GOETHE UNIVERSITÄT FRANKFURT AM MAIN

A Call-by-Need Lambda Calculus with Scoped Work Decorations

David Sabel and Manfred Schmidt-Schauß

Goethe University Frankfurt am Main, Germany

ATPS 2016, Vienna, Austria

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Reasoning on program transformations, like

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

Are transformations **optimizations** / **improvements**?

- w.r.t. time consumption, i.e. the **number of 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 an untyped call-by-need lambda calculus

- counting based on an abstract machine semantics
- **tick-algebra** for modular reasoning on improvements
- no concrete technique for list induction proofs

[Hackett & Hutton, ICFP'14]:

Improvement for worker-wrapper-transformations

- based on Moran & Sands' tick algebra
- argue for the requirement of a typed language

[Schmidt-Schauß & S., PPDP'15, IFL'15]:

Improvement in call-by-need lambda calculi: untyped LR, typed LRP

- counting essential reduction steps of a small-step semantics
- core language with seq-operator
- **proving list-laws** being improvements, **using work-decorations**

Example:

$$
\begin{array}{lcl}
s_1 & := & \texttt{letrec}\ xs{=}\texttt{0} : xs\ \texttt{in}\ xs \\
s_2 & := & \texttt{letrec}\ y{=}((\lambda x.x)\ \texttt{0}), xs{=}y : y : xs\ \texttt{in}\ xs \\
s_3 & := & \texttt{letrec}\ y{=}((\lambda x.x)\ \texttt{0}), xs{=}y : ((\lambda x.x)\ y) : xs\ \texttt{in}\ xs \\
s_4 & := & \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ \texttt{0})
\end{array}
$$

Contextual equivalence: $s \sim_c t$ iff $\forall$ contexts $C : C[s]\downarrow \iff C[t]\downarrow$

Prove $s_i \sim_c s_j$ for all $i, j \in \{1, 2, 3, 4\}$

Example:

$$
\begin{aligned}
s_1 &:= \texttt{letrec}\ xs{=}0 : xs\ \texttt{in}\ xs\\
s_2 &:= \texttt{letrec}\ y{=}((\lambda x.x)\ 0), xs{=}y : y : xs\ \texttt{in}\ xs\\
s_3 &:= \texttt{letrec}\ y{=}((\lambda x.x)\ 0), xs{=}y : ((\lambda x.x)\ y) : xs\ \texttt{in}\ xs\\
s_4 &:= \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ 0)
\end{aligned}
$$

Contextual equivalence: $s \sim_c t$ iff $\forall$ contexts $C : C[s]\downarrow \iff C[t]\downarrow$

Prove $s_i \sim_c s_j$ for all $i, j \in \{1,2,3,4\}$

- For list-expressions r_1, r_2 define:
  $r_1\ R\ r_2$ iff $r_1 \sim_c (h_1 : t_1)$, $r_2 \sim_c (h_2 : t_2)$
  such that $h_1 \sim_c h_2$ and $t_1\ R\ t_2$
- Principle of co-induction: $r_1\ \mathrm{gfp}(R)\ r_2 \implies r_1 \sim_c r_2$

Example:

$$
\begin{array}{lll}
s_1 & := & \texttt{letrec}\ xs{=}\texttt{0} : xs\ \texttt{in}\ xs \\
s_2 & := & \texttt{letrec}\ y{=}((\lambda x.x)\ \texttt{0}), xs{=}y : y : xs\ \texttt{in}\ xs \\
s_3 & := & \texttt{letrec}\ y{=}((\lambda x.x)\ \texttt{0}), xs{=}y : ((\lambda x.x)\ y) : xs\ \texttt{in}\ xs \\
s_4 & := & \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ \texttt{0})
\end{array}
$$

Contextual equivalence: $s \sim_c t$ iff $\forall$ contexts $C : C[s]\downarrow \iff C[t]\downarrow$

Prove $s_i \sim_c s_j$ for all $i, j \in \{1,2,3,4\}$

- For list-expressions r_1, r_2 define:
  $r_1\ R\ r_2$ iff $r_1 \sim_c (h_1 : t_1)$, $r_2 \sim_c (h_2 : t_2)$
  such that $h_1 \sim_c h_2$ and $t_1\ R\ t_2$
- Principle of co-induction: $r_1\ \mathrm{gfp}(R)\ r_2 \implies r_1 \sim_c r_2$

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Example:

$$
\begin{array}{lll}
s_1 & \sim_c & 0 : 0 : (\texttt{letrec}\ xs{=}0 : xs\ \texttt{in}\ xs) \\
s_2 & := & \texttt{letrec}\ y{=}((\lambda x.x)\ 0), xs{=}y : y : xs\ \texttt{in}\ xs \\
s_3 & := & \texttt{letrec}\ y{=}((\lambda x.x)\ 0), xs{=}y : ((\lambda x.x)\ y) : xs\ \texttt{in}\ xs \\
s_4 & := & \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ 0)
\end{array}
$$

Contextual equivalence: $s \sim_c t$ iff $\forall$ contexts $C : C[s]\downarrow \iff C[t]\downarrow$

Prove $s_i \sim_c s_j$ for all $i, j \in \{1, 2, 3, 4\}$

- For list-expressions r_1, r_2 define:
  $r_1\ R\ r_2$ iff $r_1 \sim_c (h_1 : t_1)$, $r_2 \sim_c (h_2 : t_2)$
  such that $h_1 \sim_c h_2$ and $t_1\ R\ t_2$
- Principle of co-induction: $r_1\ \mathrm{gfp}(R)\ r_2 \implies r_1 \sim_c r_2$

Example:

$$
\begin{array}{lcl}
s_1 & \sim_c & \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : xs\ \mathtt{in}\ xs) \\
s_2 & := & \mathtt{letrec}\ y{=}((\lambda x.x)\ \mathtt{0}), xs{=}y : y : xs\ \mathtt{in}\ xs \\
s_3 & := & \mathtt{letrec}\ y{=}((\lambda x.x)\ \mathtt{0}), xs{=}y : ((\lambda x.x)\ y) : xs\ \mathtt{in}\ xs \\
s_4 & := & \mathtt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \mathtt{in}\ (xs\ \mathtt{0})
\end{array}
$$

Contextual equivalence: $s \sim_c t$ iff $\forall$ contexts $C : C[s]\downarrow \iff C[t]\downarrow$

Prove $s_i \sim_c s_j$ for all $i, j \in \{1, 2, 3, 4\}$

- For list-expressions r_1, r_2 define:
  $r_1\ R\ r_2$ iff $r_1 \sim_c (h_1 : t_1)$, $r_2 \sim_c (h_2 : t_2)$
  such that $h_1 \sim_c h_2$ and $t_1\ R\ t_2$
- Principle of co-induction: $r_1\ \mathrm{gfp}(R)\ r_2 \implies r_1 \sim_c r_2$

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Example:

$$s_1 \sim_c \texttt{0} : \texttt{0} : (\texttt{letrec}\ xs{=}\texttt{0} : xs\ \texttt{in}\ xs)$$
$$s_2 \sim_c \texttt{0} : \texttt{0} : (\texttt{letrec}\ xs{=}\texttt{0} : \texttt{0} : xs\ \texttt{in}\ xs)$$
$$s_3 := \texttt{letrec}\ y{=}((\lambda x.x)\ \texttt{0}), xs{=}y : ((\lambda x.x)\ y) : xs\ \texttt{in}\ xs$$
$$s_4 := \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ \texttt{0})$$

Contextual equivalence: $s \sim_c t$ iff $\forall$ contexts $C : C[s]\downarrow \iff C[t]\downarrow$

Prove $s_i \sim_c s_j$ for all $i, j \in \{1, 2, 3, 4\}$

- For list-expressions r_1, r_2 define:
  $r_1\ R\ r_2$ iff $r_1 \sim_c (h_1 : t_1)$, $r_2 \sim_c (h_2 : t_2)$
  such that $h_1 \sim_c h_2$ and $t_1\ R\ t_2$
- Principle of co-induction: $r_1\ \mathrm{gfp}(R)\ r_2 \implies r_1 \sim_c r_2$

Example:

$$
\begin{array}{lll}
s_1 & \sim_c & \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : xs\ \mathtt{in}\ xs) \\
s_2 & \sim_c & \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : \mathtt{0} : xs\ \mathtt{in}\ xs) \\
s_3 & := & \mathtt{letrec}\ y{=}((\lambda x.x)\ \mathtt{0}), xs{=}y : ((\lambda x.x)\ y) : xs\ \mathtt{in}\ xs \\
s_4 & := & \mathtt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \mathtt{in}\ (xs\ \mathtt{0})
\end{array}
$$

Contextual equivalence: $s \sim_c t$ iff $\forall$ contexts $C : C[s]\downarrow \iff C[t]\downarrow$

Prove $s_i \sim_c s_j$ for all $i, j \in \{1, 2, 3, 4\}$

- For list-expressions r_1, r_2 define:
  $r_1\ R\ r_2$ iff $r_1 \sim_c (h_1 : t_1)$, $r_2 \sim_c (h_2 : t_2)$
  such that $h_1 \sim_c h_2$ and $t_1\ R\ t_2$
- Principle of co-induction: $r_1\ \mathrm{gfp}(R)\ r_2 \implies r_1 \sim_c r_2$

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Example:

$$
\begin{array}{lll}
s_1 & \sim_c & \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : xs\ \mathtt{in}\ xs) \\
s_2 & \sim_c & \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : \mathtt{0} : xs\ \mathtt{in}\ xs) \\
s_3 & \sim_c & \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : \mathtt{0} : xs\ \mathtt{in}\ xs) \\
s_4 & := & \mathtt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \mathtt{in}\ (xs\ \mathtt{0})
\end{array}
$$

Contextual equivalence: $s \sim_c t$ iff $\forall$ contexts $C : C[s]\downarrow \iff C[t]\downarrow$

Prove $s_i \sim_c s_j$ for all $i, j \in \{1, 2, 3, 4\}$

- For list-expressions r_1, r_2 define:
  $r_1\ R\ r_2$ iff $r_1 \sim_c (h_1 : t_1)$, $r_2 \sim_c (h_2 : t_2)$
  such that $h_1 \sim_c h_2$ and $t_1\ R\ t_2$
- Principle of co-induction: $r_1\ \mathrm{gfp}(R)\ r_2 \implies r_1 \sim_c r_2$

Example:

$$
\begin{array}{lll}
s_1 & \sim_c & \mathtt{0:0:}(\mathtt{letrec}\ xs{=}\mathtt{0}:xs\ \mathtt{in}\ xs) \\
s_2 & \sim_c & \mathtt{0:0:}(\mathtt{letrec}\ xs{=}\mathtt{0}:\mathtt{0}:xs\ \mathtt{in}\ xs) \\
s_3 & \sim_c & \mathtt{0:0:}(\mathtt{letrec}\ xs{=}\mathtt{0}:\mathtt{0}:xs\ \mathtt{in}\ xs) \\
s_4 & := & \mathtt{letrec}\ xs{=}\lambda y.y:(xs\ y)\ \mathtt{in}\ (xs\ \mathtt{0})
\end{array}
$$

Contextual equivalence: $s \sim_c t$ iff $\forall$ contexts $C : C[s]\downarrow \iff C[t]\downarrow$

Prove $s_i \sim_c s_j$ for all $i, j \in \{1, 2, 3, 4\}$

- For list-expressions r_1, r_2 define:
  $r_1\ R\ r_2$ iff $r_1 \sim_c (h_1 : t_1)$, $r_2 \sim_c (h_2 : t_2)$
  such that $h_1 \sim_c h_2$ and $t_1\ R\ t_2$
- Principle of co-induction: $r_1\ \mathrm{gfp}(R)\ r_2 \implies r_1 \sim_c r_2$

Example:

$$s_1 \sim_c \texttt{0} : \texttt{0} : (\texttt{letrec}\ xs{=}\texttt{0} : xs\ \texttt{in}\ xs)$$
$$s_2 \sim_c \texttt{0} : \texttt{0} : (\texttt{letrec}\ xs{=}\texttt{0} : \texttt{0} : xs\ \texttt{in}\ xs)$$
$$s_3 \sim_c \texttt{0} : \texttt{0} : (\texttt{letrec}\ xs{=}\texttt{0} : \texttt{0} : xs\ \texttt{in}\ xs)$$
$$s_4 \sim_c \texttt{0} : \texttt{0} : (\texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ \texttt{0}))$$

Contextual equivalence: $s \sim_c t$ iff $\forall$ contexts $C : C[s]\downarrow \iff C[t]\downarrow$

Prove $s_i \sim_c s_j$ for all $i, j \in \{1, 2, 3, 4\}$

- For list-expressions r_1, r_2 define:
  $r_1\ R\ r_2$ iff $r_1 \sim_c (h_1 : t_1)$, $r_2 \sim_c (h_2 : t_2)$
  such that $h_1 \sim_c h_2$ and $t_1\ R\ t_2$
- Principle of co-induction: $r_1\ \mathrm{gfp}(R)\ r_2 \implies r_1 \sim_c r_2$

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Example:

$$
\begin{array}{ll}
s_1 & \sim_c \texttt{0} : \texttt{0} : (\texttt{letrec}\ xs{=}\texttt{0} : xs\ \texttt{in}\ xs) \\
s_2 & \sim_c \texttt{0} : \texttt{0} : (\texttt{letrec}\ xs{=}\texttt{0} : \texttt{0} : xs\ \texttt{in}\ xs) \\
s_3 & \sim_c \texttt{0} : \texttt{0} : (\texttt{letrec}\ xs{=}\texttt{0} : \texttt{0} : xs\ \texttt{in}\ xs) \\
s_4 & \sim_c \texttt{0} : \texttt{0} : (\texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ \texttt{0}))
\end{array}
$$

further processing with the tails indeed shows $s_i \sim_c s_j$

Contextual equivalence: $s \sim_c t$ iff $\forall$ contexts $C : C[s]\downarrow \iff C[t]\downarrow$

Prove $s_i \sim_c s_j$ for all $i, j \in \{1, 2, 3, 4\}$

- For list-expressions r_1, r_2 define:
  $r_1\ R\ r_2$ iff $r_1 \sim_c (h_1 : t_1)$, $r_2 \sim_c (h_2 : t_2)$
  such that $h_1 \sim_c h_2$ and $t_1\ R\ t_2$
- Principle of co-induction: $r_1\ \mathrm{gfp}(R)\ r_2 \implies r_1 \sim_c r_2$

In [Schmidt-Schauß & S., IFL 2015]:

- **analogous reasoning**,
- but w.r.t. **improvement** and **cost-equivalence**

Improvement and Cost-Equivalence

Improvement:

$s \preceq t$ iff $s \sim_c t$ and $\forall$ closing contexts $C : \mathtt{rln}(C[s]) \leq \mathtt{rln}(C[t])$

where $\mathtt{rln}(\cdot)$ is the reduction length, counting essential reduction steps

Cost-Equivalence:

$s \approx t$ iff $s \preceq t$ and $t \preceq s$

Equational reasoning w.r.t. cost equivalence:

$$\begin{array}{lcl}
s_1 & := & \texttt{letrec}\ xs{=}\mathtt{0} : xs\ \texttt{in}\ xs \\
s_2 & := & \texttt{letrec}\ y{=}((\lambda x.x)\ \mathtt{0}), xs{=}y : y : xs\ \texttt{in}\ xs \\
s_3 & := & \texttt{letrec}\ y{=}((\lambda x.x)\ \mathtt{0}), xs{=}y : ((\lambda x.x)\ y) : xs\ \texttt{in}\ xs \\
s_4 & := & \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ \mathtt{0})
\end{array}$$

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Equational reasoning w.r.t. cost equivalence:

$s_1 := \texttt{letrec}\ xs{=}\texttt{0} : xs\ \texttt{in}\ xs$

$s_2 := \texttt{letrec}\ y{=}((\lambda x.x)\ \texttt{0}), xs{=}y : y : xs\ \texttt{in}\ xs$

$s_3 := \texttt{letrec}\ y{=}((\lambda x.x)\ \texttt{0}), xs{=}y : ((\lambda x.x)\ y) : xs\ \texttt{in}\ xs$

$s_4 := \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ \texttt{0})$

Equational reasoning w.r.t. cost equivalence:

$$s_1 \approx \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : xs\ \mathtt{in}\ xs)$$
$$s_2 := \mathtt{letrec}\ y{=}((\lambda x.x)\ \mathtt{0}), xs{=}y : y : xs\ \mathtt{in}\ xs$$
$$s_3 := \mathtt{letrec}\ y{=}((\lambda x.x)\ \mathtt{0}), xs{=}y : ((\lambda x.x)\ y) : xs\ \mathtt{in}\ xs$$
$$s_4 := \mathtt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \mathtt{in}\ (xs\ \mathtt{0})$$

Equational reasoning w.r.t. cost equivalence:

$$s_1 \approx \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : xs\ \mathtt{in}\ xs)$$

$$s_2 := \mathtt{letrec}\ y{=}((\lambda x.x)\ \mathtt{0}), xs{=}y : y : xs\ \mathtt{in}\ xs$$

$$s_3 := \mathtt{letrec}\ y{=}((\lambda x.x)\ \mathtt{0}), xs{=}y : ((\lambda x.x)\ y) : xs\ \mathtt{in}\ xs$$

$$s_4 := \mathtt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \mathtt{in}\ (xs\ \mathtt{0})$$

Equational reasoning w.r.t. cost equivalence:

$s_1 \approx \texttt{0} : \texttt{0} : (\texttt{letrec}\ xs{=}\texttt{0} : xs\ \texttt{in}\ xs)$

$s_2 \approx \texttt{letrec}\ y{=}\texttt{0}^{[1]}, xs{=}y : y : xs\ \texttt{in}\ xs$

$s_3 := \texttt{letrec}\ y{=}((\lambda x.x)\ \texttt{0}), xs{=}y : ((\lambda x.x)\ y) : xs\ \texttt{in}\ xs$

$s_4 := \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ \texttt{0})$

Work-decorations to keep track of `rln`-work:

- $^{[n]} := n$ essential reduction steps

Motivation: Reasoning including Resources

Equational reasoning w.r.t. cost equivalence:

$$
\begin{array}{lcl}
s_1 & \approx & \texttt{0} : \texttt{0} : (\texttt{letrec}\ xs{=}\texttt{0} : xs\ \texttt{in}\ xs) \\
s_2 & \approx & \texttt{letrec}\ xs{=}\texttt{0}^{[a\mapsto 1]} : \texttt{0}^{[a\mapsto 1]} : xs\ \texttt{in}\ xs \\
s_3 & := & \texttt{letrec}\ y{=}((\lambda x.x)\ \texttt{0}), xs{=}y : ((\lambda x.x)\ y) : xs\ \texttt{in}\ xs \\
s_4 & := & \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ \texttt{0})
\end{array}
$$

Work-decorations to keep track of `rln`-work:

- ${}^{[n]} := n$ essential reduction steps
- ${}^{[a\mapsto n]} := n$ shared essential reduction steps
  (label a marks the sharing)

Motivation: Reasoning including Resources

Equational reasoning w.r.t. cost equivalence:

$$s_1 \approx \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : xs\ \mathtt{in}\ xs)$$
$$s_2 \approx \mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1]} : (\mathtt{letrec}\ xs{=}\mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1]} : xs\ \mathtt{in}\ xs)$$
$$s_3 := \mathtt{letrec}\ y{=}((\lambda x.x)\ \mathtt{0}), xs{=}y : ((\lambda x.x)\ y) : xs\ \mathtt{in}\ xs$$
$$s_4 := \mathtt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \mathtt{in}\ (xs\ \mathtt{0})$$

Work-decorations to keep track of `rln`-work:

- $^{[n]} := n$ essential reduction steps
- $^{[a\mapsto n]} := n$ shared essential reduction steps
  (label a marks the sharing)

Motivation: Reasoning including Resources

Equational reasoning w.r.t. cost equivalence:

$$
\begin{array}{lcl}
s_1 & \approx & \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : xs\ \mathtt{in}\ xs) \\
s_2 & \approx & \mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1]} : (\mathtt{letrec}\ xs{=}\mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1]} : xs\ \mathtt{in}\ xs) \\
s_3 & := & \mathtt{letrec}\ y{=}((\lambda x.x)\ \mathtt{0}), xs{=}y : ((\lambda x.x)\ y) : xs\ \mathtt{in}\ xs \\
s_4 & := & \mathtt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \mathtt{in}\ (xs\ \mathtt{0})
\end{array}
$$

Work-decorations to keep track of `rln`-work:

- $^{[n]} := n$ essential reduction steps
- $^{[a\mapsto n]} := n$ shared essential reduction steps
  (label a marks the sharing)

Motivation: Reasoning including Resources

Equational reasoning w.r.t. cost equivalence:

$$
\begin{array}{lll}
s_1 & \approx & 0 : 0 : (\texttt{letrec}\ xs{=}0 : xs\ \texttt{in}\ xs) \\
s_2 & \approx & 0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : xs\ \texttt{in}\ xs) \\
s_3 & \approx & \texttt{letrec}\ y{=}0^{[1]}, xs{=}y : ((\lambda x.x)\ y) : xs\ \texttt{in}\ xs \\
s_4 & := & \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ 0)
\end{array}
$$

Work-decorations to keep track of `rln`-work:

- $^{[n]} := n$ essential reduction steps
- $^{[a\mapsto n]} := n$ shared essential reduction steps
  (label a marks the sharing)

Motivation: Reasoning including Resources

Equational reasoning w.r.t. cost equivalence:

$$
\begin{array}{lll}
s_1 & \approx & 0 : 0 : (\texttt{letrec}\ xs{=}0 : xs\ \texttt{in}\ xs) \\
s_2 & \approx & 0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : xs\ \texttt{in}\ xs) \\
s_3 & \approx & \texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : ((\lambda x.x)\ 0^{[a\mapsto 1]}) : xs\ \texttt{in}\ xs \\
s_4 & := & \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ 0)
\end{array}
$$

Work-decorations to keep track of `rln`-work:

- $^{[n]} := n$ essential reduction steps
- $^{[a\mapsto n]} := n$ shared essential reduction steps
  (label a marks the sharing)

Motivation: Reasoning including Resources

Equational reasoning w.r.t. cost equivalence:

$$
\begin{array}{lcl}
s_1 & \approx & \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : xs\ \mathtt{in}\ xs) \\
s_2 & \approx & \mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1]} : (\mathtt{letrec}\ xs{=}\mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1]} : xs\ \mathtt{in}\ xs) \\
s_3 & \approx & \mathtt{letrec}\ xs{=}\mathtt{0}^{[a\mapsto 1]} : (\mathtt{0}^{[a\mapsto 1]})^{[1]} : xs\ \mathtt{in}\ xs \\
s_4 & := & \mathtt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \mathtt{in}\ (xs\ \mathtt{0})
\end{array}
$$

Work-decorations to keep track of `rln`-work:

- $^{[n]}$ $:= n$ essential reduction steps
- $^{[a\mapsto n]}$ $:= n$ shared essential reduction steps
  (label a marks the sharing)

Motivation: Reasoning including Resources

Equational reasoning w.r.t. cost equivalence:

$$s_1 \approx 0 : 0 : (\texttt{letrec}\ xs{=}0 : xs\ \texttt{in}\ xs)$$
$$s_2 \approx 0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : xs\ \texttt{in}\ xs)$$
$$s_3 \approx 0^{[a\mapsto 1]} : 0^{[a\mapsto 1, b\mapsto 1]} : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1, b\mapsto 1]} : xs\ \texttt{in}\ xs)$$
$$s_4 := \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ 0)$$

Work-decorations to keep track of `rln`-work:

- $^{[n]}$ $:= n$ essential reduction steps
- $^{[a\mapsto n]}$ $:= n$ shared essential reduction steps
  (label a marks the sharing)

Motivation: Reasoning including Resources

Equational reasoning w.r.t. cost equivalence:

$$
\begin{array}{lll}
s_1 & \approx & 0 : 0 : (\texttt{letrec}\ xs{=}0 : xs\ \texttt{in}\ xs) \\
s_2 & \approx & 0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : xs\ \texttt{in}\ xs) \\
s_3 & \approx & 0^{[a\mapsto 1]} : (0^{[a\mapsto 1, b\mapsto 1]}) : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1, b\mapsto 1]} : xs\ \texttt{in}\ xs) \\
s_4 & := & \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ 0)
\end{array}
$$

Work-decorations to keep track of `rln`-work:

- $^{[n]}$ $:=$ n essential reduction steps
- $^{[a\mapsto n]}$ $:=$ n shared essential reduction steps (label a marks the sharing)

Motivation: Reasoning including Resources

Equational reasoning w.r.t. cost equivalence:

$$s_1 \approx \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : xs\ \mathtt{in}\ xs)$$
$$s_2 \approx \mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1]} : (\mathtt{letrec}\ xs{=}\mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1]} : xs\ \mathtt{in}\ xs)$$
$$s_3 \approx \mathtt{0}^{[a\mapsto 1]} : (\mathtt{0}^{[a\mapsto 1, b\mapsto 1]}) : (\mathtt{letrec}\ xs{=}\mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1, b\mapsto 1]} : xs\ \mathtt{in}\ xs)$$
$$s_4 \approx \mathtt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \mathtt{in}\ ((\lambda y.y : (xs\ y))\ \mathtt{0})$$

Work-decorations to keep track of `rln`-work:

- $^{[n]} := n$ essential reduction steps
- $^{[a\mapsto n]} := n$ shared essential reduction steps
  (label a marks the sharing)

Motivation: Reasoning including Resources

Equational reasoning w.r.t. cost equivalence:

$$s_1 \approx 0 : 0 : (\texttt{letrec}\ xs{=}0 : xs\ \texttt{in}\ xs)$$
$$s_2 \approx 0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : xs\ \texttt{in}\ xs)$$
$$s_3 \approx 0^{[a\mapsto 1]} : (0^{[a\mapsto 1, b\mapsto 1]}) : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1, b\mapsto 1]} : xs\ \texttt{in}\ xs)$$
$$s_4 \approx \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (0 : (xs\ 0))^{[1]}$$

Work-decorations to keep track of `rln`-work:

- $^{[n]} := n$ essential reduction steps
- $^{[a\mapsto n]} := n$ shared essential reduction steps
  (label a marks the sharing)

Motivation: Reasoning including Resources

Equational reasoning w.r.t. cost equivalence:

$s_1 \approx 0 : 0 : (\texttt{letrec}\ xs{=}0 : xs\ \texttt{in}\ xs)$
$s_2 \approx 0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : xs\ \texttt{in}\ xs)$
$s_3 \approx 0^{[a\mapsto 1]} : (0^{[a\mapsto 1, b\mapsto 1]}) : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1, b\mapsto 1]} : xs\ \texttt{in}\ xs)$
$s_4 \approx \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (0 : ((\lambda y.y : (xs\ y))\ 0))^{[1]}$

Work-decorations to keep track of `rln`-work:

- $^{[n]} := n$ essential reduction steps
- $^{[a\mapsto n]} := n$ shared essential reduction steps
  (label a marks the sharing)

Motivation: Reasoning including Resources

Equational reasoning w.r.t. cost equivalence:

$s_1 \approx 0 : 0 : (\texttt{letrec}\ xs{=}0 : xs\ \texttt{in}\ xs)$

$s_2 \approx 0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : xs\ \texttt{in}\ xs)$

$s_3 \approx 0^{[a\mapsto 1]} : (0^{[a\mapsto 1, b\mapsto 1]}) : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1, b\mapsto 1]} : xs\ \texttt{in}\ xs)$

$s_4 \approx \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (0 : (0 : (xs\ 0))^{[1]})^{[1]}$

Work-decorations to keep track of $\texttt{rln}$-work:

- $^{[n]} := n$ essential reduction steps
- $^{[a\mapsto n]} := n$ shared essential reduction steps
  (label a marks the sharing)

Motivation: Reasoning including Resources

Equational reasoning w.r.t. cost equivalence:

$s_1 \approx \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : xs\ \mathtt{in}\ xs)$

$s_2 \approx \mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1]} : (\mathtt{letrec}\ xs{=}\mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1]} : xs\ \mathtt{in}\ xs)$

$s_3 \approx \mathtt{0}^{[a\mapsto 1]} : (\mathtt{0}^{[a\mapsto 1, b\mapsto 1]}) : (\mathtt{letrec}\ xs{=}\mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1, b\mapsto 1]} : xs\ \mathtt{in}\ xs)$

$s_4 \approx (0 : (0 : \mathtt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \mathtt{in}\ (xs\ 0))^{[1]})^{[1]}$

Work-decorations to keep track of `rln`-work:

- $^{[n]} := n$ essential reduction steps
- $^{[a\mapsto n]} := n$ shared essential reduction steps
  (label a marks the sharing)

Equational reasoning w.r.t. cost equivalence:

$$
\begin{array}{ll}
s_1 & \approx \mathtt{0} : \mathtt{0} : (\mathtt{letrec}\ xs{=}\mathtt{0} : xs\ \mathtt{in}\ xs) \\
s_2 & \approx \mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1]} : (\mathtt{letrec}\ xs{=}\mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1]} : xs\ \mathtt{in}\ xs) \\
s_3 & \approx \mathtt{0}^{[a\mapsto 1]} : (\mathtt{0}^{[a\mapsto 1, b\mapsto 1]}) : (\mathtt{letrec}\ xs{=}\mathtt{0}^{[a\mapsto 1]} : \mathtt{0}^{[a\mapsto 1, b\mapsto 1]} : xs\ \mathtt{in}\ xs) \\
s_4 & \approx (0 : (0 : \mathtt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \mathtt{in}\ (xs\ 0))^{[1]})^{[1]}
\end{array}
$$

further processing shows $s_1 \preceq s_2 \preceq s_3 \preceq s_4$

Work-decorations to keep track of `rln`-work:

- $^{[n]} := n$ essential reduction steps
- $^{[a\mapsto n]} := n$ shared essential reduction steps
  (label a marks the sharing)

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Equational reasoning w.r.t. cost equivalence:

$$s_1 \approx 0 : 0 : (\texttt{letrec}\ xs{=}0 : xs\ \texttt{in}\ xs)$$
$$s_2 \approx 0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1]} : xs\ \texttt{in}\ xs)$$
$$s_3 \approx 0^{[a\mapsto 1]} : (0^{[a\mapsto 1, b\mapsto 1]}) : (\texttt{letrec}\ xs{=}0^{[a\mapsto 1]} : 0^{[a\mapsto 1, b\mapsto 1]} : xs\ \texttt{in}\ xs)$$
$$s_4 \approx (0 : (0 : \texttt{letrec}\ xs{=}\lambda y.y : (xs\ y)\ \texttt{in}\ (xs\ 0))^{[1]})^{[1]}$$

further processing shows $s_1 \preceq s_2 \preceq s_3 \preceq s_4$

Work-decorations to keep track of `rln`-work:

- $^{[n]}$ $:=$ n essential reduction steps
- $^{[a\mapsto n]}$ $:=$ n shared essential reduction steps
  (label a marks the sharing)

Goal of current paper:

Define and analyze the exact semantics of $^{[a\mapsto n]}$

Our Contribution

- Exact semantics of (shared) work-decorations $^{[a \mapsto n]}$ (and $^{[n]}$)
- Prove **computation rules**, like $S[s^{[a \mapsto n]}, t^{[a \mapsto n]}] \preceq S[s,t]^{[n]}$
- The notation $^{[a \mapsto n]}$ is **ambiguous**, e.g. in
  `letrec` $x{=}\lambda y.s^{[a \mapsto n]}$ `in` $C[x]$ when inlining the binding for x:

  Possibilities:
  - `letrec` $x{=}\lambda y.s^{[a \mapsto n]}$ `in` $C[\lambda y.s^{[a \mapsto n]}]$
  - `letrec` $x{=}\lambda y.s^{[a \mapsto n]}$ `in` $C[\lambda y.s^{[b \mapsto n]}]$ (where b is fresh)

Our Contribution

- Exact semantics of (shared) work-decorations $^{[a \mapsto n]}$ (and $^{[n]}$)
- Prove **computation rules**, like $S[s^{[a \mapsto n]}, t^{[a \mapsto n]}] \preceq S[s,t]^{[n]}$
- The notation $^{[a \mapsto n]}$ is **ambiguous**, e.g. in
  $\texttt{letrec}\ x{=}\lambda y.s^{[a \mapsto n]}\ \texttt{in}\ C[x]$ when inlining the binding for x:

  Possibilities:
  - $\texttt{letrec}\ x{=}\lambda y.s^{[a \mapsto n]}\ \texttt{in}\ C[\lambda y.s^{[a \mapsto n]}]$
  - $\texttt{letrec}\ x{=}\lambda y.s^{[a \mapsto n]}\ \texttt{in}\ C[\lambda y.s^{[b \mapsto n]}]$ (where b is fresh)
- We **change the notation** to add a **scoping** for work-decorations:

Instead of $^{[a \mapsto n]}$ we use a **binding** $a := n$ and a **label** $^{[a]}$

Our Contribution

- Exact semantics of (shared) work-decorations $^{[a \mapsto n]}$ (and $^{[n]}$)
- Prove **computation rules**, like $S[s^{[a \mapsto n]}, t^{[a \mapsto n]}] \preceq S[s,t]^{[n]}$
- The notation $^{[a \mapsto n]}$ is **ambiguous**, e.g. in
  $\texttt{letrec}\ x{=}\lambda y.s^{[a \mapsto n]}\ \texttt{in}\ C[x]$ when inlining the binding for x:

  Possibilities:
  - $\texttt{letrec}\ x{=}\lambda y.s^{[a \mapsto n]}\ \texttt{in}\ C[\lambda y.s^{[a \mapsto n]}]$
  - $\texttt{letrec}\ x{=}\lambda y.s^{[a \mapsto n]}\ \texttt{in}\ C[\lambda y.s^{[b \mapsto n]}]$ (where b is fresh)
- We **change the notation** to add a **scoping** for work-decorations:

> Instead of $^{[a \mapsto n]}$ we use a **binding** $a := n$ and a **label** $^{[a]}$

- Examples:
  - $\texttt{letrec}\ a := n, x{=}\lambda y.s^{[a]}\ \texttt{in}\ C[x]$
  - $\texttt{letrec}\ x{=}\lambda y.(\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]})\ \texttt{in}\ C[x]$

The Calculus LRPw

LRPw extends LRP by work decorations

Types:

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

Expressions:

$$
\begin{array}{lll}
u \in \mathit{PExpr}_F & ::= & \Lambda A_1.\ldots.\Lambda A_k.\lambda x.s \\
s, t \in \mathit{Expr}_F & ::= & u \mid x :: \rho \mid (s\ \tau) \mid (s\ t) \mid (\texttt{seq}\ s\ t) \\
& \mid & (\texttt{letrec}\ \mathit{bind}_1, \ldots, \mathit{bind}_m\ \texttt{in}\ t) \\
& \mid & (c_{K,i} :: \tau\ s_1\ \ldots\ s_{\mathrm{ar}(c_{K,i})}) \\
& \mid & (\texttt{case}_K\ s\ \texttt{of}\ (\mathit{pat}_{K,1}\ \texttt{->}\ t_1) \ldots (\mathit{pat}_{K,|D_K|}\ \texttt{->}\ t_{|D_K|})) \\
& \mid & s^{[a]}, \text{where } a \text{ is a label} \\
\mathit{pat}_{K,i} & ::= & (c_{K,i} :: \tau\ x_1 :: \tau_1 \ldots x_{\mathrm{ar}(c_{K,i})} :: \tau_{\mathrm{ar}(c_{K,i})}) \\
\mathit{bind}_i & ::= & x_i :: \rho_i = s_i \mid a := n, \text{where } n \in \mathbb{N} \text{ and } a \text{ is a label}
\end{array}
$$

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

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

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

...

(letwn) $\texttt{letrec}\ \ldots a := n \ldots C[(s^{[a]})] \to \texttt{letrec}\ \ldots a := n{-}1\ \ldots\ C[s^{[a]}]$

(letw0) $\texttt{letrec}\ \ldots a := 0 \ldots C[(s^{[a]})] \to \texttt{letrec}\ \ldots a := 0 \ldots C[s]$

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 $s \xrightarrow{\mathrm{LRPw},*} t \wedge t$ 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 in LRPw

Counting Essential Reductions

For $\{lbeta, letwn\} = A_0 \subseteq A \subseteq \mathfrak{A} = \{lbeta, case, seq, letwn\}$:

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

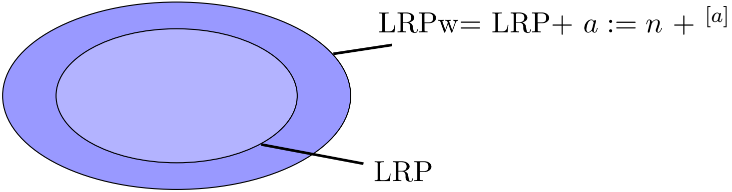
Improvement Relation

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

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

We write $s \approx_A t \iff s \preceq_A t \wedge t \preceq_A s$ (**cost-equivalence**)

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



Questions:

- Is there a change w.r.t. contextual equivalence?
- Is there a change w.r.t. improvement and cost-equivalence?

No, since:

Theorem

The embedding of LRP into LRPw w.r.t. $\sim_c$ is conservative and the calculi LRP and LRPw are isomorphic: The isomorphism is $[s]_{\sim_{c,\mathrm{LRPw}}} = [\mathrm{rmw}(s)]_{\sim_{c,\mathrm{LRP}}}$ where $\mathrm{rmw}(\cdot)$ removes the decorations.

No, if (seq)-reductions do not count for `rln`:

Theorem

Let $A_0 \subseteq A \subset \mathfrak{A}$, such that $seq \notin A$.
Then LRP and LRPw **are isomorphic w.r.t. $\approx_A$**.

Encode `letrec` $a := n, Env$ `in` s as
`letrec` $x_a :=$ `id`$^{n+1}, Env[$`seq` $x_a\ t/t^{[a]}]$ `in` $s[$`seq` $x_a\ t/t^{[a]}]$

Q2: Is there a change w.r.t. cost-equivalence?

No, if (seq)-reductions do not count for `rln`:

Theorem

Let $A_0 \subseteq A \subset \mathfrak{A}$, such that $seq \notin A$.
Then LRP and LRPw **are isomorphic w.r.t.** $\approx_A$.

Encode $\texttt{letrec}\ a := n, Env\ \texttt{in}\ s$ as
$\texttt{letrec}\ x_a := \texttt{id}^{n+1}, Env[\texttt{seq}\ x_a\ t/t^{[a]}]\ \texttt{in}\ s[\texttt{seq}\ x_a\ t/t^{[a]}]$

Yes, for the isomorphism property if $A = \mathfrak{A}$

Proposition

Let $A = \mathfrak{A}$ and let c_1 and c_2 be different constants. Then

$$\texttt{letrec}\ a := 1\ \texttt{in}\ (\texttt{Pair}\ c_1^{[a]}\ c_2^{[a]})$$

is **not equivalent w.r.t.** $\approx_A$ to any LRP-expression.

Q2: Is there a change w.r.t. cost-equivalence?

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

No, if (seq)-reductions do not count for `rln`:

Theorem

Let $A_0 \subseteq A \subset \mathfrak{A}$, such that $seq \notin A$.
Then LRP and LRPw **are isomorphic w.r.t. $\approx_A$**.

Encode $\texttt{letrec}\ a := n, Env\ \texttt{in}\ s$ as
$\texttt{letrec}\ x_a := \texttt{id}^{n+1}, Env[\texttt{seq}\ x_a\ t/t^{[a]}]\ \texttt{in}\ s[\texttt{seq}\ x_a\ t/t^{[a]}]$

Yes, for the isomorphism property if $A = \mathfrak{A}$

Proposition

Let $A = \mathfrak{A}$ and let c_1 and c_2 be different constants. Then

$$\texttt{letrec}\ a := 1\ \texttt{in}\ (\texttt{Pair}\ c_1^{[a]}\ c_2^{[a]})$$

is **not equivalent w.r.t. $\approx_A$** to any LRP-expression.

Open for conservativity: $s \approx_{\mathfrak{A},\mathrm{LRP}} t \implies s \approx_{\mathfrak{A},\mathrm{LRPw}} t$?

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Theorem

Let $A_0 \subseteq A \subseteq \mathfrak{A}$.

- If $s \xrightarrow{\text{LRPw},a} t$ where $a \in A$ then $s \approx_A t^{[1]}$
- If $s \xrightarrow{C,a} t$ where $a \in A$ then $t \preceq_A s$
- If $s \xrightarrow{C,a} t$, a is (lll), (cp), (letw0), (cpx), (cpcx), (abs), (abse), (lwas), (ucp), (gc), (gcW), then $s \approx_A t$

(lll) $\texttt{letrec}\ Env_1\ \texttt{in}\ \texttt{letrec}\ Env_2\ \texttt{in}\ s \rightarrow \texttt{letrec}\ Env_1, Env_2\ \texttt{in}\ s$

(lll) $\texttt{letrec}\ Env_1, x{=}(\texttt{letrec}\ Env_2\ \texttt{in}\ s)\ \texttt{in}\ t \rightarrow \texttt{letrec}\ Env_1, Env_2, x{=}s\ \texttt{in}\ t$

(lll) $(\texttt{letrec}\ Env\ \texttt{in}\ s)\ t \rightarrow \texttt{letrec}\ Env\ \texttt{in}\ (s\ t)$

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

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

(gcW) $\texttt{letrec}\ \{a_i := n_i\}_{i=1}^{m}, Env\ \texttt{in}\ s \rightarrow \texttt{letrec}\ Env\ \texttt{in}\ s$, if all a_i do not occur in Env, s

(gcW) $\texttt{letrec}\ \{a_i := n_i\}_{i=1}^{m}\ \texttt{in}\ s \rightarrow s$, if $a_1, \ldots, a_m$ do not occur in s

(cpx) $\texttt{letrec}\ x{=}y, \ldots C[x] \ldots \rightarrow \texttt{letrec}\ x{=}y, \ldots C[y] \ldots$

(cpcx) $\texttt{letrec}\ x{=}(c\,t_1 \ldots t_n) \ldots C[x] \ldots \rightarrow \texttt{letrec}\ x{=}(c\,y_1 \ldots y_n), \{y_i{=}t_i\}_{i=1}^{n} \ldots C[c\,y_1 \ldots y_n] \ldots$

...

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Theorem

Let $A_0 \subseteq A \subseteq \mathfrak{A}$ and S, T be surface contexts

1. $(s^{[n]})^{[m]} \approx_A s^{[n+m]}$
2. $\texttt{letrec}\ a := n\ \texttt{in}\ (s^{[a]})^{[a]} \approx_A \texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}$
3. $S[\texttt{letrec}\ a := n\ \texttt{in}\ T[s^{[a]}]] \preceq_A \texttt{letrec}\ a := n\ \texttt{in}\ S[T[s]]^{[a]}$
4. $S[\texttt{letrec}\ a := n\ \texttt{in}\ T[s^{[a]}]] \approx_A \texttt{letrec}\ a := n\ \texttt{in}\ S[T[s]]^{[a]}$, if $S[T]$ is strict.
5. $\texttt{letrec}\ a := n, b := m\ \texttt{in}\ (s^{[a]})^{[b]} \approx_A \texttt{letrec}\ a := n, b := m\ \texttt{in}\ (s^{[b]})^{[a]}$
6. $\texttt{letrec}\ a := n\ \texttt{in}\ S[s_1^{[a]}, \ldots, s_n^{[a]}] \preceq_A \texttt{letrec}\ a := n\ \texttt{in}\ S[s_1, \ldots, s_n]^{[a]}$.
7. $\texttt{letrec}\ a := n\ \texttt{in}\ S[s_1^{[a]}, \ldots, s_n^{[a]}] \approx_A \texttt{letrec}\ a := n\ \texttt{in}\ S[s_1, \ldots, s_n]^{[a]}$, if some hole in S is in strict position

Conclusion

- LRPw = Call-by-need calculus with **scoped work-decorations**
- LRPw **not obviously encodable** in LRP
- Several **improvements and cost-equivalences** hold in LRPw
- Expected **computation rules** hold in LRPw

Further Work

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- Apply the results to prove **further improvements and cost-equivalences**
- **Automation** of **program optimization**
- **Automation** of **proving** improvement
- Space-improvements