$\begin{aligned}\texttt{iterate1}\ g\ x &= x : \texttt{iterate1}\ g\ (g\ x)\\ \texttt{iterate2}\ g\ x &= \texttt{map}\ g\ (\texttt{iterate1}\ g\ x)\end{aligned}$$

- $\texttt{fibsA}\ 1\ 2 \not\preceq \texttt{fibsB}\ 1$ but $\texttt{fibsA}\ 1\ 2 \preceq \texttt{fibsC}\ 1$ where

  $$\begin{aligned}\texttt{fibsA}\ x\ y &= x : \texttt{fibsA}\ y\ (x+y)\\ \texttt{fibsB}\ n &= (\texttt{fib}\ n) : (\texttt{fibsB}\ (n+1))\\ \texttt{fibsC}\ n &= ((\texttt{fib}\ n) : (\texttt{fibsC}\ (n+1)))^{[1]}\\ \texttt{fib}\ 0 &= 1\\ \texttt{fib}\ 1 &= 1\\ \texttt{fib}\ n &= \texttt{fib}\ (n-1) + \texttt{fib}(n-2)\end{aligned}$$

Conclusion

- Proof techniques for proving improvements in the call-by-need setting
- Novel notation to explicitly **compute with shared work**
- We illustrated our techniques on interesting examples

Conclusion and Future Work

Conclusion

- Proof techniques for proving improvements in the call-by-need setting
- Novel notation to explicitly **compute with shared work**
- We illustrated our techniques on interesting examples

Further Work

- Consider further examples and variants of the proof tools
- **Automation** of optimization and showing improvement
- **Space**-improvements