Sharing Decorations for Improvements in a Functional Core Language with Call-By-Need Operational Semantics

Manfred Schmidt-Schauß and David Sabel

Goethe-University, Frankfurt, Germany

Technical Report Frank-56

Research group for Artificial Intelligence and Software Technology
Institut für Informatik,
Fachbereich Informatik und Mathematik,
Johann Wolfgang Goethe-Universität,
Postfach 11 19 32, D-60054 Frankfurt, Germany

September 7, 2015

Abstract. This report documents the extension LRPw of LRP by sharing decorations. We show correctness of improvement properties of reduction and transformation rules and also of computation rules for decorations in the extended calculus LRPw. We conjecture that conservativity of the embedding of LRP in LRPw holds.

1 Introduction

In this technical report we consider improvements in the polymorphically typed, extended call-by-need functional language LRP and its extension by shared-worked decorations LRPw. The goal of the report is to show that known improvement laws for LRP also hold in the extended calculus LRPw, that a context lemma for improvement holds in LRPw and that several computation rules which simplify the reasoning with decorated expressions are invariant w.r.t. the improvement relation. The results of this report allow to use shared work decorations as a reasoning tool, e.g. for proving improvement laws on list-processing expressions and functions.

For reasoning on the correctness of program transformations, a notion of program semantics is required. We adopt the well-known and natural notion of contextual equivalence for our investigations: Contextual equivalence identifies two programs as equal if exchanging one program by the other program in any surrounding larger program (the so-called context) is not observable. Due to the quantification over all contexts it is sufficient to only observe the termination behavior of the programs, since e.g. different values like `True` and `False` can be distinguished by plugging them into a context C s.t. $C[\texttt{True}]$ terminates while $C[\texttt{False}]$ diverges. A program transformation is correct if it preserves the semantics, i.e. it preserves contextual equivalence. For reasoning whether program transformation are also optimizations, i.e. so-called *improvements*, we adopt the improvement theory originally invented by Moran and Sands [2], but slightly modified and adapted in [7] for the calculus LR. The calculus LR [9] is a untyped call-by-need lambda calculus extended by data-constructors, `case`-expressions, `seq`-expressions, and `letrec`-expressions. This calculus e.g. models the (untyped) core language of Haskell. In [9] the calculus LR was introduced and analyzed in the setting of a strictness analysis using abstract reduction where also several results on the reduction length w.r.t. program transformation were proved. The calculus LRP is the polymorphically typed variant of LR. Typing in LRP is by `let`-polymorphism [3, 4, 1, 10]. Polymorphism is made explicit in the syntax and there are also reduction rules for computing the specific types of functions. The type erasure of reduction sequences exactly leads to the untyped reduction sequences in LR, so that the untyped and typed calculus are compatible. The transfer of the results on improvement in LR to LRP is straightforward and can be found in [8].

In [2] a tick-algebra was introduced to prove correctness of improvement laws in a modular way. A tick $\checkmark^n$ can be attached to an expression to add a fixed amount of work to the expression (i.e. n execution steps). Several laws for computing with ticks are formulated and proved correct. In this paper we introduce the calculus LRPw which extends LRP in a similar way, where ticks are called decorations, but they are extended to a formalism that can express *work* which is *shared* between several subexpressions, which makes reasoning more comfortable and also more exact. In LRPw there are the two new (compared to LRP) constructs: Bindings of the form $a := n$ and decorations of the form $s^{[a]}$. Here $s^{[a]}$ means that the work expressed by the binding for a (i.e. n essential steps, if the binding is $a := n$) has to be done before the expression s can be further evaluated. If decoration a occur at several subexpression, then the work is shared between the subexpressions (and thus at most performed once). The bindings $a := n$ occur in usual `letrec`-expressions and thus also define the scope of the sharing, and a notion of α-equivalence w.r.t. the labels a. This makes a formal treatment possible. As shorthand notation we will use the notation $s^{[a \mapsto n]}$ for shared work. However, this notation is imprecise and requires a definition of its semantics in the calculus LRPw (to fix the scoping of a).

As an example for the usefulness of shared-work decoration, consider the expression $\texttt{let from x = x : (from (x + 1)) in from (2 * 21)}$ which generates an infinite list of numbers $[\texttt{42}, \texttt{43}, \ldots]$. For simplicity in this example we assume a work amount of 1 for arithmetic operations. The work for computing the product `(2*21)` is shared between all list elements, which can be expressed by our decorations: we can rewrite this list as $(\texttt{42}^{[a\mapsto 1]}\texttt{: let from x = x : (from (x + 1)) in from (43}^{[1,a\mapsto 1]})^{[\mathsf{a'}\mapsto 1]}$ which exactly shows that there is shared work between the head and the tail of the list. Clearly, this can be iterated for further partial evaluation of the tail. Moreover, since we provide computation rules for the shared decorations, we can further compute with the decorations. Using the tick-notation of [2] such exact computations seem to be impossible.

We develop the improvement theory in the calculus LRPw and prove correctness and result w.r.t. improvement for the reduction rules and for several other program transformations. We develop computation rules for the shared-work decoration and proof their soundness.

Outline. Section 2 introduces the different calculi LRP and LRPw, and transfers the basic definitions, lemmas and correctness proofs of program transformations from LRP to LRPw. Section 3 defines the work decorations and proves a theorem that provides several computation rules for work decorations. Section 4 contains a proof that an improvement simulation on lists is correct for improvement and can be used as a tool. Some lengthy proofs can be found in the appendix.

2 The Polymorphically Typed Lazy Lambda Calculus LRPw

The extended call-by-need lambda calculus LRP (see e.g. [8, 6]), is a polymorphically typed variant of the calculus LR [9].

The calculus LRPw extends the calculus LRP by shared work decorations, where the decoration of the shared position is explicitely represented by two new constructs: There are new `letrec`-bindings $a_i := n_i$ meaning that a work load of n essential reduction steps is associated with label a where the shared position is the top of the `letrec`-expression, the construct $s^{[a]}$ means that before expression s can be evaluated the work associated with label a has to be evaluated.

Let $\mathcal{K}$ be a fixed set of type constructors, s.t. every $K \in \mathcal{K}$ has an arity $ar(K) \geq 0$ and an associated finite, non-empty set D_K of data constructors, s.t. every $c_{K,i} \in D_K$ has an arity $ar(c_{K,i}) \geq 0$. We assume that $\mathcal{K}$ includes type constructors for lists, pairs and Booleans together with the data constructors `Nil` and `Cons`, where we often use the Haskell notation of an infix colon; pairs as mixfix brackets, and the constants `True` and `False`.

The syntax of expressions and types of LRPw is defined in Fig. 1, where we assume that variables have a fixed type, written as $x :: \rho$. The calculus LRPw extends the lambda-calculus by recursive let-expressions, data constructors, `case`-expressions (for every type constructor K), `seq`-expressions and by type abstractions $\Lambda a.s$ and type applications $(s\ \tau)$ in order to express polymorphic functions and

Variables: We assume type variables $a, a_i \in TVar$ and term variables $x, x_i \in Var$

Labels: We assume label names a, b, a_i, b_i used for sharing work.

Types: Types Typ and polymorphic types $PTyp$ are generated by the following grammar:

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

Expressions: Expression $Expr_F$, patterns $pat_{K,i}$, and polymorphic abstractions $PExpr_F$ are generated by the following grammar:

$$\begin{array}{ll} s, t \in Expr_F & ::= u \mid x :: \rho \mid (s\ \tau) \mid (s\ t) \mid (\texttt{seq}\ s\ t) \mid (\texttt{letrec}\ Bind_1, \ldots, Bind_n\ \texttt{in}\ t) \\ & \quad \mid (s^{[a]}) \\ & \quad \mid (c_{K,i} :: \tau\ s_1\ \ldots\ s_{ar(c_{K,i})}) \mid (\texttt{case}_K\ s\ \texttt{of}\ (pat_{K,1}\ \texttt{->}\ t_1) \ldots (pat_{K,|D_K|}\ \texttt{->}\ t_{|D_K|})) \\ pat_{K,i} & ::= (c_{K,i} :: \tau\ x_1 :: \tau_1 \ldots x_{ar(c_{K,i})} :: \tau_{ar(c_{K,i})}) \\ Bind_i & ::= x_i :: \rho_i = s_1 \mid a_i := n_i \quad \text{where } a_i \text{ is a label and } n_i \text{ is a nonnegative integer} \\ u \in PExpr_F & ::= \Lambda a_1. \ldots .\Lambda a_k.\lambda x.s \end{array}$$

Typing rules:

$$\frac{s :: \rho}{\Lambda a.s :: \boldsymbol{\lambda} a.\rho} \qquad \frac{s :: \tau_1 \quad pat_i :: \tau_1 \quad t_i :: \tau_2}{(\texttt{case}_K\ s\ \texttt{of}\ (pat_1\ \texttt{->}\ t_1) \ldots (pat_{|D_K|}\ \texttt{->}\ t_{|D_K|})) :: \tau_2} \qquad \frac{s :: \tau_2}{(\lambda x :: \tau_1.s) :: \tau_1 \to \tau_2}$$

$$\frac{s :: \boldsymbol{\lambda} a.\rho}{(s\ \tau) :: \rho[\tau/a]} \qquad \frac{s :: \tau_1 \to \tau_2 \quad t :: \tau_1}{(s\ t) :: \tau_2}$$

$$\frac{s :: \tau}{(\texttt{letrec}\ a_1 := n_1, \ldots, a_m := n_m\ \texttt{in}\ s) :: \tau} \qquad \frac{(\texttt{letrec}\ Env\ \texttt{in}\ s) :: \tau}{(\texttt{letrec}\ Env, a_1 := n_1, \ldots, a_m := n_m\ \texttt{in}\ s) :: \tau}$$

$$\frac{s_1 :: \rho_1 \quad \ldots \quad s_n :: \rho_n \quad t :: \rho}{(\texttt{letrec}\ x_1 :: \rho_1 = s_1, \ldots, x_n :: \rho_n = s_n\ \texttt{in}\ t) :: \rho}$$

$$\frac{s :: \tau \quad t :: \tau'}{(\texttt{seq}\ s\ t) :: \tau'} \qquad \frac{\begin{array}{cc} s_1 :: \tau_1, \ldots, s_{ar(c)} :: \tau_{ar(c)} & \tau = \tau_1 \to \ldots \to \tau_{ar(c)} \to \tau_{ar(c)+1} \\ type(c) = \boldsymbol{\lambda} a_1, \ldots, a_m.\tau'' & \exists \tau'_1, \ldots, \tau'_m : \tau''[\tau'_1/a_1, \ldots, \tau'_m/a_m] = \tau \end{array}}{(c :: \tau\ s_1 \ldots s_{ar(c)}) :: \tau_{ar(c)+1}}$$

Labeling algorithm: Labeling of s starts with s^{top}. The rules from below are applied until no more labeling is possible or until a fail occurs, where $a \vee b$ means label a or label b.

$$\begin{array}{ll} (s\ t)^{\mathsf{sub}\vee\mathsf{top}} & \to (s^{\mathsf{sub}}\ t)^{\mathsf{vis}} \qquad s \neq \Lambda a.e' \\ ((\Lambda a.u)\ \tau)^{\mathsf{sub}\vee\mathsf{top}} & \to ((\Lambda a.u)^{\mathsf{sub}}\ \tau)^{\mathsf{vis}}; \qquad \text{then stop with success} \\ (\texttt{letrec}\ Env\ \texttt{in}\ s)^{\mathsf{top}} & \to (\texttt{letrec}\ Env\ \texttt{in}\ s^{\mathsf{sub}})^{\mathsf{vis}} \\ (\texttt{letrec}\ x = s, Env\ \texttt{in}\ C[x^{\mathsf{sub}}]) & \to (\texttt{letrec}\ x = s^{\mathsf{sub}}, Env\ \texttt{in}\ C[x^{\mathsf{vis}}]) \\ (\texttt{letrec}\ x = s, y = C[x^{\mathsf{sub}}], Env\ \texttt{in}\ t) & \to (\texttt{letrec}\ x = s^{\mathsf{sub}}, y = C[x^{\mathsf{vis}}], Env\ \texttt{in}\ t), \text{if } C \neq [\cdot] \\ (\texttt{letrec}\ x = s, y = x^{\mathsf{sub}}, Env\ \texttt{in}\ t) & \to (\texttt{letrec}\ x = s^{\mathsf{sub}}, y = x^{\mathsf{nontarg}}, Env\ \texttt{in}\ t) \\ (\texttt{seq}\ s\ t)^{\mathsf{sub}\vee\mathsf{top}} & \to (\texttt{seq}\ s^{\mathsf{sub}}\ t)^{\mathsf{vis}} \\ (\texttt{case}_K\ s\ \texttt{of}\ alts)^{\mathsf{sub}\vee\mathsf{top}} & \to (\texttt{case}_K\ s^{\mathsf{sub}}\ \texttt{of}\ alts)^{\mathsf{vis}} \\ \texttt{letrec}\ x = s^{\mathsf{vis}\vee\mathsf{nontarg}}, y = C[x^{\mathsf{sub}}] \ldots & \to \texttt{Fail} \\ \texttt{letrec}\ x = C[x^{\mathsf{sub}}], Env\ \texttt{in}\ t & \to \texttt{Fail} \end{array}$$

Fig. 1. Syntax of expressions and types, typing rules, and rules for labeling

type instantiation, and by the shared work-decorations $a := n$ and $^{[a]}$. Polymorphically typed variables are only permitted for usual bindings of let-environments; at other places, the language is monomorphic where the concrete types can be computed through type reductions. For example, the identity can be written as $\Lambda a.\lambda x :: a.x$, and an application to the constant $\texttt{True}$ is written $(\Lambda a.\lambda x :: a.x)$ $\texttt{Bool True}$. The reduction is $(\Lambda a.\lambda x :: a.x)$ $\texttt{Bool True} \to (\lambda x :: \texttt{Bool}.x)$ $\texttt{True} \to (\texttt{letrec}\ x = \texttt{True}\ \texttt{in}\ x)$. An expression s is *well-typed* with type τ (polymorphic type ρ, resp.), written as $s :: \tau$ (or $s :: \rho$, resp.), if s can be typed with the typing rules in Fig. 1 with type τ (ρ, resp.).

The calculus LR [9] is the untyped variant of LRPw (without shared work-decorations), where types and type-reduction are removed. In the following we often ignore the types and omit the types at variables and also sometimes omit the type reductions. We use some abbreviations: We write $\lambda x_1, \ldots, x_n.s$ instead of $\lambda x_1. \ldots .\lambda x_n.s$. A $\texttt{letrec}$-environment (or a part of it) is abbreviated by Env, and with $\{x_{g(i)} = s_{f(i)}\}_{i=j}^m$ we abbreviate the bindings $x_{g(j)} = s_{f(j)}, \ldots, x_{g(m)} = s_{g(m)}$. We write $\texttt{if}\ s\ \texttt{then}\ t_1\ \texttt{else}\ t_2$ instead of $\texttt{case}_{\texttt{Bool}}\ s\ \texttt{of}\ (\texttt{True -> } t_1)\ (\texttt{False -> } t_2)$. Alternatives of $\texttt{case}$-expressions are abbreviated by *alts*. Constructor applications $(c_{K,i}\ s_1\ \ldots\ s_{ar(c_{K,i})})$ are abbreviated using vector notation and omitting the index as $c\,\overrightarrow{s}$.

We use $FV(s)$ and $BV(s)$ to denote the free and bound variables of an expression s, and $FN(s)$ and $BN(s)$ to denote the free and bound label-names of an expression s. An expression s is closed iff $FV(s) = \emptyset$ and $FN(s) = \emptyset$. In an environment $Env = \{x_i = t_i\}_{i=1}^n$, we define $LV(Env) = \{x_1, \ldots, x_n\}$.

A *value* is an abstraction $\lambda x.s$, a type abstraction $\Lambda a.u$, or a constructor application $c\,\overrightarrow{s}$.

A *context* C is an expression with exactly one hole $[\cdot]$ at expression position. The reduction rules of the calculus are in Fig. 2. The operational semantics of LRPw is defined by the normal order reduction strategy which is a call-by-need strategy, i.e. a call-by-name strategy adapted to sharing. The labeling algorithm shown in Fig. 1 is used to detect the position to which a reduction rule is applied according to normal order, and the labelings in the expressions in Fig 2 indicate the exact place and positions of the expressions and subexpressions involved in the reduction step. It uses the labels: top,sub,vis,nontarg where top means reduction of the top term, sub means reduction of a subterm, vis marks already visited subexpressions, and nontarg marks already visited variables that are not target of a (cp)-reduction. Note that the labeling algorithm does not descend into sub-labeled $\texttt{letrec}$-expressions. The rules of the labeling algorithm are in Fig. 1. If the labeling algorithm terminates, then we say the termination is successful, and a potential normal order redex is found, which can only be the direct superterm of the sub-marked subexpression. It is possible that there is no normal order reduction: in this case either the evaluation is already finished, or it is a dynamically detected error (like a type-error), or the labeling fails.

Definition 2.1. *Let t be an expression. Then a* normal order reduction step $t \xrightarrow{\text{LRPw}} t'$ *is defined by first applying the labeling algorithm to t, and if the labeling algorithm terminates successfully, then one of the rules in Fig. 2 has to be applied resulting in t', if possible, where the labels* sub, vis *must match the labels in the expression t.*

A weak head normal form (WHNF) *is a value v, or an expression* $\texttt{letrec}$ *Env* $\texttt{in}$ *v, where v is a value, or an expression* $\texttt{letrec}\ x_1 = c\,\overrightarrow{t}, \{x_i = x_{i-1}\}_{i=2}^m, Env\ \texttt{in}\ x_m$.

An expression s converges*, denoted as $s\downarrow_{\text{LRPw}}$, iff there is a normal-order reduction $s \xrightarrow{\text{LRPw},*} s'$, where s' is a WHNF. This may also be denoted as $s \downarrow_{\text{LRPw}} s'$. With $\bot$ we denote a diverging, closed expression.*

The calculus LRP is the subcalculus of LRPw which does not have the syntactic constructs $a := n$ and $s^{[a]}$, and the operational semantics of LRP does not have the reduction rules (letwn) and (letw0). WHNFs are defined as in LRPw. Convergence $\downarrow_{\text{LRP}}$ is defined accordingly.

Lemma 2.2. *For every* LRPw*-expression s which is also an* LRP*-expression (i.e. s has no decorations and no $a := n$-construct): $s\downarrow_{\text{LRPw}} \iff s\downarrow_{\text{LRP}}$.*

Remark 2.3. The relation between the typed reduction in LRP and the untyped reduction in LR [9, 7] is that the removal of types and the reduction (Tbeta) results exactly in the untyped normal-order

(lbeta)	$C[((\lambda x.s)^{\mathsf{sub}}\ r)] \to C[\mathtt{letrec}\ x = r\ \mathtt{in}\ s]$
(Tbeta)	$((\Lambda a.u)^{\mathsf{sub}}\ \tau) \to u[\tau/a]$
(cp-in)	$\mathtt{letrec}\ x_1 = (\lambda y.t)^{\mathsf{sub}}, \{x_i = x_{i-1}\}_{i=2}^m, Env\ \mathtt{in}\ C[x_m^{\mathsf{vis}}] \to \mathtt{letrec}\ x_1 = \lambda y.t, \{x_i = x_{i-1}\}_{i=2}^m, Env\ \mathtt{in}\ C[\lambda y.t]$
(cp-e)	$\mathtt{letrec}\ x_1 = (\lambda y.t)^{\mathsf{sub}}, \{x_i = x_{i-1}\}_{i=2}^m, Env, y = C[x_m^{\mathsf{vis}}]\ \mathtt{in}\ r$ $\to \mathtt{letrec}\ x_1 = \lambda y.t, \{x_i = x_{i-1}\}_{i=2}^m, Env, y = C[\lambda y.t]\ \mathtt{in}\ r$
(llet-in)	$(\mathtt{letrec}\ Env_1\ \mathtt{in}\ (\mathtt{letrec}\ Env_2\ \mathtt{in}\ r)^{\mathsf{sub}}) \to (\mathtt{letrec}\ Env_1, Env_2\ \mathtt{in}\ r)$
(llet-e)	$\mathtt{letrec}\ Env_1, x = (\mathtt{letrec}\ Env_2\ \mathtt{in}\ t)^{\mathsf{sub}}\ \mathtt{in}\ r \to \mathtt{letrec}\ Env_1, Env_2, x = t\ \mathtt{in}\ r$
(lapp)	$C[((\mathtt{letrec}\ Env\ \mathtt{in}\ t)^{\mathsf{sub}}\ s)] \to C[(\mathtt{letrec}\ Env\ \mathtt{in}\ (t\ s))]$
(lcase)	$C[(\mathtt{case}_K\ (\mathtt{letrec}\ Env\ \mathtt{in}\ t)^{\mathsf{sub}}\ alts)] \to C[(\mathtt{letrec}\ Env\ \mathtt{in}\ (\mathtt{case}_K\ t\ alts))]$
(seq-c)	$C[(\mathtt{seq}\ v^{\mathsf{sub}}\ t)] \to C[t]$ if v is a value
(seq-in)	$(\mathtt{letrec}\ x_1 = (c\,\overrightarrow{s})^{\mathsf{sub}}, \{x_i = x_{i-1}\}_{i=2}^m, Env\ \mathtt{in}\ C[(\mathtt{seq}\ x_m^{\mathsf{vis}}\ t)]) \to (\mathtt{letrec}\ x_1 = (c\,\overrightarrow{s}), \{x_i = x_{i-1}\}_{i=2}^m, Env\ \mathtt{in}\ C[t])$
(seq-e)	$(\mathtt{letrec}\ x_1 = (c\,\overrightarrow{s})^{\mathsf{sub}}, \{x_i = x_{i-1}\}_{i=2}^m, Env, y = C[(\mathtt{seq}\ x_m^{\mathsf{vis}}\ t)]\ \mathtt{in}\ r)$ $\to (\mathtt{letrec}\ x_1 = (c\,\overrightarrow{s}), \{x_i = x_{i-1}\}_{i=2}^m, Env, y = C[t]\ \mathtt{in}\ r)$
(lseq)	$C[(\mathtt{seq}\ (\mathtt{letrec}\ Env\ \mathtt{in}\ s)^{\mathsf{sub}}\ t)] \to C[(\mathtt{letrec}\ Env\ \mathtt{in}\ (\mathtt{seq}\ s\ t))]$
(case-c)	$C[\mathtt{case}_K\ (c\,\overrightarrow{t})^{\mathsf{sub}} \ldots ((c\,\overrightarrow{y}) \to t) \ldots] \to C[\mathtt{letrec}\ \{y_i = t_i\}_{i=1}^{ar(c)}\ \mathtt{in}\ t]$ if $ar(c) \geq 1$
(case-c)	$C[(\mathtt{case}_K\ c^{\mathsf{sub}} \ldots (c \to t) \ldots)] \to C[t]$ if $ar(c) = 0$
(case-in)	$\mathtt{letrec}\ x_1 = (c\,\overrightarrow{t})^{\mathsf{sub}}, \{x_i = x_{i-1}\}_{i=2}^m, Env\ \mathtt{in}\ C[\mathtt{case}_K\ x_m^{\mathsf{vis}} \ldots ((c\,\overrightarrow{z}) \to t) \ldots]$ $\to \mathtt{letrec}\ x_1 = (c\,\overrightarrow{y}), \{y_i = t_i\}_{i=1}^{ar(c)}, \{x_i = x_{i-1}\}_{i=2}^m, Env$ $\mathtt{in}\ C[\mathtt{letrec}\ \{z_i = y_i\}_{i=1}^{ar(c)}\ \mathtt{in}\ t]$ if $ar(c) \geq 1$ and where y_i are fresh variables
(case-in)	$\mathtt{letrec}\ x_1 = c^{\mathsf{sub}}, \{x_i = x_{i-1}\}_{i=2}^m, Env\ \mathtt{in}\ C[\mathtt{case}_K\ x_m^{\mathsf{vis}} \ldots (c \to t) \ldots]$ $\to \mathtt{letrec}\ x_1 = c, \{x_i = x_{i-1}\}_{i=2}^m, Env\ \mathtt{in}\ C[t]$ if $ar(c) = 0$
(case-e)	$\mathtt{letrec}\ x_1{=}(c\,\overrightarrow{t})^{\mathsf{sub}}, \{x_i{=}x_{i-1}\}_{i=2}^m, u{=}C[\mathtt{case}_K\ x_m^{\mathsf{vis}} \ldots ((c\,\overrightarrow{z}) \to r) \ldots], Env\ \mathtt{in}\ s$ $\to \mathtt{letrec}\ x_1{=}(c\,\overrightarrow{y}), \{y_i{=}t_i\}_{i=1}^n, \{x_i{=}x_{i-1}\}_{i=2}^m, u{=}C[\mathtt{letrec}\ \{z_i{=}y_i\}_{i=1}^n\ \mathtt{in}\ r], Env$ $\mathtt{in}\ s$ where $n = ar(c) \geq 1$ and y_i are fresh variables
(case-e)	$\mathtt{letrec}\ x_1 = c^{\mathsf{sub}}, \{x_i = x_{i-1}\}_{i=2}^m, u = C[\mathtt{case}_K\ x_m^{\mathsf{vis}}\ \ldots\ (c \to r_1) \ldots], Env\ \mathtt{in}\ r_2$ $\to \mathtt{letrec}\ x_1 = c, \{x_i = x_{i-1}\}_{i=2}^m \ldots, u = C[r_1], Env\ \mathtt{in}\ r_2$ if $ar(c) = 0$
(letwn-in)	$\mathtt{letrec}\ Env, a := n,\ \mathtt{in}\ C[(s^{[a]})^{\mathsf{sub}}] \to \mathtt{letrec}\ Env, a := n-1\ \mathtt{in}\ C[s^{[a]}]$ if $n > 0$
(letwn-e)	$\mathtt{letrec}\ a := n, x = C[(s^{[a]})^{\mathsf{sub}}], Env\ \mathtt{in}\ r \to \mathtt{letrec}\ a := n-1, x = C[s^{[a]}], Env\ \mathtt{in}\ r$ if $n > 0$
(letw0-in)	$\mathtt{letrec}\ Env, a := 0,\ \mathtt{in}\ C[(s^{[a]})^{\mathsf{sub}}] \to \mathtt{letrec}\ Env, a := 0\ \mathtt{in}\ C[s]$
(letw0-e)	$\mathtt{letrec}\ a := 0, x = C[(s^{[a]})^{\mathsf{sub}}], Env\ \mathtt{in}\ r \to \mathtt{letrec}\ a := 0, x = C[s], Env\ \mathtt{in}\ r$

Fig. 2. Reduction rules

reduction. This also holds for WHNFs and the convergence notions. An immediate consequence is that the untyped contextual approximations and equivalences can be inherited to the typed LRP, since the typed contexts are also untyped ones.

We define some special context classes:

Definition 2.4. *A* reduction context *R is any context, such that its hole will be labeled with* sub *or* top *by the labeling algorithm in Fig. 1. A* weak reduction context, *R^-, is a reduction context, where the hole is not within a* `letrec`*-expression.* Surface contexts *S are contexts where the hole is not in an abstraction,* top contexts *T are surface contexts where the hole is not in an alternative of a case, and* weak top contexts *are top contexts where the hole does not occur in a* `letrec`*. A context C is* strict *iff $C[\bot] \sim_c \bot$.*

A program transformation P is binary relation on expressions. We write $s \xrightarrow{P} t$, if $(s,t) \in P$. For a set of contexts X and a transformation P, the transformation (X,P) is the closure of P w.r.t. the contexts in P, i.e. $s \xrightarrow{X,P} t$ iff there exists $C \in X$ with $C[s] \xrightarrow{P} C[t]$.

Definition 2.5. *We define several unions of the program transformations in Figs. 2 (ignoring the labels) and 3: (case) is the union of (case-c), (case-in), (case-e); (seq) is the union of (seq-c), (seq-in), (seq-e); (cp) is the union of (cp-in), (cp-e); (llet) is the union of (llet-in), (llet-e); (lll) is the union of (lapp),(lcase),(lseq),(llet-in),(llet-e); (letwn) is the union of (letwn-in), (letwn-e); (letw0) is the union of (letw0-in), (letw0-e); (letw) is the union of (letwn), (letw0); (gc) is the union of (gc1), (gc2); (cpx)*

(gc1)	$\texttt{letrec}\ \{x_i = s_i\}_{i=1}^n, Env\ \texttt{in}\ t \to \texttt{letrec}\ Env\ \texttt{in}\ t$, if for all $i : x_i \notin FV(t, Env)$
(gc2)	$\texttt{letrec}\ x_1 = s_1, \ldots, x_n = s_n\ \texttt{in}\ t \to t$, if for all $i : x_i \notin FV(t)$
(gcW1)	$\texttt{letrec}\ Env, a_1 := n_1, \ldots, a_m := n_m\ \texttt{in}\ s \to \texttt{letrec}\ Env\ \texttt{in}\ s$, if labels $a_1, \ldots, a_m$ do not occur in Env or t
(gcW2)	$\texttt{letrec}\ a_1 := n_1, \ldots, a_m := n_m\ \texttt{in}\ s \to s$, if labels $a_1, \ldots, a_m$ do not occur in t
(cpx-in)	$\texttt{letrec}\ x = y, Env\ \texttt{in}\ C[x] \to \texttt{letrec}\ x = y, Env\ \texttt{in}\ C[y]$, if $y \in Var, x \neq y$
(cpx-e)	$\texttt{letrec}\ x{=}y, z{=}C[x], Env\ \texttt{in}\ t \to \texttt{letrec}\ x{=}y, z{=}C[y], Env\ \texttt{in}\ t$, if $y \in Var, x \neq y$
(cpax)	$\texttt{letrec}\ x = y, Env\ \texttt{in}\ s \to \texttt{letrec}\ x = y, Env[y/x]\ \texttt{in}\ s[y/x]$, if $y \in Var, x \neq y, y \in FV(s, Env)$
(cpcx-in)	$\texttt{letrec}\ x = c\,\overrightarrow{t}, Env\ \texttt{in}\ C[x] \to \texttt{letrec}\ x = c\,\overrightarrow{y}, \{y_i = t_i\}_{i=1}^{ar(c)}, Env\ \texttt{in}\ C[c\,\overrightarrow{y}]$
(cpcx-e)	$\texttt{letrec}\ x = c\,\overrightarrow{t}, z = C[x], Env\ \texttt{in}\ t$ $\to \texttt{letrec}\ x = c\,\overrightarrow{y}, \{y_i = t_i\}_{i=1}^{ar(c)}, z = C[c\,\overrightarrow{y}], Env\ \texttt{in}\ t$
(abs)	$\texttt{letrec}\ x = c\,\overrightarrow{t}, Env\ \ \texttt{in}\ s \to \texttt{letrec}\ x = c\,\overrightarrow{x}, \{y_i = t_i\}_{i=1}^{ar(c)}, Env\ \texttt{in}\ s$
(abse)	$(c\,\overrightarrow{t}) \to \texttt{letrec}\ \{y_i = t_i\}_{i=1}^{ar(c)}\ \texttt{in}\ c\,\overrightarrow{x}$
(xch)	$\texttt{letrec}\ x = t, y = x, Env\ \texttt{in}\ r\ \to\ \texttt{letrec}\ y = t, x = y, Env\ \texttt{in}\ r$
(lwas)	$T[\texttt{letrec}\ Env\ \texttt{in}\ t] \to \texttt{letrec}\ Env\ \texttt{in}\ T[t]$ if T is a weak top context with hole depth 1
(letsh1)	$\texttt{letrec}\ Env, Env'\ \texttt{in}\ T[s] \to \texttt{letrec}\ Env'\ \texttt{in}\ T[(\texttt{letrec}\ Env\ \texttt{in}\ s)]$
(letsh2)	$\texttt{letrec}\ Env, Env', y = T[s]\ \texttt{in}\ r \to \texttt{letrec}\ Env', y = T[(\texttt{letrec}\ Env\ \texttt{in}\ s)]\ \texttt{in}\ r$
(letsh3)	$\texttt{letrec}\ Env\ \texttt{in}\ T[s] \to T[(\texttt{letrec}\ Env\ \texttt{in}\ s)]$ where in the (letsh)-rules, the variables $LV(Env)$ only occur in s, and T is weak top context that does not bind the variables in Env.
(ucp1)	$\texttt{letrec}\ Env, x = t\ \texttt{in}\ S[x] \to \texttt{letrec}\ Env\ \texttt{in}\ S[t]$
(ucp2)	$\texttt{letrec}\ Env, x = t, y = S[x]\ \texttt{in}\ r \to \texttt{letrec}\ Env, y = S[t]\ \texttt{in}\ r$
(ucp3)	$\texttt{letrec}\ x = t\ \texttt{in}\ S[x] \to S[t]$ where in the (ucp)-rules, $x \notin FV(S, Env, t, r)$ and S is a surface context

Fig. 3. Extra Transformation Rules

is the union of (cpx-in), (cpx-e); (cpcx) is the union of (cpcx-in), (cpcx-e); (letsh) is the union of (letsh1), (letsh2), (letsh3), and (ucp) is the union of (ucp1), (ucp2), (ucp3).

2.1 Improvement in LRP and LRPw

The main measure for estimating the time consumption of computation in this paper is a measure counting *essential* reduction steps in the normal-order reduction of expressions. We omit the type reductions in this measure, since these are always terminating and usually can be omitted after compilation. See [8] for more detailed explanations.

We define the essential reduction length for both calculi:

Definition 2.6. *Let $L \in \{\mathrm{LRP}, \mathrm{LRPw}\}$ and let t be a closed L-expression with $t{\downarrow_L}t_0$. Then $\texttt{rln}(t)$ is the number of (lbeta)-, (case)-, (seq)-, and (in* LRPw*) (letwn)-reductions in the normal order reduction $t{\downarrow_L}t_0$. It is consistent to define the measure as ∞, if $t{\uparrow_L}$. For a reduction $t \xrightarrow{L,*} t'$, we define $\texttt{rln}(t \xrightarrow{L,*} t')$ as the number of (lbeta)-, (case)-, (seq)-, and (in* LRPw*) (letwn)-reductions in it.*

We define contextual equivalence and the improvement relation for both calculi LRPw and LRP:

Definition 2.7. *For $L \in \{\mathrm{LRP}, \mathrm{LRPw}\}$ Let s, t be two L-expressions of the same type ρ.*

- *s is* contextually smaller than *t, $s \leq_{c,L} t$, iff for all L-contexts $C[\cdot :: \rho]$: $C[s]{\downarrow_L} \implies C[t]{\downarrow_L}$.*
- *s and t are* contextually equivalent*, $s \sim_{c,L} t$, iff for all L-contexts $C[\cdot :: \rho]$: $C[s]{\downarrow_L} \iff C[t]{\downarrow_L}$.*
- *s* improves *t, $s \preceq_L t$, iff $s \sim_{c,L} t$ and for all L-contexts $C[\cdot :: \rho]$ s.t. $C[s], C[t]$ are closed: $\texttt{rln}(C[s]) \leq \texttt{rln}(C[t])$. If $s \preceq_L t$ and $t \preceq_L s$, we write $s \approx_L t$.*

A program transformation P is correct *(in L) if $P \subseteq \sim_{c,L}$ and it is an* improvement *iff $\xrightarrow{P} \subseteq (\preceq_L)^{-1}$.*

The following context lemma for contextual equivalence holds in LRP and also in LRPw. The proof is standard, so we omit it.

Lemma 2.8 (Context Lemma for Equivalence). *Let $L \in \{\mathrm{LRP}, \mathrm{LRPw}\}$ and let s, t be L-expressions of the same type. Then $s \ \leq_c \ t$ iff for all $C \in \{R, S, T\}$: $C[s]\downarrow_L \implies C[t]\downarrow_L$.*

Let $\eta \in \{\leq, =, \geq\}$ be a relation on non-negative integers and X be a class of contexts X (we will instantiate X with: all contexts C; all reduction contexts R; all surface contexts S; or all top-contexts T). For expressions s, t of type ρ, let $s \bowtie_{\eta,X} t$ iff for all X-contexts $X[\cdot : \rho]$, s.t. $X[s], X[t]$ are closed: $\mathtt{rln}(X[s]) \ \eta \ \mathtt{rln}(X[t])$. In particular, $\bowtie_{\leq,C} \ = \ \preceq$, $\bowtie_{\geq,C} \ = \ \succeq$, and $\bowtie_{=,C} \ = \ \approx$.

In the following we formulate statements for the calculus LRPw, if not stated otherwise.

The context lemma for improvement shows that it suffices to take reduction contexts into account for proving improvement. Its proof is similar to the ones for context lemmas for contextual equivalence in call-by-need lambda calculi (see [7, 9, 5]).

Lemma 2.9 (Context Lemma for Improvement). *Let s, t be expressions with $s \sim_c t$, $\eta \in \{\leq, =, \geq\}$, and let $X \in \{R, S, T\}$. Then $s \ \bowtie_{\eta,X} \ t$ iff $s \ \bowtie_{\eta,C} \ t$.*

Proof. The proof is nearly a complete copy of the proof of the context lemma for improvement in LRP (see [7]). However, for the sake of completeness we include it:

One direction is trivial. For the other direction we prove a more general claim using multicontexts:

For all $n \geq 0$ and for all $i = 1, \ldots, n$ let s_i, t_i be expressions with $s_i \sim_c t_i$ and $s_i \ \bowtie_{\eta,R} \ t_i$. Then for all multicontexts M with n holes such that $M[s_1, \ldots, s_n]$ and $M[t_1, \ldots, t_n]$ are closed: $\mathtt{rln}(M[s_1, \ldots, s_n]) \ \eta \ \mathtt{rln}(M[t_1, \ldots, t_n])$.

The proof is by induction on the pair (k, k') where k is the number of normal order reductions of $M[s_1, \ldots, s_n]$ to a WHNF, and k' is the number of holes of M. If M (without holes) is a WHNF, then the claim holds. If $M[s_1, \ldots, s_n]$ is a WHNF, and no hole is in a reduction context, then also $M[t_1, \ldots, t_n]$ is a WHNF and $\mathtt{rln}(M[s_1, \ldots, s_n]) = 0 = \mathtt{rln}(M[t_1, \ldots, t_n])$.

If in $M[s_1, \ldots, s_n]$ one s_i is in a reduction context, then one hole, say i of M is in a reduction context and the context $M[t_1, \ldots, t_{i-1}, \cdot, t_{i+1}, \ldots, t_n]$ is a reduction context. By the induction hypothesis, using the multi-context $M[\ldots, \cdot, s_i, \cdot, \ldots]$, we have $\mathtt{rln}(M[s_1, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_n]) \ \ \eta \ \ \mathtt{rln}(M[t_1, \ldots, t_{i-1}, s_i, t_{i+1}, \ldots, t_n])$, and from the assumption we have $\mathtt{rln}(M[t_1, \ldots, t_{i-1}, s_i, t_{i+1}, \ldots, t_n]) \ \ \eta \ \ \mathtt{rln}(M[t_1, \ldots, t_{i-1}, t_i, t_{i+1}, \ldots, t_n])$, and hence $\mathtt{rln}(M[s_1, \ldots, s_n]) \ \ \eta \ \ \mathtt{rln}(M[t_1, \ldots, t_n])$.

If in $M[s_1, \ldots, s_n]$ there is no s_i in a reduction context, then $M[s_1, \ldots, s_n] \xrightarrow{\mathrm{LRPw},a} M'[s'_1, \ldots, s'_{n'}]$, may copy or shift some of the s_i where $s'_j = \rho(s_i)$ for some permutation ρ on variables and on the sharing labels. However, the reduction type is the same for the first step of $M[s_1, \ldots, s_n]$ and $M[t_1, \ldots, t_n]$, i.e. $M[t_1, \ldots, t_n] \xrightarrow{\mathrm{LRPw},a} M'[t'_1, \ldots, t'_{n'}]$ with $(s'_j, t'_j) = (\rho(s_i), \rho(t_i))$. We take for granted that the renaming can be carried through. The $\mathtt{rln}(.)$-count on both sides is $m = 0$ or $m = 1$, depending on a. Thus we can apply the induction hypothesis to $M'[s'_1, \ldots, s'_{n'}]$ and $M[t'_1, \ldots, t'_{n'}]$, and so we have $\mathtt{rln}(M[s_1, \ldots, s_n]) = m + \mathtt{rln}(M'[s'_1, \ldots, s'_{n'}]) \ \eta \ m + \mathtt{rln}(M'[t'_1, \ldots, t'_{n'}]) = \mathtt{rln}(M[t_1, \ldots, t_n])$.

We now use the context lemma and the context lemma for improvement to show several properties about the reduction rules and the additional transformation rules.

Lemma 2.10. *A complete set of forking and commuting diagrams for internal (letw)-transformations applied in reduction contexts can be read off the following diagrams:*

$$\begin{tikzcd} \cdot \arrow[r, "iR{,}b"] \arrow[d, "\mathrm{LRPw}{,}a"'] & \cdot \arrow[d, "\mathrm{LRPw}{,}a"] \\ \cdot \arrow[r, "iR{,}b"'] & \cdot \end{tikzcd}$$

$b \in \{letwn, letw0\},\ a\ arbitrary$

$$\begin{tikzcd} \cdot \arrow[r, "iR{,}letw"] \arrow[d, "\mathrm{LRPw}{,}a"'] & \cdot \arrow[dl, "\mathrm{LRPw}{,}a"] \\ \cdot & \end{tikzcd}$$

$a\ arbitrary$

$$\begin{tikzcd} \cdot \arrow[r, "iR{,}b"] \arrow[d, "\mathrm{LRPw}{,}a"'] & \cdot \arrow[ddl, "\mathrm{LRPw}{,}a"] \\ \cdot \arrow[d, "\mathrm{LRPw}{,}b"'] & \\ \cdot & \end{tikzcd}$$

$b \in \{letwn, letw0\},\ a\ arbitrary$

$$\begin{tikzcd} \cdot \arrow[rr, "iR{,}letw0"] \arrow[d, "\mathrm{LRPw}{,}cp"'] & & \cdot \arrow[d, "\mathrm{LRPw}{,}cp"] \\ \cdot \arrow[r, "iR{,}letw0"'] & \cdot \arrow[r, "iR{,}letw0"'] & \cdot \end{tikzcd}$$

Proof. The first diagram describes the case where the transformation and the normal order reduction commute. It also includes cases where a (letw-in)-transformation is flipped into an (letw-e)-transformation, if the normal order reduction is (LRPw,llet). The second diagram describes the case where the a-labeled expression of the (letw)-transformation is removed by the normal order reduction, which may be the case if the expression is inside an unused alternative of `case` or inside the first argument of `seq`. The third diagram describes the case where the internal (letw)-transformation becomes a normal-order reduction. There are several cases where this may happen, e.g. for expressions of the form `letrec` Env `in` `letrec` $a := n$ `in` $C[s^{[a]}]$ where the normal order reduction is (LRPw,llet). The fourth diagram describes the case where an a-labeled expression is inside an abstraction which is copied by (LRPw,cp). If the transformation is a (letwn), then the transformations commute, but if the transformation is (letw0), then the transformation is duplicated, since it has to remove the a-label twice.

Lemma 2.11. *If* $s \xrightarrow{iR,letw} t$ *then* s *is a WHNF iff* t *is a WHNF.*

Lemma 2.12. *Let* R *be a reduction context and* $s \xrightarrow{letw} t$. *Then* $R[s]\downarrow \iff R[t]\downarrow$.

Proof. We split the proof in several parts:

- $R[s]\downarrow \implies R[t]\downarrow$: Assume that $R[s]\downarrow$ holds, and let $R[s] \xrightarrow{\mathrm{LRPw},k} r$ where r is a WHNF. We show $R[t] \xrightarrow{\mathrm{LRPw},k'} r'$ where r' is a WHNF, and $k' \leq k$. We use induction on k. The base case $k = 0$ is covered by Lemma 2.11. For the induction step let $R[s] \xrightarrow{no} r_1 \xrightarrow{\mathrm{LRPw},k-1} r$. If $R[s] \xrightarrow{\mathrm{LRPw},letw} R[t]$, then $r_1 = R[t]$ and $R[t] \xrightarrow{\mathrm{LRPw},k-1} r$ and thus the claim holds. If the reduction is internal, then apply a forking diagram to $r_1 \xleftarrow{no} R[s] \xrightarrow{\mathrm{LRPw},letw} R[t]$.
  1. If the first diagram is applied, then $r_1 \xrightarrow{iR,letw} r_1'$, $R[t] \xrightarrow{no} r_1'$ and $r_1 \xrightarrow{\mathrm{LRPw},k-1} r$. We apply the induction hypothesis to r_1 and r_1' which shows $r_1' \xrightarrow{\mathrm{LRPw},k''} r'$ where r' is a WHNF and $k'' \leq k-1$. Thus $R[t] \xrightarrow{\mathrm{LRPw},k'} r'$ where r' is a WHNF and $k' \leq k$
  2. If the second diagram is applied, then $R[t] \xrightarrow{no} r_1 \xrightarrow{\mathrm{LRPw},k-1} r$ and thus the claim holds.
  3. If the third diagram is applied, then $R[t] \xrightarrow{no} r_2 \xrightarrow{\mathrm{LRPw},k-2} r$ (where $r_1 \xrightarrow{no} r_2$) and the claim holds.
  4. In case of diagram (4) we apply the induction hypothesis twice for each $(iR, letw)$-transformation, which shows that $R[t] \xrightarrow{\mathrm{LRPw},cp} r_1' \xrightarrow{\mathrm{LRPw},k''} r'$ where r' is a WHNF, $k'' \leq k-1$. Thus the claim holds.
- $R[t]\downarrow \implies R[s]\downarrow$. Let $\#cp(r)$ be the number of (LRPw,cp) reductions in the normal order reductions from r to a WHNF and $\#cp(r) = \infty$ if $r\uparrow$ Assume that $R[t] \xrightarrow{\mathrm{LRPw},k} r$ where r is a WHNF. We show $R[s]\downarrow$ and $\#cp(R[s]) \leq \#cp(R[t])$ by induction on the measure $(\#cp(R[t]), k)$. For the base case (0,0) $R[t]$ is a WHNF and thus by Lemma 2.11 also $R[s]$ is a WHNF and the claim holds. For the induction step let $(l, k) > (0, 0)$. Then $R[t] \xrightarrow{no} t' \xrightarrow{\mathrm{LRPw},k-1} r$ where r is a WHNF. If $R[s] \xrightarrow{\mathrm{LRPw},letw} R[t]$ then the claim holds: $R[s]\downarrow$ and $\#cp(R[s]) = \#cp(R[t])$. If the transformation is internal, then we apply a commuting diagram to $R[s] \xrightarrow{iR,letw} R[t] \xrightarrow{no} t_1$.

1. For the first diagram we have an expression s_1 s.t. $R[s] \xrightarrow{\text{LRPw},a} s_1$, $s_1 \xrightarrow{iR,letw} s_2$ and the measure for t_1 is $(\#cp(t_1), k-1)$ which is strictly smaller than (l, k) (since $\#cp(t_1) \leq l$). Thus we can apply the induction hypothesis and derive $s_1\downarrow$ and $\#cp(s_1) \leq \#cp(t_1)$. This shows $R[s]\downarrow$ and $\#cp(R[s]) \leq \#(R[t])$.
2. For the second diagram the claim obviously holds.
3. For the third diagram, the claim also holds.
4. For the last diagram, we apply the induction hypothesis twice, which is possible since $\#cp(\cdot)$ is strictly decreased.

Theorem 2.13. *The transformations (letw0) and (letwn) are correct.*

Proof. Correctness of the transformation (letw) follows from Lemma 2.12 and the context lemma.

Lemma 2.14. *If $s \xrightarrow{letw0} t$, then for all reduction contexts R, s.t. $R[s]$, $R[t]$ are closed:* $\mathtt{rln}(R[s]) = \mathtt{rln}(R[t])$

Proof. Since $(letw0)$ is correct we know that $\mathtt{rln}(R[s]) = \infty \iff \mathtt{rln}(R[t]) = \infty$. So suppose that $\mathtt{rln}(R[s]) = n$. We show $\mathtt{rln}(R[t]) = n$ by induction on a normal order reduction $R[s] \xrightarrow{\text{LRPw},k} s'$ where s' is a WHNF. The base case is covered by Lemma 2.11. For the induction step, let $R[s] \xrightarrow{no} s_1 \xrightarrow{\text{LRPw},k-1} s'$. If $R[s] \xrightarrow{\text{LRPw},letw0} R[t]$, then $\mathtt{rln}(R[s]) = \mathtt{rln}(R[t]) = \mathtt{rln}(s_1)$ and the claim holds. If the transformation is internal, then we apply a forking diagram to s_1. For the first diagram we have $s_1 \xrightarrow{iR,letw0} t_1$ and we apply the induction hypothesis to s_1 and thus have $\mathtt{rln}(s_1) = \mathtt{rln}(t_1)$. This also shows $\mathtt{rln}(R[s]) = \mathtt{rln}(R[t])$. For the second diagram the claim holds. For the third diagram the claim also holds, since the additional (LRPw,letw0)-reduction in the normal order reduction for $R[s]$ is not counted in the $\mathtt{rln}$-measure. For the fourth diagram we have $s_1 \xrightarrow{iR,letw0} s_1' \xrightarrow{iR,letw0} t_1 \xleftarrow{\text{LRPw},cp} R[t]$. We apply the induction hypothesis twice: For s_1 we get $\mathtt{rln}(s_1) = \mathtt{rln}(s_1')$ and for s_1' we get $\mathtt{rln}(s_1') = \mathtt{rln}(t_1)$ which finally shows $\mathtt{rln}(R[t]) = \mathtt{rln}(t_1) = \mathtt{rln}(s_1) = \mathtt{rln}(R[s])$.

The context lemma for improvement and the previous lemma imply:

Corollary 2.15. *(letw0) $\subseteq \approx$.*

Lemma 2.16. *If $s \xrightarrow{letwn} t$, then for all reduction contexts R s.t. $R[s]$ and $R[t]$ are closed:* $\mathtt{rln}(R[s]) = \mathtt{rln}(R[t])$ *or* $\mathtt{rln}(R[s]) = 1 + \mathtt{rln}(R[t])$.

Proof. Since $(letwn)$ is correct we know that $\mathtt{rln}(R[s]) = \infty \iff \mathtt{rln}(R[t]) = \infty$. So suppose that $\mathtt{rln}(R[s]) = n$. We show $\mathtt{rln}(R[t]) = n$ or $\mathtt{rln}(R[t]) = n+1$ by induction on a normal order reduction $R[s] \xrightarrow{\text{LRPw},k} s'$ where s' is a WHNF. The base case is covered by Lemma 2.11. For the induction step, let $R[s] \xrightarrow{no} s_1 \xrightarrow{\text{LRPw},k-1} s'$. If $R[s] \xrightarrow{\text{LRPw},letwn} R[t]$, then $\mathtt{rln}(R[s]) = 1 + \mathtt{rln}(R[t])$ and the claim holds. If the transformation is internal, then we apply a forking diagram to s_1. For the first diagram we have $s_1 \xrightarrow{iR,letwn} t_1$ and we apply the induction hypothesis to s_1 and thus have $\mathtt{rln}(s_1) = 1 + \mathtt{rln}(t_1)$ or $\mathtt{rln}(s_1) = \mathtt{rln}(t_1)$. This also shows $\mathtt{rln}(R[s]) = 1 + \mathtt{rln}(R[t])$ or $\mathtt{rln}(R[s]) = \mathtt{rln}(R[t])$. For the second diagram we have $\mathtt{rln}(R[s]) = \mathtt{rln}(R[t])$. For the third diagram we have $\mathtt{rln}(R[s]) = 1 + \mathtt{rln}(R[t])$. The fourth diagram is not applicable, since the given transformation is (letwn).

Corollary 2.17. *(letwn) $\subseteq \succeq$.*

Proposition 2.18. *All reduction rules are correct.*

Proof. For the (letwn)-rules this is already proved. For the other rules, correctness was shown in the untyped calculus LR in [9], which can be directly transfered to LRP. However, LRPw has shared-work decorations and the (letwn)-rules as normal order reduction. To keep the proof compact, we only consider these new cases. The reasoning to show correctness of the reduction rules in LRPw is the same as for LR, since all additional diagrams between an internal transformation step (i,b) and a $(\mathrm{LRPw}, letw)$-reduction are:

$$\begin{tikzcd} \cdot \arrow[r, "i{,}b"] \arrow[d, "\mathrm{LRPw}{,}a"'] & \cdot \arrow[d, "\mathrm{LRPw}{,}a"] \\ \cdot \arrow[r, "i{,}b"'] & \cdot \end{tikzcd}$$

$a \in \{letwn, letw0\}, b \in \{lbeta, cp, case, seq, lll\}$

$$\begin{tikzcd} \cdot \arrow[r, "i{,}b"] \arrow[d, "\mathrm{LRPw}{,}letw0"'] & \cdot \arrow[ddl, "\mathrm{LRPw}{,}letw0"] \\ \cdot \arrow[d, "\mathrm{LRPw}{,}T"'] & \\ \cdot & \end{tikzcd}$$

$b \in \{lbeta, cp, case, seq, lll\}$

The first case is the case where the (LRPw,letw) and the transformation commute, the second case is that the internal transformation becomes a normal order reduction after removing the a label. However, these cases are already covered by the diagram proofs in LR (see [9]) and thus can easily added.

We define a translation from expressions with work-decorations into decoration-free expressions, by removing the work-decorations and the corresponding bindings:

Definition 2.19. *Let t be an expression in* LRPw*, and $\mathtt{rmw}(t)$ be derived from t by removing the work-syntax, i.e.*

$$\begin{array}{ll}
\mathtt{rmw}(\mathtt{letrec}\ x_1 = s_1, \ldots, x_n = s_n, a_1 := n_1, \ldots, a_m := n_m\ \mathtt{in}\ s) = & \\
\qquad \mathtt{letrec}\ x_1 = \mathtt{rmw}(s_1), \ldots, x_n = \mathtt{rmw}(s_n)\ \mathtt{in}\ \mathtt{rmw}(s) & \textit{for } m \geq 0, n \geq 1 \\
\mathtt{rmw}(\mathtt{letrec}\ a_1 := n_1, \ldots, a_m := n_m\ \mathtt{in}\ s) & = \mathtt{rmw}(s), \\
\mathtt{rmw}(s^{[a]}) & = \mathtt{rmw}(s) \\
\mathtt{rmw}(f[s_1, \ldots, s_n]) & = f[\mathtt{rmw}(s_1), \ldots, \mathtt{rmw}(s_n)] \\
 & \quad \textit{for all other language constructs } f.
\end{array}$$

Proposition 2.20. *Let t be an expression in* LRPw*, then $t\downarrow_{\mathrm{LRPw}} \iff \mathtt{rmw}(t)\downarrow_{\mathrm{LRPw}}$.*

Proof. Observing that $t \xrightarrow{\mathrm{LRPw}} t'$ implies $\mathtt{rmw}(t) = \mathtt{rmw}(t')$ or $\mathtt{rmw}(t) \xrightarrow{\mathrm{LRPw}} \mathtt{rmw}(t')$, the proof is obvious.

An immediate consequence is the following theorem:

Theorem 2.21. *The embedding of* LRP *into* LRPw *w.r.t. $\sim_c$ is conservative.*

We do not know whether the embedding of LRP into LRPw is conservative w.r.t. the improvement relation $\preceq$. We conjecture that the embedding of LRP into LRPw is conservative w.r.t. the improvement relation $\preceq$. However, we did not find a proof. A naive proof which tries to encode the work decorations by usual expressions fails, since there are work decorations which cannot be encoded (see Proposition 3.8). However, conservativity is not really necessary. It would allow to lift results on improvements from LRP to LRPw more easily. Our goal to use the calculus LRPw as a proof technique to show results on improvements for LRP is possible:

Lemma 2.22. *Let s, t be* LRP*-expressions s.t. $s \preceq_{\mathrm{LRPw}} t$. Then also $s \preceq_{\mathrm{LRP}} t$ holds.*

Proof. This holds, since every LRP-context is also an LRPw-context and on decoration-free expressions the `rln`-length is the same in both calculi.

We prove correctness and (invariance w.r.t. $\approx$) for (gcW), the transformation which performs garbage collection of $a := n$-bindings which have no corresponding ${}^{[a]}$-label.

Lemma 2.23. *A complete set of forking and commuting diagrams for (S,gcW) can be read off the following diagrams:*

$$\begin{tikzcd} \cdot \arrow[r, "S{,}gcW"] \arrow[d, "\mathrm{LRPw}{,}a"'] & \cdot \arrow[d, "\mathrm{LRPw}{,}a"] \\ \cdot \arrow[r, "S{,}gcW"'] & \cdot \end{tikzcd} \qquad \begin{tikzcd} \cdot \arrow[r, "S{,}gcW"] \arrow[d, "\mathrm{LRPw}{,}a"'] & \cdot \arrow[dl, "\mathrm{LRPw}{,}a"] \\ \cdot & \end{tikzcd} \qquad \begin{tikzcd} \cdot \arrow[r, "S{,}gcW2"] \arrow[d, "\mathrm{LRPw}{,}lll"'] & \cdot \\ \cdot \arrow[ur, "S{,}gcW2"'] & \end{tikzcd}$$

a arbitrary *a arbitrary*

Proof. The first diagram covers the case where the transformation and the reduction commute. There are also cases where a (gcW2) becomes a (gcW1)-transformation, e.g. in $\texttt{letrec}\ x = (\texttt{letrec}\ a := n\ \texttt{in}\ s)\ \texttt{in}\ r \xrightarrow{S,gcW2} \texttt{letrec}\ x = s\ \texttt{in}\ r$ where $\texttt{letrec}\ x = (\texttt{letrec}\ a := n\ \texttt{in}\ s)\ \texttt{in}\ r \xrightarrow{\mathrm{LRPw},llet} \texttt{letrec}\ x = s, a := n\ \texttt{in}\ r$. The second diagram covers the case where the (gcW)-redex is removed by the normal order reduction, e.g. if it is in an unused alternative of `case` or inside the first argument of `seq`. The last diagram covers the case where the `letrec`-expression of the redex of (LRPw,lll) is removed by (gcW2).

Lemma 2.24. *If* $s \xrightarrow{S,gcW} t$ *then*

– *If s is a WHNF, then t is a WHNF.*
– *If t is a WHNF, then* $s \xrightarrow{\mathrm{LRPw},llet,0\vee 1} s'$ *where s' is a WHNF*

Proof. The first item can be easily verified. For the second item it may be the case that s is not a WHNF, but t is a WHNF, e.g. $\texttt{letrec}\ a := n\ \texttt{in}\ r \xrightarrow{gcW2} r$ where r is a WHNF.

Proposition 2.25. *The transformation (gcW) is correct and (gcW) $\subseteq \approx$.*

Proof. We first show correctness. Let $s \xrightarrow{S,gcW} t$

– $s\downarrow \implies t\downarrow$: This can be shown by induction on the length k in $s \xrightarrow{\mathrm{LRPw},k} s'$ where s' is a WHNF. For the base case Lemma 2.24 shows $t\downarrow$. For the induction step we apply a forking diagram. For the first diagram we have $s \xrightarrow{\mathrm{LRPw},a} s_1$, $s_1 \xrightarrow{S,gcW} t_1$, $t \xrightarrow{\mathrm{LRPw},a} t_1$. Applying the induction hypothesis to s_1 and t_1 shows $t_1\downarrow$ and thus $t\downarrow$. For the second diagram $t\downarrow$ obviously holds. For the third diagram we have $s \xrightarrow{\mathrm{LRPw},lll} s_1$, $s_1 \xrightarrow{gcW2} t$. We apply the induction hypothesis to s_1 and t which shows $t\downarrow$.
– $t\downarrow \implies s\downarrow$: We use an induction in the length k in $t \xrightarrow{\mathrm{LRPw},k} t'$ where t' is a WHNF. For the base case $k = 0$ Lemma 2.24 shows that $s\downarrow$. For the induction step we apply a forking diagram. For the first and the second diagram the cases are analogous to the previous part. For the third diagram we apply the diagram as long as possible which terminates, since there are no infinite sequences of $(\mathrm{LRPw},,lll)$-reductions. Then we get an expression s' with either $s \xrightarrow{\mathrm{LRPw},lll,+} s'$ where s' is a WHNF and thus $s\downarrow$, or we apply the first or second diagram to t and s', and then the induction hypothesis (in case of diagram 1). In any case we derive $s\downarrow$.

The two items and the context lemma for $\sim_c$ show that (gcW) is correct. Now we consider improvement. Let $s \xrightarrow{S,gcW} t$. We show $\texttt{rln}(s) = \texttt{rln}(t)$. The context lemma for improvement then implies $(gcW) \subseteq \approx$. Since (gcW) is correct we already have $\texttt{rln}(s) = \infty \iff \texttt{rln}(t) = \infty$. Now let $s \downarrow s'$ (where $s \xrightarrow{\mathrm{LRPw},k} s'$) and $\texttt{rln}(s) = n$. We show $\texttt{rln}(t) = n$ by induction on k. If $k = 0$ then Lemma 2.24 shows $\texttt{rln}(s) = 0 = \texttt{rln}(t)$. If $k > 0$ then we again apply the forking diagrams. The cases are completely analogous as for the correctness proof, where have to verify, that the first and the second diagram do either introduce nor remove normal order reductions, and the third diagram may only remove $(\mathrm{LRPw},,lll)$-reduction which are not counted by the `rln`-measure.

The following results from [9, 8] on the lengths of reductions also hold in the calculus LRPw, since the overlappings for (LRPw, *letw*) and the corresponding transformation are analogous to already covered cases.

Theorem 2.26. *Let t be a closed* LRP*-expression with $t \downarrow t_0$.*

1. *If $t \xrightarrow{C,\ a} t'$, and $a \in \{case, seq, lbeta\}$, then $\mathtt{rln}(t) \geq \mathtt{rln}(t')$.*
2. *Let t be a closed* LR*-expression with $t \downarrow t_0$ and $t \xrightarrow{C,cp} t'$, then $\mathtt{rln}(t) = \mathtt{rln}(t')$.*
3. *If $t \xrightarrow{S,a} t'$, and $a \in \{case, seq, lbeta\}$, then $\mathtt{rln}(t) \geq \mathtt{rln}(t') \geq \mathtt{rln}(t) - 1$.*
4. *If $t \xrightarrow{C,\ a} t'$, and $a \in \{lll, gc\}$, then $\mathtt{rln}(t) = \mathtt{rln}(t')$.*
5. *If $t \xrightarrow{C,a} t'$, and $a \in \{cpx, cpax, xch, cpcx, abs, lwas\}$, then $\mathtt{rln}(t) = \mathtt{rln}(t')$.*
6. *If $t \xrightarrow{C,\ ucp} t'$, then $\mathtt{rln}(t) = \mathtt{rln}(t')$.*

Corollary 2.27. *1. If $s \xrightarrow{S,a} s'$ where a is any rule from Figs. 2 and 3, then $s' \preceq s$.*
2. If $s \xrightarrow{C,a} s'$ where a is (lll), (cp), (letw0) or any rule of Fig. 3. Then $s' \approx s$.

Proof. The claims follow from Theorem 2.26 and the context lemma, and for the rule (letsh) the claim holds, since it is a composition of (*lwas*) and (*llet*) and their inverses. For (gcW) this follows from Proposition 2.25. For (letw0) it follows from Corollary 2.15, and for (letwn) it follows from Corollary 2.17.

3 Work Decorations

In this section we consider another notation for work decorations.

Definition 3.1. *For* LRPw *we use the following notation:*

rln-decoration *: If $n \in \mathbb{N}$, then $s^{[n]}$ is an expression, where $[n]$ is called a* rln-decoration*. The semantics of $s^{[n]}$ is $\mathtt{letrec}\ a := n\ \mathtt{in}\ s^{[a]}$ where a is a fresh label.*

sharing decoration *: If a is label and $n \in \mathbb{N}$, then $C[s_1^{[a \mapsto n]}, \ldots, s_m^{[a \mapsto n]}]$ is an expression. The semantics of $C[s_1^{[a \mapsto n]}, \ldots, s_m^{[a \mapsto n]}]$ is $\mathtt{letrec}\ a := n\ \mathtt{in}\ C[s_1^{[a]}, \ldots, s_m^{[a]}]$*

further notation: *For convenience, we also write several decorations in the form $[n, a_1 \mapsto m_1, \ldots, a_k \mapsto m_k]$ (where the a_i are distinct). The semantics of the expressions can be derived from the previous cases, where the nondeterminism in the translation is irrelevant, since (lll)-transformations allow to reorder and combine the corresponding environments without changing the $\mathtt{rln}$-measure. We may also use the abstract notation $[n, p]$ for a sharing decoration with constant n, and further sharing decorations p.*

Note that LRPw contains expressions, which cannot be expressed by this notation. E.g., the expression $\lambda x.\mathtt{letrec}\ a = n\ \mathtt{in}\ C[s^{[a]}, t^{[a]}]$, since the semantic translation of $\lambda x.C[s^{[a]}, t^{[a]}]$ is $\mathtt{letrec}\ a = n\ \mathtt{in}\ \lambda x.C[s^{[a]}, t^{[a]}]$ which is a different expression.

We show that the (non-shared) rln-decorations are redundant, and can be encoded by usual LRP-expressions.

Proposition 3.2. *The (sharing) rln-decorations $s^{[n]}$ can be encoded as $\mathtt{letrec}\ x = (id^n)\ \mathtt{in}\ (x\ s)$ and thus are redundant.*

Proof. The proof is in Appendix A.

3.1 Computation Rules for Decorations

In this section we develop the computation rules with decorations.

First we define a combination of labels, since addition has to be modified. Here we assume that labels are sets consisting of exactly one nonnegative integer (a rln-decoration) and several rln-sharing decorations.

Definition 3.3. *The combination $p_1 \oplus p_2$ of two decorations $p_1 = [n_1, p_1']$ and $p_2 = [n_2, p_2']$ is defined as $[n_1 + n_2, p_3]$, where $p_3 = p_1 \cup p_2$.* □

For example, $[1, a_1 \mapsto 3, a_2 \mapsto 5] \oplus [2, a_1 \mapsto 3, a_3 \mapsto 7] = [3, a_1 \mapsto 3, a_2 \mapsto 5, a_3 \mapsto 7]$.

A corollary from the theorem on reduction lengths (Theorem 2.26) is:

Corollary 3.4. *Let S be a surface context. If $s \xrightarrow{S} s'$ by any reduction or transformation rule from Figs. 2 and 3, then $s' \preceq s$ and $s \preceq s'^{[1]}$.*

In Appendix Bthe following computation rules are proved:

Theorem 3.5. *1. If $s \xrightarrow{\mathrm{LRPw},a} t$ with $a \in \{lbeta, case, seq, letwn\}$, then $s \approx t^{[1]}$.*

2. *$R[\mathtt{letrec}\ a := n\ \mathtt{in}\ s^{[a]}] \approx \mathtt{letrec}\ a := n\ \mathtt{in}\ R[s]^{[a]}$ and thus in particular $R[s^{[n]}] \approx R[s]^{[n]}$.*
3. *$\mathtt{rln}(\mathtt{letrec}\ a := n\ \mathtt{in}\ s^{[a]}) = n + \mathtt{rln}(s')$ where s' is s where all $^{[a]}$-labels are removed. In particular this also shows $\mathtt{rln}(s^{[n]}) = n + \mathtt{rln}(s)$*
4. *For every reduction context R: $\mathtt{rln}(R[\mathtt{letrec}\ a := n\ \mathtt{in}\ s^{[a]}]) = n + \mathtt{rln}(R[s'])$ where s' is s where all $^{[a]}$-labels are removed. In particular, this shows $\mathtt{rln}(R[s^{[n]}]) = n + \mathtt{rln}(R[s])$.*
5. *$(s^{[n]})^{[m]} \approx s^{[n+m]}$*
6. *For all surface contexts S_1, S_2: $S_1[\mathtt{letrec}\ a := n\ \mathtt{in}\ S_2[s^{[a]}]] \preceq \mathtt{letrec}\ a := n\ \mathtt{in}\ S_1[S_2[s]]^{[a]}$ and if $S_1[S_2]$ is strict, also $S_1[\mathtt{letrec}\ a := n\ \mathtt{in}\ S_2[s^{[a]}]] \approx \mathtt{letrec}\ a := n\ \mathtt{in}\ S_1[S_2[s]]^{[a]}$.*
   In particular, this shows for all surface contexts S and expressions s: $S[s^{[k]}] \preceq S[s]^{[k]}$, and if S is strict, then $S[s^{[k]}] \approx S[s]^{[k]}$.
7. *$\mathtt{letrec}\ a := n, b := m\ \mathtt{in}\ (s^{[a]})^{[b]} \approx \mathtt{letrec}\ a := n, b := m\ \mathtt{in}\ (s^{[b]})^{[a]}$*
8. *$\mathtt{letrec}\ a := n\ \mathtt{in}\ (s^{[a]})^{[a]} \approx \mathtt{letrec}\ a := n\ \mathtt{in}\ (s^{[a]})$*
9. *$(t^{p_1})^{p_2} \approx t^{p_1 \oplus p_2}$.*
10. *Let $S[\cdot, \ldots, \cdot]$ be a multi-context where all holes are in surface position. Then $\mathtt{letrec}\ a := n\ \mathtt{in}\ S[s_1^{[a]}, \ldots, s_n^{[a]}] \preceq \mathtt{letrec}\ a := n\ \mathtt{in}\ S[s_1, \ldots, s_n]^{[a]}$. If some hole $\cdot_i$ with $i \in \{1, \ldots, n\}$ is in strict position in $S[\ldots, .]$, then $\mathtt{letrec}\ a := n\ \mathtt{in}\ S[s_1^{[a]}, \ldots, s_n^{[a]}] \approx \mathtt{letrec}\ a := n\ \mathtt{in}\ S[s_1, \ldots, s_n]^{[a]}$.*
11. *Let $S[\cdot, \ldots, \cdot]$ be a multi-context where all holes are in surface position. Let $S[s_1, \ldots, s_n]$ be closed. Then $S[s_1^{[p_1, a \mapsto m]}, \ldots, s_n^{[p_n, a \mapsto m]}] \preceq S[s_1^{p_1}, \ldots, s_n^{p_n}]^{[m]}$.*
    If some hole $\cdot_i$ with $i \in \{1, \ldots, n\}$ is in strict position in $S[\ldots, .]$, then $S[s_1^{[p_1, a \mapsto m]}, \ldots, s_n^{[p_n, a \mapsto m]}] \approx S[s_1^{p_1}, \ldots, s_n^{p_n}]^{[m]}$.
    By iteratively applying the claim this shows for all surface contexts S and expressions s: $S[s^p] \preceq S[s]^p$, and if S is strict, then $S[s^p] \approx S[s]^p$.
12. *The following transformation is correct w.r.t. $\approx$: Replace $(\mathtt{letrec}\ x = s^{[n,p]}, Env\ \mathtt{in}\ t)$ by $\mathtt{letrec}\ x = s[x^{[a \mapsto n,p]}/x], Env[x^{[a \mapsto n,p]}/x]\ \mathtt{in}\ t[x^{[a \mapsto n,p]}/x]$, where a is a fresh label and all occurrences of x are in surface position.*
13. *If a label-name a occurs exactly once in a surface context, then it can be changed into an unshared decoration.*

An immediate consequence is:

Proposition 3.6. *The following variant of reduction is correct w.r.t. the* LRPw*-semantics:*

> *The reduction on* LRP *expressions with work decorations is as follows: If $n > 0$ and $t = R[t_1^{[n]}]$, then $R[t_1^{[n]}] \xrightarrow{\mathrm{LRPw}} R[t_1^{[n-1]}]$ where this reduction counts to the $\mathtt{rln}$-measure. If $n = 0$ then $R[t_1^{[0]}] \xrightarrow{\mathrm{LRPw}} R[t_1]$ where this reduction is not counted.*

(cp-in) $\texttt{letrec}\ x_1 = (\lambda y.t)^{[n,p_1]}, \{x_i = x_{i-1}\}_{i=2}^m, Env\ \texttt{in}\ C[x_m^{p_2}]$
$\rightarrow \texttt{letrec}\ x_1 = \lambda y.t^{[a\mapsto n,p_1]}, \{x_i = x_{i-1}\}_{i=2}^m, Env\ \texttt{in}\ C[(\lambda y.t)^{([a\mapsto n,p_1]\oplus p_2]}]$
where p_2 is nontrivial only if $x_m^{p_2}$ is not the right hand side of a binding
(llet-e) $(\texttt{letrec}\ Env_1, x = (\texttt{letrec}\ Env_2\ \texttt{in}\ t)^p\ \texttt{in}\ r) \rightarrow (\texttt{letrec}\ Env_1, Env_2, x = t^p\ \texttt{in}\ r)$

The standard cases are usually dealt with shifting the decoration up, since the decoration is in a strict position, and/or using further rules (locally) from Theorem 3.5.

Fig. 4. The non-standard cases of decoration modification of reduction rules of LRPw (variants omitted)

If $n > 0$ and $t = R[t_1^{[a\mapsto n]}]$ where R is a reduction context, where no decorations are on the path to the hole, then $R[t^{[a\mapsto n]}] \xrightarrow{\text{LRPw}} R'[t^{[a\mapsto n-1]}]$, where all $[a \mapsto n]$-decorations in R and t are changed into $[a \mapsto n-1]$. The reduction step also counts as one $\texttt{rln}$*-reduction step, i.e.* $\texttt{rln}(R[t^{[a\mapsto n]}]) = 1 + \texttt{rln}(R[t^{[a\mapsto n-1]}])$. *If $t = R[t_1^{[a\mapsto 0]}]$ where R is a reduction context, where no decorations are on the path to the hole, then $R[t^{[a\mapsto 0]}] \xrightarrow{\text{LRPw}} R'[t]$, this reduction is not counted by the* $\texttt{rln}$*-measure.*

Remark 3.7. For a surface context C, the rln-sharing decorated expression $s := C[s_1^{[n_1,a\mapsto h]}, \ldots, s_m^{[n_m,a\mapsto h]}]$ can be given a semantics in the case $n_i > 0$ for all i.: Define

$$sem(s) := \texttt{letrec}\ x_0 = \texttt{id}^{[h]}\ \texttt{in}\ C[(x_0\ s_1^{[n_1-1])}, \ldots, (x_0\ s_m^{[n_m-1]})]).$$

In the case $n_i = 0$ for some i, the equivalence classes of expressions w.r.t. $\approx$ are properly extended (see Proposition 3.8)

It can easily be verified, that $sem(s) \xrightarrow{T,*} s'$ with $s' \approx C[s_1, \ldots, s_m]$, where the reduction requires $h + \sum_{i=1}^m n_i$ rln-reduction steps: $h + n_i$ steps for each $(x_0\ s_i^{[n_i-1]})$ where h is the number of shared rln-reduction steps.

In the exceptional case $n_1 = 0$ there are some cases, which can be given a semantics: for example $(Z^{[a\mapsto 1]}, Z^{[a\mapsto 1]}) \approx \texttt{letrec}\ x = \texttt{id}\ Z\ \texttt{in}\ (x, x)$. Generalizing, if the sharing decorated subexpressions are syntactically equal, then the construction may be applied in certain cases.

Proposition 3.8. *The decorated expression $(Z^{[a\mapsto 1]}, \texttt{Nil}^{[a\mapsto 1]})$ is not equivalent w.r.t. $\approx$ to any* LRP-*expression.*

Proof. Assume there is such an expression s. Then $s \sim_c (Z, \texttt{Nil})$ and $\texttt{rln}(s) = 0$, so we can assume that s is a WHNF. Using the correctness w.r.t. $\approx$ of program transformations and that $Z \not\sim_c \texttt{Nil}$, we can assume that s is of the form $\texttt{let}\ x = s_1, y = s_2, Env\ \texttt{in}\ (x, y)$. We see that s_1 as well as s_2 alone have $\texttt{rln}$-count 1. Using that (lll) is a correct program transformation, we obtain that s_1, s_2 can only be applications, a $\texttt{seq}$- or a case-expressions. But then every of them requires at least one $\texttt{rln}$-reduction that is independent of the other to become a WHNF. Hence the context $C := \texttt{let}\ z = [\cdot]\ \texttt{in seq}\ (\texttt{fst}\ z)\ (\texttt{snd}\ z)$ applied to $(Z^{[a\mapsto 1]}, \texttt{Nil}^{[a\mapsto 1]})$ requires $6 = 5 + 1$ steps: 2 for $\texttt{fst}$, 2 for $\texttt{snd}$, 1 for $\texttt{seq}$, and 1 for the shared evaluation of $Z^{[a\mapsto 1]}$, whereas s requires at least $7 = 5 + 2$: the 2 reductions are the minimum to reach a WHNF for the first as well for the second component.

We show how the decorations are implicitly modified under reductions and transformations, where the reduction are invariant under $\approx$. See figure 4 for the reduction rules of LRP w.r.t. decorations.

3.2 More Transformations and Improvements

Let (caseId) be defined as:

$$(\texttt{case}_K\ s\ \texttt{of}\ (pat_1 \rightarrow pat_1) \ldots (pat_{|D_K|} \rightarrow pat_{|D_K|})) \rightarrow s$$

(cpcx-in)	$\texttt{letrec}\ x = c\,\overrightarrow{t}^{[n,p_1]}, Env\ \texttt{in}\ C[x^{p_2}] \to \texttt{letrec}\ x = c\,\overrightarrow{y}^{[a\mapsto n,p_1]}, \{y_i = t_i\}_{i=1}^{ar(c)}, Env\ \texttt{in}\ C[c\,\overrightarrow{y}^{[a\mapsto n,p_1]\oplus p_2}]$
(xch)	$\texttt{letrec}\ x = t^p, y = x, Env\ \texttt{in}\ r\ \to\ \texttt{letrec}\ y = t^p, x = y, Env\ \texttt{in}\ r$ where $y = x^q$ is not permitted.
(lwas)	$T[\texttt{letrec}\ Env\ \texttt{in}\ t^p] \to \texttt{letrec}\ Env\ \texttt{in}\ T[t^p]$ if T is a weak top context with hole depth 1
(ucp1)	$\texttt{letrec}\ Env, x = t^p\ \texttt{in}\ S[x] \to \texttt{letrec}\ Env\ \texttt{in}\ S[t^p]$

The standard cases are usually dealt with shifting the decoration up, since the decoration is in a strict position, and/or using further rules (locally) from Theorem 3.5.

Fig. 5. Non-standard cases of decoration modification in the Extra Transformation Rules (variants omitted)

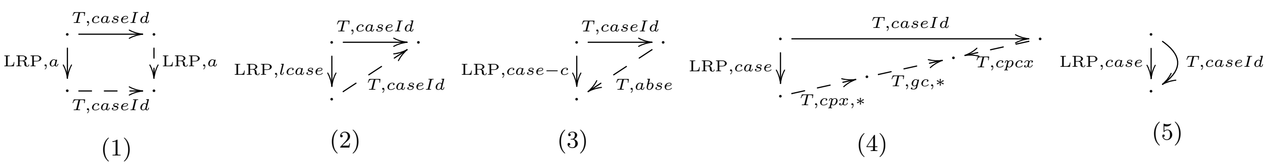

Fig. 6. Diagrams for (caseId)

The rule (caseId) is the heart (of the correctness proof) of other type-dependent transformations, like rules involving map, filter, fold, asf., and it is only correct under typing, i.e. in LRP and LRPw, but not in LR, which can be seen by trying the case $s = \lambda x.t$.

We show that (caseId) is an improvement in LRPw.

Lemma 3.9. *Let* $s \xrightarrow{T,caseId} t$. *If* s *is a WHNF, then* t *is a WHNF. If* t *is a WHNF, then* $s \xrightarrow{\mathrm{LRPw},lll,*} \xrightarrow{\mathrm{LRPw},case,0\vee 1} \xrightarrow{\mathrm{LRPw},lll,*} s'$ *where* s' *is a WHNF.*

Lemma 3.10. *If* $s{\downarrow} \wedge s \xrightarrow{T,caseId} t$, *then* $t{\downarrow}$ *and* $\mathtt{rln}(s) \geq \mathtt{rln}(t)$.

Proof. Let $s \xrightarrow{T,caseId} t$ and $s \xrightarrow{\mathrm{LRPw},k} s'$ where s' is a WHNF. We use induction on k. For $k = 0$ Lemma 3.9 shows the claim. For the induction step, let $s \xrightarrow{\mathrm{LRPw}} s_1$. The diagrams in Fig. 6 describe all cases how the fork $s_1 \xleftarrow{\mathrm{LRPw}} s \xrightarrow{T,caseId}$ can be closed. For diagram (1) we apply the induction hypothesis to $s_1 \xrightarrow{T,caseId} t_1$ which shows $t_1{\downarrow}$, $\mathtt{rln}(s_1) \geq \mathtt{rln}(t_1)$ and thus also $t{\downarrow}$ and $\mathtt{rln}(s) \geq \mathtt{rln}(t)$. For diagram (2) the induction hypothesis shows the claim. For diagram (3) we have $t{\downarrow}$, since (abse) is correct. Moreover, $t \xrightarrow{T,abse} s'$ is equivalent to $s' \xrightarrow{T,ucp\vee gc,*} t$ and Theorem 2.26 shows $\mathtt{rln}(s') = \mathtt{rln}(t)$. Thus also $\mathtt{rln}(s) \geq \mathtt{rln}(t)$. For diagram (4) we have $t{\downarrow}$, since (cpcx), (gc), and (cpx) are correct. Theorem 2.26 shows that $\mathtt{rln}(s) > \mathtt{rln}(s') = \mathtt{rln}(t)$, since (cpcx), (cpx) and (gc) do not change the measure $\mathtt{rln}(\cdot)$. For diagram(5) the claim obviously holds.

Theorem 3.11. *(caseId) is an improvement.*

Proof. Lemma 3.9 and the diagrams in Fig. 6 can be used to show (by induction on the sequence for t) that if $s \xrightarrow{T,caseId} t$ and $t{\downarrow}$, then $s{\downarrow}$, since the used existentially quantified transformations are correct and diagram 2 can only be applied finitely often. Then the context lemma for $\sim_c$ (which states that convergence preservation and reflection in reduction contexts suffices to $\sim_c$, see e.g. [5]) and Lemma 3.10 show that (caseId) is correct. Finally, the context lemma for improvement (Lemma 2.9) and Lemma 3.10 show that (caseId) is an improvement.

4 A Head-Centered Improvement Simulation for Lists

We define an improvement simulation $\sqsubseteq_{h,\tau}$ on lists of the same type, $\texttt{List}\ \tau$, for proving $\preceq$-relations between functions on lists.

Definition 4.1. *Let τ be a type, and $\mathfrak{L}_\tau := \{(s,t) \mid s,t$ are closed, $s,t :: \texttt{List}(\tau), s \sim_c t\}$. We define the following operator $F_h :: \mathfrak{L}_\tau \to \mathfrak{L}_\tau$: Let $\eta \subseteq \mathfrak{L}_\tau$, and $s\ \eta\ t$.*

1. *If $s \sim_c \bot \sim_c t$, then $s\ F_h(\eta)\ t$.*
2. *If $s \approx \texttt{Nil}^{[k]}$, $t \approx \texttt{Nil}^{[k']}$ and $k \leq k'$, then $s\ F_h(\eta)\ t$.*
3. *If $s \preceq (s_1^{[k_1, a_j \mapsto m_j \text{ for } j=1,\ldots]} : s_2^{[k_2]})^{[k_3]}$, and $(t_1^{[k_1', a_j \mapsto m_j' \text{ for } j=1,\ldots]} : t_2^{[k_2']})^{[k_3']} \preceq t$, for some expressions s_1, s_2, t_1, t_2, and numbers where s_2, t_2 may contain further sharing decorations, but only in surface context positions; and the following conditions hold:*
   - *$k_i \leq k_i'$ for $i = 1,2,3$,*
   - *$m_j \leq m_j'$ for all j.*
   - *$s_1 \preceq t_1$ and s_1, t_1 are decoration-free, and*
   - *$s_2\ \eta\ t_2$.*

   Then $s\ F_h(\eta)\ t$.

Let $\sqsubseteq_{h,\tau}$ be the greatest fixpoint of F_h. □

To ease reading we leave out the index τ in the following and simply write $\sqsubseteq_h$ instead of $\sqsubseteq_{h,\tau}$ unless the type τ becomes relevant.

Clearly, the operator F_h is monotone, and thus $\sqsubseteq_h$ is well-defined, i.e. the fixpoint exists.

Moreover, due to determinism of normal-order reduction, F_h is lower-continuous, and thus Kleene's fixpoint theorem can be applied, which implies the following inductive characterization of $\sqsubseteq_h$: Let $\sqsubseteq_{h,0} = \mathfrak{L}_\tau$, and $\sqsubseteq_{h,i} = F(\sqsubseteq_{h,i-1})$ for $i > 0$. Then $\sqsubseteq_h = \bigcap_{i=0}^{\infty} \sqsubseteq_{h,i}$. Thus for $(s,t) \in \mathfrak{L}_\tau$ we can show $s \sqsubseteq_h t$ by proving $s \sqsubseteq_{h,i} t$ for all i.

Note that the same sharing label may occur in several elements of the list.

The following is required in the proof below, where the formulation of the second claim is w.r.t. the inductively generated relations.

Theorem 4.2. *If $s \sqsubseteq_h t$, then also $s \preceq t$.*

Proof. We show a generalized claim. Using this claim with a single-hole surface-context T and $n = 1$ shows that $s \preceq_T t$, and thus using the context lemma for improvement, also the claim of the theorem follows. The claim is:

> Let $C[\cdot,\ldots,\cdot]$ be a multicontext, where the holes are in surface-contexts, for $i = 1,\ldots,n$ let s_i, t_i be closed and of the same type such that for each pair s_i, t_i either $s_i \sqsubseteq_h t_i$ (and thus $s_i, t_i :: \texttt{List}(\tau)$), or $s_i \preceq t_i$ (and thus $s_i, t_i :: \tau$) holds. Let the expressions s_i be decorated with d_i and the expressions t_i with e_i, where d_i is $[n_{s,i}, a_{i,j} \mapsto m_{s,i,j}]$ or no label, and e_i is $[n_{t,i}, a_{i,j} \mapsto m_{t,i,j}]$ or no label according to the rules above: i.e., $n_{s,i} \leq n_{t,i}$ holds and $m_{s,i,j} \leq m_{t,i,j}$ in all cases.
> Then $\texttt{rln}(C[s_1^{d_1},\ldots,s_n^{d_n}]) \leq \texttt{rln}(C[t_1^{e_1},\ldots,t_n^{e_n}])$ holds.

For the proof we assume that for the first input pair (s,t), the infinite sequence of the expansion (including the decorations asf.) according to Definition 4.1 is fixed, and so we make the same choices even if copies of s,t appear in the expressions. So we can use the Kleene-criterion for computing the fixed point.
First observe that $\texttt{rln}(C[t_1^{e_1},\ldots,t_n^{e_n}]) = \infty$ if, and only if $\texttt{rln}(C[s_1^{d_1},\ldots,s_n^{d_n}]) = \infty$, which follows from finiteness of decorations and from $s_i \sim_c t_i$.

In other cases we show the claim by induction on the lexicographically ordered measure $(\mu_1,\mu_2,\mu_3,\mu_4)$ where $\mu_1 = \texttt{rln}(C[t_1^{e_1},\ldots,t_n^{e_n}])$, $\mu_2 = \texttt{rln}(C[s_1^{d_1},\ldots,s_n^{d_n}])$, μ_3 is the number of holes in C and $\mu_4 = \texttt{rlnall}(C[s_1^{d_1},\ldots,s_n^{d_n}])$.

For the **base case**, let $\mu_1 = \mu_2 = 0$, i.e. $\texttt{rln}(C[s_1^{d_1},\ldots,s_n^{d_n}]) = 0$. Then the claim $0 = \texttt{rln}(C[s_1^{d_1},\ldots,s_n^{d_n}]) \leq \texttt{rln}(C[t_1^{e_1},\ldots,t_n^{e_n}])$ is obvious.

For the **induction step**, assume that $(\mu_1, \mu_2) \neq (0,0)$.

If no hole of C is in a reduction context, then $C[t_1^{e_1}, \ldots, t_n^{e_n}] \xrightarrow{no} C'[t_1^{e_1}, \ldots, t_n^{e_n}]$ as well as $C[s_1^{d_1}, \ldots, s_n^{d_n}] \xrightarrow{no} C'[s_1^{d_1}, \ldots, s_n^{d_n}]$, where C' has n or less than n holes, since all holes are in surface contexts. We can apply the induction hypothesis after the reduction, since μ_1,μ_2 remain equal or both are decreased by 1, μ_3 remains equal or is decreased, and μ_4 is strictly decreased. Note that the case $\mu_1 = 0, \mu_2 > 0$ is not possible in this case.

Now we consider the case that some $t_j^{e_j}$ is in a reduction context in $C[t_1^{e_1}, \ldots, t_n^{e_n}]$ or $s_j^{d_j}$ is in a reduction context in $C[s_1^{d_1}, \ldots, s_n^{d_n}]$. Then we can assume w.l.o.g. that the hole j is in a reduction context in C, independent of the expressions in the holes. Hence $s_j^{d_j}$ as well as $t_j^{e_j}$ are in a reduction context in $C[s_1^{d_1}, \ldots, s_n^{d_n}]$ and $C[t_1^{d_1}, \ldots, t_n^{d_n}]$, respectively.

1. If t_j (and / or) s_j are nontrivially decorated, then there are two cases:
   (a) The decoration may be constants m_d, m_e with $m_e > 0$. Then $m_d < m_e$, and we can use the induction hypothesis, since the measure μ_1 is strictly smaller.
   (b) The decoration is a sharing one: $a \mapsto m_d$ in s and $a \mapsto m_e$ in t. Then we have $m_d \leq m_e$, and $m_e > 0$. Then remove the sharing for a in $C[s_1^{d_1}, \ldots, s_n^{d_n}]$ and also in $C[t_1^{e_1}, \ldots, t_n^{e_n}]$.
   Let d'_i, e'_i be the accordingly modified decorations at the other expressions s_i and t_i, and s'_i, t'_i be the expressions after the removal. The expressions are $C[s_1'^{d'_1}, \ldots, s'_j, \ldots, s_n'^{d'_n}]$ and $C[t_1'^{e'_1}, \ldots, t'_j, \ldots, t_n'^{e'_n}]$.
   Since $n_{t,j} > 0$ and / or $n_{s,j} > 0$ the measure $(\mu_1, \mu_2, \mu_3, \mu_4)$ is strictly decreased. The induction hypothesis applies, since μ_1 is strictly reduced. We also have to verify that the modified sequence of the derived sequences after setting a label value to zero, still satisfies the conditions of $\sqsubseteq_h$. But this holds, since the inequations are still valid, and the labels are only in surface contexts, so the computation rules say that $\preceq$ of the components is invariant (see Theorem 3.5.)
   The induction hypothesis shows $\mathtt{rln}(C[s_1'^{d'_1}, \ldots, s'_j, \ldots, s_n'^{d'_n}]) \leq \mathtt{rln}(C[t_1'^{e'_1}, \ldots, t'_j, \ldots, t_n'^{e'_n}]$ which also implies $\mathtt{rln}(C[s_1^{d_1}, \ldots, s_n^{d_n}]) \leq \mathtt{rln}(C[t_1^{e_1}, \ldots, t_n^{e_n}])$.
2. Now assume that s_j and t_j are not decorated, and $s_j \preceq t_j$, and there are no further decorations in s_j, t_j. Then the context $C' = C[\cdot_1, \ldots, \cdot_{j-1}, s_j, \cdot_{j+1}, \ldots, \cdot_n]$ has $(n-1)$ holes and the induction hypothesis shows $\mathtt{rln}(C[s_1^{d_1}, \ldots, s_j, \ldots, s_n^{d_n}] \leq \mathtt{rln}(C[t_1^{e_1}, \ldots, s_j, \ldots, t_n^{e_n}])$. Note that the induction hypothesis is applicable, since $\mathtt{rln}(C[t_1^{e_1}, \ldots, s_j, \ldots, t_n^{e_n}]) \leq \mu_1$, μ_2 is unchanged, but μ_3 is strictly decreased). Since $s_j \preceq t_j$, we also have $\mathtt{rln}(C''[s_j]) \leq \mathtt{rln}(C''[t_j])$ for $C'' = C'[t_1^{e_1}, \ldots, t_{j-1}^{e_{j-1}}, \cdot_j, t_{j+1}^{e_{j+1}}, \ldots, t_n^{e_j}]$ and thus $\mathtt{rln}(C[s_1^{d_1}, \ldots, s_j, \ldots, s_n^{d_n}]) \leq m + \mathtt{rln}(C[t_1^{e_1}, \ldots, t_j, \ldots, t_n^{e_n}])$.
3. Now assume that s_j and t_j are not decorated and $s_j \sqsubseteq_h t_j$.
   By our assumptions, $s_j \sqsubseteq_h t_j$ holds. We check the cases from the definition of $\sqsubseteq_h$.
   (a) If $s_j \sim_c \bot$, then $t_j \sim_c \bot$, and $C[s_1^{d_1}, \ldots, s_j, \ldots s_n^{d_n}] \sim_c \bot \sim_c C[t_1^{e_1}, \ldots t_j, \ldots, t_n^{e_n}]$.
   (b) If $s_j \approx \mathtt{Nil}$, then $t_j \approx \mathtt{Nil}$. (Due to our assumptions, the decorations are already dealt with) and we can integrate s_j, t_j into the context, which makes μ_3 strictly smaller.
   (c) If $s_j \preceq (s_{j,1}^{d_{j,1}} : s_{j,2}^{d_{j,2}})$ with $d_{j,1} = [n_{s,j,1}, p_{s,j,1}]$ and $d_{j,2} = [n_{s,j,2}, p_{s,j,2}]$, then due to the preconditions there is a representation $(t_{j,1}^{e_{j,1}} : t_{j,2}^{e_{j,2}})^{[k']} \preceq t_j$ with $e_{j,1} = [n_{t,j,1}, p_{t,j,1}]$ and $e_{j,2} = [n_{t,j,2}, p_{t,j,2}]$) such that $n_{s,j,1} \leq n_{t,j,1}, n_{s,j,2} \leq n_{t,j,2}$, and $p_{s,j,1} \leq p_{t,j,1}$, $p_{s,j,2} \leq p_{t,j,2}$, and $s_{j,1} \preceq t_{j,1}$ and $s_{j,2} \sqsubseteq_h t_{j,2}$.
   It suffices to show that $\mathtt{rln}(C[s_1^{d_1}, \ldots, (s_{j,1}^{d_{j,1}} : s_{j,2}^{d_{j,2}}), \ldots, s_n^{d_n}]) \leq \mathtt{rln}(C[t_1^{e_1}, \ldots, (t_{j,1}^{e_{j,1}} : t_{j,2}^{e_{j,2}}), \ldots, t_n^{e_n}])$ to prove the claim.
   Now consider the next normal order reduction for $C[t_1^{e_1}, \ldots, (t_{j,1}^{e_{j,1}} : t_{j,2}^{e_{j,2}}), \ldots, t_n^{e_n}]$. If there is no such reduction, then $C[t_1^{e_1}, \ldots, (t_{j,1}^{e_{j,1}} : t_{j,2}^{e_{j,2}}), \ldots, t_n^{e_n}]$ is a WHNF. Then $C[s_1^{d_1}, \ldots, (s_{j,1}^{d_{j,1}} : s_{j,2}^{d_{j,2}}), \ldots, s_n^{d_n}]$ is also a WHNF and the measure (μ_1, μ_2) is strictly decreased. Applying the

induction hypothesis to $C[s_1^{d_1}, \ldots, (s_{j,1}^{d_{j,1}} : s_{j,2}^{d_{j,2}}), \ldots, s_n^{d_n}]$ and $C[t_1^{e_1}, \ldots, (t_{j,1}^{e_{j,1}} : t_{j,2}^{e_{j,2}}), \ldots, t_n^{e_n}]$ shows the claim.

If a normal order reduction for $C[t_1^{e_1}, \ldots, (t_{j,1}^{e_{j,1}} : t_{j,2}^{e_{j,2}}), \ldots, t_n^{e_n}]$ exists, then – due to typing – it must be a (seq)- or (case)-reduction.

If the reduction is a (seq)-reduction, then it strictly decreases the measure μ_1. If the (seq)-reduction removes $(s_{j,1}^{d_{j,1}} : s_{j,2}^{d_{j,2}})$ and $(t_{j,1}^{e_{j,1}} : t_{j2}^{e_{j,2}})$, then we can apply the induction hypothesis. If the (seq)-reduction does not remove $(s_{j,1}^{d_{j,1}} : s_{j,2}^{d_{j,2}})$ and $(t_{j,1}^{e_{j,1}} : t_{j2}^{e_{j,2}})$, then the number of holes is increased, but all preconditions also hold for $s_{j,i}$ and $t_{j,i}$ and the induction hypothesis is applicable.

If the reduction is a (case)-reduction, then the expressions $s_{j,1}, s_{j,2}$ and also $t_{j,1}, t_{j,2}$ are moved into a `letrec`-environment and remain in surface-context position. Since μ_1 is strictly decreased, the preconditions hold for $s_{j,i}$ and $t_{j,i}$, we can apply the induction hypothesis which shows the claim. □

5 Conclusion

We have provided the necessary proofs of all the computation rules for unshared and shared decorations. There is also a proof of the simulation proof method for improvement.

References

1. Simon Marlow, editor. *Haskell 2010 – Language Report.* 2010. `www.haskell.org`.
2. Andrew Moran and David Sands. Improvement in a lazy context: An operational theory for call-by-need. In *Proc. POPL 1999*, pages 43–56. ACM Press, 1999.
3. Benjamin C. Pierce. *Types and Programming Languages.* The MIT Press, 2002.
4. Andrew M. Pitts. Parametric polymorphism and operational equivalence. *Math. Structures Comput. Sci.*, 10:321–359, 2000.
5. Manfred Schmidt-Schauß and David Sabel. On generic context lemmas for higher-order calculi with sharing. *Theoret. Comput. Sci.*, 411(11-13):1521 – 1541, 2010.
6. Manfred Schmidt-Schauß and David Sabel. Contextual equivalences in call-by-need and call-by-name polymorphically typed calculi (preliminary report). In *Proc. WPTE 2014*, volume 40 of *OASICS*, pages 63–74. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.
7. Manfred Schmidt-Schauß and David Sabel. Improvements in a functional core language with call-by-need operational semantics. In Elvira Albert, editor, *Proc. PPDP '15*, pages 220–231, New York, NY, USA, 2015. ACM.
8. Manfred Schmidt-Schauß and David Sabel. Improvements in a functional core language with call-by-need operational semantics. Frank report 55, Institut für Informatik, Goethe-Universität Frankfurt am Main, March 2015. `http://www.ki.informatik.uni-frankfurt.de/papers/frank/`.
9. Manfred Schmidt-Schauß, Marko Schütz, and David Sabel. Safety of Nöcker's strictness analysis. *J. Funct. Programming*, 18(04):503–551, 2008.
10. Dimitrios Vytiniotis and Simon Peyton Jones. Evidence Normalization in System FC (Invited Talk). In Femke van Raamsdonk, editor, *Proc. RTA 2013*, volume 21 of *LIPIcs*, pages 20–38, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

A Redundancy of rln-Decorations

We prove Proposition 3.2:

Proposition A.1. *The sharing rln-decorations* $s^{[n]}$ *can be encoded as* $\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)$ *and thus are redundant.*

Proof. We show that $s^{[n]} = \texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]} \approx \texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)$:
Let R be a reduction context. It suffices to show $R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] \sim_c R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)]$ and $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]) = \texttt{rln}(R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)])$, since then the context lemma for $\sim_c$ and the context lemma for improvement show the claim.

- If R is a weak reduction context, then

  $$R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] \xrightarrow{\mathrm{LRPw},lll,*} \texttt{letrec}\ a := n\ \texttt{in}\ R[s^{[a]}]$$

  and thus $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]) = \texttt{rln}(\texttt{letrec}\ a := n\ \texttt{in}\ R[s^{[a]}])$. Now

  $$\texttt{letrec}\ a := n\ \texttt{in}\ R[s^{[a]}] \xrightarrow{\mathrm{LRPw},letwn,n} \texttt{letrec}\ a := 0\ \texttt{in}\ R[s^{[a]}] \xrightarrow{\mathrm{LRPw},letw0} \texttt{letrec}\ a := 0\ \texttt{in}\ R[s]$$

  and thus $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]) = n + (R[\texttt{letrec}\ a := 0\ \texttt{in}\ s])$. Now Proposition 2.25 shows $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]) = n + (R[s])$.
  For $R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)]$ one can verify that

  $$\begin{array}{l} R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)] \\ \xrightarrow{\mathrm{LRPw},lll,*} (\xrightarrow{\mathrm{LRPw},lbeta}\xrightarrow{\mathrm{LRPw},llet})^{n-1} \xrightarrow{\mathrm{LRPw},cp}\xrightarrow{\mathrm{LRPw},lbeta}\xrightarrow{\mathrm{LRPw},lll,*} \\ \texttt{letrec}\ x = x_1, x_1 = x_2, x_{n-1} = id, x_n = s\ \texttt{in}\ R[x_n] \end{array}$$

  and thus $\texttt{rln}(R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)]) = n + \texttt{rln}(\texttt{letrec}\ x = x_1, x_1 = x_2, x_{n-1} = id, x_n = s\ \texttt{in}\ R[x_n])$. Finally, since

  $$\begin{array}{ll} & \texttt{letrec}\ x = x_1, x_1 = x_2, x_{n-1} = id, x_n = s\ \texttt{in}\ R[x_n] \\ \xrightarrow{ucp} & \texttt{letrec}\ x = x_1, x_1 = x_2, x_{n-1} = id, x_n = s\ \texttt{in}\ R[s] \\ \xrightarrow{gc2} & R[s] \end{array}$$

  and Theorem 2.26 shows that (ucp) and (gc2) do not change the $\texttt{rln}$-measure, we have $\texttt{rln}(R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)]) = n + \texttt{rln}(R[s])$. Concluding, this shows $R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] \sim_c R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)]$ since the left expression can be transformed into the right expression by correct program transformations and it also shows $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]) = \texttt{rln}(R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)])$.
- If R is not a weak reduction context, then there are two cases:
  1. $R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] \xrightarrow{\mathrm{LRPw},lll,*} \texttt{letrec}\ Env, a := n\ \texttt{in}\ R_1^-[s^{[a]}]$ and
     $R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)] \xrightarrow{\mathrm{LRPw},lll,*} \texttt{letrec}\ Env, x = (id^n)\ \texttt{in}\ R_1^-[(x\ s)]$ where R_1^- is a weak reduction context.
     For the left expression:

     $$\begin{array}{ll} & \texttt{letrec}\ Env, a := n\ \texttt{in}\ R_1^-[s^{[a]}] \\ \xrightarrow{\mathrm{LRPw},letwn,n} & \texttt{letrec}\ Env, a := 0\ \texttt{in}\ R_1^-[s^{[a]}] \\ \xrightarrow{\mathrm{LRPw},letw0} & \texttt{letrec}\ Env, a := 0\ \texttt{in}\ R_1^-[s] \\ \xrightarrow{gcW} & \texttt{letrec}\ Env\ \texttt{in}\ R_1^-[s] \end{array}$$

     and thus $R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]) \sim_c \texttt{letrec}\ Env\ \texttt{in}\ R_1^-[s]$ and $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]) = n + \texttt{rln}(\texttt{letrec}\ Env\ \texttt{in}\ R_1^-[s])$.

For the right expression

$$\begin{array}{ll} & \texttt{letrec}\ Env, x = (id^n)\ \texttt{in}\ R_1^-[(x\ s)] \\ (\xrightarrow{\text{LRPw},lbeta}\xrightarrow{\text{LRPw},llet})^{n-1} & \texttt{letrec}\ Env, x = x_1, x_1 = x_2, \ldots, x_{n-1} = id\ \texttt{in}\ R_1^-[(x\ s)] \\ \xrightarrow{\text{LRPw},cp} & \texttt{letrec}\ Env, x = x_1, x_1 = x_2, \ldots, x_{n-1} = id\ \texttt{in}\ R_1^-[(id\ s)] \\ \xrightarrow{\text{LRPw},lbeta} & \texttt{letrec}\ Env, x = x_1, x_1 = x_2, \ldots, x_{n-1} = id\ \texttt{in}\ R_1^-[\texttt{letrec}\ x_n = s\ \texttt{in}\ x_n] \\ \xrightarrow{\text{LRPw},lll,*} & \texttt{letrec}\ Env, x = x_1, x_1 = x_2, \ldots, x_{n-1} = id, x_n = s\ \texttt{in}\ R_1^-[x_n] \\ \xrightarrow{ucp} & \texttt{letrec}\ Env, x = x_1, x_1 = x_2, \ldots, x_{n-1} = id\ \texttt{in}\ R_1^-[s] \\ \xrightarrow{gc} & \texttt{letrec}\ Env\ \texttt{in}\ R_1^-[s] \end{array}$$

and thus $R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)] \sim_c \texttt{letrec}\ Env\ \texttt{in}\ R_1^-[s]$ and $\texttt{rln}(R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)]) = n + \texttt{rln}(\texttt{letrec}\ Env\ \texttt{in}\ R_1^-[s])$
Together this shows $R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] \sim_c R[\texttt{letrec}\ x = (id^n)\ \texttt{in}(x\ s)]$ and $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]) = \texttt{rln}(R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)])$.

2.
$$\begin{array}{ll} & R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] \\ \xrightarrow{\text{LRPw},lll,*} & \texttt{letrec}\ Env, a := n, y_1 = R_1^-[s^{[a]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m,\ \texttt{in}\ R_0^-[y_m] \end{array}$$

and

$$\begin{array}{ll} & R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)] \\ \xrightarrow{\text{LRPw},lll,*} & \texttt{letrec}\ Env, x = (id^n), y_1 = R_1^-[(x\ s)], \{y_i = R_i^{y_{i-1}}\}_{i=2}^m\ \texttt{in}\ R_0^-[y_m] \end{array}$$

where $R_0^-, \ldots, R_m$ are weak reduction contexts.
For the left expression:

$$\begin{array}{ll} & \texttt{letrec}\ Env, a := n, y_1 = R_1^-[s^{[a]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \texttt{in}\ R_0^-[y_m] \\ \xrightarrow{\text{LRPw},letwn,n} & \texttt{letrec}\ Env, a := 0, y_1 = R_1^-[s^{[a]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \texttt{in}\ R_0^-[y_m] \\ \xrightarrow{\text{LRPw},letw0} & \texttt{letrec}\ Env, a := 0, y_1 = R_1^-[s], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \texttt{in}\ R_0^-[y_m] \\ \xrightarrow{gcW} & \texttt{letrec}\ Env, y_1 = R_1^-[s], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \texttt{in}\ R_0^-[y_m] \end{array}$$

and thus

$$R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] \sim_c \texttt{letrec}\ Env, y_1 = R_1^-[s], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \texttt{in}\ R_0^-[y_m]$$

and

$$\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]) = n + \texttt{rln}(\texttt{letrec}\ Env, y_1 = R_1^-[s], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \texttt{in}\ R_0^-[y_m])$$

For the right expression:

$$\begin{array}{ll} & \texttt{letrec}\ Env, x = (id^n), y_1 = R_1^-[(x\ s)], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \texttt{in}\ R_0^-[y_m] \\ (\xrightarrow{\text{LRPw},lbeta}\xrightarrow{\text{LRPw},llet})^{n-1} & \texttt{letrec}\ Env, x = x_1, \{x_i = x_{i+1}\}_{i=1}^{n-2}, x_{n-1} = id, \\ & \qquad y_1 = R_1^-[(x\ s)], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m \\ & \texttt{in}\ R_0^-[y_m] \\ \xrightarrow{\text{LRPw},cp}\xrightarrow{\text{LRPw},lbeta}\xrightarrow{lll,*} & \texttt{letrec}\ Env, x = x_1, \{x_i = x_{i+1}\}_{i=1}^{n-2}, x_{n-1} = id, x_n = s, \\ & \qquad y_1 = R_1^-[x_n], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m \\ & \texttt{in}\ R_0^-[y_m] \\ \xrightarrow{ucp}\xrightarrow{gc} & \texttt{letrec}\ Env, y_1 = R_1^-[s], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \texttt{in}\ R_0^-[y_m] \end{array}$$

and thus

$$R[\texttt{letrec}\ x = (id^n)\ \texttt{in}\ (x\ s)] \sim_c \texttt{letrec}\ Env, y_1 = R_1^-[s], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \texttt{in}\ R_0^-[y_m]$$

and

$$\texttt{rln}(R[\texttt{letrec}\,x = (id^n)\;\texttt{in}\,(x\;s)]) = n + \texttt{rln}(\texttt{letrec}\,Env, y_1 = R_1^-[s], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\;\texttt{in}\,R_0^-[y_m])$$

Together this shows

$$R[\texttt{letrec}\,a := n\;\texttt{in}\;s^{[a]}] \sim_c R[\texttt{letrec}\,x = (id^n)\;\texttt{in}\;(x\;s)]$$

and

$$\texttt{rln}(R[\texttt{letrec}\,a := n\;\texttt{in}\;s^{[a]}]) = \texttt{rln}(R[\texttt{letrec}\,x = (id^n)\;\texttt{in}\;(x\;s)])$$

B Proofs of Computation Rules

The following theorem summarizes the results proved in this section.

Theorem B.1. *1. If $s \xrightarrow{\text{LRPw},a} t$ with $a \in \{lbeta, case, seq, letwn\}$, then $s \approx t^{[1]}$.*

2. *$R[\texttt{letrec}\,a := n\;\texttt{in}\;s^{[a]}] \approx \texttt{letrec}\,a := n\;\texttt{in}\;R[s]^{[a]}$ and thus in particular $R[s^{[n]}] \approx R[s]^{[n]}$.*
3. *$\texttt{rln}(\texttt{letrec}\,a := n\;\texttt{in}\;s^{[a]}) = n + \texttt{rln}(s')$ where s' is s where all $^{[a]}$-labels are removed. In particular this also shows $\texttt{rln}(s^{[n]}) = n + \texttt{rln}(s)$*
4. *For every reduction context R: $\texttt{rln}(R[\texttt{letrec}\,a := n\;\texttt{in}\;s^{[a]}]) = n + \texttt{rln}(R[s'])$ where s' is s where all $^{[a]}$-labels are removed. In particular, this shows $\texttt{rln}(R[s^{[n]}]) = n + \texttt{rln}(R[s])$.*
5. *$(s^{[n]})^{[m]} \approx s^{[n+m]}$*
6. *For all surface contexts S_1, S_2: $S_1[\texttt{letrec}\,a := n\;\texttt{in}\;S_2[s^{[a]}]] \preceq \texttt{letrec}\,a := n\;\texttt{in} S_1[S_2[s]]^{[a]}$ and if $S_1[S_2]$ is strict, also $S_1[\texttt{letrec}\,a := n\;\texttt{in}\;S_2[s^{[a]}]] \approx \texttt{letrec}\,a := n\;\texttt{in} S_1[S_2[s]]^{[a]}$.*
7. *$\texttt{letrec}\,a := n, b := m\;\texttt{in}\;(s^{[a]})^{[b]} \approx \texttt{letrec}\,a := n, b := m\;\texttt{in}\;(s^{[b]})^{[a]}$*
8. *$\texttt{letrec}\,a := n\;\texttt{in}\;(s^{[a]})^{[a]} \approx \texttt{letrec}\,a := n\;\texttt{in}\;(s^{[a]})$*
9. *$(t^{p_1})^{p_2} \approx t^{p_1 \oplus p_2}$.*
10. *Let $S[\cdot, \ldots, \cdot]$ be a multi-context where all holes are in surface position. Then $\texttt{letrec}\;a := n\;\texttt{in}\;S[s_1^{[a]}, \ldots, s_n^{[a]}] \preceq \texttt{letrec}\;a := n\;\texttt{in}\;S[s_1, \ldots, s_n]^{[a]}$. If some hole $\cdot_i$ with $i \in \{1, \ldots, n\}$ is in strict position in $S[\ldots, .]$, then $\texttt{letrec}\;a := n\;\texttt{in}\;S[s_1^{[a]}, \ldots, s_n^{[a]}] \approx \texttt{letrec}\;a := n\;\texttt{in}\;S[s_1, \ldots, s_n]^{[a]}$.*
11. *Let $S[\cdot, \ldots, \cdot]$ be a multi-context where all holes are in surface position. Let $S[s_1, \ldots, s_n]$ be closed. Then $S[s_1^{[p_1, a \mapsto m]}, \ldots, s_n^{[p_n, a \mapsto m]}] \preceq S[s_1^{p_1}, \ldots, s_n^{p_n}]^{[m]}$.*
    If some hole $\cdot_i$ with $i \in \{1, \ldots, n\}$ is in strict position in $S[\ldots, .]$, then $S[s_1^{[p_1, a \mapsto m]}, \ldots, s_n^{[p_n, a \mapsto m]}] \approx S[s_1^{p_1}, \ldots, s_n^{p_n}]^{[m]}$.
12. *The following transformation is correct w.r.t. $\approx$: Replace $(\texttt{letrec}\;x = s^{[n,p]}, Env\;\texttt{in}\;t)$ by $\texttt{letrec}\;x = s[x^{[a \mapsto n,p]}/x],\; Env[x^{[a \mapsto n,p]}/x]\;\texttt{in}\;t[x^{[a \mapsto n,p]}/x]$, where a is a fresh label and all occurrences of x are in surface position.*
13. *If a label-name a occurs exactly once in a surface context, then it can be changed into an unshared decoration.*

Proof. (1) is proved in Theorem B.8.
(2) is proved in Proposition B.3.
(3) is proved in Lemma B.4.
(4) is proved in Corollary B.5.
(5) is proved in Proposition B.6.
(6) is proved in Corollary B.12.
(7) and (8) are proved in Proposition B.13, and (9) holds by iteratively applying items (4), (7) and (8) and by applying (lll) and (gcW)-transformations which are invariant w.r.t. $\approx$.
(10) is proved in Proposition B.14.
(11) is proved in Corollary B.15.
(12) is proved in Proposition B.16.
(13) follows from the semantics of the labels.

Proposition B.2. *If* $s \xrightarrow{\mathrm{LRPw},letwn} t$, *then* $s \approx t^{[1]}$

Proof. We use the context lemma for improvement and thus have to show for all reduction contexts R:

$\mathtt{rln}(R[s]) = \mathtt{rln}(R[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}])$

There are three general cases for the reduction context R and two cases for s and t:

1. $s = \mathtt{letrec}\ b := n, Env\ \mathtt{in}\ R_0^-[r^{[b]}]$,
   $t = \mathtt{letrec}\ b := n-1, Env\ \mathtt{in}\ R_0^-[r^{[b]}]$

   (a) R is a weak reduction context. Then $\mathtt{rln}(R[s]) = \mathtt{rln}(R[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}])$, since:

$$\begin{array}{lll}
R[s] & \xrightarrow{\mathrm{LRPw},lll,*} & \mathtt{letrec}\ b := n, Env\ \mathtt{in}\ R[R_0^-[r^{[b]}]] \\
 & \xrightarrow{\mathrm{LRPw},letwn} & \mathtt{letrec}\ b := n-1, Env\ \mathtt{in}\ R[R_0^-[r^{[b]}]] \\
R[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}] & \xrightarrow{\mathrm{LRPw},lll,*} & \mathtt{letrec}\ a := 1\ \mathtt{in}\ R[(\mathtt{letrec}\ b := n-1, Env\ \mathtt{in}\ R_0^-[r^{[b]}])^{[a]}] \\
 & \xrightarrow{\mathrm{LRPw},lll,*} & \mathtt{letrec}\ a := 1, b := n-1, Env\ \mathtt{in}\ R[R_0^-[r^{[b]}])^{[a]}] \\
 & \xrightarrow{\mathrm{LRPw},letwn} & \mathtt{letrec}\ a := 0, b := n-1, Env\ \mathtt{in}\ R[R_0^-[r^{[b]}])^{[a]}] \\
 & \xrightarrow{\mathrm{LRPw},letw0} & \mathtt{letrec}\ a := 0, b := n-1, Env\ \mathtt{in}\ R[R_0^-[r^{[b]}])] \\
 & \xrightarrow{gcW} & \mathtt{letrec}\ b := n-1, Env\ \mathtt{in}\ R[R_0^-[r^{[b]}])]
\end{array}$$

   (b) $R = \mathtt{letrec}\ Env'\ \mathtt{in}\ R'[\cdot]$, where R' is a weak reduction context. Then $\mathtt{rln}(R[s]) = \mathtt{rln}(R[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}])$, since:

$$\begin{array}{l}
\mathtt{letrec}\ Env'\ \mathtt{in}\ R'[s] \xrightarrow{\mathrm{LRPw},lll,*} \mathtt{letrec}\ b := n, Env, Env'\ \mathtt{in}\ R'[R_0^-[r^{[b]}]] \\
\qquad \xrightarrow{\mathrm{LRPw},letwn} \mathtt{letrec}\ b := n-1, Env, Env'\ \mathtt{in}\ R'[R_0^-[r^{[b]}]] \\
\mathtt{letrec}\ Env'\ \mathtt{in}\ R'[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}] \\
\xrightarrow{\mathrm{LRPw},lll,*} \mathtt{letrec}\ Env', a := 1\ \mathtt{in}\ R'[(\mathtt{letrec}\ b := n-1, Env\ \mathtt{in}\ R_0^-[r^{[b]}])^{[a]}] \\
\xrightarrow{\mathrm{LRPw},lll,*} \mathtt{letrec}\ Env', a := 1, b := n-1, Env\ \mathtt{in}\ R'[R_0^-[r^{[b]}])^{[a]}] \\
\xrightarrow{\mathrm{LRPw},letwn} \mathtt{letrec}\ Env', a := 0, b := n-1, Env\ \mathtt{in}\ R'[R_0^-[r^{[b]}])^{[a]}] \\
\xrightarrow{\mathrm{LRPw},letw0} \mathtt{letrec}\ Env', a := 0, b := n-1, Env\ \mathtt{in}\ R'[R_0^-[r^{[b]}])] \\
\xrightarrow{gcW} \mathtt{letrec}\ Env', b := n-1, Env\ \mathtt{in}\ R'[R_0^-[r^{[b]}])]
\end{array}$$

   (c) $R = \mathtt{letrec}\ Env', x = R'[\cdot]\ \mathtt{in}\ u$, where R' is a weak reduction context. Then $\mathtt{rln}(R[s]) = \mathtt{rln}(R[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}])$, since:

$$\begin{array}{l}
\mathtt{letrec}\ Env', x = R'[s]\ \mathtt{in}\ u \xrightarrow{\mathrm{LRPw},lll,*} \mathtt{letrec}\ b := n, Env, Env', x = R'[R_0^-[r^{[b]}]]\ \mathtt{in}\ u \\
\qquad \xrightarrow{\mathrm{LRPw},letwn} \mathtt{letrec}\ b := n-1, Env, Env', x = R'[R_0^-[r^{[b]}]]\ \mathtt{in}\ u \\
\mathtt{letrec}\ Env', x = R'[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}]\ \mathtt{in}\ u \\
\xrightarrow{\mathrm{LRPw},lll,*} \mathtt{letrec}\ Env', a := 1, x = R'[(\mathtt{letrec}\ b := n-1, Env\ \mathtt{in}\ R_0^-[r^{[b]}])^{[a]}]\ \mathtt{in}\ u \\
\xrightarrow{\mathrm{LRPw},lll,*} \mathtt{letrec}\ Env', a := 1, b := n-1, Env, x = R'[R_0^-[r^{[b]}]^{[a]}]\ \mathtt{in}\ u \\
\xrightarrow{\mathrm{LRPw},letwn} \mathtt{letrec}\ Env', a := 0, b := n-1, Env, x = R'[R_0^-[r^{[b]}]^{[a]}]\ \mathtt{in}\ u \\
\xrightarrow{\mathrm{LRPw},letw0} \mathtt{letrec}\ Env', a := 0, b := n-1, Env, x = R'[R_0^-[r^{[b]}]]\ \mathtt{in}\ u \\
\xrightarrow{\mathrm{LRPw},gcW} \mathtt{letrec}\ Env', b := n-1, Env, x = R'[R_0^-[r^{[b]}]]\ \mathtt{in}\ u
\end{array}$$

2. $s = \mathtt{letrec}\ b := n, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \mathtt{in}\ R_{m+1}^-[y_m]$
   $t = \mathtt{letrec}\ b := n-1, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \mathtt{in}\ R_{m+1}^-[y_m]$

(a) R is a weak reduction context. Then $\mathtt{rln}(R[s]) = \mathtt{rln}(R[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}])$, since:

$$
\begin{array}{l}
R[s] \xrightarrow{\text{LRPw},lll,*} \mathtt{letrec}\ b := n, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \mathtt{in}\ R[R_{m+1}^-[y_m]] \\
\quad \xrightarrow{\text{LRPw},letwn} \mathtt{letrec}\ b := n-1, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \mathtt{in}\ R[R_{m+1}^-[y_m]] \\
R[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}] \\
\xrightarrow{\text{LRPw},lll,*} \mathtt{letrec}\ a := 1\ \mathtt{in} \\
\qquad R[(\mathtt{letrec}\ b := n-1, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \mathtt{in}\ R_{m+1}^-[y_m])^{[a]}] \\
\xrightarrow{\text{LRPw},letwn} \mathtt{letrec}\ a := 0\ \mathtt{in} \\
\qquad R[(\mathtt{letrec}\ b := n-1, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \mathtt{in}\ R_{m+1}^-[y_m])^{[a]}] \\
\xrightarrow{\text{LRPw},letw0} \mathtt{letrec}\ a := 0\ \mathtt{in} \\
\qquad R[(\mathtt{letrec}\ b := n-1, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \mathtt{in}\ R_{m+1}^-[y_m])] \\
\xrightarrow{gcW} R[(\mathtt{letrec}\ b := n-1, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \mathtt{in}\ R_{m+1}^-[y_m])] \\
\xrightarrow{\text{LRPw},lll,*} \mathtt{letrec}\ b := n-1, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \mathtt{in}\ R[R_{m+1}^-[y_m]])
\end{array}
$$

(b) $R = \mathtt{letrec}\ Env'\ \mathtt{in}\ R'[\cdot]$, where R' is a weak reduction context. Then $\mathtt{rln}(R[s]) = \mathtt{rln}(R[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}])$, since:

$$
\begin{array}{l}
\mathtt{letrec}\ Env'\ \mathtt{in}\ R'[s] \\
\xrightarrow{\text{LRPw},lll,*} \mathtt{letrec}\ Env', b := n, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \mathtt{in}\ R'[R_{m+1}^-[y_m]] \\
\xrightarrow{\text{LRPw},letwn} \mathtt{letrec}\ Env', b := n-1, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \mathtt{in}\ R'[R_{m+1}^-[y_m]] \\
\mathtt{letrec}\ Env'\ \mathtt{in}\ R'[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}] \\
\xrightarrow{\text{LRPw},lll,*} \mathtt{letrec}\ Env', a := 1\ \mathtt{in}\ R'[t^{[a]}] \\
\xrightarrow{\text{LRPw},letwn} \mathtt{letrec}\ Env', a := 0\ \mathtt{in}\ R'[t^{[a]}] \\
\xrightarrow{\text{LRPw},letw0} \mathtt{letrec}\ Env', a := 0\ \mathtt{in}\ R'[t] \\
\xrightarrow{gcW} \mathtt{letrec}\ Env'\ \mathtt{in}\ R'[t] \\
\xrightarrow{\text{LRPw},lll,*} \mathtt{letrec}\ Env', b := n-1, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m\ \mathtt{in}\ R'[R_{m+1}^-[y_m]]
\end{array}
$$

(c) $R = \mathtt{letrec}\ Env', x = R'[\cdot]\ \mathtt{in}\ u$, where R' is a weak reduction context. Then $\mathtt{rln}(R[s]) = \mathtt{rln}(R[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}])$, since:

$$
\begin{array}{l}
\mathtt{letrec}\ Env', x = R'[s]\ \mathtt{in}\ u \\
\xrightarrow{\text{LRPw},lll,*} \mathtt{letrec}\ Env', b := n, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m, x = R'[R_{m+1}^-[y_m]]\ \mathtt{in}\ u \\
\xrightarrow{\text{LRPw},letwn} \mathtt{letrec}\ Env', b := n-1, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m, x = R'[R_{m+1}^-[y_m]]\ \mathtt{in}\ u \\
\mathtt{letrec}\ Env', x = R'[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}]\ \mathtt{in}\ u \\
\xrightarrow{\text{LRPw},lll,*} \mathtt{letrec}\ Env', a := 1, x = R'[t^{[a]}]\ \mathtt{in}\ u \\
\xrightarrow{\text{LRPw},letwn} \mathtt{letrec}\ Env', a := 0, x = R'[t^{[a]}]\ \mathtt{in}\ u \\
\xrightarrow{\text{LRPw},letw0} \mathtt{letrec}\ Env', a := 0, x = R'[t]\ \mathtt{in}\ u \\
\xrightarrow{gcW} \mathtt{letrec}\ Env', x = R'[t]\ \mathtt{in}\ u \\
\xrightarrow{\text{LRPw},lll,*} \mathtt{letrec}\ Env', b := n-1, Env, y_1 = R_0^-[r^{[b]}], \{y_i = R_i^-[y_{i-1}]\}_{i=2}^m, x = R'[R_{m+1}^-[y_m]]\ \mathtt{in}\ u
\end{array}
$$

Proposition B.3. $R[\mathtt{letrec}\ a := n\ \mathtt{in}\ s^{[a]}] \approx \mathtt{letrec}\ a := n\ \mathtt{in}\ R[s]^{[a]}$ *and thus in particular* $R[s^{[n]}] \approx R[s]^{[n]}$

Proof. We show $R[\mathtt{letrec}\ a := n\ \mathtt{in}\ s^{[a]}] \approx \mathtt{letrec}\ a := n\ \mathtt{in}\ R[s]^{[a]}$ by induction on n. If $n = 0$ then the claim holds, since $(letw0) \subseteq \approx$.

For the induction step assume that the claim holds for all k, with $k \leq n - 1$.

We make a case distinction on the reduction context R.

1. R is a weak reduction context. Then

$$\begin{array}{ll}
\, R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] & \\
\approx \texttt{letrec}\ a := n\ \texttt{in}\ R[s^{[a]}] & \text{(since } (lll) \subseteq\approx) \\
\approx \texttt{letrec}\ b := 1\ \texttt{in}\ (\texttt{letrec}\ a := n-1\ \texttt{in}\ R[s^{[a]}])^{[b]} & \text{(by Proposition B.2)} \\
\approx \texttt{letrec}\ b := 1\ \texttt{in}\ R[\texttt{letrec}\ a := n-1\ \texttt{in}\ s^{[a]}]^{[b]} & \text{(since } (lll) \subseteq\approx) \\
\approx \texttt{letrec}\ b := 1\ \texttt{in}\ (\texttt{letrec}\ a := n-1\ \texttt{in}\ (R[s]^{[a]}))^{[b]} & \text{(induction hypothesis)} \\
\approx \texttt{letrec}\ a := n\ \texttt{in}\ (R[s]^{[a]}) & \text{(by Proposition B.2)}
\end{array}$$

2. $R = \texttt{letrec}\ \mathit{Env}\ \texttt{in}\ R'[\cdot]$ where R' is a weak reduction context.

$$\begin{array}{ll}
\, R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] & \\
= \texttt{letrec}\ \mathit{Env}\ \texttt{in}\ R'[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] & \\
\approx \texttt{letrec}\ a := n, \mathit{Env},\ R'[s^{[a]}] & \text{(since } (lll) \subseteq\approx) \\
\approx \texttt{letrec}\ b := 1\ \texttt{in}\ (\texttt{letrec}\ a := n-1, \mathit{Env}\ \texttt{in}\ R'[s^{[a]}])^{[b]} & \text{(by Proposition B.2)} \\
\approx \texttt{letrec}\ b := 1\ \texttt{in}\ R[\texttt{letrec}\ a := n-1\ \texttt{in}\ s^{[a]}]^{[b]} & \text{(since } (lll) \subseteq\approx) \\
\approx \texttt{letrec}\ b := 1\ \texttt{in}\ (\texttt{letrec}\ a := n-1\ \texttt{in}\ R[s]^{[a]})^{[b]} & \text{(induction hypothesis)} \\
= \texttt{letrec}\ b := 1\ \texttt{in}\ (\texttt{letrec}\ a := n-1\ \texttt{in}\ (\texttt{letrec}\ \mathit{Env}\ \texttt{in}\ R'[s])^{[a]})^{[b]} & \\
\approx \texttt{letrec}\ a := n\ \texttt{in}\ (\texttt{letrec}\ \mathit{Env}\ \texttt{in}\ R'[s])^{[a]}) & \text{(by Proposition B.2)} \\
= \texttt{letrec}\ a := n\ \texttt{in}\ (R[s])^{[a]} &
\end{array}$$

3. $R = \texttt{letrec}\ \mathit{Env}, x = R'[\cdot]\ \texttt{in}\ u$ where R' is a weak reduction context.

$$\begin{array}{ll}
\, R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] & \\
= \texttt{letrec}\ \mathit{Env}, x = R'[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]\ \texttt{in}\ u & \\
\approx \texttt{letrec}\ \mathit{Env}, a := n, x = R'[s^{[a]}]\ \texttt{in}\ u & \text{(since } (lll) \subseteq\approx) \\
\approx \texttt{letrec}\ b := 1\ \texttt{in}\ (\texttt{letrec}\ \mathit{Env}, a := n-1, x = R'[s^{[a]}]\ \texttt{in}\ u)^{[b]} & \text{(by Proposition B.2)} \\
\approx \texttt{letrec}\ b := 1\ \texttt{in}\ (\texttt{letrec}\ \mathit{Env}, x = (R'[\texttt{letrec}\ a := n-1\ \texttt{in}\ s^{[a]}])\ \texttt{in}\ u)^{[b]} & \text{(since } (lll) \subseteq\approx) \\
= \texttt{letrec}\ b := 1\ \texttt{in}\ (R[\texttt{letrec}\ a := n-1\ \texttt{in}\ s^{[a]}])^{[b]} & \\
\approx \texttt{letrec}\ b := 1\ \texttt{in}\ (\texttt{letrec}\ a := n-1\ \texttt{in}\ R[s]^{[a]}])^{[b]} & \text{(induction hypothesis)} \\
= \texttt{letrec}\ b := 1\ \texttt{in}\ (\texttt{letrec}\ a := n-1\ \texttt{in}\ (\texttt{letrec}\ \mathit{Env}, x = R'[s]\ \texttt{in}\ u)^{[a]})^{[b]} & \\
\approx \texttt{letrec}\ a := n\ \texttt{in}\ (\texttt{letrec}\ \mathit{Env}, x = R'[s]\ \texttt{in}\ u)^{[a]} & \text{(by Proposition B.2)} \\
= \texttt{letrec}\ a := n\ \texttt{in}\ R[s]^{[a]} &
\end{array}$$

Lemma B.4. $\texttt{rln}(\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}) = n + \texttt{rln}(s')$ *where* s' *is* s *where all* ${}^{[a]}$*-labels are removed. In particular this also shows* $\texttt{rln}(s^{[n]}) = n + \texttt{rln}(s)$

Proof. The reduction $\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]} \xrightarrow{\text{LRPw},letwn,n} \xrightarrow{C,letw0,*} \texttt{letrec}\ a := 0\ \texttt{in}\ s'$ shows $\texttt{rln}(s^{[n]}) = n + \texttt{rln}(\texttt{letrec}\ a := 0\ \texttt{in}\ s)$. Finally, $(gcW) \subseteq\approx$ shows the claim.

Corollary B.5. *For every reduction context R:* $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]) = n + \texttt{rln}(R[s'])$ *where* s' *is* s *where all* ${}^{[a]}$*-labels are removed. In particular, this shows* $\texttt{rln}(R[s^{[n]}]) = n + \texttt{rln}(R[s])$.

Proof. By Proposition B.3 we have $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]) = \texttt{rln}(\texttt{letrec}\ a := n\ \texttt{in}\ R[s]^{[a]})$ and by Lemma B.4 we have $\texttt{rln}(\texttt{letrec}\ a := n\ \texttt{in}\ R[s]^{[a]}) = n + \texttt{rln}(R[s'])$.

Proposition B.6. $(s^{[n]})^{[m]} \approx s^{[n+m]}$

Proof. Clearly $(s^{[n]})^{[m]} \sim_c s^{[n+m]}$ Let R be a reduction context, then $\texttt{rln}(R[(s^{[n]})^{[m]}]) = m + \texttt{rln}([R[s^{[n]}]) = m + n + \texttt{rln}(R[s]) = R[s^{[n+m]}]$ by Corollary B.5. Now the context lemma for improvement shows the claim.

Lemma B.7. *If* $s \xrightarrow{\text{LRPw},a} t$ *with* $a \in \{lbeta, case-c, seq-c\}$, *then* $s \approx t^{[1]}$

Proof. We use the context lemma for improvement and thus have to show for all reduction contexts R:

$\mathtt{rln}(R[s]) = \mathtt{rln}(R[t^{[1]}])$
By Corollary B.5 we have $\mathtt{rln}(R[t^{[1]}]) = 1 + \mathtt{rln}(R[t])$ and thus it suffices to show
$\mathtt{rln}(R[s]) = 1 + \mathtt{rln}(R[t])$
Let $s_0 \xrightarrow{a} t_0$ for $a \in \{lbeta, case-c, seq\}$. We go through the case for s and t;

1. $s = R_0^-[s_0]$, Then $R[s] \xrightarrow{LRPw,a} R[t]$ and thus $\mathtt{rln}(R[s]) = 1 + \mathtt{rln}(R[t])$.
   $t = R_0^-[t_0]$
2. $s = \mathtt{letrec}\ Env\ \mathtt{in}\ R_0^-[s_0]$, We go through the cases for R:
   $t = \mathtt{letrec}\ Env\ \mathtt{in}\ R_0^-[t_0]$
   (a) R is a weak reduction context. Then

$$\begin{array}{lll} R[s] & \xrightarrow{LRPw,lll,*} & \mathtt{letrec}\ Env\ \mathtt{in}\ R[R_0^-[s_0]] \\ & \xrightarrow{LRPw,a} & \mathtt{letrec}\ Env\ \mathtt{in}\ R[R_0^-[t_0]] \\ & \xleftarrow{LRPw,lll,*} & R[\mathtt{letrec}\ Env\ \mathtt{in}\ R_0^-[t_0]] = R[t] \end{array}$$

   (b) $R = \mathtt{letrec}\ Env'\ \mathtt{in}\ R'$, where R' is a weak reduction context. Then

$$\begin{array}{lll} R[s] & \xrightarrow{LRPw,lll,*} & \mathtt{letrec}\ Env, Env'\ \mathtt{in}\ R'[R_0^-[s_0]] \\ & \xrightarrow{LRPw,a} & \mathtt{letrec}\ Env, Env'\ \mathtt{in}\ R'[R_0^-[t_0]] \\ & \xleftarrow{C,lll,*} & \mathtt{letrec}\ Env'\ \mathtt{in}\ R'[\mathtt{letrec}\ Env\ \mathtt{in}\ R_0^-[t_0]] \\ & = & R[t] \end{array}$$

   (c) $R = \mathtt{letrec}\ Env', u = R'\ \mathtt{in}\ r$, where R' is a weak reduction context. Then $\mathtt{rln}(R[s]) = \mathtt{rln}(R[\mathtt{letrec}\ a := 1\ \mathtt{in}\ t^{[a]}])$, since:

$$\begin{array}{lll} R[s] & \xrightarrow{LRPw,lll,*} & \mathtt{letrec}\ Env, Env', u = R'[R_0^-[s_0]]\ \mathtt{in}\ r \\ & \xrightarrow{LRPw,a} & \mathtt{letrec}\ Env, Env', u = R'[R_0^-[t_0]]\ \mathtt{in}\ r \\ & \xleftarrow{C,lll,*} & \mathtt{letrec}\ Env', u = R'[\mathtt{letrec}\ Env\ \mathtt{in} R_0^-[t_0]]\ \mathtt{in}\ r \\ & = & R[t] \end{array}$$

3. $s = \mathtt{letrec}\ Env, y = R_0^-[s_0]\ \mathtt{in}\ u_0$ We go through the cases for R:
   $t = \mathtt{letrec}\ Env, y = R_0^-[t_0]\ \mathtt{in}\ u_0$
   (a) R is a weak reduction context. Then

$$\begin{array}{lll} R[s] & \xrightarrow{LRPw,lll,*} & \mathtt{letrec}\ Env, y = R_0^-[s_0]\ \mathtt{in}\ R[u_0] \\ & \xrightarrow{LRPw,a} & \mathtt{letrec}\ Env, y = R_0^-[t_0]\ \mathtt{in}\ R[u_0] \\ & \xleftarrow{C,lll,*} & R[t] \end{array}$$

   (b) $R = \mathtt{letrec}\ Env'\ \mathtt{in}\ R'$, where R' is a weak reduction context.

$$\begin{array}{lll} R[s] & \xrightarrow{LRPw,lll,*} & \mathtt{letrec}\ Env', Env, y = R_0^-[s_0]\ \mathtt{in}\ R'[u_0] \\ & \xrightarrow{LRPw,a} & \mathtt{letrec}\ Env', Env, y = R_0^-[t_0]\ \mathtt{in}\ R'[u_0] \\ & \xleftarrow{C,lll,*} & R[t] \end{array}$$

   (c) $R = \mathtt{letrec}\ Env', u = R'\ \mathtt{in}\ r$, where R' is a weak reduction context. Then

$$\begin{array}{lll} R[s] & \xrightarrow{LRPw,lll,*} & \mathtt{letrec}\ Env', Env, y = R_0^-[s_0], u = R'[u_0]\ \mathtt{in}\ r \\ & \xrightarrow{LRPw,a} & \mathtt{letrec}\ Env', Env, y = R_0^-[t_0], u = R'[u_0]\ \mathtt{in}\ r \\ & \xleftarrow{C,lll,*} & R[t] \end{array}$$

Theorem B.8. *If* $s \xrightarrow{LRPw,a} t$ *with* $a \in \{lbeta, case, seq, letwn\}$, *then* $s \approx t^{[1]}$

Proof. For (letwn) this was proved in Proposition B.2, for (case-c), (seq-c), and (lbeta) this was proved in Lemma B.7. For the remaining (case) and (seq)-reductions, it suffices to observe that these transformation can be expressed by using one (LRPw,case-c)-reduction (or (LRPw,seq-c)-reduction respectively) and (cpcx), (gc), and (lll) transformations. Since for all these transformation we have $u \xrightarrow{C,cpcx \vee gc \vee lll} u'$ implies $u \approx u'$ (Theorem 2.26) the claim follows.

Proposition B.9. *For any strict surface context* S: $S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] \approx \texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}$ *and thus in particular* $S[s^{[n]}] \approx S[s]^{[n]}$

Proof. If $S[r] \sim_c \bot$ for all r, then $S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] \sim_c \bot \sim_c \texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}$ and since for any reduction context R: $R[\bot]\uparrow$, $R[S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]]\uparrow$ and $R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}]\uparrow$ and thus $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]) = \infty = \texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}])$ and the context lemma for improvement shows $S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] \approx \texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}$.

Otherwise, for every r and any reduction context R: $R[S[r]] \xrightarrow{LRPw,k} R'[r]$ where R' is a reduction context. and $\texttt{rln}(R[S[r]] \xrightarrow{LRPw,k} R'[r]) = m$ for some $m \leq k$.

For $R[S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]]$, we have $\texttt{rln}(R[S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]]) = m + \texttt{rln}(R'[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}])$ and by Proposition B.3 we have $\texttt{rln}(R'[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]) = \texttt{rln}(\texttt{letrec}\ a := n\ \texttt{in}\ R'[s]^{[a]})$. By Lemma B.4 we have $\texttt{rln}(\texttt{letrec}\ a := n\ \texttt{in}\ R'[s]^{[a]}) = n + \texttt{rln}(R'[s'])$ where s' is s where all $^{[a]}$-labels are removed. Thus $R[S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]] = m + n + \texttt{rln}(R'[s'])$

For $R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}]$, we have by Corollary B.5 $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}]) = n + \texttt{rln}(R[S[s']])$ where s' is s where all $^{[a]}$-labels are removed. Since $\texttt{rln}(R[S[s']]) = m + \texttt{rln}(R'[S[s'])$ we have $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}]) = n + m + \texttt{rln}(R'[S[s'])$.

Concluding we have shown $\texttt{rln}(R[S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]]) = \texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}])$ and clearly also $R[S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]] \sim_c R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}]$ holds. Thus the context lemma for improvement shows the claim.

Proposition B.10. *Let* S *be a surface context. Then* $S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] \preceq \texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}$ *and in particular* $S[s^{[n]}] \preceq S[s]^{[n]}$.

Proof. Let R be a reduction context. If $R[S]$ is strict, then Proposition B.9 shows $R[S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]] \preceq R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}]$ and thus $\texttt{rln}(R[S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]]) \leq \texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}])$ for all reduction contexts.

If $R[S]$ is non-strict, then $\texttt{rln}(R[S[r]]) = m_R$ for any R and where m_R depends only depends the context $R[S]$. Then $\texttt{rln}(R[S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}]]) = m_R$. From Corollary B.5 we have $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}]) = n + \texttt{rln}(R[S[s']])$ where s' is s where all $^{[a]}$-labels are removed. Thus $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}]) = n + m_R$. Since $S[\texttt{letrec}\ a := n\ \texttt{in}\ s^{[a]}] \sim_c \texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}$ (by correctness of $(letw)$ and (gcW)), the context lemma for improvement shows the claim.

Proposition B.11. *Let* S *be a surface context. Then* $\texttt{letrec}\ a := n\ \texttt{in}\ S[s^{[a]}] \preceq \texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}$, *and if* S *is strict, then* $\texttt{letrec}\ a := n\ \texttt{in}\ S[s^{[a]}] \approx \texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}$.

Proof. First assume that S is strict. Let R be a reduction context. Then $S' := R[\texttt{letrec}\ a := n\ \texttt{in}\ S[\cdot]$ is also strict. If $S'[r] \sim_c \bot$ for all r, then $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s^{[a]}]])) = \infty$ and $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s])^{[a]}) = n + \texttt{rln}(R[S[s]]) = n + \infty = \infty$

Now assume that S is not strict. Let R be a reduction context. By (lll)-transformations we have $R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s^{[a]}]] \approx \texttt{letrec}\ a := n\ \texttt{in}\ R[S[s^{[a]}]$.

If $R[S[\cdot]]$ is strict, then we have $\texttt{letrec}\ a := n\ \texttt{in}\ R[S[s^{[a]}]] \approx \texttt{letrec}\ a := n\ \texttt{in}\ R[S[s]]^{[a]}$ (since $R[S[\cdot]]$ is a strict surface context) and $\texttt{letrec}\ a := n\ \texttt{in}\ R[S[s]]^{[a]} \approx \texttt{letrec}\ a := n\ \texttt{in}\ R[S[s]^{[a]}]$ (since R is a strict surface context).

By (lll)-transformations we have $\texttt{letrec}\ a := n\ \texttt{in}\ R[S[s]^{[a]}] \approx R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}]$. thus $\texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s^{[a]}]]) = \texttt{rln}(R[\texttt{letrec}\ a := n\ \texttt{in}\ S[s]^{[a]}])$.

If $R[S[\cdot]]$ is non-strict, then $\mathtt{rln}(R[S[r]]) = m_R$ for any r and where m_R only depends on the context $R[S]$. Then $\mathtt{rln}(R[\mathtt{letrec}\ a := n\ \mathtt{in}\ S[s^{[a]}]]) = \mathtt{rln}(\mathtt{letrec}\ a := n\ \mathtt{in}\ R[S[s^{[a]}]]]) = m_R$, since $\mathtt{rln}$-length of the normal order reduction for $R[S[r]]$ is the same for $\mathtt{letrec}\ a := n\ \mathtt{in}\ R[S[r]]$, since only (lll)-reduction may be added. We also have $\mathtt{rln}(R[\mathtt{letrec}\ a := n\ \mathtt{in}\ S[s]^{[a]}]) = n + \mathtt{rln}(R[S[s]]) = n + m_R$ by Corollary B.5.

Thus in any case $\mathtt{rln}(R[\mathtt{letrec}\ a := n\ \mathtt{in}\ S[s^{[a]}]]) \leq \mathtt{rln}(R[\mathtt{letrec}\ a := n\ \mathtt{in}\ S[s]^{[a]}])$ and the expressions are contextually equivalent and thus the context lemma for improvement shows the claim.

Corollary B.12. *For all surface contexts S_1, S_2: $S_1[\mathtt{letrec}\ a := n\ \mathtt{in}\ S_2[s^{[a]}]] \preceq \mathtt{letrec}\ a := n\ \mathtt{in} S_1[S_2[s]]^{[a]}$ and if $S_1[S_2]$ is strict, also $S_1[\mathtt{letrec}\ a := n\ \mathtt{in}\ S_2[s^{[a]}]] \approx \mathtt{letrec}\ a := n\ \mathtt{in} S_1[S_2[s]]^{[a]}$.*

Proof. This follows from Propositions B.10 and B.11.

Proposition B.13. *1.* $\mathtt{letrec}\ a := n, b := m\ \mathtt{in}\ (s^{[a]})^{[b]} \approx \mathtt{letrec}\ a := n, b := m\ \mathtt{in}\ (s^{[b]})^{[a]}$
2. $\mathtt{letrec}\ a := n\ \mathtt{in}\ (s^{[a]})^{[a]} \approx \mathtt{letrec}\ a := n\ \mathtt{in}\ (s^{[a]})$
3. $(t^{p_1})^{p_2} \approx t^{p_1 \oplus p_2}$.

Proof. 1. Let R be a reduction context. Then $R[\mathtt{letrec}\ a := n, b := m\ \mathtt{in}\ (s^{[a]})^{[b]}] \approx R[\mathtt{letrec}\ b := m\ \mathtt{in}\ (\mathtt{letrec}\ a := n\ \mathtt{in}\ (s^{[a]})^{[b]}])$, since $(llet) \subseteq \approx$. Applying Corollary B.5 two times shows $\mathtt{rln}(R[\mathtt{letrec}\ b := m\ \mathtt{in}\ (\mathtt{letrec}\ a := n\ \mathtt{in}\ (s^{[a]})^{[b]}])) = m + \mathtt{rln}(R[\mathtt{letrec}\ a := n\ \mathtt{in}\ (s'^{[a]})]) = m + n\mathtt{rln}(R[s''])$ where s'' is s where all label $^{[a]}$ and $^{[b]}$ are removed. Completely analogously it can be shown that $\mathtt{rln}(R[\mathtt{letrec}\ a := n, b := m\ \mathtt{in}\ (s^{[b]})^{[a]}]) = n + m + \mathtt{rln}(R[s''])$. Clearly, $\mathtt{letrec}\ a := n, b := m\ \mathtt{in}\ (s^{[a]})^{[b]} \sim_c \mathtt{letrec}\ a := n, b := m\ \mathtt{in}\ (s^{[b]})^{[a]}$ and thus the context lemma for improvement shows the claim.

2. Corollary B.5 shows that for all reduction contexts R the equation $\mathtt{rln}(R[\mathtt{letrec}\ a := n\ \mathtt{in}\ (s^{[a]})^{[a]}]) = n + \mathtt{rln}(R[s']) = \mathtt{rln} R[\mathtt{letrec}\ a := n\ \mathtt{in}\ (s^{[a]})]$ holds, where s' is s where all $^{[a]}$-labels are removed. The expressions are also contextually equivalent and thus the context lemma for improvement shows the claim.
3. This follows from the previous parts and from Proposition B.6.

Proposition B.14. *Let $S[\cdot, \ldots, \cdot]$ be a multi-context where all holes are in surface position. Then $\mathtt{letrec}\ a := n\ \mathtt{in}\ S[s_1^{[a]}, \ldots, s_n^{[a]}] \preceq \mathtt{letrec}\ a := n\ \mathtt{in}\ S[s_1, \ldots, s_n]^{[a]}$. If some hole $\cdot_i$ with $i \in \{1, \ldots, n\}$ is in strict position in $S[\ldots, .]$, then $\mathtt{letrec}\ a := n\ \mathtt{in}\ S[s_1^{[a]}, \ldots, s_n^{[a]}] \approx \mathtt{letrec}\ a := n\ \mathtt{in}\ S[s_1, \ldots, s_n]^{[a]}$.*

Proof. This follows by repeated application of Corollary B.12 and Proposition B.13.

Corollary B.15. *Let $S[\cdot, \ldots, \cdot]$ be a multi-context where all holes are in surface position. Let $S[s_1, \ldots, s_n]$ be closed. Then $S[s_1^{[p_1, a \mapsto m]}, \ldots, s_n^{[p_n, a \mapsto m]}] \preceq S[s_1^{p_1}, \ldots, s_n^{p_n}]^{[m]}$.*
If some hole $\cdot_i$ with $i \in \{1, \ldots, n\}$ is in strict position in $S[\ldots, .]$, then $S[s_1^{[p_1, a \mapsto m]}, \ldots, s_n^{[p_n, a \mapsto m]}] \approx S[s_1^{p_1}, \ldots, s_n^{p_n}]^{[m]}$.

Proposition B.16. *The following transformation is correct w.r.t. $\approx$: Replace $(\mathtt{letrec}\ x = s^{[n,p]}, Env\ \mathtt{in}\ t)$ by $\mathtt{letrec}\ x = s[x^{[a \mapsto n,p]}/x],\ Env[x^{[a \mapsto n,p]}/x]\ \mathtt{in}\ t[x^{[a \mapsto n,p]}/x]$, where a is a fresh label and all occurrences of x are in surface position.*

Proof. Let R be a reduction context. If all occurences of x in $R[(\mathtt{letrec}\ x = s^{[n,p]}, Env\ \mathtt{in}\ t)]$ are in non-strict positions, then $\mathtt{rln}(R[(\mathtt{letrec}\ x = s^{[n,p]}, Env\ \mathtt{in}\ t)]) = \mathtt{rln}(R[(\mathtt{letrec}\ x = s, Env\ \mathtt{in}\ t)]) = \mathtt{rln}(R[\mathtt{letrec}\ x = s[x^{[a \mapsto n,p]}/x], Env[x^{[a \mapsto n,p]}/x]\ \mathtt{in}\ t[x^{[a \mapsto n,p]}/x]])$. If there is a strict position of x in $R[(\mathtt{letrec}\ x = s^{[n,p]}, Env\ \mathtt{in}\ t)]$, then $\mathtt{rln}(R[(\mathtt{letrec}\ x = s^{[n,p]}, Env\ \mathtt{in}\ t)]) = \mathtt{letrec}\ x = s[x^{[a \mapsto n,p]}/x], Env[x^{[a \mapsto n,p]}/x]\ \mathtt{in}\ t[x^{[a \mapsto n,p]}/x]$, since the work corresponding to labels in p are evaluated once and also the work n is only evaluated once. The context lemma for improvement thus shows the claim.