Characterizing Languages by Normalization and Termination in String Rewriting*
(Extended Abstract)

Jeroen Ketema[1] and Jakob Grue Simonsen[2]

[1] Faculty EEMCS, University of Twente
PO Box 217, 7500 AE Enschede, the Netherlands
j.ketema@ewi.utwente.nl
[2] Department of Computer Science, University of Copenhagen (DIKU)
Njalsgade 126–128, Building 24.5.46, 2300 Copenhagen S, Denmark
simonsen@diku.dk

Abstract. We characterize sets of strings using two central properties from rewriting: normalization and termination. We recall the well-known result that any recursively enumerable set of strings can occur as the set of normalizing strings over a "small" alphabet if the rewriting system is allowed access to a "larger" alphabet (and extend the result to termination). We then show that these results do not hold when alphabet extension is disallowed. Finally, we prove that for every reasonably well-behaved deterministic time complexity class, there is a set of strings complete for the class that also occurs as the set of normalizing or terminating strings, without alphabet extension.

1 Motivation

This paper considers the following fundamental question: If R is a string rewriting system, what must the set of normalizing (alternatively, terminating) strings of R look like?

Rewriting systems are commonly used to characterize sets of objects, for example the sets of strings generated by formal grammars [7], the sets of constructor terms that, when embedded in certain "basic" terms, give rise to reductions to a normal form [8,1], and so forth. However, all of these approaches either assume the entire rewriting system to be terminating, or *extend the signature* that objects can be built from, for example using a larger alphabet to construct strings. As an alternative, we would like to see if any insight can be gained by appealing to notions and methods particular to (string) rewriting: normalization and termination of arbitrary strings, and investigate the sets of strings that normalize or terminate. This paper is a first step in this direction; for non-empty alphabets Σ and Γ with $\Sigma \subseteq \Gamma$, we write $\mathrm{NORM}_{R(\Gamma)}(\Sigma)$ (resp. $\mathrm{TERMIN}_{R(\Gamma)}(\Sigma)$) for the set of strings over Σ that are normalizing (resp. terminating) wrt. the (finite)

* Jakob Grue Simonsen is partially supported by the Sapere Aude grant "Complexity through Logic and Algebra" (COLA).

rewriting system $R(\Gamma)$ whose rules may use symbols from Γ. The main focus of the paper is to characterize the set of languages L that arise as $\mathrm{NORM}_{R(\Gamma)}(\Sigma)$ or $\mathrm{TERMIN}_{R(\Gamma)}(\Sigma)$, in particular in the case $\Gamma = \Sigma$. We loosely call such languages L *characterizable* by normalization (resp. termination).

Related work. McNaughton et al. considered languages accepted by finite, length-reducing and confluent string rewriting systems using extra non-terminal symbols [12], and this work was later heavily generalized by Beaudry et al. [4]. In the setting of [12,4], a language $L \subseteq \Sigma^*$ is accepted (a "McNaughton language" in the terminology of [4]) if there is a $\Gamma \supsetneq \Sigma$ and a finite string rewriting system R over Γ, two strings $t_1, t_2 \in (\Gamma \setminus \Sigma)^* \cap \mathrm{IRR}(R)$, and a symbol $\circ \in (\Gamma \setminus \Sigma) \cap \mathrm{IRR}(R)$ such that for all $w \in \Sigma^*$ we have $w \in L$ iff $t_1 w t_2 \to_R^* \circ$. This construction is very similar to ours in the case where $\circ$ is the only string $v \in \mathrm{IRR}(R)$ such that there is a $w \in \Sigma^*$ with $t_1 w t_2 \to_R^* v$. However, there are two crucial differences: [12,4] allow for other normal forms (however, this is a fairly superficial difference for decidable languages), and they do not treat the case where alphabet extensions are not allowed, i.e., where $\Gamma = \Sigma$.

The idea of disallowing alphabet extensions has cropped up occasionally in the literature, but has apparently not been treated systematically: Minsky has a short discussion of the problem in his classic book [13, Section 12.7], but otherwise the literature is scant. We believe the reason for this is that the foremost applications of string rewriting, proof systems for semi-groups and monoids, attach no special importance to normalizing strings: Indeed, there the interesting property is not reduction $\to^*$ (to normal form or otherwise), but *conversion* $\leftrightarrow^*$ where rules are also allowed to be used "in reverse", hence, rendering normal forms much less important.

There is a wealth of work on length-decreasing string rewriting systems, particularly confluent such systems, where a language L over alphabet Σ is typically characterized by $v \in L$ iff $v \to^* \epsilon$ where ϵ is the empty string; the impetus was Nivat's fundamental work on languages akin to the Dyck language [14] and developed by later researchers, see [5]. Contrary to our work, the rewriting systems considered in the above work are almost invariably terminating (i.e., *all* strings terminate) and confluent.

2 Preliminaries

We refer to the classic textbook [6] for basics on string rewriting, to [16] for general background on rewriting, and to [9,15] for basics of computability. To fix notation, we repeat a few basic definitions below.

Definition 2.1. *An* abstract reduction system *is a pair $(A, \to)$ with A is a non-empty set of* objects *and $\to$ a binary relation on A, called the* reduction relation. *We denote by $\to^+$ the transitive closure of $\to$ and by $\to^*$ the reflexive, transitive closure. An object $a \in A$ is a ($\to$-)*normal form *if there do not exist $b \in A$ with $a \to b$. We denote by* $\mathrm{IRR}(\to)$ *the set of normal forms of A. An object*

$a \in A$ *is* normalizing *(wrt.* $\to$*) if there are* $b \in \mathrm{IRR}(\to)$ *such that* $a \to^* b$*;* a *is* terminating *(wrt.* $\to$*) if there is no infinite sequence* $a = b_1 \to b_2 \to b_2 \to \cdots$.

Let $\Sigma = \{a_1, \ldots, a_n\}$ *be a finite set of symbols, called the* alphabet*; we denote by* Σ^+ *the set of non-empty finite strings over* Σ *and by* Σ^* *the set of finite strings over* Σ*. A* string rewrite rule *over* Σ *is a pair* (l, r) *of strings, invariably written* $l \to r$*; throughout the paper, it is assumed that* $l, r \in \Sigma^+$*. A* string rewriting system *(or* semi-Thue system*), abbreviated SRS, over alphabet* Σ *is a set of string rewrite rules over* Σ*. The* reduction relation $\to_R \subseteq \Sigma^* \times \Sigma^*$ *induced by a string rewriting system* R *is defined by* $v \to w$ *if there are* $x, y \in \Sigma^*$ *and* $l \to r \in R$ *such that* $v = xly$ *and* $w = xry$*. If* R *is clear from the context, we write* $\to$ *instead of* $\to_R$.

One important difference from ordinary string rewriting: All rules $l \to r$ will have both $l \neq \epsilon$ and $r \neq \epsilon$ to avoid a host of special cases. Unless explicitly stated otherwise, R is a *finite* set of rules throughout the paper. Additionally, Σ and Γ will denote finite alphabets (usually with $\Sigma \subseteq \Gamma$) throughout the paper. If R is an SRS over alphabet Σ, we will usually write $R(\Sigma)$ to avoid confusion.

The following definition fixes our two main objects of study:

Definition 2.2. *Let* Σ *be an alphabet and* $R(\Gamma)$ *a string rewriting system (over an alphabet* $\Gamma \supseteq \Sigma$*). By* $\mathrm{NORM}_{R(\Gamma)}(\Sigma)$ *we denote the set of non-empty strings over* Σ *that are normalizing wrt.* $\to_R$*. By* $\mathrm{TERMIN}_{R(\Gamma)}(\Sigma)$ *we denote the set of non-empty, terminating strings over* Σ *wrt.* $\to_R$.

Example 2.3. Let $\Sigma = \{0, 1\}$. We define GOLDEN as the set of non-empty strings over Σ that do not contain 00 as a substring, and SPRIME as the set of non-empty strings over Σ that contain no substring of the form 10^p1 with p a prime number. Moreover, we define PALINDROME as the set of non-empty, even-length palindromes over Σ, and PARITY as the set of non-empty, even-length strings over Σ containing exactly the same number of 0s and 1s.

The following definition is standard (see e.g. [3,2]):

Definition 2.4. *A language* $L \subseteq \Sigma^+$ *is* factorial *if* $s \in L$ *and* $s = uvw$ *(for* $u, w \in \Sigma^*$ *and* $v \in \Sigma^+$*) implies that* $v \in L$.

Note that GOLDEN and SPRIME are factorial, while PALINDROME and PARITY are not.

3 Sets of strings with alphabet extension

We start by considering which sets of strings over Σ can be characterized if we allow the rewrite system to be defined over an extended alphabet $\Gamma \supseteq \Sigma$. For normalization, the following theorem is well-known in different guises, e.g. as a statement about Post Normal Systems [9, Ch. 6.5].

Theorem 3.1. *Let* $L \subseteq \Sigma^+$*. There exists a string rewriting system* $R(\Gamma)$ *(with* $\Gamma \supseteq \Sigma$*), resp.* $R'(\Gamma')$ *(with* $\Gamma' \supseteq \Sigma$*) such that* $L = \mathrm{NORM}_{R(\Gamma)}(\Sigma)$ *iff* L *is recursively enumerable, resp.* $L = \mathrm{TERMIN}_{R'(\Gamma')}(\Sigma)$ *iff* L *is recursively enumerable and factorial.*

Factoriality is essential in the case of termination:

Lemma 3.2. *If R is a string rewriting system, then* $\mathrm{TERMIN}_R(\Sigma)$ *is factorial.*

4 Reducing the alphabet to Σ

A fundamental question is what happens when $\Gamma = \Sigma$, i.e., when we restrict the building blocks of our rewriting systems to the alphabet of the language we wish to characterize. In general, this is impossible:

Example 4.1. Let $\Sigma = \{0, 1\}$ and consider the set PARITY. Clearly, L is recursively enumerable. We claim that there is no $R(\Sigma)$ such that PARITY $= \mathrm{NORM}_{R(\Gamma)}(\Sigma)$. To see this, suppose there is an R and note that $0 \notin L$ and $1 \notin L$. Hence, there must be rules $0 \to r, 1 \to r' \in R$; but then there are no R-normal forms in Σ^+, a contradiction. Note that the same argument can be repeated for any language that contains neither 0 nor 1.

The reader may well ponder what we have gained by the characterization $\mathrm{NORM}_{R(\Gamma)}(\Sigma) = L$ when the rewriting system R employs symbols from a "large" alphabet Γ—surely we generally have $\mathrm{NORM}_{R(\Gamma)}(\Sigma) \subsetneq \mathrm{NORM}_{R(\Gamma)}(\Gamma)$ and, hence, $L \subsetneq \mathrm{NORM}_{R(\Gamma)}(\Gamma)$. In fact, a stronger result holds: If the set of normalizing strings over Γ are built solely with symbols from Σ, then we could have built all rules of R solely with symbols from Σ.

Lemma 4.2. *Let $L \subseteq \Sigma^+$ and let $R(\Gamma)$ satisfy $L = \mathrm{NORM}_R(\Gamma)$. Then there exists $R'(\Sigma)$ satisfying $L = \mathrm{NORM}_{R'(\Sigma)}(\Sigma)$.*

Lemma 4.2 makes clear that when characterizing a set of strings L by devising an SRS R having L as *exactly* its set of normalizing strings, then R can only use the symbols of Σ. This observation naturally leads to the question "can all recursively enumerable sets be characterized this way?"—but Example 4.1 has answered this question in the negative. The next natural question is "*which* recursively enumerable sets cannot be characterized this way?"

We have no full characterization of the languages that are *not* characterizable as $\mathrm{NORM}_{R(\Sigma)}(\Sigma)$ or $\mathrm{TERMIN}_{R(\Sigma)}(\Sigma)$. However, criteria can be given that rule out certain languages. In particular, neither PALINDROME, nor PARITY is characterizable (indeed, PALINDROME must always have infinite symmetric difference with any characterizable language). GOLDEN *is* characterizable; the status of SPRIME is not known to the authors.

5 Complexity of languages characterizable in Σ

A naïve—and wrong—conjecture is that $\mathrm{NORM}_{R(\Sigma)}(\Sigma)$ and $\mathrm{TERMIN}_{R(\Sigma)}(\Sigma)$ must be very simple. We shall prove that this is not the case by showing that $\mathrm{NORM}_{R(\Sigma)}(\Sigma)$ and $\mathrm{TERMIN}_{R(\Sigma)}(\Sigma)$ can be hard and complete for arbitrarily hard complexity classes.

We start with a few preliminaries on computational complexity (see [10,15] for examples and further explanation).

Definition 5.1. *Let $\mathcal{F}$ be a class of functions $f : \Gamma^+ \longrightarrow \Gamma^+$. We say that a set $A \subseteq \Gamma^+$ is* $\mathcal{F}$-hard *for a class of subsets $\mathcal{C} \subseteq \mathcal{P}(\Gamma^+)$ if for every set B in $\mathcal{C}$ there exists $f \in \mathcal{F}$ such that $x \in B$ iff $f(x) \in A$ for all $x \in \Gamma^+$. A is* complete for $\mathcal{C}$ under $\mathcal{F}$-reduction *(usually just abbreviated $\mathcal{C}$-complete) if $A \in \mathcal{C}$ and A is $\mathcal{F}$-hard for $\mathcal{C}$.*

For a function f and a set $\mathcal{G}$ of functions $\mathbb{N} \longrightarrow \mathbb{N}$, we say that f is globally bounded by a function in $\mathcal{G}$ *(written $f \leq \mathcal{G}$) if there exists a function $g \in \mathcal{G}$ such that for all $n \in \mathbb{N}$, we have $f(n) \leq g(n)$. If $\mathcal{F}$ is a class of functions of type $\Gamma^+ \longrightarrow \Gamma^+$, we say that $\mathcal{F}$ is* time-defined *(resp.* space-defined*) by $\mathcal{G}$ if $\mathcal{F}$ is exactly the class of functions of type $\Gamma^+ \longrightarrow \Gamma^+$ that are computed by a multi-tape deterministic Turing Machine running in time (resp. space) bounded above by a function in $\mathcal{G}$. If $\mathcal{C} \subseteq \mathcal{P}(\Gamma^+)$ is a class of languages, we say that $\mathcal{C}$ is time-defined (resp. space-defined) by $\mathcal{G}$ if $\mathcal{C}$ is the set of languages L such that $L = f^{-1}(1)$ for some $f \in \mathcal{F}$ where $\mathcal{F}$ is a class of functions of type $\Gamma^+ \longrightarrow \{0,1\}$ time-defined (resp. space-defined) by $\mathcal{G}$.*

The set $\mathcal{G}$ is closed under polynomial slowdown *if, for any $g \in \mathcal{G}$ and any polynomial P with coefficients from $\mathbb{N}$, we have $f \leq \mathcal{G}$ for $f(x) = P(g(x))$. If $\mathcal{F}$ (resp. $\mathcal{C}$) are classes of functions (resp. sets) that are time- or space-defined by $\mathcal{G}$ we say that $\mathcal{F}$ (resp. $\mathcal{C}$) is closed under polynomial slowdown if $\mathcal{G}$ is.*

$\mathcal{G}$ is O-closed if, for each $f \leq \mathcal{G}$ and each positive integer a, we have $a \cdot f \leq \mathcal{G}$ (note that if $f(n) > 0$ for all n, then O-closure implies that $(n \mapsto f(n) + c) \leq \mathcal{G}$ for all integers c, i.e., additive constants "don't matter"). If a class of functions $\mathcal{F}$ or sets $\mathcal{C}$ are time- or space-defined by $\mathcal{G}$, then $\mathcal{F}$ or $\mathcal{C}$ is O-closed if $\mathcal{G}$ is.

We now have the following result about the normalizing, resp. terminating, strings over an alphabet Σ:

Theorem 5.2. *Let $\mathcal{F}$ be an O-closed class of functions and let $\mathcal{C} \subseteq \mathcal{P}(\Sigma^+)$ be a class of languages time-defined by $\mathcal{G}$ and closed under polynomial slowdown. If there exists a $\mathcal{C}$-complete set under $\mathcal{F}$-reduction, then there is an $S(\Sigma)$, resp. $S'(\Sigma)$, such that* $\mathrm{NORM}_{S(\Sigma)}(\Sigma)$*, resp.* $\mathrm{TERMIN}_{S'(\Sigma)}(\Sigma)$*, is $\mathcal{C}$-complete.*

Thus, there are complete sets L, L' for PTIME and EXPTIME under log-space-reductions such that $L = \mathrm{NORM}_{S(\Sigma)}(\Sigma)$, resp. $L' = \mathrm{TERMIN}_{S'(\Sigma)}(\Sigma)$ for appropriate SRS $S(\sigma)$, resp. $S'(\Sigma)$.

6 Conclusion and future work

We have considered the problem of characterizing sets of strings as the sets of normalizing, resp. terminating, strings over a finite string rewriting system. A number of open questions remain, the most important of which is to give precise necessary and sufficient conditions for a set of strings to be characterizable in this way. Other interesting problems include: (a) Which sets can be characterized by non-overlapping rewriting systems? (b) Which sets can be characterized by confluent rewriting systems? (c) How *large* is the class of characterizable sets? (The exact notion of largeness is debatable, one suggestion is to use a suitable

form of constructive measure [11]). (d) The authors of [4] identify an extensive hierarchy of classes of McNaughton languages; can a similar hierarchy be obtained in our case when $\Gamma = \Sigma$? (e) What are the exact closure properties under standard operations of the class of languages characterized by normalization, resp. termination?

References

1. M. Avanzini and G. Moser. Closing the gap between runtime complexity and polytime computability. In *Proceedings of the 21st International Conference on Rewriting Techniques and Applications (RTA 2010)*, volume 6 of *Leibniz International Proceedings in Informatics*, pages 33–48, 2010.
2. M.-P. Béal, M. Crochemore, F. Mignosi, A. Restivo, and M. Sciortino. Computing forbidden words of regular languages. *Fundamenta Informaticae*, 56(1–2):121–135, 2003.
3. M.-P. Béal, F. Mignosi, A. Restivo, and M. Sciortino. Forbidden words in symbolic dynamics. *Advances in Applied Mathematics*, 25:163–193, 2000.
4. M. Beaudry, M. Holzer, G. Niemann, and F. Otto. McNaughton families of languages. *Theoretical Computer Science*, 290(3):1581–1628, 2003.
5. R. Book. Thue systems as rewriting systems. *Journal of Symbolic Computation*, 3(1–2):39–68, 1987.
6. R. Book and F. Otto. *String Rewriting*. Texts and Monographs in Computer Science. Springer-Verlag, 1993.
7. N. Chomsky. *Syntactic Structures*. Mouton & Co., 1957.
8. C. Choppy, S. Kaplan, and M. Soria. Complexity analysis of term-rewriting systems. *Theoretical Computer Science*, 67(2&3):261–282, 1989.
9. M. Davis. *Computability and Unsolvability*. Dover Publications Inc., 1982. Originally published in 1958 by McGraw-Hill Book Company.
10. N. D. Jones. *Computability and Complexity from a Programming Perspective*. The MIT Press, 1997.
11. J. H. Lutz. The dimensions of individual strings and sequences. *Information and Computation*, 187(1):49–79, 2003.
12. R. McNaughton, P. Narendran, and F. Otto. Church-Rosser Thue systems and formal languages. *Journal of the Association for Computing Machinery*, 35(2):324–344, 1988.
13. M. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall Series in Automatic Computation. Prentice-Hall, 1967.
14. M. Nivat. On some families of languages related to the Dyck language. In *Proceedings of the 2nd Annual ACM Symposium on Theory of Computing (STOC '70)*, pages 221–225, 1970.
15. M. Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology, 2nd edition, 2006.
16. Terese, editor. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.