Symbolic Enumeration of One-Rule String Rewriting Systems

Alfons Geser[1], Johannes Waldmann[2], and Mario Wenzel[3]

1 **Fakultät EIT, HTWK Leipzig, Germany** alfons.geser@htwk-leipzig.de
2 **Fakultät IMN, HTWK Leipzig, Germany** johannes.waldmann@htwk-leipzig.de
3 **Fakultät IMN, HTWK Leipzig, Germany**

Abstract

The purpose of the enumeration of one-rule string rewriting systems is to benchmark methods for proving termination automatically, in particular, to extract interesting cases that merit further attention. We report on a new enumeration approach that represents sets of rewriting systems as the set of models of a binary decision diagram. We relate this to methods and results from the literature, and present preliminary results of experiments.

Digital Object Identifier 10.4230/LIPIcs...

1 Motivation

Rewriting is a model of computation. The termination status of a rewriting system — does it terminate or not? — is a practically relevant piece of information. Small, hard examples of a restricted shape play a crucial role. They allow to uncover, demonstrate and communicate weaknesses of existing approaches and they drive the invention of new methods. One example of a shape restriction is the restriction to unary symbols which means the switching to string rewriting.

The restriction of size and shape may or may not weaken the descriptive power. E.g., termination is decidable for one-rule string rewriting systems (SRSs) $l \to r$ with $l \in 0^*1^*$ [13] whence it is, particularly, not Turing-complete. On the other hand, termination of one-rule term rewriting is undecidable. And there are one-element bases for combinatory logic, which are Turing-complete. The study of restricted systems per se is justified by finding out the thresholds between these classes.

Small string rewriting termination problems have indeed triggered new approaches.

- The first automated termination proof for Zantema's problem [16] $a^2b^2 \to b^3a^3$ obtained from (RFC) matchbounds [6] was later generalized to term rewriting [8].
- The first termination proof (automated or not) for Zantema's other problem $a^2 \to bc, b^2 \to ac, c^2 \to ab$ by matrix interpretations [7] was also generalized later to term rewriting [2] and to complexity analysis [12].

2 Explicit Enumeration

Small hard examples are found by enumerating all small instances, and filtering out those that are

- **easy**, in the sense that they belong to a class that is known to have a decidable termination problem; or
- **redundant**, in the sense that there is a smaller system that is known to have the same termination status. Here, "smaller" is with respect to a well-founded order that is a refinement of the order by size.

For instance, if $|l| \geq |r|$ then $l \to r$ is easy: it terminates iff $l \neq r$. Or, if there is a bijective renaming ϕ of letters such that $\phi(l) \to \phi(r)$ is lexicographically smaller, then $l \to r$ is redundant. A system $l \to r$ is also redundant if there is a bijective renaming ϕ of letters such that $\phi(\tilde{l}) \to \phi(\tilde{r})$ is smaller, where $\tilde{s}$ denotes the reversal of string s. We call $l \to r$ **canonical** if it is not redundant in either of these two ways.

The overhead of an enumeration can be reduced substantially if one avoids some of the systems that are easy or redundant. Kurth [9] enumerates all length-increasing, canonical one-rule SRSs $l \to r$ for $|r| \leq 6$. Geser [4] extends this enumeration to $|r| \leq 9$. Both enumerations follow this approach:

```
foreach System s in canonical_systems { if not (easy (s)) then print (s) }
```

3 Symbolically Representing Sets of Rewriting Systems as BDDs

We present a radically different approach that avoids explicit enumeration: We represent SRSs as models of binary decision diagrams (BDDs [1]). We represent all rules $l \to r$ of a certain shape (fixed length of l and r) and a fixed alphabet as assignments of Boolean variables, using some encoding scheme. We formulate criteria $P_1, P_2, \ldots$ of rewriting systems as Boolean formulas $P'_1, P'_2, \ldots$ compatible with the chosen encoding.

Instead of explicitly enumerating all $l \to r$ and then checking criteria $P_1, P_2, \ldots$ one after another, we compute the BDD representation P' of $P'_1 \wedge P'_2 \wedge \ldots$ and then enumerate the models of P':

```
foreach Assignment a in models (P1 and P2 and ...) { print (decode (a)) }
```

Additional advantages of this approach are:

- we can count the number of models without actually enumerating them,
- we can use any Boolean combination of criteria to investigate relations between them, e.g., implications.

4 Criteria related to Termination of Standard Rewriting

The following criteria are used. These are either obvious or well-known, except for (two-letter) coding.

Redundancy criteria:

- $l \to r$ is not canonical. A canonical rule is lexicographically minimal in the equivalence class of rules w.r.t. renaming or reversal.
  - reversal: $ab \to baa$ is transformed to $ba \to aab$
  - renaming: $ab \to baa$ is transformed by $\{a \mapsto b, b \mapsto a\}$ to $ba \to abb$

  The equivalence class of $ab \to baa$, restricted to alphabet $\{a, b\}$ is $\{ab \to baa, ba \to abb, ba \to aab, ab \to bba\}$. The minimal element w.r.t. the order $rl <_{\text{lex}} r'l'$ is $ba \to aab$.
- $l \to r$ is *bordered*, i.e. both l and r begin and end with the same non-empty string [4]. Example: $abba \to abaaba$ is bordered by a, and the termination problem is reduced to $[bb] \to [b][][b]$, over alphabet $\{[], [b], [bb]\}$.
- two-letter-coding. For example, $bca \to aabc$ is reduced to $[bc]a \to aa[bc]$ via the code $\{a, bc\}$, where $[bc]$ is treated as a single letter.

Ease criteria:

- $l \to r$ deletes a letter: $\Sigma(l) \not\subseteq \Sigma(r)$.

- Kurth's Criterion A: a letter occurs more often in l than in r. This class includes the deleting rules.
- Kurth's Criterion D: l is not a factor of r, and either there are no overlaps between the end of l and the begin of r or there are no overlaps between the end of r and the begin of l. Example: $aba \to aaabb$. The end of $aaabb$ has no overlaps with the begin of aba.
- Loops of length one: l is a factor of r.
- Loops of length two (by analysis of overlaps).
- McNaughton's criterion [11]: there exists an inhibitor $i \in \Sigma(r) \setminus \Sigma(l)$.
- Sénizergues' criterion [13]: l has the shape a^*b^*.
- $l \to r$ is grid [5]: there is a letter c with $|l|_c > 0$ and $|l|_c \geq |r|_c$ Example: $bbab \to abbaaabaa$. This class includes the Criterion A rules.

5 Implementation

Our implementation (`https://gitlab.imn.htwk-leipzig.de/waldmann/srs-count`) uses Haskell and the well-known BDD C-library CUDD [14].

We use the "one-hot" encoding for letters where the i-th variable being true means this letter is the i-th letter of Σ while all other variables for that letter are false. A word is a list of letters and a rule is a pair of words. In total, the encoding of $l \to r$ uses $(|l| + |r|) \cdot |\Sigma|$ propositional variables.

Criteria from Section 4 are expressed with the help of predicates for equality and order on letters, for the prefix relation on words, and so on. A consistency predicate expresses the one-hot property. It is always part of the main conjunction. Other predicates, or their negation, can be included via command line arguments. The most expensive criterion is canonicity w.r.t. reversal and renaming, where the number of BDD operations depends exponentially on the size of the alphabet.

The implementation computes the BDD and enumerates its models and decodes them to SRSs. Termination provers `matchbox` [15] and TTT2 [10] can be called for further filtering.

```
srs-count -n True -R True -a True -i False, -g False -o False
  --results 20 --matchbox no 3 6 9
```

This example call computes the first 20 systems with a left-hand side of size 6, a right-hand side of size 9 and a size-3 alphabet that are canonical by re(n)aming and (R)eversal-and-(R)enaming, use (a)ll 3 letters of the alphabet, do not have an (i)nhibitor, are not a (g)rid-rule and do not have a loop of length (o)ne, while `matchbox` still has a non-termination proof.

Additionally, we allow the enumeration to be split or restricted using patterns (globs) like `-globleft="ab*"`, which would restrict the left-hand side to words of the language $a \cdot b \cdot \Sigma^*$. This replaces Boolean variables by constants, and makes for smaller BDDs. For a complete enumeration, we apply several such patterns to distribute the computation across multiple computers.

6 Results

We confirmed that symbolic and explicit enumeration agree for $|r| \leq 9$. Table 1 shows the numbers obtained by an explicit enumeration, using Geser's original implementation, of all length-increasing, canonical one-rule SRSs ("all"), and of those SRSs that satisfy both $|l| \geq |\Sigma|$ and $|r| \geq |l| + |\Sigma|$ ("restricted"). The number of non-grid, non-inhibitor systems, obtained through filtering, is the same in both cases. Further filtering out 1-loop and Criterion D yields the next column. The final column shows the number after further filtering out 2-loop and bordered ("fast check criteria"). The table illustrates that the explicit generate-and-filter approach quickly becomes prohibitively expensive and less useful since the share of interesting systems becomes smaller as the system size grows.

Using symbolic enumeration, we were able to reproduce the results from the second-to-last column up to $|r| \leq 8$ in less than 10 seconds ($|r| \leq 9$ in 3½ minutes) on a 3.2 GHz processor.

$\|r\|$	all	restricted	non-grid, non-inhibitor	..., non-1-loop, non-crit-D	non-fast-criteria
2	2	1	0	0	0
3	21	2	2	0	0
4	226	20	8	1	0
5	3 929	103	30	7	4
6	96 029	1 699	207	68	45
7	3 151 054	18 345	1 618	540	440
8	130 792 338	396 184	16 594	4 994	4 265
9	6 641 134 837	6 642 933	196 476	49 814	43 535
10	?	173 514 078	2 710 745	562 258	493 855
11	?	4 039 563 892	42 735 641	7 213 316	6 346 721

Table 1 Numbers of length-increasing, canonical one-rule SRSs

In order to obtain fresh hard termination problems, we have enumerated and filtered all one-rule SRS with $|r| \leq 14$ and $|\Sigma|$ no larger than 3, using all stated criteria except criterion D (which was a recent addition to our implementation).

This left about $7.66 \cdot 10^9$ systems, which we have filtered using `matchbox` [15], applying only RFC match bounds for termination, and forward closure enumeration for non-termination, and spending no more than 1 second per problem (on our machines). Enumeration and filtering took 30.000 CPU hours, approximately.

This left 671 systems, on which we ran $\mathsf{T_TT_2}$ [10] and AProVE [3] on starexec, using 300 seconds as a timeout. We obtained 226 systems where termination currently cannot be shown automatically, and which we will submit for TPBD. Four random examples are:

$$aabaaaa \to aaaaaabaab, \qquad babbaabba \to abbaabbabba,$$
$$bababababaa \to aababababababa, \qquad cababababa \to abababcccca.$$

7 Extension to Termination of Cycle Rewriting

Recently, there has been an interest in cycle rewriting [17]. A string rewriting system R over Σ defines a *cycle rewriting relation* $\overset{\circ}{\to}_R$ on Σ^* that is the composition of the standard *conjugacy* relation $uv \equiv vu$ with the standard rewrite relation $\to_R$.

Our approach for symbolically enumerating interesting one-rule rewriting systems is easily applicable for cycle rewriting, and in fact we simply use our existing implementation, and switch off a few criteria. From the list of properties in Section 4, we omit the following because their applicability needs further research: Kurth's criterion D, the grid criterion and Sénizergues' criterion. Note that we can use the inhibitor criterion for reduction: If R has an inhibitor, then cycle termination of R is equivalent to standard termination of R, which is (in that case) decidable.

For cycle termination, there was no previous enumeration. We generated $3.1 \cdot 10^6$ length-increasing systems without the applicable properties with $|r| \leq 9$ and $|\Sigma|$ no larger than 3. The initial generation took 5½ minutes on a 2.1 GHz i3 processor.

These are the 6 smallest one-rule SRSs for which matchbox could not determine the status of cycle termination:

$$\begin{array}{lll} baba \to abaaabab, & ababba \to aabbabab, & abaaba \to aababaab, \\ baba \to abaaaabab, & baabba \to aabbaaabb, & ababbab \to abbababba. \end{array}$$

References

1 Sheldon B. Akers. Binary Decision Diagrams. *IEEE Trans. Computers*, 27(6):509–516, 1978.

2 Jörg Endrullis, Johannes Waldmann, and Hans Zantema. Matrix Interpretations for Proving Termination of Term Rewriting. *J. Autom. Reasoning*, 40(2-3):195–220, 2008.

3 Jürgen Giesl et al. Automated Program Verification Environment. `http://aprove.informatik.rwth-aachen.de/`, 2016.

4 Alfons Geser. *Is Termination Decidable for String Rewriting With Only One Rule.* Habilitationsschrift, Universität Tübingen, 2001.

5 Alfons Geser. Decidability of Termination of Grid String Rewriting Rules. *SIAM J. Comput.*, 31(4):1156–1168, 2002.

6 Alfons Geser, Dieter Hofbauer, and Johannes Waldmann. Match-Bounded String Rewriting Systems. *Appl. Algebra Eng. Commun. Comput.*, 15(3-4):149–171, 2004.

7 Dieter Hofbauer and Johannes Waldmann. Termination of String Rewriting with Matrix Interpretations. In Frank Pfenning, editor, *RTA 2006*, volume 4098 of *LNCS*, pages 328–342. Springer, 2006.

8 Martin Korp and Aart Middeldorp. Match-bounds revisited. *Inf. Comput.*, 207(11):1259–1283, 2009.

9 Winfried Kurth. *Termination und Konfluenz von Semi-Thue-Systemen mit nur einer Regel.* Dissertation, Technische Universität Clausthal, 1990.

10 Harald Zankl Martin Korp, Christian Sternagel and Aart Middeldorp. Tyrolean Termination Tool 2. `http://cl-informatik.uibk.ac.at/software/ttt2/`, 2014.

11 Robert McNaughton. Semi-Thue Systems with an Inhibitor. *J. Autom. Reasoning*, 26:409–431, 1997.

12 Georg Moser, Andreas Schnabl, and Johannes Waldmann. Complexity Analysis of Term Rewriting Based on Matrix and Context Dependent Interpretations. In Ramesh Hariharan, Madhavan Mukund, and V. Vinay, editors, *FSTTCS 2008*, volume 2 of *LIPIcs*, pages 304–315. Schloss Dagstuhl - LZI, 2008.

13 Géraud Sénizergues. On the Termination Problem for One-Rule Semi-Thue System. In Harald Ganzinger, editor, *RTA-96*, volume 1103 of *LNCS*, pages 302–316. Springer, 1996.

14 Fabio Somenzi. CUDD: CU Decision Diagram Package Release 3.0.0. `http://vlsi.colorado.edu/~fabio/CUDD`, 2015.

15 Johannes Waldmann. Pure Matchbox. `https://gitlab.imn.htwk-leipzig.de/waldmann/pure-matchbox`, 2016.

16 Hans Zantema and Alfons Geser. A Complete Characterization of Termination of $0^p1^q \to 1^r0^s$. *Appl. Algebra Eng. Commun. Comput.*, 11(1):1–25, 2000.

17 Hans Zantema, Barbara König, and H. J. Sander Bruggink. Termination of Cycle Rewriting. In Gilles Dowek, editor, *RTA-TLCA 2014*, volume 8560 of *LNCS*, pages 476–490. Springer, 2014.