## Non-termination using regular languages

## Citation for published version (APA):

Endrullis, J., \& Zantema, H. (2014). Non-termination using regular languages. (arXiv; Vol. 1405.5662 [cs.LO]). s.n.

## Document status and date:

Published: 01/01/2014

## Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

## Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.
Link to publication


## General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25 fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

## Take down policy

If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

# Non-termination using Regular Languages* - International Workshop on Termination - 

Jörg Endrullis ${ }^{1}$ and Hans Zantema ${ }^{2}$<br>1 VU University Amsterdam, The Netherlands<br>2 Eindhoven University of Technology, The Netherlands


#### Abstract

We describe a method for proving non-termination of term rewriting systems that do not admit looping reductions. As certificates of non-termination, we employ regular (tree) automata.


1998 ACM Subject Classification D.1.1, D.3.1, F.4.1, F.4.2, I.1.1, I.1.3

Keywords and phrases non-termination, finite automata, regular languages

## 1 Introduction

We describe a method for proving non-termination of term rewriting systems that do not admit looping reductions, that is, reductions from a term $t$ to a term $C[t \sigma]$ containing a substitution instance of $t$. For this purpose, we employ tree automata as certificates of non-termination. For proving non-termination of a term rewriting system $R$, we search a tree automaton $A$ whose language $\mathcal{L}(A)$ is not empty, weakly closed under rewriting and every term of the language contains a redex occurrence. We have fully automated the search for these certificates employing SAT-solvers.

All automata that we use as example in this paper have been found automatically; this concerns in particular fully automated proofs of non-termination for the following two rewrite systems.

- Example 1. We consider the following string rewriting system:
$z L \rightarrow L z$
$R z \rightarrow z R$
$b L \rightarrow b R$
$R b \rightarrow L z b$

This rewrite system admits no reductions of the form $s \rightarrow^{*} \ell s r$.

- Example 2. We consider the $S$-rule from combinatory logic:

$$
\operatorname{ap}(\operatorname{ap}(\operatorname{ap}(S, x), y), z) \rightarrow \operatorname{ap}(\operatorname{ap}(x, z), a p(y, z))
$$

For the $S$-rule it is known that there are no reductions $t \rightarrow{ }^{*} C[t]$ for ground terms $t$, see [15]. For open terms $t$ the existence of reductions $t \rightarrow{ }^{*} C[t \sigma]$ is open.

It turns out that the method can be fruitfully applied to obtain non-termination proofs of several string rewriting systems that have remained unsolved in the last full run of the termination competition.

[^0]
## Related Work

The paper 11 investigates necessary conditions for the existence of loops. The work [17] employs SAT solvers to find loops, 18 uses forward closures to find loops efficiently, and the wook [16] introduces 'compressed loops' to find certain forms of (possibly very long) loops.

Non-termination beyond loops has been investigated in [14] and [2]; we note that Example 2 cannot be handled by these techniques.

Here we prove non-looping non-termination on regular languages. The converse, local termination on regular languages, has been investigated in 3. Regular (tree) automata have been fruitfully applied to a wide rage of properties of term rewriting systems: for proving termination [10, 8, 12], for infinitary normalization [4, for proving liveness [13], and for analysing reachability and deciding the existance of common reducts [9 5.

## 2 Non-termination and Weakly Closed Languages

- Definition 3. Let $L \subseteq T(\Sigma, \varnothing)$ a language and $R$ a TRS over $\Sigma$. Then $L$ is called:
- closed under rewriting if for every $t \in L$ and $s$ such that $t \rightarrow s$, one has $s \in L$, and
- weakly closed under rewriting if for every $t \in L$ that is not in normal form, there exists $s \in L$ such that $t \rightarrow_{R} s$.

The following theorem describes the basic idea that we employ for proving non-termination.

- Theorem 4. A term rewriting system $R$ over $\Sigma$ is non-terminating if and only if there exists a non-empty language $L \subseteq T(\Sigma, \mathcal{X})$ such that
(i) every $t \in L$ contains a redex (that $i s, t \rightarrow s$ for some term $s$ ), and
(ii) $L$ is weakly closed under rewriting.

A language fulfilling the properties of Theorem 4 is also called a recurrence set, see [1].
To automate this method, we need to restrict to a certain family of languages. In this paper, we consider regular tree languages. To guarantee that the language of a tree automaton is weakly closed under rewriting, we check that the language is not empty and that the automaton is a quasi-model (see Definition 13) for the rewrite system. The latter condition is actually too strict; it implies that the languages is not only weakly closed, but also closed under rewriting. In future, we plan to relieve this restriction.

## 3 Tree Automata

- Definition 5. A (nondeterministic finite) tree automaton $A$ over a signature $\Sigma$ is a tuple $A=\langle Q, \Sigma, F, \delta\rangle$ where
(i) $Q$ is a finite set of states,
(ii) $F \subseteq Q$ is a set of accepting states, and
(iii) $\left\{\delta_{f}\right\}_{f \in \Sigma}$ is a family of transition relations such that for every $f \in \Sigma$ :

$$
\delta_{f} \subseteq Q^{n} \times Q
$$

where $n$ is the arity of $f$.
In examples, we often write the transition relation $\delta_{f}$ as $\rightarrow_{f}$.

- Example 6. The following is a tree automaton for the signature in Example 1 We consider string rewriting systems as term rewriting systems by interpreting all symbols as unary and adding a special constant $\varepsilon$ to denote the end of the word. Let $A_{L R}=\langle Q, \Sigma, F, \rightarrow\rangle$ where $Q=\{0,1,2,3\}, \Sigma=\{b, L, R, 0, \varepsilon\}, F=\{3\}$ and
$\rightarrow_{\varepsilon} 0$
$1 \rightarrow_{z} 1$
$0 \rightarrow_{b} 1$
$1 \rightarrow_{R} 2$
$1 \rightarrow_{L} 2$

$$
2 \rightarrow_{z} 2
$$

$$
2 \rightarrow_{b} 3
$$

The transition relation for $\varepsilon$ can be thought of as defining the initial states (here 0 ) of a word automaton.

- Example 7. The following is a tree automaton for Example 2 Let $A_{S}=\langle Q, \Sigma, F, \rightarrow\rangle$ where $Q=\{0,1,2,3,4\}, \Sigma=\{a p, S\}, F=\{4\}$ and

$$
\begin{array}{rlrl}
\rightarrow_{S} 0 & (0,0) & \rightarrow_{a p} 1 & (1,0) \rightarrow_{a p} 2 \\
& (2,2) \rightarrow_{a p} 3 & (3,3) \rightarrow_{a p} 3 \\
& (0,2) \rightarrow_{a p} 2 & & (2,3) \rightarrow_{a p} 3 \\
& (0,3) \rightarrow_{a p} 2 & &
\end{array}
$$

In Example 12 we show that this automaton accepts the term $\operatorname{SSS}(S S S)(S S S(S S S))$.

- Definition 8. Let $A=\langle Q, \Sigma, F, \delta\rangle$ be a tree automaton over $\Sigma$. For terms $t \in T(\Sigma, \mathcal{X})$ and assignments $\alpha: \mathcal{X} \rightarrow \mathcal{P}(Q)$ we define the interpretation $[t, \alpha]_{A}$ by:

$$
\begin{aligned}
{[x, \alpha]_{A} } & =\alpha(x) \\
{\left[f\left(t_{1}, \ldots, t_{n}\right), \alpha\right]_{A} } & =\left\{q \mid\left(q_{1}, \ldots, q_{n}\right) \in\left[t_{1}, \alpha\right]_{A} \times \ldots \times\left[t_{n}, \alpha\right]_{A},\left\langle\left(q_{1}, \ldots, q_{n}\right), q\right\rangle \in \delta_{f}\right\}
\end{aligned}
$$

Whenever $A$ is clear from the context, we write $[t, \alpha]$ as shorthand for $[t, \alpha]_{A}$. For ground terms $t$, the interpretation is independent of $\alpha$, allowing is to write $[t]_{A}$ or $[t]$ for short.

- Example 9. We use the automaton $A_{S}$ from Example 7 Let $\alpha(x)=\{2\}$, then we have:

$$
\begin{aligned}
& {[S, \alpha]=\{0\} \quad[\operatorname{ap}(S, S), \alpha]=\{1\} \quad[\operatorname{ap}(\operatorname{ap}(S, S), S), \alpha]=\{2\}} \\
& {[\operatorname{ap}(x, x), \alpha]=\{3\} \quad[\operatorname{ap}(\operatorname{ap}(x, x), a p(x, x)), \alpha]=\{3,4\}}
\end{aligned}
$$

- Definition 10. Let $A=\langle Q, \Sigma, F, \delta\rangle$ be a tree automaton over $\Sigma$. The language $\mathcal{L}(A)$ accepted by $A$ is the set $\mathcal{L}(A)=\left\{t \mid t \in T(\Sigma, \varnothing),[t]_{A} \cap F \neq \varnothing\right\}$ of ground terms.

Example 11. The automaton in Example 6 accepts all words of the form $b z^{*}(L \mid R) z^{*} b$, that is, all words that start with $b$, end with $b$, contain one $L$ or $R$ and otherwise only $z$.

- Example 12. We continue Example 9

$$
\begin{aligned}
& {[\operatorname{ap}(\operatorname{ap}(S, S), S)]=\{2\} \quad[\operatorname{ap}(\operatorname{ap}(\operatorname{ap}(S, S), S), \operatorname{ap}(\operatorname{ap}(S, S), S))]=\{3\}} \\
& {[\operatorname{ap}(\operatorname{ap}(\operatorname{ap}(\operatorname{ap}(S, S), S), \operatorname{ap}(\operatorname{ap}(S, S), S)), \operatorname{ap}(\operatorname{ap}(\operatorname{ap}(S, S), S), \operatorname{ap}(\operatorname{ap}(S, S), S)))]=\{3,4\}}
\end{aligned}
$$

Thus $F \cap[S S S(S S S)(S S S(S S S))]=\{4\} \neq \varnothing$ and hence the term is accepted by the automaton.

## 4 Closure under Rewriting

- Definition 13. A tree automaton $A=\langle Q, \Sigma, F, \delta\rangle$ is a quasi-model for a term rewriting system $R$ over $\Sigma$ if $[\ell, \alpha]_{A} \subseteq[r, \alpha]_{A}$ for every $\ell \rightarrow r \in R$ and $\alpha: \mathcal{X} \rightarrow \mathcal{P}(Q)$.

Actually, it suffices to check the property $[\ell, \alpha]_{A} \subseteq[r, \alpha]_{A}$ for assignments $\alpha: \mathcal{X} \rightarrow \mathcal{P}(Q)$ that map variables to singleton sets.

- Lemma 14. A tree automaton $A=\langle Q, \Sigma, F, \delta\rangle$ is a quasi-model for a term rewriting system $R$ over $\Sigma$ iff $[\ell, \alpha]_{A} \subseteq[r, \alpha]_{A}$ for every $\ell \rightarrow r \in R$ and $\alpha: \mathcal{X} \rightarrow\{\{q\} \mid q \in Q\}$.
- Example 15. It is not difficult to check that the automaton $A_{L R}$ from Example 6 is a quasi-model for rewrite system in Example 1
- Example 16. We consider the automaton $A_{S}$ from Example 7 . We write $(a, b, c) \rightarrow d$ if $d \in[\ell, \alpha]$ when $\alpha(x)=\{a\}, \alpha(y)=\{b\}, \alpha(z)=\{c\}$. Then for $[\ell, \alpha]$ we have:
$(0,0,2) \rightarrow 1$
$(2,2,3) \rightarrow 3$
$(2,3,3) \rightarrow 3$
$(3,2,3) \rightarrow 3$
$(3,3,3) \rightarrow 3$
$(0,0,3) \rightarrow 1$
$(2,2,3) \rightarrow 4$
$(2,3,3) \rightarrow 4$
$(3,2,3) \rightarrow 4$
$(3,3,3) \rightarrow 4$

The interpretation $[r, \alpha]$ has all the above and additionally:
$(0,2,2) \rightarrow 3$
$(1,1,0) \rightarrow 3$
$(2,2,2) \rightarrow 3$
$(0,2,3) \rightarrow 3$
$(2,2,2) \rightarrow 4$
$(0,3,3) \rightarrow 3$

As a consequence $A_{S}$ is a quasi-model for the $S$-rule.
The following theorem is immediate:

- Theorem 17. Let $A=\langle Q, \Sigma, F, \delta\rangle$ be a tree automaton and $R$ a term rewriting system over $\Sigma$. If $A$ is a quasi-model for $R$ then the language of $A$ is closed under rewriting.


## 5 Ensuring Redex Occurrences

Next, we want to guarantee that every term in the language $\mathcal{L}(A)$ of an automaton $A$ contains a redex with respect to the term rewriting system $R$. For left-linear systems $R$, this problem can be reduced to deciding the inclusion of regular languages.

Let $R$ be a left-linear term rewriting system. Then the set of ground terms containing a redex is a regular tree language. A deterministic automaton $B$ for this language can be constructed using the overlap-closure of subterms of left-hand sides, see further [6, 7].

- Example 18. The following tree automaton $C=\langle Q, \Sigma, F, \rightarrow\rangle$ accepts the language of ground terms that contain a redex occurrence with respect to the $S$-rule. Here $Q=\{0,1,2,3\}$, $\Sigma=\{a p, S\}, F=\{3\}$ and

$$
\rightarrow_{S} 0 \quad(0, q) \rightarrow_{a p} 1 \quad(1, q) \rightarrow_{a p} 2 \quad(2, q) \rightarrow_{a p} 3 \quad(3, q) \rightarrow_{a p} 3 \quad(q, 3) \rightarrow_{a p} 3
$$

for all $q \in\{0,1,2\}$.
As a consequence the problem of checking whether every term in $\mathcal{L}(A)$ contains a redex boils down to checking that $\mathcal{L}(A) \subseteq \mathcal{L}(B)$. For non-deterministic $A$ and deterministic $B$, this property can be decided by constructing the product automaton and considering the reachable states.

- Definition 19. The product $A \cdot B$ of tree automata $A=\langle Q, \Sigma, F, \delta\rangle$ and $B=\left\langle Q^{\prime}, \Sigma, F^{\prime}, \delta^{\prime}\right\rangle$ is the automaton $C=\left\langle Q \times Q^{\prime}, \Sigma, \varnothing, \gamma\right\rangle$ where for every $f \in \Sigma$ of arity $n$, we define the transition relation $\gamma_{f} \subseteq\left(Q \times Q^{\prime}\right)^{n} \times\left(Q \times Q^{\prime}\right)$ by

$$
\left\langle\left(q_{1}, p_{1}\right), \ldots,\left(q_{n}, p_{n}\right)\right\rangle \gamma\left(q^{\prime}, p^{\prime}\right) \Longleftrightarrow\left\langle q_{1}, \ldots, q_{n}\right\rangle \delta_{f} q^{\prime} \wedge\left\langle p_{1}, \ldots, p_{n}\right\rangle \delta_{f}^{\prime} p^{\prime}
$$

- Definition 20. The set of reachable states of a tree automaton $A=\langle Q, \Sigma, F, \delta\rangle$ is the smallest set $S \subseteq Q$ such that $q \in S$ whenever $\left\langle q_{1}, \ldots, q_{n}\right\rangle \delta_{f} q$ for some $q_{1}, \ldots, q_{n} \in S$ and $f \in \Sigma$ with arity $n$.

The following theorem gives a method for checking $\mathcal{L}(A) \subseteq \mathcal{L}(B)$ without the need for determinising $A$ (only $B$ needs to be deterministic).

- Theorem 21. Let $A=\langle Q, \Sigma, F, \delta\rangle$ and $B=\left\langle Q^{\prime}, \Sigma, F^{\prime}, \delta^{\prime}\right\rangle$ be tree automata such that $B$ is deterministic. Let $S$ be the set of reachable states of the product automaton $A \cdot B$. Then $\mathcal{L}(A) \subseteq \mathcal{L}(B)$ if and only if for all $(q, p) \in S$ it holds that $q \in F \Longrightarrow p \in F^{\prime}$.
- Example 22. The reachable states of product automaton $A_{S} \cdot C$ of the automata $A_{S}$ from Example 7 and $C$ from Example 18 are $(0,0),(1,1),(2,2),(2,1),(3,3),(3,2),(2,3),(4,3)$. The only state $\left(q, q^{\prime}\right)$ such that $q$ is accepting in $A_{S}$ is $(4,3)$ and 3 is an accepting state of $C$. Thus the conditions of Theorem 21 are fulfilled and hence $\mathcal{L}\left(A_{S}\right) \subseteq \mathcal{L}(C)$. Thus every term accepted by $A_{S}$ contains a redex.


## 6 Future Work

We plan to investigate whether the method described in this paper can be fruitfully extended from regular automata to pushdown automata, that is, context-free languages. For this purpose, it is important that it is decidable whether a context-free language is a subset of a regular language (the language of terms containing left-linear redex occurrences). However, it remains to be investigated whether context-free certificates can be found efficiently.

## ——References

1 B. Cook. Priciples of program termination. http://research.microsoft.com/en-us/um/ cambridge/projects/terminator/principles.pdf
2 F. Emmes, T. Enger, and J. Giesl. Proving Non-looping Non-termination Automatically. In International Joint Conference on Automated Reasoning (IJCAR 2012), volume 7364 of Lecture Notes in Computer Science, pages 225-240. Springer, 2012.
3 J. Endrullis, R.C. de Vrijer, and J. Waldmann. Local Termination: Theory and Practice. Logical Methods in Computer Science, 6(3), 2010.
4 J. Endrullis, C. Grabmayer, D. Hendriks, J.W. Klop, and R.C. de Vrijer. Proving Infinitary Normalization. In Postproc. Int. Workshop on Types for Proofs and Programs (TYPES 2008), volume 5497 of Lecture Notes in Computer Science, pages 64-82. Springer, 2009.

5 J. Endrullis, C. Grabmayer, J.W. Klop, and V. van Oostrom. On Equal $\mu$-Terms. Theoretical Computer Science, 412(28):3175-3202, 2011.
6 J. Endrullis and D. Hendriks. Transforming Outermost into Context-Sensitive Rewriting. Logical Methods in Computer Science, 6(2), 2010.
7 J. Endrullis and D. Hendriks. Lazy Productivity via Termination. Theoretical Computer Science, 412(28):3203-3225, 2011.
8 J. Endrullis, D. Hofbauer, and J. Waldmann. Decomposing Terminating Rewrite Relations. In Proc. Workshop on Termination (WST '06), pages 39-43, 2006.
9 B. Felgenhauer and R. Thiemann. Reachability Analysis with State-Compatible Automata. In Proc. Conf. on Language and Automata Theory and Applications (LATA 2014), volume 8370 of Lecture Notes in Computer Science, pages 347-359. Springer, 2014.
10 A. Geser, D. Hofbauer, J. Waldmann, and H. Zantema. On tree automata that certify termination of left-linear term rewriting systems. Inf. Comput., 205(4):512-534, 2007.
11 A. Geser and H. Zantema. Non-looping string rewriting. ITA, 33(3):279-302, 1999.

12 M. Korp and A. Middeldorp. Match-bounds revisited. Inf. Comput., 207(11):1259-1283, 2009.

13 M. Mousazadeh, B. T. Ladani, and H. Zantema. Liveness verification in trss using tree automata and termination analysis. Computing and Informatics, 29(3):407-426, 2010.
14 M. Oppelt. Automatische Erkennung von Ableitungsmustern in nichtterminierenden Wortersetzungssystemen. Technical report, HTWK Leipzig, Germany, 2008. Diploma Thesis.
15 J. Waldmann. The Combinator S. Information Computation, 159(1-2):2-21, 2000.
16 Johannes Waldmann. Compressed loops (draft).
17 H. Zankl and A. Middeldorp. Nontermination of String Rewriting using SAT. 2007.
18 Harald Zankl, Christian Sternagel, Dieter Hofbauer, and Aart Middeldorp. Finding and certifying loops. In Proc. Conf. on Theory and Practice of Computer Science (SOFSEM 2010), volume 5901 of Lecture Notes in Computer Science, pages 755-766. Springer, 2010.


[^0]:    * Published at the International Workshop on Termination 2014.
    © J. Endrullis and H. Zantema
    Leibniz International Proceedings in Informatics
    LIPICS Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

