Linear Depth Increase of Lambda Terms in Leftmost-Outermost Rewrite Sequences

CLEMENS GRABMAYER

Abstract

Accattoli and Dal Lago have recently proved that the number of steps in a leftmost-outermost β-reduction rewrite sequence to normal form provides an invariant cost model for the Lambda Calculus. They sketch how to implement leftmost-outermost rewrite sequences on a reasonable machine, with a polynomial overhead, by using simulating rewrite sequences in the linear explicit substitution calculus.

I am interested in an implementation that demonstrates this result, but uses graph reduction techniques similar to those that are employed by runtime evaluators of functional programs. As a crucial stepping stone I prove here the following property of leftmost-outermost β-reduction rewrite sequences in the Lambda Calculus: For every λ-term M with depth d it holds that in every step of a leftmost-outermost β-reduction rewrite sequence starting on M the term depth increases by at most d, and hence that the depth of the n-th reduct of M in such a rewrite sequence is bounded by $d \cdot (n+1)$.

Dedicated to Albert Visser on the occasion of his retirement,
with much gratitude for my time in his group in Utrecht,
and with my very best wishes for the future!

1 Introduction

Recently Accattoli and Dal Lago [1, 2] have proved that the number of steps in a leftmost-outermost rewrite sequence to normal form provides an invariant cost model for the Lambda Calculus. That is, there is an implementation I on a reasonable machine (e.g., a Turing machine, or a random access machine) of the partial function that maps a λ-term to its normal form whenever that exists such that I has the following property: there is a bivariate integer polynomial $p(x, y)$ such that if a λ-term M of size m has a leftmost-outermost rewrite sequence of length n to a normal form N, then I obtains a compact representation of N from M in time bounded by $p(n, m)$. Accattoli and Dal Lago first simulate leftmost-outermost rewrite sequences by 'leftmost-outermost useful' rewrite sequences in the linear explicit substitution calculus, using the restriction that only those steps are performed that facilitate the simulation of leftmost-outermost β-reduction steps. Subsequently they show that such rewrite sequences can be implemented on a reasonable machine.

I am interested in obtaining a graph rewriting implementation for leftmost-outermost β-reduction in the Lambda Calculus that demonstrates this result, but that is close in spirit to graph reduction as it is widely used for the implementation of functional programming languages. In particular, my aim is to describe a port graph grammar [7] implementation that is based on TRS (term rewrite system) representations of λ-terms. Such λ-term representations correspond closely to supercombinator systems that are obtained by lambda-lifting, or fully-lazy lambda-lifting, as first described by to Hughes [6].

That such an implementation is feasible by employing subterm sharing is suggested by the following property of (plain, unshared) leftmost-outermost β-reduction rewrite sequences in the Lambda Calculus, which will be shown here. The depth increase in the steps of an arbitrarily long leftmost-outermost rewrite sequence from a λ-term M is uniformly bounded by $|M|$, the depth of M. As a consequence, for the depth of the n-th reduct L_n of a λ-term M in a leftmost-outermost rewrite sequence it holds: $|L_n| \leq |M| \cdot (n+1)$.

In the terminology of [1, 2] this property shows that leftmost-outermost rewrite sequences do not cause 'depth explosion' in λ-terms. General $\rightarrow_\beta$ rewrite sequences do not enjoy this property, as along them the depth can increase exponentially, which is shown by the example below.

EXAMPLE 1.1 ('depth-exploding' family, from Asperti and Lévy [3]). Consider the following λ-terms:

$$M_0 := xx \qquad M_{i+1} := \mathit{two}(\lambda x.M_i)x \qquad\qquad N_0 := M_0 = xx \qquad N_{i+1} := N_i[x := N_i]$$

where $\mathit{two} := \lambda x.\lambda y.x(xy)$ is the Church numeral for 2. By induction on i it can be verified that:

$$|M_i| = \left\{ \begin{array}{rl} 1 & \text{if } i = 0 \\ 3(i+1) & \text{if } i \geq 1 \end{array} \right\} \in O(i) \qquad\qquad M_i \quad \rightarrow_\beta^{4i} \; N_i \qquad\qquad |N_i| = 2^i$$

and that the syntax tree of N_i is the complete binary application tree of depth $|N_i| = 2^i$ with at depth 2^i occurrences of x. The induction step for the statement on the rewrite sequence can be performed as follows:

$$\begin{aligned} M_{i+1} = \mathit{two}(\lambda x.M_i)x \;\; & \rightarrow_\beta^{4i} \mathit{two}(\lambda x.N_i)x = (\lambda x.\lambda y.x(xy))(\lambda x.N_i)x \qquad\qquad \text{(by the ind. hyp.)} \\ & \rightarrow_\beta \; (\lambda y.(\lambda x.N_i)((\lambda x.N_i)y))x \\ & \rightarrow_\beta \; (\lambda x.N_i)((\lambda x.N_i)x) \;\rightarrow_\beta\; (\lambda x.N_i)N_i \;\rightarrow_\beta\; N_i[x := N_i] = N_{i+1} \end{aligned}$$

Note that this $\rightarrow_\beta$ rewrite sequence is not leftmost-outermost, but essentially inside-out. Now let $i \geq 1$. Then for $n = 4i$ and $M := M_i$ we find: $M = M_i = L_0 \rightarrow_\beta^n L_n = N_i$ with $|M| = |M_i| = 3(i+1) \leq 4i = n$ and $|L_n| = |N_i| = |N_i| = 2^i = 2^{n/4}$. Such an exponential depth increase contradicts the depth increase result that we will show for leftmost-outermost rewrite sequences, since, in the situation here, $|L_n| \leq |M| \cdot (n+1)$ would imply that $2^{n/4} \leq n(n+1)$ holds for infinitely many n.

The result on the linear depth increase of λ-terms along leftmost-outermost rewrite sequences will be shown here for TRS-representations of λ-terms, which will be called 'λ-TRSs'. These representations of λ-terms as orthogonal TRSs correspond to systems of 'supercombinators', which are obtained by the lambda-lifting transformation that is widely used in the implementation of functional programming languages [6]. Lambda-lifting transforms higher-order terms with binding such as λ-terms, or indeed functional programs (which can be viewed as generalized λ-terms with case and letrec constructs) into first-order terms, namely systems of combinator definitions that are called supercombinators. λ-TRSs are TRS-versions of systems of supercombinators. They are well-suited for the purpose of showing the linear depth increase result for much of the same reason why supercombinators are so useful for the evaluation of functional programs: after representing the initial term by a finite number of first-order rewrite rules (through lambda-lifting), the evaluation proceeds by repeatedly searching, and then contracting, the next leftmost-outermost redex with respect to one of these rules. This easy form of the evaluation procedure facilitates a straightforward proof of the linear depth increase invariant for steps of leftmost-outermost rewrite sequences.

While the linear depth increase statement will be shown for rewrite sequences in a TRS for simulating leftmost-outermost β-reduction, its transfer to λ-terms via a lifting theorem along lambda-lifting will only be sketched. The lifting and projection statements needed for this part are largely analogous to proofs for the correctness of fully-lazy lambda-lifting as described by Balabonski [4].

The linear depth increase property for leftmost-outermost rewrite sequences contrasts starkly with the fact that 'size explosion' can actually take place. There are λ-terms M_n of size $O(n)$ (linear size in n) such that M_n reduces in n leftmost-outermost β-reduction steps to a term of size $\Omega(2^n)$ (proper exponential size in n). As an example Accattoli and Dal Lago [1, 2] exhibit the family $\{M_n\}_n$ of $O(n)$-sized λ-terms with $M_0 = yxx$, and $M_{n+1} = (\lambda x.M_n)M_0$ for $n > 1$, which reduce in n leftmost-outermost β-reduction steps to the corresponding, $\Omega(2^n)$-sized term of the family $\{N_n\}_n$ with $N_0 = M_0$, and $N_{n+1} = N_n[x := N_0]$.

The property of the linear depth increase along leftmost-outermost rewrite sequences suggests an alternative proof of the result by Accattoli and Dal Lago, now based on a graph rewriting implementation. The crucial idea is to use (directed acyclic) graph representations of terms in λ-TRSs, which can safeguard that the implementation preserves the linear depth increase property, and to employ the power of sharing to avoid size explosion of the graph representations.[1] If additionally the overhead for the search and the simulation of the next redex contraction can be bounded polynomially in the present graph size, then leftmost-outermost rewrite sequences can be simulated efficiently (first by graph rewrite steps, which subsequently can be implemented efficiently on a reasonable machine). In Section 4 we sketch the basic idea for a graph implementation in which subterm sharing guarantees that the size increase of the graph that represents a λ-term is polynomial in the number of simulated leftmost-outermost β-reduction steps.

Notation. By $\mathbb{N} = \{0, 1, 2, \ldots\}$ we denote the natural numbers including 0. For term rewriting systems, terminology and notation from the book [8] is used, shortly summarized here. A signature Σ is a set of function symbols together with an arity function. For signature Σ we denote by $\mathrm{Ter}(\Sigma)$ the set of terms over Σ, and by $\mathrm{Ter}^{\infty}(\Sigma)$ the set of infinite terms over Σ. For a term s, $|s|$ denotes the depth of s, that is, the longest path in the syntax tree of s from the root to a leaf. A term rewriting system (TRS) is a pair $\langle \Sigma, R \rangle$ consisting of a signature Σ, and a set R of rules for terms over Σ (subject to the usual restrictions). For a TRS with rewrite relation $\to$, the many-step (zero, one or more step) rewrite relation is denoted by $\twoheadrightarrow$, and the n step rewrite relation by $\to^n$, for $n \in \mathbb{N}$. Constrasting with terms in a TRS (first-order terms), λ-terms are viewed as α-equivalence classes of pseudo-term representations with names for bound variables. For λ-terms, $\to_\beta$ denotes β-reduction, and $\to_{\mathrm{lo}\beta}$ leftmost-outermost β-reduction.

[1] In the proof by Accattoli and Dal Lago the chosen representations are terms in the linear explicit substitution calculus, and size explosion is avoided by showing that linear size increase holds for 'leftmost-outermost useful' rewrite sequences.

2 Simulation of leftmost-outermost rewrite sequences

We start with the formal definition of first-order representations of λ-terms, called λ-term representations, before describing a TRS for simulating leftmost-outermost β-reduction on λ-term representations.

DEFINITION 2.1 (λ-term representations, denoted λ-terms). Let $\Sigma_\lambda := \{\mathsf{v}_j \mid j \in \mathbb{N}\} \cup \{@\} \cup \{(\lambda\mathsf{v}_j) \mid j \in \mathbb{N}\}$ be the signature that consists of the *variable* symbols v_j, with $j \in \mathbb{N}$, which are constants (nullary function symbols), the binary *application symbol* $@$, and the unary *named abstraction* symbols $(\lambda\mathsf{v}_j)$, for $j \in \mathbb{N}$.

Now by a *λ-term representation* (a *(first-order) representation of a λ-term*) we mean a closed term in Ter(Σ_λ). A λ-term representation s denotes, by reading its symbols in the obvious way, and interpreting occurrences of variable symbols v_j that are not bound, as the variable names x_j, a unique λ-term $[\![s]\!]_\lambda$.

EXAMPLE 2.2. $(\lambda\mathsf{v}_0)(\mathsf{v}_0)$, $(\lambda\mathsf{v}_1)((\lambda\mathsf{v}_2)(\mathsf{v}_1))$, and $(\lambda\mathsf{v}_0)((\lambda\mathsf{v}_1)((\lambda\mathsf{v}_2)(@(@(\mathsf{v}_0,\mathsf{v}_1),@(\mathsf{v}_1,\mathsf{v}_2)))))$ are λ-term representations that denote the λ-terms $I = \lambda x.x$, $K = \lambda xy.x$, and $S = \lambda xyz.xz(yz)$, respectively.

The TRS below is designed to enable the simulation on λ-term representations of the evaluation of λ-terms according to the leftmost-outermost strategy. We formulate it as a motivation for a similar simulation TRS on refined λ-term representations that is introduced later in Definition 2.10, and which will be crucial for obtaining the main result. The idea behind the TRS below is as follows. Applicative terms are uncurried (by steps of rule (search$_2$)) until on the spine of the term a variable or an abstraction is encountered (detected by rules (search$_3$) or (search$_4$)). If an abstraction occurs, and the expression contains an argument for this abstraction, then a step corresponding to a β-contraction is performed (applying rule (contract)), and the procedure continues similarly from there. If there is no argument for such an abstraction, then it is part of a head normal form context, and the evaluation can descend into this abstraction (applying rule (search$_3$)) to proceed in a similar fashion. If a variable occurs on the spine (detected by rule (search$_5$), it and the recently uncurried applications form a head normal form context, and the simulating evaluation can continue (after applying (search$_5$), (search$_6$), and repeatedly (search$_7$)), possibly in parallel, from any immediate subterm of one of the recently uncurried applications. The rules:

$$
\begin{array}{rcll}
\mathit{losim}(x) & \to & \mathit{losim}_0(x) & (\text{search}_1)\\
\mathit{losim}_n(@(x,y),y_1,\ldots,y_n) & \to & \mathit{losim}_{n+1}(x,y,y_1,\ldots,y_n) & (\text{search}_2)\\
\mathit{losim}_0((\lambda\mathsf{v}_j)(x)) & \to & (\lambda\mathsf{v}_j)(\mathit{losim}_0(x)) & (\text{search}_3)\\
\mathit{losim}_{n+1}((\lambda\mathsf{v}_j)(x),y_1,y_2,\ldots,y_{n+1}) & \to & \mathit{losim}_n(\mathit{subst}(x,\mathsf{v}_j,y_1),y_2,\ldots,y_{n+1}) & (\text{contract})\\
\mathit{losim}_0(\mathsf{v}_j) & \to & \mathsf{v}_j & (\text{search}_4)\\
\mathit{losim}_{n+1}(\mathsf{v}_j,y_1,\ldots,y_{n+1}) & \to & \mathit{curry}_{n+1}(\mathsf{v}_j,y_1,\ldots,y_{n+1}) & (\text{search}_5)\\
\mathit{curry}_1(x,y_1) & \to & @(x,\mathit{losim}_0(y_1)) & (\text{search}_6)\\
\mathit{curry}_{n+2}(x,y_1,y_2,\ldots,y_{n+2}) & \to & \mathit{curry}_{n+1}(@(x,\mathit{losim}_0(y_1)),y_2,\ldots,y_{n+2}) & \\
 & & & (\text{search}_7)
\end{array}
$$

have to be extended with appropriate rules for *subst* that implement capture-avoiding substitution, but which are not provided here. By $\to_{\text{subst}}$ we denote the rewrite relation induced by these rules for *subst*. By $\rightarrow_{\text{contract}}$ we mean the rewrite relation induced by the rule scheme (contract), which defines steps that initiate the simulation of a β-reduction step that proceeds with $\rightarrow_{\text{subst}}$ steps for carrying out the substitution. By $\rightarrow_{\text{search}}$ we designate the rewrite relation induced by the rules labeled with 'search', which defines steps that search for the next representation of a leftmost-outermost redex below the current position. Finally $\rightarrow_{\text{losim}}$ denotes the rewrite relation that is induced by the entire TRS.

EXAMPLE 2.3. We consider the λ-term $M = \lambda x.(\lambda y.y)((\lambda z.\lambda w.wz)x)$. Evaluating M with the leftmost-outermost rewrite strategy, symbolized by the rewrite relation $\rightarrow_{\text{lo}}$, gives rise to the rewrite sequence:

$$\lambda x.(\lambda y.y)((\lambda z.\lambda w.wz)x) \rightarrow_{\text{lo}\beta} \lambda x.(\lambda z.\lambda w.wz)x \rightarrow_{\text{lo}\beta} \lambda x.\lambda w.wx \tag{1}$$

The term $s = (\lambda \mathsf{v}_0)(@((\lambda \mathsf{v}_1)(\mathsf{v}_1), @((\lambda \mathsf{v}_2)((\lambda \mathsf{v}_3)(@(\mathsf{v}_3, \mathsf{v}_2))), \mathsf{v}_0)))$ denotes M, that is, $[\![s]\!]_\lambda = M$; other variable names are possible modulo 'α-conversion'. Simulating this leftmost-outermost rewrite sequence by means of the simulation TRS above amounts to the following $\rightarrow_{\text{losim}}$ rewrite sequence starting on *losim*(s):

$$\begin{aligned}
\textit{losim}(s) &\rightarrow_{\text{search}} \textit{losim}_0((\lambda \mathsf{v}_0)(@((\lambda \mathsf{v}_1)(\mathsf{v}_1), @((\lambda \mathsf{v}_2)((\lambda \mathsf{v}_3)(@(\mathsf{v}_3, \mathsf{v}_2))), \mathsf{v}_0))))\\
&\rightarrow_{\text{search}} (\lambda \mathsf{v}_0)(\textit{losim}_0(@((\lambda \mathsf{v}_1)(\mathsf{v}_1), @((\lambda \mathsf{v}_2)((\lambda \mathsf{v}_3)(@(\mathsf{v}_3, \mathsf{v}_2))), \mathsf{v}_0))))\\
&\rightarrow_{\text{search}} (\lambda \mathsf{v}_0)(\textit{losim}_1((\lambda \mathsf{v}_1)(\mathsf{v}_1), @((\lambda \mathsf{v}_2)((\lambda \mathsf{v}_3)(@(\mathsf{v}_3, \mathsf{v}_2))), \mathsf{v}_0)))\\
&\rightarrow_{\text{contract}} (\lambda \mathsf{v}_0)(\textit{losim}_0(\textit{subst}(\mathsf{v}_1, \mathsf{v}_1, @((\lambda \mathsf{v}_2)((\lambda \mathsf{v}_3)(@(\mathsf{v}_3, \mathsf{v}_2))), \mathsf{v}_0))))\\
&\rightarrow_{\text{subst}} (\lambda \mathsf{v}_0)(\textit{losim}_0(@((\lambda \mathsf{v}_2)((\lambda \mathsf{v}_3)(@(\mathsf{v}_3, \mathsf{v}_2))), \mathsf{v}_0)))\\
&\rightarrow_{\text{search}} (\lambda \mathsf{v}_0)(\textit{losim}_1((\lambda \mathsf{v}_2)((\lambda \mathsf{v}_3)(@(\mathsf{v}_3, \mathsf{v}_2))), \mathsf{v}_0))\\
&\rightarrow_{\text{contract}} (\lambda \mathsf{v}_0)(\textit{losim}_0(\textit{subst}((\lambda \mathsf{v}_3)(@(\mathsf{v}_3, \mathsf{v}_2)), \mathsf{v}_2, \mathsf{v}_0)))\\
&\twoheadrightarrow_{\text{subst}} (\lambda \mathsf{v}_0)(\textit{losim}_0((\lambda \mathsf{v}_3)(@(\mathsf{v}_3, \mathsf{v}_0))))\\
&\rightarrow_{\text{search}} (\lambda \mathsf{v}_0)((\lambda \mathsf{v}_3)(\textit{losim}_0(@(\mathsf{v}_3, \mathsf{v}_0))))\\
&\rightarrow_{\text{search}} (\lambda \mathsf{v}_0)((\lambda \mathsf{v}_3)(\textit{losim}_1(\mathsf{v}_3, \mathsf{v}_0)))\\
&\rightarrow_{\text{search}} (\lambda \mathsf{v}_0)((\lambda \mathsf{v}_3)(\textit{curry}_1(\mathsf{v}_3, \mathsf{v}_0)))\\
&\rightarrow_{\text{search}} (\lambda \mathsf{v}_0)((\lambda \mathsf{v}_3)(@(\mathsf{v}_3, \textit{losim}_0(\mathsf{v}_0))))\\
&\rightarrow_{\text{search}} (\lambda \mathsf{v}_0)((\lambda \mathsf{v}_3)(@(\mathsf{v}_3, \mathsf{v}_0)))
\end{aligned}$$

Note that the $\rightarrow_{\text{contract}}$ steps indeed initiate, and the $\rightarrow_{\text{subst}}$ steps complete, the simulation of corresponding β-reduction steps in the $\rightarrow_{\text{lo}}$ rewrite sequence on λ-terms above, while $\rightarrow_{\text{search}}$ steps organize the search for the next (λ-term representation of a) leftmost-outermost β-redex. The $\rightarrow_{\text{lo}\beta}$ rewrite sequence (1) can be viewed as the projection of the $\rightarrow_{\text{losim}}$ rewrite sequence above under an extension of the denotation operation $[\![\cdot]\!]_\lambda$ on λ-term representations yielding λ-terms (which works out substitutions, and interprets uncurried application expressions $\textit{losim}_n(s, t_1, \ldots, t_n)$ appropriately). Along this projection $\rightarrow_{\text{search}}$ and $\rightarrow_{\text{subst}}$ steps vanish, but $\rightarrow_{\text{contract}}$ steps project to $\rightarrow_{\text{lo}\beta}$ steps.

While the TRS above facilitates the faithful representation of leftmost-outermost rewrite sequences on λ-terms (in analogy with Lemma 2.15, see page 134), it does not lend itself well to the purpose of proving the linear depth increase result. In particular, it is not readily clear which invariant for reducts t of a term s in rewrite sequences $\sigma : s \twoheadrightarrow_{\text{losim}} t \to_{\text{losim}} u$ could make it possible to prove that the depth increase in the final step of σ is bounded by a constant d that only depends on the initial term s of the sequence (but not on t). Therefore it is desirable to consider extensions of first-order λ-term representations in which representations of leftmost-outermost β-redexes are built up from contexts that trace back to contexts in the initial term of the rewrite sequence.

λ-TRSs

We now formally define λ-TRSs as orthogonal TRSs that are able to represent λ-terms. The basic idea is that, for a λ-term M, function symbols that are called 'scope symbols' are used to represent abstractions scopes. Hereby the scope of an abstraction $\lambda x.L$ in M includes the abstraction λx and all occurrences of the bound variable x, but may leave room for subterms in L without occurrences of x bound by the abstraction. For example, the λ-term $\lambda x.zxyx$ may be denoted, with the binary scope symbol f that represents the scope of x, as the term $f(z, y)$. (In our formalization of λ-term representations the free variables z and y will be replaced by variable constants, yielding for example the term $f(\mathsf{v}_2, \mathsf{v}_1)$.) Furthermore, scopes are assumed to be strictly nested. Every scope symbol defines a rewrite rule that governs the behavior of the application of the scope to an argument. In the case of the λ-term $\lambda x.zxyx$ this leads to the first-order rewrite rule $@(f(z, y), x) \to @(@(@(z, x), y), x)$.

As mentioned earlier, λ-TRSs are TRS-representations of systems of supercombinators that are obtained by the lambda-lifting transformation. I have been familiarized with these λ-term representations by Vincent van Oostrom who pointed me to the studies of optimal reduction for weak β-reduction (β-reduction outside of abstractions or in 'maximal free' subexpressions) by Blanc, Lévy, and Maranget [5], and encouraged work by Balabonski [4] on characterizations of optimal-sharing implementations for weak β-reduction by term labellings, and on the relation with lambda-lifting.

DEFINITION 2.4 (λ-TRSs). A *λ-TRS* is a pair $\mathcal{L} = \langle \Sigma, R \rangle$, where Σ is a signature containing the binary application symbol @, and the *scope symbols* in $\Sigma^- := \Sigma \setminus \{@\}$, and where $R = \{\rho_f \mid f \in \Sigma^-\}$ consists of the *defining rules* ρ_f for scope symbols $f \in \Sigma^-$ with arity k that are of the form:

$$(\rho_f) \quad @(f(x_1, \ldots, x_k), y) \;\to\; F[x_1, \ldots, x_k, y]$$

with F a $(k+1)$-ary context that is called the *scope context* for f. For scope symbols $f, g \in \Sigma^-$ we say that f *depends on* the scope symbol g, denoted by $f \multimapinv g$, if g occurs in the scope context F for f. We say that $\mathcal{L}$ is *finitely nested* if the converse relation of $\multimapinv$, the *nested-into* relation $\multimap$, is well-founded, or equivalently (using axiom of dependent choice), if there is no infinite chain of the form $f_0 \multimapinv f_1 \multimapinv f_2 \multimapinv \ldots$ on scope symbols $f_0, f_1, f_2, \ldots \in \Sigma^-$.

EXAMPLE 2.5. Let $\mathcal{L} = \langle \Sigma, R \rangle$ be the λ-TRS with $\Sigma^- = \{f/2, g/0, h/0, i/1\}$, and set of rules R as follows:

$$\begin{array}{llll} (\rho_f) & @(f(x_1, x_2), x) \to @(x_1, @(x_2, x)) & (\rho_h) & @(h, x) \to i(x) \\ (\rho_g) & @(g, x) \to x & (\rho_i) & @(i(x_1), x) \to @(x, x_1) \end{array}$$

This finite λ-TRS, which will facilitate to denote the λ-term M in Example 2.3, is also finitely nested, as the depends-on relation consists of merely a single link: $h \multimapinv i$.

In order to explain how λ-TRS terms (Definition 2.4) denote λ-term representations (Definition 2.1), we introduce, for every λ-TRS $\mathcal{L}$, an expansion TRS that makes use of the defining rules for the scope symbols in $\mathcal{L}$. Then 'denoted λ-term representations' are defined as normal forms of terms in the expansion TRS.

DEFINITION 2.6 (expansion TRS for a λ-TRS). Let $\mathcal{L} = \langle \Sigma, R \rangle$ be a λ-TRS. The *expansion TRS* $\mathcal{E}(\mathcal{L}) = \langle \Sigma_{\text{exp}}, R_{\text{exp}} \rangle$ *for* $\mathcal{L}$ has the signature $\Sigma_{\text{exp}} := \Sigma \cup \Sigma_\lambda \cup \Sigma_{\text{expand}}$ with $\Sigma_{\text{expand}} := \{\mathit{expand}_i \mid i \in \mathbb{N}\}$ where expand_i is unary for $i \in \mathbb{N}$, and $\Sigma^- \cap (\Sigma_\lambda \cup \Sigma_{\text{expand}}) = \varnothing$, and its set of rules R_{exp} consists of the rules:

$$\begin{aligned} \mathit{expand}_i(@(x_1, x_2)) &\to @(\mathit{expand}_i(x_1), \mathit{expand}_i(x_2)) \\ \mathit{expand}_i(f(x_1, \ldots, x_k)) &\to (\lambda \mathsf{v}_i)(\mathit{expand}_{i+1}(F[x_1, \ldots, x_k, \mathsf{v}_i])) \qquad \text{(where } F \text{ scope context for } f\text{)} \\ \mathit{expand}_i(\mathsf{v}_j) &\to \mathsf{v}_j \end{aligned}$$

By $\to_{\text{exp}}$ we denote the rewrite relation of $\mathcal{E}(\mathcal{L})$.

Since expansion TRSs are orthogonal TRSs, finite or infinite normal forms are unique. Furthermore they are constructor TRSs, i.e. they have rules whose right-hand sides are guarded by constructors. This can be used to show that all terms in an expansion TRS rewrite to a unique finite or infinite normal form.

DEFINITION 2.7 (λ-term representations denoted by λ-TRS-terms). Let $\mathcal{L} = \langle \Sigma, R \rangle$ be a λ-TRS. For $s \in \text{Ter}(\Sigma)$ we denote by $[\![s]\!]^{\mathcal{L}}$ the finite or infinite $\to_{\text{exp}}$-normal form of the term $\mathit{expand}_0(s)$ in $\mathcal{E}(\mathcal{L})$. If it is a λ-term representation, we say $[\![s]\!]^{\mathcal{L}}$ is the *denoted λ-term representation* of s, and write $[\![s]\!]^{\mathcal{L}}_\lambda$ for the λ-term $[\![[\![s]\!]^{\mathcal{L}}]\!]_\lambda$.

PROPOSITION 2.8. *Let $\mathcal{L}$ be a finitely nested λ-TRS. Then for every closed term s of $\mathcal{L}$, $[\![s]\!]^{\mathcal{L}}$ is a finite closed term over Σ_λ, hence a λ-term representation of the λ-term $[\![s]\!]^{\mathcal{L}}_\lambda$.*

EXAMPLE 2.9. With the λ-TRS $\mathcal{L}$ from Example 2.5 the λ-term M in Example 2.3 can be denoted as the term $f(g, h)$ expands to a λ-term representation of M (the final $\twoheadrightarrow_{\text{exp}}$ step consists of two parallel $\to_{\text{exp}}$ steps):

$$\begin{aligned} \mathit{expand}_0(f(g,h)) &\to_{\text{exp}} (\lambda \mathsf{v}_0)(\mathit{expand}_1(@(g, @(h, \mathsf{v}_0)))) \\ &\to_{\text{exp}} (\lambda \mathsf{v}_0)(@(\mathit{expand}_1(g), \mathit{expand}_1(@(h, \mathsf{v}_0)))) \\ &\to_{\text{exp}} (\lambda \mathsf{v}_0)(@((\lambda \mathsf{v}_1)(\mathit{expand}_2(\mathsf{v}_1)), \mathit{expand}_1(@(h, \mathsf{v}_0)))) \\ &\to_{\text{exp}} (\lambda \mathsf{v}_0)(@((\lambda \mathsf{v}_1)(\mathsf{v}_1), \mathit{expand}_1(@(h, \mathsf{v}_0)))) \\ &\to_{\text{exp}} (\lambda \mathsf{v}_0)(@((\lambda \mathsf{v}_1)(\mathsf{v}_1), @(\mathit{expand}_1(h), \mathit{expand}_1(\mathsf{v}_0)))) \\ &\to_{\text{exp}} (\lambda \mathsf{v}_0)(@((\lambda \mathsf{v}_1)(\mathsf{v}_1), @(\mathit{expand}_1(h), \mathsf{v}_0))) \\ &\to_{\text{exp}} (\lambda \mathsf{v}_0)(@((\lambda \mathsf{v}_1)(\mathsf{v}_1), @((\lambda \mathsf{v}_1)(\mathit{expand}_2(i(\mathsf{v}_1))), \mathsf{v}_0))) \\ &\to_{\text{exp}} (\lambda \mathsf{v}_0)(@((\lambda \mathsf{v}_1)(\mathsf{v}_1), @((\lambda \mathsf{v}_1)((\lambda \mathsf{v}_2)(\mathit{expand}_3(@(\mathsf{v}_2, \mathsf{v}_1)))), \mathsf{v}_0))) \\ &\to_{\text{exp}} (\lambda \mathsf{v}_0)(@((\lambda \mathsf{v}_1)(\mathsf{v}_1), @((\lambda \mathsf{v}_1)((\lambda \mathsf{v}_2)(@(\mathit{expand}_3(\mathsf{v}_2), \mathit{expand}_3(\mathsf{v}_1)))), \mathsf{v}_0))) \\ &\twoheadrightarrow_{\text{exp}} (\lambda \mathsf{v}_0)(@((\lambda \mathsf{v}_1)(\mathsf{v}_1), @((\lambda \mathsf{v}_1)((\lambda \mathsf{v}_2)(@(\mathsf{v}_2, \mathsf{v}_1))), \mathsf{v}_0))) \end{aligned}$$

Hence $[\![f(g,h)]\!]^{\mathcal{L}} = (\lambda \mathsf{v}_0)(@((\lambda \mathsf{v}_1)(\mathsf{v}_1), @((\lambda \mathsf{v}_1)((\lambda \mathsf{v}_2)(@(\mathsf{v}_2, \mathsf{v}_1))), \mathsf{v}_0)))$. This λ-term representation coincides with the term s in Example 2.3 'modulo α-conversion', and $[\![f(g,h)]\!]^{\mathcal{L}}_\lambda = \lambda x.(\lambda y.y)((\lambda z.\lambda w.wz)x) = M$.

Simulation of leftmost-outermost rewrite sequences on λ-TRS-terms

We adapt the TRS for the simulation of leftmost-outermost $\to_\beta$ rewrite sequences on λ-term representations (see page 128) to 'losim-TRSs' that facilitate this simulation on terms of λ-TRSs. For every λ-TRS $\mathcal{L}$, we introduce a 'losim-TRS' with rules that are similar as before but differ for steps involving abstractions. A simulation starts on a term *losim*(s) where s is a closed λ-TRS term. Therefore initially all abstractions are represented by scope symbols. During the simulation, abstraction representations $(\lambda \mathsf{v}_i)$ are produced in stable parts of the term. The final term in the simulation of a leftmost-outermost $\to_\beta$ rewrite sequence on λ-terms will be a λ-term representation (thus with named abstraction symbols, but without scope symbols).

The altered rules concern $\to_{\text{search}}$ steps that descend into an abstraction, and $\to_{\text{contract}}$ steps that simulate the reduction of β-redexes. In both cases before the step the pertaining abstractions are represented by terms with a scope symbol at the root, and then the expansion of this scope symbol as stipulated in the expansion TRS is used. Additional substitution rules are not necessary any more, because the substitution involved in the contraction of a (represented) β-redex can now be carried out by a single first-order rewrite step. This is because such a step includes the transportation of the argument of a redex into the scope context that defines the body of the abstraction. An additional parameter i of the operation symbols $\mathit{losim}_{n,i}$, $\mathit{curry}_{n,i}$ is used to prevent that any two nested named abstractions refer to the same variable name, safeguarding that rewrite sequences denote meaningful reductions on λ-terms.

DEFINITION 2.10 (losim-TRS for λ-TRSs). Let $\mathcal{L} = \langle \Sigma, R \rangle$ be a λ-TRS. The *losim-TRS (leftmost-outermost reduction simulation TRS)* $\mathcal{LO}(\mathcal{L}) = \langle \Sigma_{\text{losim}}, R_{\text{losim}} \rangle$ *for* $\mathcal{L}$ has signature $\Sigma_{\text{losim}} := \Sigma \cup \Sigma_{\text{lored}} \cup \Sigma_\lambda$ with $\Sigma_{\text{lored}} := \{\mathit{losim}\} \cup \{\mathit{losim}_{n,i},\ \mathit{curry}_{n,i} \mid n, i \in \mathbb{N}\}$, a signature of *operation* symbols (for simulating leftmost-outermost reduction) consisting of the unary symbol *losim*, and the symbols $\mathit{losim}_{n,i}$ and $\mathit{curry}_{n,i}$ with arity $n+1$, for $n, i \in \mathbb{N}$; the rule set R_{losim} of $\mathcal{LO}(\mathcal{L})$ consists of the following (schemes of) rules, which are indexed by scope symbols $f \in \Sigma^-$, and where F is the scope context for scope symbol f:

$$
\begin{array}{rcll}
\mathit{losim}(x) & \to & \mathit{losim}_{0,0}(x) & (\text{search})_{\text{init}} \\
\mathit{losim}_{n,i}(@(x,y), y_1, \ldots, y_n) & \to & \mathit{losim}_{n+1,i}(x, y, y_1, \ldots, y_n) & (\text{search})^{@}_{n,i} \\
\mathit{losim}_{0,i}(f(x_1, \ldots, x_k)) & \to & (\lambda \mathsf{v}_i)(\mathit{losim}_{0,i+1}(F[x_1, \ldots, x_k, \mathsf{v}_i])) & \\
 & & & (\text{search})^{f}_{0,i} \\
\mathit{losim}_{n+1,i}(f(x_1, \ldots, x_k), y_1, y_2, \ldots, y_{n+1}) & \to & \mathit{losim}_{n,i}(F[x_1, \ldots, x_k, y_1], y_2, \ldots, y_{n+1}) & \\
 & & & (\text{contract})^{f}_{n+1,i} \\
\mathit{losim}_{0,i}(\mathsf{v}_j) & \to & \mathsf{v}_j & (\text{search})^{\text{var}}_{0,i} \\
\mathit{losim}_{n+1,i}(\mathsf{v}_j, y_1, \ldots, y_{n+1}) & \to & \mathit{curry}_{n+1,i}(\mathsf{v}_j, y_1, \ldots, y_{n+1}) & (\text{search})^{\text{var}}_{n+1,i} \\
\mathit{curry}_{1,i}(x, y_1) & \to & @(x, \mathit{losim}_{0,i}(y_1)) & (\text{search})_{1,i} \\
\mathit{curry}_{n+2,i}(x, y_1, y_2, \ldots, y_{n+2}) & \to & \mathit{curry}_{n+1,i}(@(x, \mathit{losim}_{0,i}(y_1)), y_2, \ldots, y_{n+2}) & \\
 & & & (\text{search})_{n+2,i}
\end{array}
$$

By $\to_{\text{losim}}$ we denote the rewrite relation of $\mathcal{LO}(\mathcal{L})$. By $\to_{\text{contract}}$ we denote the rewrite relation induced by the rule scheme $(\text{contract})^f$ where $f \in \Sigma^-$ varies among scope symbols of $\mathcal{L}$. By $\to_{\text{search}}$ we denote the rewrite relation induced by the other rules of $\mathcal{LO}(\mathcal{L})$.

EXAMPLE 2.11. For the λ-TRS $\mathcal{L}$ in Example 2.5, we reduce the term $f(g,h)$, which denotes the λ-term M in Example 2.3, in the losim-TRS $\mathcal{LO}(\mathcal{L})$ for $\mathcal{L}$:

$$\begin{aligned}
\mathit{losim}(f(g,h)) &\to_{\text{search}} \mathit{losim}_{0,0}(f(g,h)) \\
&\to_{\text{search}} (\lambda \mathsf{v}_0)(\mathit{losim}_{0,1}(@(g,@(h,\mathsf{v}_0)))) \\
&\to_{\text{search}} (\lambda \mathsf{v}_0)(\mathit{losim}_{1,1}(g,@(h,\mathsf{v}_0))) \\
&\to_{\text{contract}} (\lambda \mathsf{v}_0)(\mathit{losim}_{0,1}(@(h,\mathsf{v}_0))) \\
&\to_{\text{search}} (\lambda \mathsf{v}_0)(\mathit{losim}_{1,1}(h,\mathsf{v}_0)) \\
&\to_{\text{contract}} (\lambda \mathsf{v}_0)(\mathit{losim}_{0,1}(i(\mathsf{v}_0))) \\
&\to_{\text{search}} (\lambda \mathsf{v}_0)((\lambda \mathsf{v}_1)(\mathit{losim}_{0,1}(@(\mathsf{v}_1,\mathsf{v}_0)))) \\
&\to_{\text{search}} (\lambda \mathsf{v}_0)((\lambda \mathsf{v}_1)(\mathit{losim}_{1,2}(\mathsf{v}_1,\mathsf{v}_0))) \\
&\to_{\text{search}} (\lambda \mathsf{v}_0)((\lambda \mathsf{v}_1)(\mathit{curry}_{1,2}(\mathsf{v}_1,\mathsf{v}_0))) \\
&\to_{\text{search}} (\lambda \mathsf{v}_0)((\lambda \mathsf{v}_1)(@(\mathsf{v}_1,\mathit{losim}_{0,2}(\mathsf{v}_0)))) \\
&\to_{\text{search}} (\lambda \mathsf{v}_0)((\lambda \mathsf{v}_1)(@(\mathsf{v}_1,\mathsf{v}_0)))
\end{aligned}$$

obtaining an 'α-equivalent' version of the λ-term representation at the end of the simulated leftmost-outermost reduction on λ-term representations in Example 2.3.

In order to define how terms in the losim-TRS denote λ-term representations we have to extend the expansion TRS from Definition 2.6 with rules that deal with operation and named-abstraction symbols.

DEFINITION 2.12 (expansion TRS for losim-TRS-terms). Let $\mathcal{L} = \langle \Sigma, R\rangle$ be a λ-TRS. The *expansion TRS* $\mathcal{E}_{\text{losim}}(\mathcal{L}) = \langle \Sigma_{\text{losim}} \cup \Sigma_{\text{expand}}, R_{\text{exp}} \cup R_{\text{exp}'}\rangle$ *for losim-TRS-terms* has as its signature the union of the signature Σ_{losim} of $\mathcal{LO}(\mathcal{L})$ and the signature Σ_{expand} of $\mathcal{E}(\mathcal{L})$, and as rules the rules R_{exp} of $\mathcal{E}(\mathcal{L})$ together with the set of rules $R_{\text{exp}'}$ that consists of:

$$\begin{aligned}
\mathit{expand}_i((\lambda \mathsf{v}_j)(x)) &\to (\lambda \mathsf{v}_j)(\mathit{expand}_{\max\{i,j\}+1}(x)) \\
\mathit{expand}_i(\mathit{losim}(x)) &\to \mathit{expand}_i(x) \\
\mathit{expand}_i(\mathit{losim}_{0,j}(x)) &\to \mathit{expand}_{\max\{i,j\}}(x) \\
\left.\begin{aligned}\mathit{expand}_i(\mathit{losim}_{n+1,j}(x,y_1,\ldots y_{n+1})) \\ \mathit{expand}_i(\mathit{curry}_{n+1,j}(x,y_1,\ldots,y_{n+1}))\end{aligned}\right\} &\to @(\cdots @(\mathit{expand}_{i'}(x),\mathit{expand}_{i'}(y_1))\ldots,\mathit{expand}_{i'}(y_{n+1})) \\
&\qquad\qquad \text{(where } i' = \max\{i,j\})
\end{aligned}$$

The rewrite relation of $\mathcal{E}_{\text{losim}}(\mathcal{L})$ will again be denoted by $\to_{\text{exp}}$.

DEFINITION 2.13 (denoted λ-term (representation), extended to losim-TRS-terms). Let $\mathcal{L} = \langle \Sigma, R\rangle$ be a λ-TRS. For terms $s \in \text{Ter}(\Sigma_{\text{losim}})$ in $\mathcal{LO}(\mathcal{L})$, we also denote by $[\![s]\!]^{\mathcal{L}}$ the finite or infinite $\to_{\text{exp}}$-normal form of the term $\mathit{expand}_0(s)$. If it is a λ-term representation, then we say that $[\![s]\!]^{\mathcal{L}}$ is the *denoted λ-term representation* of s, and we again write $[\![s]\!]^{\mathcal{L}}_{\lambda}$ for the λ-term $[\![[\![s]\!]^{\mathcal{L}}]\!]_{\lambda}$.

We now sketch the relationship between rewrite sequences in losim-TRSs with β-reduction rewrite sequences on the denoted λ-terms. For this we formulate statements about the projections of $\to_{\text{losim}}$ steps to steps on λ-terms, and about the lifting of leftmost-outermost β-reduction rewrite sequences to leftmost-outermost rewrite sequences in losim-TRSs. These statements can be illustrated by means of the running example. We do not prove these statements here, as they are closely analogous to the correctness statement for fully-lazy lambda-lifting, and in particular, to the correspondence between weak β-reduction steps on λ-terms and combinator reduction steps on supercombinator representations obtained by fully-lazy lambda-lifting. The latter result was formulated and proved by by Balabonski in [4]. The statements below can be established in a very similar manner.

The first statement concerns the projection of $\to_{\text{losim}}$ steps to $\to_\beta$ or empty steps on λ-terms.

LEMMA 2.14 (Projection of $\to_{\text{losim}}$ steps via $[\![\cdot]\!]^{\mathcal{L}}_\lambda$). *Let $\mathcal{L} = \langle \Sigma, R\rangle$ be a λ-TRS. Let $s \in \text{Ter}(\Sigma_{\text{losim}})$ be a closed term in $\mathcal{LO}(\mathcal{L})$ such that $[\![s]\!]^{\mathcal{L}}_\lambda = M$ for a λ-term M. Then the following statements hold concerning the projection of $\to_{\text{losim}}$ steps via $[\![\cdot]\!]^{\mathcal{L}}_\lambda$ to steps on λ-terms, for all $s_1 \in \text{Ter}(\Sigma_{\text{losim}})$:*

(i) If $s \to_{\text{search}} s_1$, then $[\![s]\!]^{\mathcal{L}}_\lambda = [\![s_1]\!]^{\mathcal{L}}_\lambda$, that is, the projection of a $\to_{\text{search}}$ step via $[\![\cdot]\!]^{\mathcal{L}}_\lambda$ vanishes.

(ii) If $s \to_{\text{contract}} s_1$, then $[\![s]\!]^{\mathcal{L}}_\lambda \to_\beta [\![s_1]\!]^{\mathcal{L}}_\lambda$, that is, every $\to_{\text{contract}}$ step projects via $[\![\cdot]\!]^{\mathcal{L}}_\lambda$ to a $\to_\beta$ step.

(iii) If $s \to_{\text{contract}} s_1$ is actually a leftmost-outermost step, then $[\![s]\!]^{\mathcal{L}}_\lambda \to_{\text{lo}\beta} [\![s_1]\!]^{\mathcal{L}}_\lambda$ holds, that is, leftmost-outermost $\to_{\text{contract}}$ steps project to leftmost-outermost β-reduction steps.

The next lemma states that every leftmost-outermost β-reduction step $M \to_{\text{lo}\beta} M_1$ can be lifted to a leftmost-outermost many-step $s \twoheadrightarrow_{\text{search}} \cdot \to_{\text{contract}} s_1$ in a losim-TRS, provided that s denotes M, and s has been obtained by the simulation of a $\to_{\text{lo}\beta}$ rewrite sequence.

LEMMA 2.15 (Lifting of $\to_{\text{lo}\beta}$ steps to $\twoheadrightarrow_{\text{search}} \cdot \to_{\text{contract}}$ steps w.r.t. $[\![\cdot]\!]^{\mathcal{L}}_\lambda$). *Let $\mathcal{L} = \langle \Sigma, R\rangle$ be a λ-TRS. Let $s \in \text{Ter}(\Sigma)$ be a closed term such that $[\![s]\!]^{\mathcal{L}}_\lambda = M_0$ for a λ-term M_0. Furthermore let $u \in \text{Ter}(\Sigma_{\text{losim}})$ with $[\![u]\!]^{\mathcal{L}}_\lambda = M$ for a λ-term M be the final term of a leftmost-outermost rewrite sequence losim$(s) \twoheadrightarrow_{\text{losim}} u$.*

Then for a $\to_{\text{lo}\beta}$ step $\rho : [\![u]\!]^{\mathcal{L}}_\lambda = M \to_{\text{lo}\beta} M_1$ with λ-term M_1 as target there are terms $u', u_1 \in \text{Ter}(\Sigma_{\text{losim}})$ and a leftmost-outermost $\to_{\text{losim}}$ rewrite sequence $\hat{\rho} : u \twoheadrightarrow_{\text{search}} u' \to_{\text{contract}} u_1$ whose projection via $[\![\cdot]\!]^{\mathcal{L}}_\lambda$ amounts to the step ρ, and hence, $[\![u']\!]^{\mathcal{L}}_\lambda = M$, and $[\![u_1]\!]^{\mathcal{L}}_\lambda = M_1$.

Now by using this lemma in a proof by induction on the length of a $\to_{\text{lo}\beta}$ rewrite sequence the theorem below can be obtained. It justifies the use of losim-TRSs for the simulation of $\to_{\text{lo}\beta}$ rewrite sequences.

THEOREM 2.16 (Lifting of $\to_{\text{lo}\beta}$ to leftmost-outermost $\to_{\text{losim}}$ rewrite sequences). *Let $\mathcal{L} = \langle \Sigma, R\rangle$ be a λ-TRS. Let $s \in \text{Ter}(\Sigma)$ be a closed term with $[\![s]\!]^{\mathcal{L}}_\lambda = M$ for a λ-term M. Then every $\to_{\text{lo}\beta}$ rewrite sequence:*

$$\sigma : M = L_0 \to_{\text{lo}\beta} L_1 \to_{\text{lo}\beta} \cdots \to_{\text{lo}\beta} L_k \ (\to_{\text{lo}\beta} L_{k+1} \to_{\text{lo}\beta} \cdots)$$

of finite or infinite length l lifts via $[\![\cdot]\!]^{\mathcal{L}}_\lambda$ to a leftmost-outermost $\to_{\text{losim}}$ rewrite sequence:

$$\begin{aligned} \hat{\sigma} : \textit{losim}(s) = u_0 \twoheadrightarrow_{\text{search}} \cdot \to_{\text{contract}} u_1 \twoheadrightarrow_{\text{search}} \ \cdots \\ \cdots \ \to_{\text{contract}} u_k \ (\twoheadrightarrow_{\text{search}} \cdot \to_{\text{contract}} u_{k+1} \twoheadrightarrow_{\text{search}} \cdots) \end{aligned}$$

with precisely l $\to_{\text{contract}}$ steps such that furthermore $[\![u_i]\!]^{\mathcal{L}}_\lambda = L_i$ holds for all $i \in \{0, 1, \ldots, l\}$.

3 Linear depth increase

In this section we establish the main result by first deriving bounds for the depth increase of the denoted λ-terms in $\rightarrow_{\text{losim}}$ rewrite sequences in losim-TRSs. In order to reason directly on terms of the losim-TRS, we define the notion of 'λ-term depth' for these terms as the depth of the denoted λ-term representations.

DEFINITION 3.1 (λ-term depth). Let $\mathcal{L} = \langle \Sigma, R \rangle$ be a λ-TRS, and let $\mathcal{LO}(\mathcal{L})$ be the losim-TRS for $\mathcal{L}$. For terms s in $\mathcal{LO}(\mathcal{L})$, the *$\lambda$-term (representation) depth* $|s|_\lambda$ of s is defined as the depth of the λ-term representation denoted by s, giving rise to the function $|\cdot|_\lambda : \text{Ter}(\Sigma_{\text{lo}}) \to \mathbb{N} \cup \{\infty\}$, $s \mapsto |s|_\lambda := |[\![s]\!]^{\mathcal{L}}|$.

Since a λ-term representation s and the λ-term $[\![s]\!]_\lambda$ denoted by it have the same depth, the λ-term depth of a term s that denotes a λ-term M is the depth of M.

PROPOSITION 3.2. *Let $\mathcal{L} = \langle \Sigma, R \rangle$ be a λ-TRS, and let $\mathcal{LO}(\mathcal{L})$ be the losim-TRS for $\mathcal{L}$. If for a term s in $\mathcal{LO}(\mathcal{L})$ it holds that $[\![s]\!]^{\mathcal{L}}_\lambda = M$ for a λ-term M, then $|s|_\lambda = |[\![s]\!]^{\mathcal{L}}| = |[\![s]\!]^{\mathcal{L}}_\lambda| = |M|$.*

The following proposition formulates clauses for the λ-term depth depending on the outermost symbol of a term in a losim-TRS. For finitely nested λ-TRSs, these clauses can be read as an inductive definition. They can be proved in a straightforward manner by making use of the definition via the expansion TRS of the λ-term representations $[\![s]\!]^{\mathcal{L}}$ for terms s of the losim-TRS for a λ-TRS $\mathcal{L}$.

PROPOSITION 3.3. *Let $\mathcal{L} = \langle \Sigma, R \rangle$ be a λ-TRS, and let $\mathcal{LO}(\mathcal{L})$ be the losim-TRS for $\mathcal{L}$. The λ-term depth of terms in $\mathcal{LO}(\mathcal{L})$ satisfies the following clauses, for $i, j, n \in \mathbb{N}$, terms $x, t, t_1, t_2, s_1, \ldots s_k$, and $f \in \Sigma^-$:*

$$
\begin{aligned}
|x|_\lambda &= 0 && (x \textit{ variable}) \\
|@(t_1, t_2)|_\lambda &= 1 + \max\{|t_1|_\lambda, |t_2|_\lambda\} \\
|f(s_1, \ldots, s_k)|_\lambda &= |(\lambda \mathsf{v}_j)(F[s_1, \ldots, s_k, \mathsf{v}_j])|_\lambda && (f \in \Sigma^-,\ F \textit{ as in the rule } \rho_f,\ \mathsf{v}_j \textit{ fresh}) \\
|\mathsf{v}_j|_\lambda &= 0 && (j \in \mathbb{N}) \\
|(\lambda \mathsf{v}_j)(t)|_\lambda &= 1 + |t|_\lambda \\
|\textit{losim}(t)|_\lambda &= |t|_\lambda \\
\left.\begin{array}{r} |\textit{losim}_{n,i}(s, t_1, \ldots, t_n)|_\lambda \\ |\textit{curry}_{n,i}(s, t_1, \ldots, t_n)|_\lambda \end{array}\right\} &= |@(\cdots @(s, t_1) \ldots, t_n)|_\lambda \\
&= \max\{|s|_\lambda + n, |t_1|_\lambda + n, \ldots, |t_n|_\lambda + 1\}
\end{aligned}
$$

PROPOSITION 3.4. Let $\mathcal{L}$ be a finitely nested λ-TRS, and let $\mathcal{LO}(\mathcal{L})$ be the losim-TRS for $\mathcal{L}$. Then every term $t \in \text{Ter}(\Sigma_{\text{lo}})$ has finite λ-term depth $|t|_\lambda \in \mathbb{N}$, and hence the λ-term depth function on terms of $\mathcal{LO}(\mathcal{L})$ is well-defined of type $|\cdot|_\lambda : \text{Ter}(\Sigma_{\text{lo}}) \to \mathbb{N}$.

We extend the concept of λ-term depth also to scope symbols. Let $\mathcal{L}$ be a λ-TRS. The *λ-term depth* $|f|_\lambda$ of a scope symbol $f \in \Sigma^-$ with arity k is defined as $|f(x_1, \ldots, x_k)|_\lambda \in \mathbb{N} \cup \{\infty\}$, the λ-term depth of the term $f(x_1, \ldots, x_k)$. Note that if $\mathcal{L}$ is finitely nested, then Proposition 3.4 entails $|f|_\lambda = |f(x_1, \ldots, x_k)|_\lambda \in \mathbb{N}$. We also define $|\mathcal{L}|_\lambda := \max\{|f|_\lambda \mid f \in \Sigma^-\} \in \mathbb{N} \cup \{\infty\}$, the *maximal λ-term depth* of a scope symbol in $\mathcal{L}$. Note that if, in addition to being finitely nested, $\mathcal{L}$ is also finite, then it holds that $|\mathcal{L}|_\lambda < \infty$.

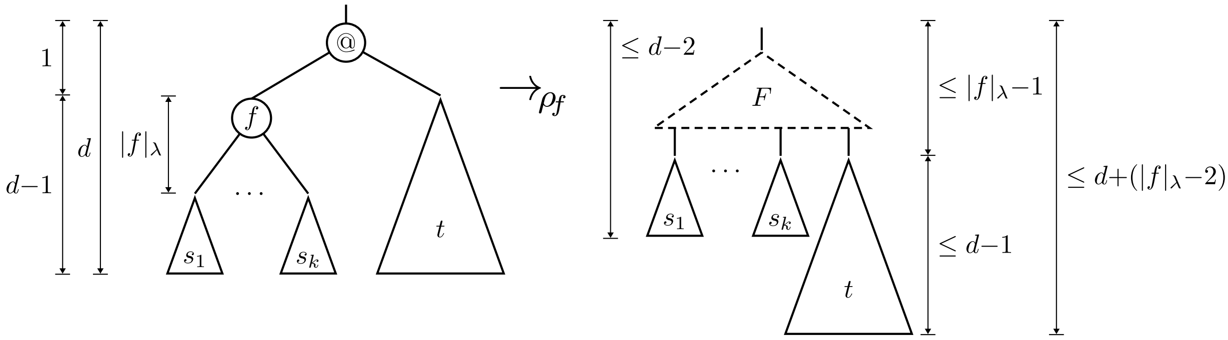

Figure 3.1. Illustration of the λ-term depth increase that is caused by the simulation of a $\rightarrow_\beta$step at the root of a λ-term M on a λ-TRS-term that denotes M: the depth increase in a step $@(f(s_1, \ldots, s_k), t) \rightarrow F[s_1, \ldots, s_k, t]$ according to the defining rule ρ_f for the scope symbol f is at most $|f|_\lambda - 2$. The subterm t could be duplicated in the step and occur several times below F, but only one such occurrence is displayed.

PROPOSITION 3.5. *Let $\mathcal{L} = \langle \Sigma, R \rangle$ be a λ-TRS, and let $\mathcal{LO}(\mathcal{L})$ be the losim-TRS for $\mathcal{L}$. If for a term s in $\mathcal{LO}(\mathcal{L})$ it holds that $[\![s]\!]^{\mathcal{L}}_\lambda = M$ for a λ-term M, then $|\mathcal{L}|_\lambda \leq |M|$.*

For analyzing the depth increase of steps in losim-TRSs, the following two lemmas will be instrumental. They relate the λ-term depth of contexts filled with terms to the λ-term depths of occurring terms.

LEMMA 3.6. *Let $\mathcal{L} = \langle \Sigma, R \rangle$ be a finitely nested λ-TRS. For all unary contexts C over Σ, terms $s, t \in \mathrm{Ter}(\Sigma)$, and $d \in \mathbb{N}$ the following statements hold:*

$$|s|_\lambda \;\leq\; |t|_\lambda + d \;\;\Rightarrow\;\; |C[s]|_\lambda \;\leq\; |C[t]|_\lambda + d\;, \tag{2}$$

$$|s|_\lambda \;=\; |t|_\lambda \;\;\Rightarrow\;\; |C[s]|_\lambda \;=\; |C[t]|_\lambda\;. \tag{3}$$

Proof. Statement (2) can be established by straightforward induction on the structure of the context C, using the clauses concerning the λ-term depth from Proposition 3.3. Statement (3) is an easy consequence. $\dashv$

LEMMA 3.7. *Let $\mathcal{L} = \langle \Sigma, R \rangle$ be a finitely nested λ-TRS. Then for all $(k+1)$-ary contexts C over Σ, where $k \in \mathbb{N}$, and for all terms $s_1, \ldots, s_k, t \in \mathrm{Ter}(\Sigma)$ it holds:*

$$|C[s_1, \ldots, s_k, t]|_\lambda \;\;\leq\;\; \max\left\{|C[s_1, \ldots, s_k, x]|_\lambda, |C[x_1, \ldots, x_{k+1}]|_\lambda + |t|_\lambda\right\}\;.$$

Proof. By a straightforward induction on the structure of the context C. $\dashv$

Now we can formulate, and prove, a crucial lemma (Lemma 3.8). Its central statement is that the depth increase in a $\rightarrow_{\text{contract}}$ step (with respect to a losim-TRS) at the root of a term is bounded by the depth of the scope context of the scope symbol that is involved in the step. See Figure 3.1 for an illustration of the underlying intuition for the analogous case of a step according to the defining rule of a scope symbol. Then we obtain a lemma (Lemma 3.9) concerning the depth increase in general $\rightarrow_{\text{contract}}$ and $\rightarrow_{\text{search}}$ steps.

LEMMA 3.8. *Let $\mathcal{L} = \langle \Sigma, R \rangle$ be a finitely nested λ-TRS. Then for every scope symbol $f \in \Sigma^-$ with arity k and scope context F, and for all terms $s_1, \ldots, s_k, t \in \mathrm{Ter}(\Sigma)$, and all $i \in \mathbb{N}$, it holds:*

(i) $|F[s_1, \ldots, s_k, t]|_\lambda \;\leq\; |@(f(s_1, \ldots, s_k), t)|_\lambda \;+\; |f|_\lambda - 2$.

(ii) $|\textit{losim}_{n,i}(F[s_1, \ldots, s_k, t_1], t_2, \ldots, t_{n+1})|_\lambda \;\leq$
$\leq\; |\textit{losim}_{n+1,i}(f(s_1, \ldots, s_k), t_1, \ldots, t_{n+1})|_\lambda + |f|_\lambda - 2$.

Proof. For all f, F, $s_1, \ldots, s_k, t$, and i as assumed in the statement of the lemma, we find:

$$\left.\begin{aligned} |@(f(s_1, \ldots, s_k), t)|_\lambda - 1 \;&=\; \max\{|f(s_1, \ldots, s_k)|_\lambda,\, |t|_\lambda\} \\ &=\; \max\{|(\lambda \mathsf{v}_j)(F[s_1, \ldots, s_k, \mathsf{v}_j])|_\lambda,\, |t|_\lambda\} \\ &=\; \max\{1 \,+\, |F[s_1, \ldots, s_k, \mathsf{v}_j]|_\lambda,\, |t|_\lambda\} \end{aligned}\right\} \tag{4}$$

$$|F[x_1, \ldots, x_{k+1}]|_\lambda = |(\lambda \mathsf{v}_j)(F[s_1, \ldots, s_k, \mathsf{v}_j])|_\lambda - 1 = |f(x_1, \ldots, x_k)|_\lambda - 1 = |f|_\lambda - 1 \tag{5}$$

by using clauses from Proposition 3.3. By applying this inequality, we obtain the statement in item (i):

$$\begin{aligned} |F[s_1, \ldots, s_k, t]|_\lambda \;&\leq\; \max\{|F[s_1, \ldots, s_k, x_{k+1}]|_\lambda,\, |F[x_1, \ldots, x_k, x_{k+1}]|_\lambda + |t|_\lambda\} \\ &\leq\; \max\{|F[s_1, \ldots, s_k, x_{k+1}]|_\lambda,\, |t|_\lambda\} \,+\, |F[x_1, \ldots, x_k, x_{k+1}]|_\lambda \\ &\leq\; \max\{1 + |F[s_1, \ldots, s_k, x_{k+1}]|_\lambda,\, |t|_\lambda\} \,+\, |F[x_1, \ldots, x_k, x_{k+1}]|_\lambda \\ &=\; |@(f(s_1, \ldots, s_k), t)|_\lambda \,-\, 1 \,+\, |f|_\lambda \,-\, 1 \\ &=\; |@(f(s_1, \ldots, s_k), t)|_\lambda \,+\, (|f|_\lambda \,-\, 2) \end{aligned}$$

where the first step is justified by Lemma 3.7, and the forth step by (4) and (5). For the statement in item (ii) we first note, using Proposition 3.3 again, that:

$$\left.\begin{aligned} &|\textit{losim}_{n+1,i}(f(s_1, \ldots, s_k), t_1, t_2, \ldots, t_{n+1})|_\lambda \\ &\qquad = |@(\ldots @(@(f(s_1, \ldots, s_k), t_1), t_2) \ldots, t_{n+1})|_\lambda \\ &\qquad = |\textit{losim}_{n,i}(@(f(s_1, \ldots, s_k), t_1), t_2, \ldots, t_{n+1})|_\lambda \end{aligned}\right\} \tag{6}$$

holds due to the definition of the λ-term depth via $\to_{\text{exp}}$ steps. With (6) the statement in (ii) follows by using Lemma 3.6 with item (i), and context $C := \textit{losim}_{n,i}(\Box, t_2, \ldots, t_{n+1})$, and constant $d := |f|_\lambda - 2$. ⊣

LEMMA 3.9. *Let $\mathcal{L} = \langle \Sigma, R \rangle$ be a finite, and finitely nested λ-TRS. Then the following statements hold concerning the preservation or the increase of the λ-term depth in steps of the losim-TRS $\mathcal{LO}(\mathcal{L})$ for $\mathcal{L}$ between terms $t_1, t_2 \in \mathrm{Ter}_{\textit{lo-red}}(\Sigma_{\mathrm{lo}})$:*

(i) If $t_1 \to_{\text{contract}} t_2$, then $|t_2|_\lambda \;\leq\; |t_1|_\lambda + (|f|_\lambda - 2)$, where f is the scope symbol involved in the step.

(ii) If $t_1 \to_{\text{search}} t_2$, then $|t_1|_\lambda \;=\; |t_2|_\lambda$.

Proof. We first establish the inequality in item (i). For $\to_{\text{contract}}$ steps at the root, which are of the form:

$$\textit{losim}_{n+1,i}(f(s_1,\ldots,s_k),t_1,\ldots,t_{n+1}) \;\to_{\text{contract}}\; \textit{losim}_{n,i}(F[s_1,\ldots,s_k,t_1],t_2,\ldots,t_{n+1})$$

the desired inequality follows by using $|F[x_1,\ldots,x_{k+1}]|_\lambda = |f(x_1,\ldots,x_{k+1})|_\lambda - 1 \leq D-1$ from Lemma 3.8, (ii). For non-root $\to_{\text{contract}}$ steps this inequality is lifted into a rewriting context by appealing to Lemma 3.6.

We continue with showing item (ii). By means of the clauses of Proposition 3.3 it is straightforward to check that $\to_{\text{search}}$ steps at the root preserve the λ-term depth. This can be extended to all $\to_{\text{search}}$ steps by Lemma 3.6, (3). ⊣

Using this lemma we now can obtain, quite directly, our main result concerning the depth increase of terms in $\to_{\text{losim}}$ rewrite sequences.

THEOREM 3.10. *Let $\mathcal{L} = \langle \Sigma, R\rangle$ be a finite, and finitely nested λ-TRS, and let $D := |\mathcal{L}|_\lambda$. Then for all finite or infinite $\to_{\text{losim}}$ rewrite sequences σ with initial term s and length $l \in \mathbb{N}\cup\{\infty\}$, which can be construed as:*

$$\begin{aligned}\sigma \;:\; s = u_0 \twoheadrightarrow_{\text{search}} u'_0 \to_{\text{contract}} u_1 \twoheadrightarrow_{\text{search}} \cdots \\ \cdots \twoheadrightarrow_{\text{search}} u'_{k-1} \to_{\text{contract}} u_k \;(\twoheadrightarrow_{\text{search}} u'_k \to_{\text{contract}} \cdots)\;,\end{aligned}$$

the following statements hold for all $n \in \mathbb{N}$ with $n \leq l$:

(i) $|u_n|_\lambda = |u'_n|_\lambda$, and $|u_{n+1}|_\lambda \leq |u'_n|_\lambda + (D-2)$ if $n+1 \leq l$, that is more precisely, the λ-term depth remains the same in the $\to_{\text{search}}$ steps, and it increases by at most $D-2$ in the $\to_{\text{contract}}$ steps.

(ii) $|u_n|_\lambda, |u'_n|_\lambda \;\leq\; |s|_\lambda + (D-2)\cdot n$, that is, the increase of the λ-term depth along σ is linear in the number of $\to_{\text{contract}}$ steps performed, with $(D-2)$ as multiplicative constant.

Proof. Statement (i) follows directly from Lemma 3.9, (i) and (ii). Statement (ii) is obtained by adding up the uniform bound D on the λ-term depth increase of each of the n $\to_{\text{contract}}$ steps in the rewrite sequence σ. ⊣

From this theorem we obtain our main theorem, the linear depth increase result for leftmost-outermost β-reduction rewrite sequences, by making use of three auxiliary results we have obtained. Namely first that $\to_{\text{lo}\beta}$ rewrite sequences lift to $\to_{\text{losim}}$ rewrite sequences (Proposition 2.16); second, that λ-term and λ-term representation depths coincide (Proposition 3.2), and third, that the depth of a λ-TRS that represents a λ-term M is bounded by the depth of M (Proposition 3.5).

THEOREM 3.11 (Linear depth increase in $\to_{\text{lo}\beta}$ rewrite sequences). *Let M be a λ-term. Then for every finite or infinite leftmost-outermost rewrite sequence $\sigma : M = L_0 \to_{\text{lo}\beta} L_1 \to_{\text{lo}\beta} \cdots \to_{\text{lo}\beta} L_k \;(\to_{\text{lo}\beta} L_{k+1} \to_{\text{lo}\beta} \cdots)$ from M with length $l \in \mathbb{N}\cup\{\infty\}$ it holds:*

(i) $|L_{n+1}| \leq |L_n| + |M|$ for all $n \in \mathbb{N}$ with $n+1 \leq l$, that is, the depth increase in each step of σ is uniformly bounded by $|M|$.

(ii) $|L_n| \leq |M| + n\cdot|M| = (n+1)\cdot|M|$, and hence $|L_n| - |M| \in O(n)$, for all $n \in \mathbb{N}$ with $n \leq l$, that is, the depth increase along σ to the n-th reduct is linear in n, with $|M|$ as multiplicative constant.

4 Idea for a Graph Rewriting Implementation

The linear depth increase result suggests a directed-acyclic-graph implementation of leftmost-outermost β-reduction that keeps subterms shared as much as possible, particularly in the search for the representation of the next leftmost-outermost redex. The idea is that steps used in the search for the next leftmost-outermost redex do not perform any unsharing, but only use markers to organize the search and keep track of its progress. All such search steps together only increase the size of the graph by at most a constant multiple. The number of search steps necessary for finding the next leftmost-outermost redex is linear in the size of the current graph. Unsharing of the graph only takes place once the next (representation of the) leftmost-outermost redex is found: then the part of the graph between this redex and the root is unshared (copied), and subsequently the (represented) redex is contracted.

Since by Theorem 3.11 the depth of the term L_n after $n \rightarrow_{\text{contract}}$ steps in a $\rightarrow_{\text{losim}}$ rewrite sequence (and hence after n already performed simulated $\rightarrow_{\text{lo}\beta}$ steps) is bounded linearly in n, this also holds for the directed acyclic graph that represents L_n after n simulated $\rightarrow_{\text{lo}\beta}$ steps on the sharing graphs. Hence unsharing work necessary for the simulation of the $(n+1)$-th $\rightarrow_{\text{lo}\beta}$ step is linear in n. This can be used to show that the size increase of the graph after n contractions of (represented) leftmost-outermost redexes is at most quadratic in n. Since consequently the work for searching and contracting the n-th leftmost-outermost redex is also quadratic in n, such an implementation can make it possible to simulate n leftmost-outermost β-reduction steps on sharing graphs in time that is cubic in n.

Acknowledgement. I want to sincerely thank: Vincent van Oostrom, for familiarizing me with TRS-representations of λ-terms, and with the simulation of weak-β reduction by orthogonal TRSs; Dimitri Hendriks, for his many comments on my drafts, and for his questions about it that always helped me; and Jörg Endrullis, for his suggestion about how to typeset Figure 3.1 with TikZ.

BIBLIOGRAPHY

[1] Beniamino Accattoli and Ugo Dal Lago. Beta Reduction is Invariant, Indeed. In *Proceedings of the joint conference CSL-LICS '14*, pages 8:1–8:10, New York, NY, USA, 2014. ACM.

[2] Beniamino Accattoli and Ugo Dal Lago. Beta Reduction is Invariant, Indeed (Long Version). Technical report, arXiv.org, 2014. `http://arxiv.org/abs/1405.3311`.

[3] Andrea Asperti and Jean-Jacques Levy. The Cost of Usage in the λ-Calculus. In *Proceedings of LICS 2013*, LICS '13, pages 293–300, Washington, DC, USA, 2013. IEEE Computer Society.

[4] Thibaut Balabonski. A Unified Approach to Fully Lazy Sharing. In *Proceedings of the Symposium POPL '12*, pages 469–480, New York, NY, USA, 2012. ACM.

[5] Tomasz Blanc, Jean-Jacques Lévy, and Luc Maranget. Sharing in the Weak Lambda-Calculus. In *Processes, Terms and Cycles: Steps on the Road to Infinity. Essays dedicated to Jan Willem Klop*, number 3838 in LNCS. Springer, 2005.

[6] John Hughes. Graph Reduction with Supercombinators. Technical Report PRG28, Oxford University Computing Laboratory, June 1982.

[7] Charles Stewart. Reducibility between Classes of Port Graph Grammar. *Journal of Computer and System Sciences*, 65(2):169 – 223, 2002.

[8] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.