Infinitary Standardisation – Computed

Jeroen Ketema

Imperial College London
j.ketema@imperial.ac.uk

Abstract
An constructive parallel standardisation method for infinitary rewriting is provided.

1 Introduction

In [4] computability is introduced to the world of infinitary rewriting. The setup is simple: Infinite terms are represented by Turing machines that take positions as input and that output function symbols and variables. Strongly convergent reductions are represented by Turing machines that take computable ordinals [5] as input and that output computable terms and rewrite steps.

A shortcoming of the above setup is that the rate of convergence of a strongly convergent reduction is in general not computable, while knowledge of this rate is required in many places. For this reason one ingredient is added to the mix in [4]: Each reduction ρ of length α is equipped with a *modulus of convergence* φ. This is a computable function which, given a depth d and a limit ordinal $\beta \leq \alpha$, yields from which step in ρ onwards we have that all steps up to β occur at depth $> d$ (see Figure 1 for an example inspired by [1]). A strongly convergent reduction equipped with a modulus of convergence is called a *computably strongly convergent reduction*.

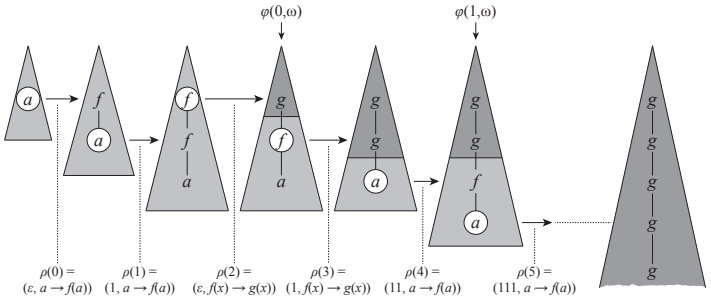

Figure 1: A strongly convergent reduction ρ with modulus φ. Rewrite steps are represented by (position, rule)-pairs.

The main aim of [4] is to provide constructive (or computable) versions of the central theorems from infinitary rewriting. In particular, computable versions of the compression theorem for left-linear systems and the confluence theorem for non-collapsing, orthogonal systems are provided. Missing are computable versions of the standardisation theorems for left-linear systems as developed in [3]. The current work partially bridges this gap by presenting a constructive version of the *parallel standardisation theorem* of [3] (the simplest form of standardisation discussed in [3]). An implementation in Haskell is provided at:

`https://github.com/jeroenk/iTRSsImplemented/tree/compression`

Here, as in [3], a parallel standard reduction is defined as follows:

Definition 1.1. Let $t_0 \twoheadrightarrow t_\alpha$ be strongly convergent with $(p_\beta, l_\beta \to r_\beta)_{\beta<\alpha}$ the sequence of rewrite steps of $t_0 \twoheadrightarrow t_\alpha$. The reduction $t_0 \twoheadrightarrow t_\alpha$ is *parallel standard* iff for every $\beta < \alpha$ either:

a. $p_\beta \parallel p_\kappa$ or $p_\beta \leq p_\kappa$ for all $\beta < \kappa < \alpha$, or

b. $p_\beta = p_\kappa \cdot p'_\beta$ with $p'_\beta \in \{q \in \mathcal{P}os(l_\kappa) \mid l_\kappa(q) \in \Sigma\}$ and $\kappa = \min\{\gamma \in (\beta, \alpha) \mid p_\beta > p_\gamma\}$.

Thus, for each step from a reduction, each preceding step (β) either (a) occurs parallel or at lesser depth, or (b) contributes to the creation of a redex from another preceding step (κ).

Main aim. We aim to prove Theorem 4.10 of [3] in a form that facilitates computability:

Theorem 1.2 (Computable Parallel Standardisation)**.** *There exists a Turing machine M with the following behaviour:*
Input: *A left-linear iTRS R and a computably strongly convergent reduction $s \twoheadrightarrow_R t$ of arbitrary, computable ordinal length α.*
Output: *A parallel standard, computably strongly convergent reduction $s \twoheadrightarrow_R t$ of length at most ω.*

Whence, given a Turing machine that represents a reduction between two terms, M outputs another Turing machine that represents a parallel standard reduction between those terms. As briefly alluded to in the opening paragraph of this paper, all these Turing machines are as one expects: they take finite inputs and compute their finite outputs in finite time.

Preliminaries. Although we omit the definition of computably strongly convergent reductions, as we do not depend on the exact details, a number of other notions from [4] are essential. In particular, a *computable parallel rewrite step* $s \Vdash t$ is a pair $(\pi, l \to r)$, with π a total computable function from natural numbers to sets of positions of s, and $l \to r$ a rewrite rule. The function π is such that for all $d \in \mathbb{N}$ we have that $\pi(d)$ yields *precisely* those positions at depth d at which redexes are contracted in the parallel step. Observe that the usual definition of a parallel rewrite step is obtained by taking $(\bigcup_{d\in\mathbb{N}} \pi(d), l \to r)$. A *computable parallel reduction*, i.e. a finite sequence of computable parallel rewrite steps, is denoted by $s \Vdash^* t$.

Below we employ the following permutation result for parallel rewrite steps, which is Proposition 8.3 of [4]. In the proposition, $l(q)$ denotes the function symbol at position q in l.

Proposition 1.3. *There exists a Turing machine M with the following behaviour:*
Input: *A left-linear iTRS R, a computable parallel rewrite step $s \Vdash_R s'$, and a computable rewrite step $s' \to_R t$, contracting a redex at a position $p \in \mathcal{P}os(s')$ employing a rule $l \to r$, such that no redex from $s \Vdash s'$ occurs at a prefix position of a position in $\{p \cdot q \mid l(q) \in \Sigma\}$.*
Output: *A reduction step $s \to_R t'$, contracting a redex at the position $p \in \mathcal{P}os(s)$ and employing the rewrite rule $l \to r$, and a parallel rewrite step $t' \Vdash t$.*

We also require needed reductions, which depend on origins (the inverse of descendants [2]):

Definition 1.4. Let $s \to t$ be a rewrite step contracting a redex at a position $p \in \mathcal{P}os(s)$ employing a rule $l \to r$. If $q \in \mathcal{P}os(t)$, then the set of *origins of q across $s \to t$*, denoted $(s \to t)^{\backslash}q$, is the subset of positions of s defined as

- $\{q\}$ if $q \parallel p$ or $q < p$,
- $\{p \cdot q' \mid l(q') \in \Sigma\}$ if $q = p \cdot p'$ with $r(p') \in \Sigma$, and
- $\{p \cdot q' \cdot p'' \mid l(q') = r(p')\} \cup \{p \cdot q' \mid l(q') \in \Sigma \wedge r(\epsilon) \in V\}$ if $q = p \cdot p' \cdot p''$ with $r(p') \in V$.

The definition is easily extended to finite reductions and finite sets of positions $Q \subseteq \mathcal{P}os(t)$. Given these extensions, the extension to strongly convergent reductions follows trivially by exploiting the limit behaviour of such reductions (see [4, 3] for details and further explanation).

Let $t_0 \twoheadrightarrow t_\alpha$ be strongly convergent and let $Q \subseteq \mathcal{P}os(t_\alpha)$ be finite and prefix-closed (i.e. $q \in Q$ and $p \leq q$ implies $p \in Q$). A step $t_\beta \to t_{\beta+1}$ of $t_0 \twoheadrightarrow t_\alpha$ contracting a redex at position p_β is called *needed for Q* if $p_\beta \in (t_\beta \twoheadrightarrow t_\alpha)^{\backslash}Q$. A *needed reduction for Q* removes from $t_0 \twoheadrightarrow t_\alpha$ all steps *not* needed for Q; this yields a finite reduction $t_0 \to^* t'_\alpha$ such that $t'_\alpha(q) = t_\alpha(q)$ for all $q \in Q$ (i.e. the function symbols of t'_α and t_α correspond for all positions in Q). We have the following relation between needed reductions and parallel reductions.

Lemma 1.5 (Lemma 8.4 of [4]). *There exists a Turing machine M with the following behaviour:*
Input: *A left-linear iTRS R, a computably strongly convergent reduction $s \twoheadrightarrow_R t$, and $n \in \mathbb{N}$.*
Output: *A parallel reduction $t_n \Vdash^*_R t_{n+1}$ such that $s \to^*_R t_i$ is the needed reduction for the finite, prefix-closed set $Q_i = \{q \in \mathcal{P}os(t) \mid |q| \leq i\}$ with $i \in \{n, n+1\}$.*

It is also possible to obtain the needed steps of a parallel reduction in a constructive manner.

Lemma 1.6 (Lemma 8.5 of [4]). *There exists a Turing machine M with the following behaviour:*
Input: *A left-linear iTRS R, a computable parallel reduction $s \Vdash^*_R t$, and a finite, prefix-closed set $Q \subseteq \mathcal{P}os(t)$.*
Output: *A finite reduction $s \to^*_R t'$ and a parallel reduction $t' \Vdash^*_R t$ such that $t'(q) = t(q)$ for all $q \in Q$ and such that the steps of $s \to^* t'$ are precisely the steps of $s \Vdash^* t$ needed for Q and where the number of parallel steps in $t' \Vdash^* t$ is equal to the number of parallel steps in $s \Vdash^* t$.*

Although not explicit in the above lemma, it is important to observe that if all steps in the parallel reduction $s \Vdash^* t$ occur at depth $> d$, then also all steps in $s \to^* t'$ and $t' \Vdash^* t$ occur at depth $> d$. That this is the case is immediate by prefix-closedness of Q and the definition of needed reductions (see [4] for details). Below, we make heavy use of this observation.

2 Standardisation

To prove standardisation in a constructive manner, the main idea now is to work in a depth-wise fashion starting from least depth and to step by step transform the reduction that is inputted in to one which is parallel standard for the redexes that occur at increasingly greater depths. To this end, we require two intermediate results which help us to separate a given reduction into a parallel standard and a parallel reduction, both satisfying certain depth constrains. We start by considering the case where the given reduction is a finite one.

Lemma 2.1. *There exists a Turing machine M with the following behaviour:*
Input: *A natural number $d \in \mathbb{N}$, a left-linear iTRS R, and a non-empty, finite reduction $s \to^*_R t$ such that all contracted redexes except the final one occur at depth $> d$ and such that the final redex occurs at depth d.*
Output: *A finite, parallel standard reduction $s \to^*_R t'$ and a parallel reduction $t' \Vdash^*_R t$ such that such that all contracted redexes of $s \to^*_R t'$ occur at depth $\geq d$ with the final redex occurring at depth d and such that all contracted redexes of $t' \Vdash^*_R t$ occur at depth $> d$.*

Proof. Iteratively compute reductions of the form $s \Vdash^* s' \to^* t' \Vdash^* t$, with (a) all redexes contracted in $s \Vdash^* s'$ and $t' \Vdash^* t$ occurring at depth $> d$ and with (b) $s' \to^* t'$ parallel standard such that all redexes occurring at positions $q \geq p$ with $|p| = d$ and p the position of the final redex of the original reduction $s \to^* t$.

Initially, (a) take $s \Vdash^* s'$ to be $s \to^* t$ with exception of the final step, (b) take $s' \to^* t'$ to be the final step of $s \to^* t$, and (c) take $t' \Vdash^* t$ to be empty. This assignment obviously satisfies all requirements when defining each parallel step $s_i \Vdash^* s_{i+1}$ from $s \Vdash^* s'$ by taking the step $s_i \to s_{i+1}$ from $s \to^* t$, contracting the redex $(p_i, l_i \to r_i)$, and computing $(\pi_i, l_i \to r_i)$ with $\pi_i(|p_i|) = \{p_i\}$, and $\pi_i(d) = \emptyset$ in case $d \neq |p_i|$.

If $s \Vdash^* s'$ is empty, halt whilst outputting $s' \to^* t'$ and $t' \Vdash^* t$. Otherwise, consider the final parallel step of $s \Vdash^* s'$ and permute this step over the steps from $s' \to^* t'$ in the manner below, assuming that at some point we have a parallel rewrite step $t'_{i-1} \Vdash s'_i$ that we want to permute over a step $s'_i \to s'_{i+1}$ from $s' \to^* t'$ contracting a redex $(p_i, l_i \to r_i)$. Initially, $t'_{i-1} \Vdash s'_i$ is the final step of $s \Vdash^* s'$ and $s'_i \to s'_{i+1}$ is the first step of $s' \to^* t'$.

1. Compute the finite, prefix-closed set $Q_i = \{q \mid q \leq p_i\} \cup \{p_i \cdot q \mid q \in \mathcal{P}os(l) \text{ and } l(q) \in \Sigma\}$, where computability of the set Q_i follows by finiteness of p_i and finiteness of l_i in $l_i \to r_i$.
2. Apply Lemma 1.6 to $t'_{i-1} \Vdash s'_i$ and Q_i to obtain $t'_{i-1} \to^* t_i \Vdash s'_i$ and, hence, $t'_{i-1} \to^* t_i \Vdash^* s'_i \to s'_{i+1}$. As the steps of $t'_{i-1} \to^* t_i$ are parallel, because those of $t'_{i-1} \Vdash s'_i$ are, they can be put in parallel standard order by sorting them based on the length of the positions of the contracted redexes (starting with positions of least depth). For the next step assume $t'_{i-1} \to^* t_i$ is sorted as such.
3. Apply Lemma 1.3 to $t_i \Vdash s'_i \to s'_{i+1}$ — this is possible as no redexes at prefix positions of positions in $\{p \cdot q \mid l(q) \in \Sigma\}$ occur in $t_i \Vdash s'_i$ by construction — to obtain $t_i \to t'_i \Vdash s'_{i+1}$ and, hence, $t'_{i-1} \to^* t_i \to t'_i \Vdash s'_{i+1}$. Observe that $t'_{i-1} \to^* t_i \to t'_i$ is parallel standard, as $t'_{i-1} \to^* t_i$ is and as all redexes contracted in this reduction are needed for Q_i and, hence, the redex contracted in $t_i \to t'_i$.

It follows easily that the concatenation of the reductions $t'_{i-1} \to^* t_i \to t'_i$ from above (in the order obtained) is parallel standard, as $s' \to^* t'$ and each $t'_{i-1} \to^* t_i \to t'_i$ is parallel standard.

Once the final step of $s \Vdash^* s'$ has been permuted over all steps from $s' \to^* t'$ by means of the above permutation procedure, obtaining a parallel step $s'' \Vdash t'$ with $(\pi, l \to r)$, compute the set $\pi(d)$. As the above procedure considers needed redexes only and as all steps in the final step of $s \Vdash^* s'$ occur at depth $> d$, it follows that $\pi(d)$ is either empty or a singleton set consisting of a position p which is also the position of the redex contracted in the final step of $s' \to^* t'$ (note that this is only possible in case the final step of $s' \to^* t'$ is collapsing). If the set is empty define $s'' = t''$ and let $t'' \Vdash t'$ be $s'' \Vdash t'$. Otherwise, let $s'' \to t''$ contract $(p, l \to r)$ and let $t'' \Vdash t'$ be $(\pi', l \to r)$ with $\pi'(d) = \emptyset$, and $\pi'(d') = \pi(d')$ in case $d' \neq d$.

The new reduction $s \Vdash^* s' \to^* t' \Vdash^* t$ is now defined by (a) taking $s \Vdash^* s'$ equal to the previous $s \Vdash^* s'$ whilst removing the final step, (b) taking $s' \to^* t'$ equal to the concatenation of the reductions $t'_{i-1} \to^* t_i \to t'_i$ (in order obtained) suffixed with $s'' \to^= t''$, and (c) taking $t' \Vdash^* t$ equal to the previous $t' \Vdash^* t$ prefixed with $t'' \Vdash t'$. That the new reduction $s \Vdash^* s' \to^* t' \Vdash^* t$ satisfies all requirements is immediately by the use of neededness in permutation procedure.

The computation eventually halts, as the reduction $s \to^* t$, given as input, is finite. □

We next consider separation in the case the given reduction is a parallel one.

Lemma 2.2. *There exists a Turing machine M with the following behaviour:*
Input: *A natural number $d \in \mathbb{N}$, a left-linear iTRS R, and a computable parallel reduction $s \Vdash^*_R t$ such that all contracted redexes occur at depth $\geq d$.*
Output: *A finite, parallel standard reduction $s \to^*_R t'$ and a parallel reduction $t' \Vdash^*_R t$ such that all steps of $s \to^*_R t'$ occur at depth $\geq d$ and the final step occurs at depth d and all contracted redexes of $t' \Vdash^*_R t$ occur at depth $> d$.*

Proof. Iterate over the parallel steps from $s \Vdash^* t$ to find the first step $s_i \Vdash s_{i+1}$ contracting a redex at depth d. For each step the (non-)existence of such a redex at depth d is computable by definition of parallel rewrite steps. Moreover, if no parallel step contracting a redex at depth d exists, we can halt after a finite number of iterations, as $s \Vdash^* t$ has computable, finite length.

If no redex is contracted at depth d, all redexes of $s \Vdash^* t$ must occur at depth $> d$. Define $t' = s$ and halt outputting the empty reduction $s \to^* s$ and the parallel reduction $s \Vdash^* t$.

If a the first redex at depth d is contracted in the step $s_i \Vdash s_{i+1}$ defined by $(\pi_i, l_i \to r_i)$ with $p_i \in \pi_i(d)$ for some p_i, then compute $s_i \Vdash s'_i \Vdash s_{i+1}$ such that (a) $s_i \Vdash s'_i$ contracts $(\pi'_i, l_i \to r_i)$ with $\pi'_i(d) = \{p_i\}$, and $\pi'_i(d') = \emptyset$ in case $d' \neq d$, and such that (b) $s'_i \Vdash s_{i+1}$ contracts $(\pi''_i, l_i \to r_i)$ with $\pi''_i(d) = \pi_i(d) \setminus \{p_i\}$, and $\pi''_i(d') = \pi_i(d')$ in case $d' \neq d$. Observe that the reduction $s_i \Vdash s'_i \Vdash s_{i+1}$ is computable as the parallel step $s_i \Vdash s_{i+1}$ is.

Compute the finite, prefix-closed set $Q_i = \{q \mid q \leq p_i\}$, with p_i as above, and apply Lemma 1.6 to $s \Vdash^* s_i \Vdash s_i'$ and Q_i to obtain $s \to^* t_i \Vdash^* s_i'$. As all redexes in $s \Vdash^* s_i \Vdash s_i'$, except for the one contracted in $s_i \Vdash s_i'$ occur at depth $> d$, and as the redex contracted in $s_i \Vdash s_i'$ is needed for Q_i, it follows by definition of needed redexes that all redexes contracted in $s \to^* t_i$, except for the final one, and all redexes contracted in $t_i \Vdash^* s_i'$ occur at depth $> d$. Moreover, the final redex contracted in $s \to t_i$ occurs at position p_i with $|p_i| = d$.

Apply Lemma 2.1 to $s \to^* t_i$ to obtain $s \to^* t_i' \Vdash^* t_i$ and recursively repeat the computation with $t_i' \Vdash^* t_i \Vdash^* s_i' \Vdash^* s_{i+1} \Vdash^* t$ in place of $s \Vdash^* t$. Suppose this yields a parallel standard reduction $t_i' \to^* t'$ and parallel reduction $t' \Vdash^* t$; halt outputting $s \to^* t_i' \to^* t'$ and $t' \Vdash^* t$. As the final step of the parallel standard reductions $s \to^* t_i'$ and $t_i' \to^* t'$ both contract a redex at depth d, and as no redexes at lesser depth are contracted, it follows that $s \to^* t_i' \to^* t'$ is parallel standard. Finally, as all redexes contracted in $t_i' \Vdash^* t_i \Vdash^* s_i'$ occur at depth $> d$ and as $s_i' \Vdash^* s_{i+1}$ has one redex less at depth d than $s_i \Vdash^* s_{i+1}$, it follows by finiteness of $s \Vdash^* t$ that the performed recursive computation eventually halts. □

Concretely, our constructive standardisation theorem now works as follows: Given a strongly convergent reduction $s \twoheadrightarrow t$, compute for each $n \in \mathbb{N}$ the set of positions $Q_n = \{q \in \mathcal{P}os(t) \mid |q| \leq n\}$ of the final term t of the reduction. Observing that Q_n is prefix-closed, the needed reduction $s \twoheadrightarrow t_n$ for Q_n is computed. Subsequently, $s \twoheadrightarrow t_n$ is filtered based on $s \twoheadrightarrow t_{n-1}$ (Lemma 1.5) and the definition of parallel standardness (Lemma 2.2). Finally, the standard reduction is extended with the finite subset of the steps that remain after filtering.

Write ν_n and ρ for, respectively, the computable functions represented by the Turing machines from Lemmas 1.5 and 2.2. The filtering is then performed by a partial computable function ϕ_n, which is defined as follows and where $n \in \mathbb{N}$ can be taken as parameter:

$$\phi_n(s \twoheadrightarrow t, s_n \Vdash^* t_{n-1}) = \rho(n, s_n \Vdash^* t_{n-1} \cdot \nu_{n-1}(s \twoheadrightarrow t)),$$

where t_{n-1} is the final term of the needed reduction for $s \twoheadrightarrow t$ and Q_{n-1}. The output of ϕ_n with respect to any other input is undefined.

By definition of ρ, the function ϕ_n yields a finite reduction $s_n \to^* s_{n+1}$ and parallel reduction $s_{n+1} \Vdash^* t_n$. The reduction $s_n \to^* s_{n+1}$ is the reduction that will be appended to the standard reduction being constructed. The parallel reduction $s_{n+1} \Vdash^* t_n$ will be used when applying ϕ_{n+1} to obtain $s_{n+1} \to^* s_{n+2}$. We now have:

Proof (Theorem 1.2). Identical to the proof of computable compression in [4], as sketched above, where instead of Lemma 8.5 from [4] we use Lemma 2.2. □

Thus, we obtain a constructive parallel standardisation theorem. An open question is if a similar result can be obtained for the notion of depth leftmost standardisation from [3].

References

[1] P. Bahr. *Modular Implementation of Programming Languages and a Partial-Order Approach to Infinitary Rewriting.* PhD thesis, University of Copenhagen, 2012.

[2] I. Bethke et al. Descendants and origins in term rewriting. *I&C*, 159(1-2):59–124, 2000.

[3] J. Ketema. Reinterpreting compression in infinitary rewriting. In *RTA'12*, volume 15 of *LIPIcs*, pages 209–224, 2012.

[4] J. Ketema and J. G. Simonsen. Computing with infinite terms and infinite reductions, 2012. Submitted.

[5] H. Rogers Jr. *Theory of Recursive Functions and Effective Computability.* The MIT Press, 1987.