Theoretical Computer Science 672 (2017) 36–63

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# On abstract normalisation beyond neededness

Eduardo Bonelli [a,b], Delia Kesner [c], Carlos Lombardi [a,*], Alejandro Ríos [d]

[a] *Departamento de Ciencia y Tecnología, Univ. Nacional de Quilmes, Roque Sáenz Peña 352 (1876), Bernal, Prov. de Buenos Aires, Argentina*
[b] *CONICET, Av. Rivadavia 1917 (1033), C.A. Buenos Aires, Argentina*
[c] *IRIF, CNRS and Univ. Paris-Diderot, Case 7014, 75205 Paris Cedex 13, France*
[d] *Univ. de Buenos Aires, Pabellón I, Ciudad Universitaria (1428), C.A. Buenos Aires, Argentina*

A R T I C L E   I N F O

*Article history:*
Received 31 December 2014
Received in revised form 30 December 2016
Accepted 30 January 2017
Available online 20 February 2017
Communicated by P.-A. Melliès

*Keywords:*
Rewriting
Normalisation
Neededness
Sequentiality
Pattern calculi

A B S T R A C T

We study normalisation of multistep strategies, strategies that reduce a set of redexes at a time, focusing on the notion of *necessary sets*, those which contain at least one redex that cannot be avoided in order to reach a normal form. This is particularly appealing in the setting of non-sequential rewrite systems, in which terms that are not in normal form may not have any *needed* redex. We first prove a normalisation theorem for abstract rewrite systems (ARS), a general rewriting framework encompassing many rewriting systems developed by P-A. Melliès [20]. The theorem states that multistep strategies reducing so called *necessary* and *never-gripping* sets of redexes at a time are normalising in any ARS. Gripping refers to an abstract property reflecting the behaviour of higher-order substitution. We then apply this result to the particular case of **PPC**, a calculus of patterns and to the lambda-calculus with parallel-or.

## 1. Introduction

This paper is about computing normal forms in rewrite systems. Consider the λ-calculus. Let $K$ stand for the term $\lambda x.\lambda y.x$, $I$ for $\lambda x.x$ and $\Omega$ for $(\lambda x.xx)\,(\lambda x.xx)$. Then $s := K\,I\,\Omega$ admits an infinite reduction sequence of $\beta$-steps, namely the one obtained by repeatedly reducing $\Omega$, and hence $s$, to itself. However, it also reduces in two $\beta$-steps to the normal form $I$ by repeatedly reducing the leftmost–outermost redex:

$$K\,I\,\Omega \underset{\beta}{\to} (\lambda y.I)\,\Omega \underset{\beta}{\to} I \qquad (1)$$

The reason this strategy normalises is that the redexes it selects are unavoidable or *needed* in any reduction to normal form. Indeed, leftmost–outermost redexes are needed in λ-calculus [6]. This paper studies normalisation for the broader case of rewriting systems where needed redexes may not exist. It does so by adapting Melliès' abstract rewriting framework [20] to encompass Sekar and Ramakrishnan's notion of *needed sets of redexes* [22]. In doing so, the relatively unfamiliar notion of *gripping*, used only marginally in the work of Melliès, is shown to play a crucial rôle, thus giving it an interest of its own.

**Normalisation in TRS.** Although in the λ-calculus the leftmost–outermost strategy does indeed attain a normal form (if it exists) [12], the same cannot be said for term rewriting systems (TRS). For example, consider the TRS:

[*] Corresponding author.
*E-mail addresses:* eabonelli@gmail.com (E. Bonelli), Delia.Kesner@pps.univ-paris-diderot.fr (D. Kesner), clombardi@unq.edu.ar (C. Lombardi), rios@dc.uba.ar (A. Ríos).

http://dx.doi.org/10.1016/j.tcs.2017.01.025
0304-3975/© 2017 Elsevier B.V. All rights reserved.

$$\begin{aligned} a &\to b \\ c &\to c \\ f(x,b) &\to d \end{aligned}$$

and the term $t := f(c,a)$. The leftmost–outermost strategy selects redex $c$ in $t$ producing an infinite reduction sequence. Yet this term admits a normal form:

$$f(c,a) \to f(c,b) \to d \tag{2}$$

For *left-normal* TRS (those in which the constant and function symbols precede, in the linear term notation, the variable occurrences in the left-hand side of rewrite rules), the leftmost–outermost strategy does indeed normalise [21]; the same applies to left-normal higher-order rewrite systems [19]. Alternatively, one might decide to reduce all *outermost* redexes at once: parallel-outermost reduction is normalising for (almost) orthogonal TRS [21], where an *orthogonal* TRS is one whose rewrite rules are left-linear and non-overlapping, and an *almost orthogonal* TRS has trivial critical pairs at the root, if at all. Parallel-outermost reduction is also normalising for almost orthogonal systems in higher-order rewriting [24].

**Needed redexes.** There is indeed a deep connection between redexes reduced in (1) and (2): they are unavoidable in the sense that in any reduction sequence from $s$ and $t$ to normal form, they (or their residuals) must be reduced at some point. Such redexes are called *needed* and a theory of needed redexes was developed by Huet and Lévy [14] for orthogonal TRS. In [14] it is shown that in these TRS, terms that are not in normal form always have at least one needed redex and that reduction of needed redexes is normalising. They also showed that determining whether a redex is needed or not is undecidable in general; this led them to study restrictions of the class of orthogonal TRS (the *strongly sequential* TRS) in which needed redexes could be identified effectively.

A fundamental limitation of Huet and Lévy's work is the requirement of orthogonality. This requirement does have its reasons though: in non-orthogonal TRS, terms that are not in normal form may not have needed redexes. A paradigmatic example is the "parallel-or" TRS:

$$\begin{aligned} \mathtt{or}(x,\mathtt{tt}) &\to \mathtt{tt} \\ \mathtt{or}(\mathtt{tt},x) &\to \mathtt{tt} \end{aligned}$$

The term $u := \mathtt{or}(\mathtt{or}(\mathtt{tt},\mathtt{tt}),\mathtt{or}(\mathtt{tt},\mathtt{tt}))$ has four redexes: the occurrence of $\mathtt{or}(\mathtt{tt},\mathtt{tt})$ on the left is an instance of the first and second rules of the parallel-or TRS, and the one on the right is also an instance of both of these rules. None of these is needed since one can always obtain a normal form without reducing it. For example, the reduction sequence:

$$\mathtt{or}(\mathtt{or}(\mathtt{tt},\mathtt{tt}),\mathtt{or}(\mathtt{tt},\mathtt{tt})) \to \mathtt{or}(\mathtt{or}(\mathtt{tt},\mathtt{tt}),\mathtt{tt}) \to \mathtt{tt} \tag{3}$$

never reduces any of the two redexes on the left. A similar argument applies to the two redexes on the right in $u$. In fact $u$ seems to suggest that no sensible normalising *strategy*, picking one redex at a time by looking solely at the term, can be constructed. The impossibility to construct an effective needed strategy can even occur in orthogonal TRSs, a paradigmatic example being Gustave's TRS [7].

Any almost orthogonal TRS,[1] such as the parallel-$\mathtt{or}$ example above, does admit a normalising one-step reduction strategy [18,3]. There is a price to pay though, namely that such a strategy has to perform lookahead (in the form of cycle detection within terms of a given size).

Another example of the absence of needed redexes in non-orthogonal rewrite systems are *pattern calculi*. Let $\mathtt{p}$ be a data constructor representing a person including her/his name, gender and marital status. For example, $\mathtt{p\,j\,m\,s}$ represents the person named $\mathtt{j}$ (for "Jack") who is male and single. A function such as $\lambda \mathtt{p}\,x\,\mathtt{m\,s}.x$ returns the name of any person that is male and single. It computes by matching the *pattern* $\mathtt{p}\,x\,\mathtt{m\,s}$ against its argument: reporting an appropriate substitution, if it is successful, or a distinguished constant $\mathfrak{f}$, if it fails (*cf.* Sec. 2). Consider the following term which results from applying the above mentioned function to a person called $\mathtt{a}$ (for "Alice") that is female and divorced (recall from above that $I$ is the identity function $\lambda x.x$):

$$t_0 := (\lambda \mathtt{p}\,x\,\mathtt{m\,s}.x)(\mathtt{p\,a}\,(I\,\mathtt{f})(I\mathtt{d})) \tag{4}$$

This term has two redexes, namely $I\mathtt{f}$ and $I\mathtt{d}$. Note that the term itself is not a redex since the success or failure of the match between pattern and argument cannot be determined. We have two possible reduction sequences to normal form:

$$\begin{aligned} (\lambda \mathtt{p}\,x\,\mathtt{m\,s}.x)(\mathtt{p\,a}\,(I\mathtt{f})(I\mathtt{d})) &\to (\lambda \mathtt{p}\,x\,\mathtt{m\,s}.x)(\mathtt{p\,a}\,(I\mathtt{f})\,\mathtt{d}) \to \mathfrak{f} \\ (\lambda \mathtt{p}\,x\,\mathtt{m\,s}.x)(\mathtt{p\,a}\,(I\mathtt{f})(I\mathtt{d})) &\to (\lambda \mathtt{p}\,x\,\mathtt{m\,s}.x)(\mathtt{p\,a\,f}(I\,\mathtt{d})) \to \mathfrak{f} \end{aligned}$$

The first reduction sequence does not reduce $I\mathtt{f}$; the second does not reduce $I\mathtt{d}$. Therefore the term $t_0$ does not contain any needed redexes.

**Beyond neededness.** This prompts one to consider whether it is possible to obtain normalisation results for possibly overlapping, and more generally *non-sequential* ([14]) rewrite systems. The following avenues have been pursued in this direction:

[1] In fact, any almost orthogonal Combinatory Reduction Systems [19].