On the Structure and Complexity of Rational Sets of Regular Languages

Andreas Holzer[1], Christian Schallhart[2], Michael Tautschnig[3], and Helmut Veith[1]

1 Vienna University of Technology, Austria
2 University of Oxford, UK
3 Queen Mary, University of London, UK

Abstract

In the recently designed and implemented test specification language FQL, relevant test goals are specified as regular expressions over program locations. To transition from single test goals to test suites, FQL describes suites as regular expressions over finite alphabets where each symbol corresponds to a regular expression over program locations. Hence, each word in a test suite expression yields a test goal specification. Such test suite specifications are in fact rational sets of regular languages (RSRLs). We show closure properties of general and finite RSRLs under common set theoretic operations. We also prove complexity results for checking equivalence and inclusion of star-free RSRLs and for checking whether a regular language is a member of a general or star-free RSRL. As the star-free (and thus finite) case underlies FQL specifications, the closure and complexity results provide a systematic foundation for FQL test specifications.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Rational Sets, Regular Languages, Test Specification in FQL, Closure Properties, Decision Problems

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2013.377

1 Introduction

Despite the success of model checking and theorem proving, software testing has a dominant role in industrial practice. In fact, state-of-the-art development guidelines such as the avionic standard DO-178B [27] are heavily dependent on test coverage criteria. It is therefore quite surprising that the formal specification of coverage criteria has been a blind spot in the formal methods and software engineering communities for a long time.

In a recent thread of papers [14, 12, 17, 16, 15, 6], we have addressed this situation and introduced the FShell Query Language (FQL) to specify and tailor coverage criteria, together with FShell, a tool to generate matching test suites for ANSI C programs. At the semantic core of FQL, test goals are described as regular expressions whose alphabet are the edges of the program control flow graph (CFG). For example, to cover a particular CFG edge c, one can use the regular expression $\Sigma^\star\ c\ \Sigma^\star$. Importantly, however, a coverage criterion usually induces not just a single test goal, but a (possibly large) number of test goals – e.g. *all* basic blocks of a program. FQL therefore employs regular languages which can express sets of regular expressions. To this end, the alphabet contains not only the CFG edges but also *postponed regular expressions* over these edges, written within quotes.

For example, “$\Sigma^\star$” $(a+b+c+d)$ “$\Sigma^\star$” describes the language {“$\Sigma^\star$” a “$\Sigma^\star$”, “$\Sigma^\star$” b “$\Sigma^\star$”, “$\Sigma^\star$” c “$\Sigma^\star$”, “$\Sigma^\star$” d “$\Sigma^\star$”}. Each of these words is a regular expression that will then serve as a test goal. Following [1], we call such languages *rational sets of regular languages (RSRL)*.

33rd Int’l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013).
Editors: Anil Seth and Nisheeth K. Vishnoi; pp. 377–388
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The goal of this paper is to initiate a systematic study of RSRLs from a theoretical point of view, considering closure properties and complexity of common set-theoretic operations. Thus, this paper is a first step towards a systematic foundation of FQL. In particular, a good understanding of set-theoretic operations is necessary for systematic algorithmic optimization and manipulation of test specifications. First results on query optimization for FQL have been obtained in [6].

A rational set of regular languages is given by a regular language K over alphabet Δ, and a *regular language substitution* $\varphi : \Delta \to 2^{\Sigma^\star}$, mapping each symbol $\delta \in \Delta$ to a regular language $\varphi(\delta)$ over alphabet Σ. We extend φ to words $w \in \Delta^+$ with $\varphi(\delta \cdot w) = \varphi(\delta) \cdot \varphi(w)$, and set $\varphi(L) = \bigcup_{w \in L} \varphi(w)$ for $L \subseteq \Delta^+$. The class of rational sets of a monoid $(M, \cdot, e)$ is the smallest subclass of M such that *(i)* $\emptyset$ is a rational set, *(ii)* each singleton set $\{m\}$ for $m \in M$ is a rational set, and if N_1 and N_2 are rational sets *(iii)* then $N_1 \cdot N_2$ is a rational set where $\cdot$ on rational sets is defined by the point-wise application of the monoid's $\cdot$ operation, *(iv)* $N_1 \cup N_2$ is a rational set, and *(v)* $N_1^\star$ is a rational set [9, 22].

▶ **Definition 1** (Rational Sets of Regular Languages, RSRLs [1])**.** Given a finite alphabet Σ, the *rational sets of regular languages* are the rational sets over the monoid $(2^{\Sigma^\star}, \cdot, \{\varepsilon\})$, where ε denotes the empty word. We represent a rational set of regular languages $\mathcal{R}$ as tuple (K, φ), where $K \subseteq \Delta^+$ is a regular language over a finite alphabet Δ, and φ is a regular language substitution $\varphi : \Delta \to 2^{\Sigma^\star}$, such that $\mathcal{R} = \{\varphi(w) \mid w \in K\}$. We say that RSRL $\mathcal{R}$ is *Kleene star free,* if there exists $(K, \varphi) = \mathcal{R}$ such that K is finite (and hence Kleene-star free).

Depending on context, we refer to $\mathcal{R}$ as a set of languages or as a pair (K, φ), but we always write $L \in \mathcal{R}$ iff $\exists w \in K : L = \varphi(w)$. Consider the above specification "$\Sigma^\star$" $(a+b+c+d)$ "$\Sigma^\star$" over base alphabet $\Sigma = \{a, b, c, d\}$. To represent this specification as RSRL $\mathcal{R} = (K, \varphi)$, we set $\Delta = \{\delta_{\Sigma^\star}\} \cup \Sigma$, containing a fresh symbol $\delta_{\Sigma^\star}$ for the quoted expression "$\Sigma^\star$". We set $K = L(\delta_{\Sigma^\star}\ (a + b + c + d)\ \delta_{\Sigma^\star})$ with $\varphi(\delta_{\Sigma^\star}) = \Sigma^\star$ and $\varphi(\sigma) = \{\sigma\}$ for $\sigma \in \Sigma$. Thus K contains the words $\delta_{\Sigma^\star}\ a\ \delta_{\Sigma^\star}, \ldots$ with $\varphi(\delta_{\Sigma^\star}\ a\ \delta_{\Sigma^\star}) = L(\Sigma^\star\ a\ \Sigma^\star) \in \mathcal{R}$, as desired.

Note that the RSRL above is finite with exactly four elements. This is of course not atypical: in concrete testing applications, FQL generates finite sets of test goals, since it relies on *Kleene star free* RSRLs only. For future applications, however, it is well possible to consider infinite sets of test goals e.g. for unbounded integer and real valued variables or for path coverage criteria which are either matched partially, or by abstract executions. In this paper, we are therefore considering the general, finite, and Kleene star free case.

▶ **Example 2.** Consider the alphabets $\Delta = \{\delta_1, \delta_2\}$ and $\Sigma = \{a, b\}$. Then, **(1)** with $\varphi(\delta_1) = L(a^\star)$, $\varphi(\delta_2) = \{ab\}$, and $K = L(\delta_1 \delta_2^\star \delta_1)$, we obtain the rational set of regular languages $\{L(a^\star (ab)^i a^\star) \mid i \in \mathbb{N}\}$; **(2)** with $\varphi(\delta_1) = L(a^\star)$, $\varphi(\delta_2) = \{a\}$, and $K = L(\delta_1 \delta_2^\star)$, we obtain $\varphi(w_1) \supset \varphi(w_2)$ for all $w_1, w_2 \in K$ with $|w_1| < |w_2|$; **(3)** with $\varphi(\delta_1) = \{\varepsilon, a\}$, $\varphi(\delta_2) = \{aa\}$, and $K = L(\delta_1 \delta_2^\star)$, we have $|\varphi(w)| = 2$ and $\varphi(w) \cap \varphi(w') = \emptyset$ for all $w \neq w' \in K$.

In the finite case we make an additional distinction for the subcase where the regular expressions in Δ, i.e., the set of postponed regular expressions, are fixed. This has practical relevance, because in the context of FQL, the results of the operations on RSRL will be better readable by engineers if Δ is unchanged.

Contributions and Organization

In Section 3, we show *closure properties* for general and finite RSRLs, considering the operators product, Kleene star, complement, union, intersection, set difference, and symmetric difference.

We also consider the case of finite RSRLs with a fixed language substitution φ, as this case is of particular interest for testing applications. In Section 4, we prove the *complexity results* of the decision problems equivalence, inclusion, and membership for Kleene star free RSRLs. To prove an upper bound on the complexity of the membership problem, we expand the decidability proof in [1] and give a first complete and explicit algorithm for the problem. We close in Section 5 in discussing how our results reflect back to design decisions for FQL.

2 Related Work

Afonin et al. [1] introduced RSRLs and studied the decidability of whether a regular language is contained in an RSRL and the decidability of whether an RSRL is finite. Although Afonin et al. shortly discuss possible upper bounds for the membership decision problem, their analysis is incomplete due to gaps in their algorithmic presentation (see also a more detailed discussion in Section 4.5). Closely connected to the membership problem is the question, whether a regular language L is expressible via a combination of a given set of regular languages L_i. Motivated by query rewriting for graph databases, Calvanese et al. [7] show the complexity of determining the maximal rewriting of a regular language L with given regular languages L_i. In earlier work, Hashiguchi [11] shows that it is decidable whether a regular language L is expressible via a finite application of a subset of the regular operators concatenation, union, and star to regular languages L_i. Afonin et al. [1] realized that distance automata [10] enable a decision algorithm for the membership problem for RSRL. Although this construction relies on distance automata, the properties analyzed by Krob [23] and Colcombet and Daviaud [8] are not applicable in our context. Kirsten [20, 21] generalizes distance automata to distance desert automata and uses these automata to show the first complexity result for determining whether a regular language is of a certain star height. Berstel [5] surveys closure properties of rational and recognizable subsets of monoids and thereby also the relationship between rational and recognizable subsets. Yet, most stated results do not apply to RSRLs, hence we investigate closure properties of RSRLs. Pin [26] introduced the term *extended automata* for RSRLs as an example of recognizable languages that can be characterized by constraint systems over symbols and substrings occurring in words of the language, but he did not further investigate any of their properties. In our own related work on FQL [14, 13, 12, 17, 16, 15, 6], we deal with practical issues arising in test case generation. Beyond RSRLs, FQL provides an additional language layer to extract suitable alphabets from the programs e.g. referring with a single symbol to all basic blocks of the program under scrutiny.

Let us finally discuss other work whose terminology is similar to RSRLs without direct technical relation. Barceló et al. define *rational relations*, which are relations between words over a common alphabet, whereas we consider sets of regular languages [3]. Barceló et al. also investigate *parameterized regular languages* [4], where words are obtained by replacing variables in expressions with alphabet symbols. *Metaregular languages* deal with languages recognized by automata with a time-variant structure [2, 28]. Lattice Automata [24] only consider lattices that have a unique complement element, whereas RSRLs are not closed under complement (no RSRL has an RSRL as complement).

3 Closure Properties

We investigate the closure properties of RSRLs, considering standard set theoretic operators, such as union, intersection, and complement, and variants thereof, fitting RSRLs. In particular, we apply those operators also to pairs in the *Cartesian product* of RSRLs, and *point-wise* to each element in an RSRL and another given regular language.

▶ **Definition 3** (Operations on RSRL). Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be RSRLs and let R be a regular language. Then, we define the following operations on RSRLs:

Operation	Definition			
Product	$\mathcal{R}_1 \cdot \mathcal{R}_2$	$=$	$\{L_1 \cdot L_2 \mid L_1 \in \mathcal{R}_1, L_2 \in \mathcal{R}_2\}$	
Kleene Star	$\mathcal{R}_1^\star$	$=$	$\bigcup_{i\in\mathbb{N}} \mathcal{R}_1^i$	
Point-wise	$\dot{\mathcal{R}}_1^\star$	$=$	$\{L^\star \mid L \in \mathcal{R}_1\}$	
Complement	$\overline{\mathcal{R}_1}$	$=$	$\{L \subseteq 2^{\Sigma^\star} \mid L \notin \mathcal{R}_1\}$	
Point-wise	$\dot{\overline{\mathcal{R}_1}}$	$=$	$\{\overline{L} \mid L \in \mathcal{R}_1\}$	
Binary Operators	$\mathcal{R}_1 \cap \mathcal{R}_2,\ \mathcal{R}_1 \cup \mathcal{R}_2,\ \mathcal{R}_1 - \mathcal{R}_2$			(standard def.)
Point-wise	$\mathcal{R}_1 \dot{\cup} / \dot{\cap} / \dot{-} R$	$=$	$\{L \cup / \cap / - R \mid L \in \mathcal{R}_1\}$	
Cartesian	$\mathcal{R}_1 \overset{\times}{\cup} / \overset{\times}{\cap} / \overset{\times}{-} \mathcal{R}_2$	$=$	$\{L_1 \cup / \cap / - L_2 \mid L_1 \in \mathcal{R}_1, L_2 \in \mathcal{R}_2\}$	
Symmetric Difference	$\mathcal{R}_1 \Delta \mathcal{R}_2$	$=$	$\{L \mid L \in ((\mathcal{R}_1 \cup \mathcal{R}_2) - (\mathcal{R}_1 \cap \mathcal{R}_2))\}$	

We analyze *three different classes* of RSRLs for being closed under these operators: **(1)** General RSRLs, **(2)** finite RSRLs, and **(3)** finite RSRLs with a fixed language substitution φ. For closure properties, we do *not distinguish* between Kleene star free and finite RSRLs, since every finite RSRL is expressible as Kleene star free RSRL (however, given an RSRL with Kleene star, it is non-trivial to decide whether the given RSRL is finite or not [1]). Therefore, all closure properties for finite RSRLs apply to Kleene star free RSRLs as well. Hence, cases **(2-3)** correspond to FQL. Case **(3)** is relevant for usability in practice, allowing to apply the corresponding operators without constructing a new language substitution. This does not only significantly reduce the search space but also provides more intuitive results to users.

▶ **Theorem 4** (Closure Properties of RSRL). *The following table summarizes the closure properties for RSRLs.*

Operation	Closure Property		
	General	Finite RSRLs	
(+ closed − not closed ? unknown)		General	Fixed Subst.
Product	+	+	+
Kleene Star	+	−	−
Point-wise	−	+	−
Complement	−	−	−
Point-wise	−	+	−
Union	+	+	+
Point-wise	−	+	−
Cartesian	−	+	−
Intersection	?	+	+
Point-wise	−	+	−
Cartesian	−	+	−
Difference	?	+	+
Point-wise	−	+	−
Cartesian	−	+	−
Symmetric	?	+	+

As most proofs for Theorem 4 are straightforward, we only exemplify the proofs for point-wise operators using the point-wise union operator (cf. Proposition 6) and show the rest of the proofs in an extended online version of this paper [18]. The following set of regular languages is not an RSRL and we use it to prove the non-closure of RSRLs under the point-wise union operator.

▶ **Example 5.** Consider the set $\mathcal{M} = \{\{b\} \cup \{a^i \mid 1 \leq i \leq n+1\} \mid n \in \mathbb{N}\} \subseteq 2^{\{a,b\}^\star}$. $\mathcal{M}$ contains infinitely many languages, therefore, any RSRL $\mathcal{R} = (K, \varphi)$, with $\mathcal{M} = \mathcal{R}$, requires a regular language K containing infinitely many words. By L_n we denote the set $\{b\} \cup \{a^i \mid 1 \leq i \leq n+1\}$. Then, $L_0 \subsetneq L_1 \subsetneq \dots L_{i-1} \subsetneq L_i \subsetneq L_{i+1} \subsetneq \dots$. There must be a word $w = uvz \in K$ such that $uv^iz \in K$, for all $i \geq 1$ (cf. pumping lemma for regular languages [19]). Furthermore, there must be such a word $w = uvz$ such that $\varphi(u) \neq \emptyset$, $\varphi(v) \neq \emptyset$, $\varphi(v) \neq \{\varepsilon\}$, and $\varphi(z) \neq \emptyset$. This is due to the fact that we have to generate arbitrary long words a^i. We can assume that $b \notin \varphi(v)$ because otherwise $b^i \in \varphi(v^i)$, for all $i \geq 1$. Therefore, $a^k \in \varphi(v)$ for some $k \geq 1$. Since $b \in \varphi(uvz)$ has to be true, we can assume w.l.o.g. that $b \in \varphi(u)$. But, then $ba^k \dots \in \varphi(uvz)$. This is a contradiction to the fact that, for all $n \geq 1$, $ba^k \dots \notin L_n$.

▶ **Proposition 6** (Closure of Point-wise Union)**.** *The set $\mathcal{R}_1 \uplus R$ is, in general, not an RSRL.*

Proof. Let $\mathcal{R}_1 = (L(\delta_1\delta_2^\star), \varphi)$ with $\varphi(\delta_1) = \{a\}$ and $\varphi(\delta_2) = L(a + \varepsilon)$ and let $R = \{b\}$. Then, $\mathcal{R}_1 \uplus R = \{\{b\} \cup \{a^i \mid 1 \leq i \leq n+1\} \mid n \in \mathbb{N}\}$ which is not an RSRL, as shown in Example 5. ◀

4 Decision Problems

Given a regular language $R \subseteq \Sigma^\star$ and an RSRL $\mathcal{R} = (K, \varphi)$ over the alphabets Δ and Σ, the *membership problem* is to decide whether $R \in \mathcal{R}$ holds. Given another $\mathcal{R}' = (K', \varphi')$, also over the alphabets Δ' and Σ, the *inclusion problem* asks whether $\mathcal{R} \subseteq \mathcal{R}'$ holds, and the *equivalence problem,* whether $\mathcal{R} = \mathcal{R}'$ holds.

▶ **Theorem 7** (Equivalence, Inclusion, and Membership for Kleene star free RSRLs)**.** *Membership, inclusion, and equivalence are* PSPACE*-complete for Kleene star free represented RSRLs.*

This holds true, since in case of Kleene star free represented RSRLs (given explicitly as (K, φ) with K finite), we can enumerate the regular expressions defining all member languages in PSPACE. Given the PSPACE-completeness of regular language equivalence, we compare a given regular expression with all member languages, solving the membership problem in PSPACE. Doing so for all languages of another RSRL solves the inclusion problem, and checking mutual inclusion yields an algorithm for equivalence. This approach does *not* immediately generalize to finite RSRLs, since finite RSRLs $\mathcal{R} = \{\varphi(w) \mid w \in K\}$ may be generated from an infinite K with Kleene stars.

In the general case, the situation is quite different: Previous work shows that the membership problem is decidable [1], but without turning the construction into a concrete algorithm or determining an upper bound for complexity of the problem. Taking this work as starting point, in the remainder of this section, we give an 2EXPSPACE upper bound on the complexity of the problem, discussing the relationship with [1] at the end of the section. The decidability of inclusion and equivalence remains open.

4.1 Membership for general RSRLs

By definition, the membership problem is equivalent to asking whether there exists a $w \in K$ with $\varphi(w) = R$. For checking the existence of such a w, we have to check possibly infinitely many words in K efficiently. To render this search feasible, we **(A)** rule out irrelevant parts of K, and **(B)** treat subsets of K at once. This leads to the procedure $\mathsf{membership}(K, R, \varphi)$ shown in Algorithm 1, which first enumerates with $M' \in \mathsf{enumerate}(K, R, \varphi)$ a sufficient set of sublanguages (Line 1), and then checks each of those sublanguages individually (Line 2).

Algorithm 1: membership(R, K, φ)

input : regular languages $R \subseteq \Sigma^\star$, $K \subseteq \Delta^\star$,
regular language substitution φ with $\varphi(\delta) \subseteq \Sigma^\star$ for all $\delta \in \Delta$

returns : **true** iff $\exists w \in K : \varphi(w) = R$ (i.e., iff $R \in (K, \varphi)$)

```
foreach M′ ∈ enumerate(R, K, φ) do
    if basiccheck(R, M′, φ) then return true;
return false;
```

More specifically, we employ the following optimizations: We rule out **(A.1)** all words w with $\varphi(w) \not\subseteq R$, and **(A.2)** all words w whose language $\varphi(w)$ differs from R in the *length of its shortest word.* We subdivide the remaining search space **(B)** into finitely many suitable languages M' and check the existence of a $w \in M'$ with $\varphi(w) = R$ in a single step.

We discuss a mutually fitting design of these steps below and consider the resulting complexity. However, due to space limitations, we put the necessary proofs into an extended online version of this paper [18].

(A.1) Maximal Rewriting

To rule out all w with $\varphi(w) \not\subseteq R$, we rely on the notion of a *maximal φ-rewriting $M_\varphi(R)$ of* R, taken from [7]. $M_\varphi(R)$ consists of the words w with $\varphi(w) \subseteq R$, i.e., we set $M_\varphi(R) = \{w \in \Delta^+ \mid \varphi(w) \subseteq R\}$. Furthermore, all subsets $M \subseteq M_\varphi(R)$ are called *rewritings* of R, and if $\varphi(M) = R$ holds, M is called *exact* rewriting.

▶ **Proposition 8** (Regularity of maximal rewritings [7][1]). *Let $\varphi : \Delta \to 2^{\Sigma^\star}$ be a regular language substitution. Then the* maximal φ-rewriting $M_\varphi(R)$ *of a regular language $R \subseteq \Sigma^\star$ is a regular language over Δ.*

As all words w with $\varphi(w) = R$ must be element of $M_\varphi(R)$, we restrict our search to $M = M_\varphi(R) \cap K$.

(A.2) Minimal Word Length

We restrict the search space further by checking the *minimal word length,* i.e., we compare the length of the respectively shortest word in R and $\varphi(w)$. If R and $\varphi(w)$ have different minimal word lengths, $R \neq \varphi(w)$ holds, and hence, we rule out w. We define the minimal word length $\mathsf{minlen}(L)$ of a language L with $\mathsf{minlen}(L) = \min\{|w| \mid w \in L\}$, leading to the definition of language strata.

▶ **Definition 9** (Language Stratum). Let L be a language over Δ, and $\varphi : \Delta \to 2^{\Sigma^\star}$ be a regular language substitution, then the *B-stratum of L,* denoted as $L[B, \varphi]$, is the set of words in L which generate via φ languages of minimal word length B, i.e., $L[B, \varphi] = \{w \in L \mid \mathsf{minlen}(\varphi(w)) = B\}$.

Starting with $M = M_\varphi(R) \cap K$, we restrict our search further to $M[\mathsf{minlen}(R), \varphi]$.

---

[1] This proposition is not trivial, as φ is not a homomorphism mapping each word to a single word, but a substitution mapping each word w to a language $\varphi(w)$; if $\varphi(w)$ would yield only words, we would immediately obtain $M_\varphi(R) = \overline{\varphi^{-1}(\overline{R})}$ for $\varphi^{-1}(L) = \{w \mid \varphi(w) \cap L \neq \emptyset\}$.

(B) 1-Word Summaries

It remains to subdivide $M[\mathsf{minlen}(R), \varphi]$ into finitely many subsets M', which are then checked efficiently without enumerating their words $w \in M'$. Here, we only discuss the property of these subsets M' which enables such an efficient check, and later we will describe an enumeration of those subsets M'. When we check a subset M', we do not search for a single word $w \in M'$ with $\varphi(w) = R$ but for a finite set $F \subseteq M'$ with $\varphi(F) = R$. The soundness of this approach will be guaranteed by the existence of *1-word summaries:* A language $M' \subseteq \Delta^\star$ has 1-word summaries, if for all finite subsets $F \subseteq M'$ there exists a summary word $w \in M'$ with $\varphi(F) \subseteq \varphi(w)$. The property we exploit is given by the following proposition.

▶ **Proposition 10** (Membership Condition for Summarizable Languages, adapting [1]). *Let $M' \subseteq \Delta^\star$ be a regular language with 1-word summaries and $\varphi(M') \subseteq R$. Then there exists a $w \in M'$ with $\varphi(w) = R$ iff there exists a finite subset $F \subseteq M'$ with $\varphi(F) = \varphi(M') = R$.*

Putting it together

First, combining **A.2** and **B**, we obtain Lemma 11, to subdivide the search space $M[B, \varphi]$ into a set $\mathsf{rep}(M, B, \varphi)$ of languages M' with 1-word summaries. Second, in Theorem 12, building upon Lemma 11 and **A.1**, we fix $B = \mathsf{minlen}(R)$ and iterate through these languages M'. We check each of them at once with our membership condition from Proposition 10. In terms of Algorithm 1, Lemma 11 provides the foundation for $\mathsf{enumerate}(K, R, \varphi)$ and Proposition 10 underlies $\mathsf{basiccheck}(R, M', \varphi)$.

▶ **Lemma 11** (Summarizable Language Representation, adapting [1]). *Let $M \subseteq \Delta^\star$ be a regular language and $\varphi : \Delta \to 2^{\Sigma^\star}$ be a regular language substitution. Then, for each bound $B \geq 0$, there exists a family $\mathsf{rep}(M, B, \varphi)$ of union-free regular languages $M' \in \mathsf{rep}(M, B, \varphi)$ with 1-word summaries, such that $M[B, \varphi] \subseteq \bigcup_{M' \in \mathsf{rep}(M,B,\varphi)} M' \subseteq M$ holds.*

▶ **Theorem 12** (Membership Condition, following [1]). *Let $\mathcal{R} = (K, \varphi)$ be a RSRL and $\varphi : \Delta \to 2^{\Sigma^\star}$ be a regular language substitution. Then, for a regular language $R \subseteq \Sigma^\star$, we have $R \in \mathcal{R}$, iff there exists an $M' \in \mathsf{rep}(M_\varphi(R) \cap K, \mathsf{minlen}(R), \varphi)$ with a finite subset $F \subseteq M'$ with $\varphi(F) = \varphi(M') = R$.*

We obtain the space complexity of $\mathsf{membership}$, depending on the *size of the expressions* which represent the involved languages. More specifically, the complexity depends on the expression sizes $||R||$ and $||K||$ and the summed size $||\varphi|| = \Sigma_{\delta \in \Delta} ||\varphi(\delta)||$ of the expressions in the co-domain of φ.

▶ **Theorem 13** ($\mathsf{membership}(R, K, \varphi)$ runs in 2EXPSPACE). *More precisely, it runs in* $\mathrm{DSPACE}\left(||K||^r 2^{2^{(||R||+||\varphi||)^s}}\right)$ *for some constants r and s.*

We prove Theorem 13 in Section 4.4, relying on the algorithms presented in Sections 4.2 and 4.3.

4.2 Implementing $\mathsf{basiccheck}(R, M', \varphi)$

Since Lemma 11 produces only languages $M' = N_1 S_1^\star N_2 \ldots N_m S_m^\star N_{m+1}$ with 1-word summaries, we restrict our implementation to such languages and exploit these restrictions subsequently. So, given such a language M' over Δ, and a regular language substitution $\varphi : \Delta \to 2^{\Sigma^\star}$, we need to check whether there exists a finite $F \subseteq M'$ with $\varphi(F) = \varphi(M') = R$.

Algorithm 2: basiccheck(R, M', φ)

input : regular languages $R \subseteq \Sigma^\star$, $M' \subseteq \Delta^\star$, and regular language substitution φ with $\varphi(\delta) \subseteq \Sigma^\star$ for all $\delta \in \Delta$

requires : M' is union-free and $\varphi(M') \subseteq R$

returns : **true** iff $\exists$ finite $F \subseteq M' : \varphi(F) = \varphi(M') = R$

1 **build** $A_{M'}$;

2 **if** $A_{M'}$ *limited* **then**

3   **if** $\varphi(M') = R$ **then return true**;

4 **return false**;

We implement this check with the procedure basiccheck(R, M', φ), splitting the condition of Proposition 10 into two parts, namely **(1)** whether there exists a finite $F \subseteq M'$ with $\varphi(F) = \varphi(M')$, and **(2)** whether $\varphi(M') = R$ holds. While the latter condition amounts to regular language equivalence, the former requires distance automata as additional machinery.

▶ **Definition 14** (Distance Automaton [10])**.** A *distance automaton* over an alphabet Δ is a tuple $\mathcal{A} = \langle \Delta, Q, \rho, q_0, F, d\rangle$ where $\langle \Delta, Q, \rho, q_0, F\rangle$ is an NFA and $d : \rho \to \{0, 1\}$ is a distance function, which can be extended to a function on words as follows. The distance function $d(\pi)$ of a path π is the sum of the distances of all edges in π. The distance $\mu(w)$ of a word $w \in L(\mathcal{A})$ is the minimum of $d(\pi)$ for all paths π accepting w.

A distance automaton $\mathcal{A}$ is called *limited* if there exists a constant U such that $\mu(w) < U$ for all words $w \in L(\mathcal{A})$.

In our check for **(1)**, we build a distance automaton which is limited iff a finite F with $\varphi(F) = \varphi(M')$ exists. Then, we rely on the PSpace-decidability [25] of the limitedness of distance automata to check whether F exists or not.

Distance-automaton Construction

Here, we exploit the assumption that M' is a union-free language over Δ: Given the regular expression defining M', we construct the distance automaton $A_{M'}$ following the form of this regular expression:

- $\delta \in \Delta$: We construct the finite automaton A_δ with $L(A_\delta) = \varphi(\delta)$. We extend A_δ to a distance automaton by labeling each transition in A_{δ_i} with 0.
- $e \cdot f$: Given distance automata A_e and A_f with $A_e = (Q_e, \Sigma, \rho_e, q_{0,e}, F_e, d_e)$ and $A_f = (Q_f, \Sigma, \rho_f, q_{0,f}, F_f, d_f)$, we set $A_{e\cdot f} = (Q_e \uplus Q_f, \Sigma, \rho_e \cup \rho_f \cup \rho, q_{0,e}, F_f, d_{e\cdot f})$ where $\rho = \{(q, \varepsilon, q_{0,f}) \mid q \in F_e\}$ and $d_{e\cdot f} = d_e \cup d_f \cup \{(t, 0) \mid t \in \rho\}$, i.e., we connect each final state of A_e to the initial state of A_f and assign the distance 0 to these connecting transitions.
- $e^\star$: We construct the distance automaton $A_e = (Q_e, \Sigma, \rho_e, q_{0,e}, F_e, d_e)$. Then, $A_{e^\star} = (Q_e, \Sigma, \rho_e \cup \rho, q_{0,e}, F_e \cup \{q_{0,e}\}, d_{e^\star})$, where $\rho = \{(q, \varepsilon, q_{0,e}) \mid q \in F_e\}$ and $d_{e^\star} = d_e \cup \{((q, \varepsilon, p), 1) \mid (q, \varepsilon, p) \in \rho\}$, i.e., we connect each final state of A_e to the initial states of A_e and assign the corresponding transitions the distance 1.

If the resulting distance automaton $A_{M'}$ is limited, then there exists a finite subset $F \subseteq M'$ such that $\varphi(F) = \varphi(M')$. This implies that **(1)** holds.

So, given M' and R together with all languages in the domain of φ as regular expressions, basiccheck(R, M', φ) in Algorithm 2 first builds $A_{M'}$ (Line 1) and checks its limitedness (Line 2), amounting to condition **(1)**. For condition **(2)**, basiccheck verifies that $\varphi(M')$ and R are equivalent (Line 3) and returns **true** if both checks succeed.

Algorithm 3: enumerate(R, K, φ)

input : regular languages $R \subseteq \Sigma^\star$, $K \subseteq \Delta^\star$, and
regular language substitution φ with $\varphi(\delta) \subseteq \Sigma^\star$ for all $\delta \in \Delta$

yields : $L \in \mathsf{rep}(M, \mathsf{minlen}(R), \varphi)$ for $M = M_\varphi(R) \cap K$

1 $M := M_\varphi(R) \cap K$;

2 **for** $L \in \mathsf{unionfreedecomp}(M)$ **do** $\mathsf{unfold}(L, \varphi, \mathsf{minlen}(R))$;

Algorithm 4: unfold(L, φ, B)

input : union-free regular language $L \subseteq \Delta^\star$, written as
$L = N_1 S_1^\star N_2 \dots N_m S_m^\star N_{m+1} \subseteq \Delta^\star$ with $N_i \in \Delta^\star$ and union-free $S_h \subseteq \Delta^\star$,
regular language substitution φ with $\varphi(\delta) \subseteq \Sigma^\star$ for all $\delta \in \Delta$, and bound B

yields : $L' \in \mathsf{rep}(L, B, \varphi)$

if $\forall S_h \forall w \in S_h \ : \ \varepsilon \in \varphi(w)$ **then yield** L;

else

  fix S_h arbitrarily with $\exists w \in S_h \ : \ \varepsilon \notin \varphi(w)$;

  $E := S_h \cap \Delta_\varepsilon^\star$; // $\Delta_\varepsilon = \{\delta \in \Delta \mid \varepsilon \in \varphi(\delta)\}$

  $L_0 := N_1 S_1^\star N_2 \dots \ N_h E^\star N_{h+1} \dots N_m S_m^\star N_{m+1}$;

  $\mathsf{unfold}(L_0, \varphi, B)$;

  // $L_p := N_1 S_1^\star N_2 \dots \ N_h E^\star \bar{E}_p S_h^\star N_{h+1} \dots N_m S_m^\star N_{m+1}$ (see text)

  for $p \in \mathsf{critical}(S_h)$ *with* $\mathsf{minlen}(\varphi(L_p)) \leq B$ **do** $\mathsf{unfold}(L_p, \varphi, B)$;

▶ **Lemma 15** (basiccheck(R, M', φ) runs in PSPACE). *basiccheck(R, M', φ) runs in* PSPACE, *which is optimal up to the assumption that* PSPACE *does not collapse with a lower class, as it solves a* PSPACE*-complete problem.*

4.3 Implementing enumerate(K, R, φ)

Our enumeration algorithm must produce the languages $\mathsf{rep}(M, B, \varphi)$, guaranteeing that all $M' \in \mathsf{rep}(M, B, \varphi)$ have 1-word summaries, and that $M[B, \varphi] \subseteq \bigcup_{M' \in \mathsf{rep}(M,B,\varphi)} M' \subseteq M$ holds (as specified by Lemma 11). To this end, we rely on a sufficient condition for the existence of 1-word summaries. First we show this condition with Proposition 16, before turning to the enumeration algorithm itself.

▶ **Proposition 16** (Sufficient Condition for 1-Word Summaries). *Let L be a union-free language over Δ, given as $L = N_1 S_1^\star N_2 \dots N_m S_m^\star N_{m+1}$, with words $N_h \in \Delta^\star$ and union-free languages $S_h \subseteq \Delta^\star$. If $\varepsilon \in \varphi(w)$ for all $w \in S_h$ and all S_h, then L has 1-word summaries.*

We are ready to design our enumeration algorithm, shown in Algorithm 3, and its recursive subprocedure in Algorithm 4. Both algorithms do not return a result but yield their result as an enumeration: Upon invocation, both algorithms run through a sequence of **yield** statements, each time appending the argument of **yield** to the enumerated sequence. Thus, the algorithm never stores the entire sequence but only the stack of the invoked procedures.

Initializing the recursive enumeration, Algorithm 3 obtains the maximum rewriting $M := M_\varphi(R) \cap K$ of R (Line 1) and iterates over the languages L in the union-free decomposition of M (Line 2) to call for each L the recursive procedure unfold, shown in Algorithm 4. In turn, Algorithm 4 takes a union free language $L = N_1 S_1^\star N_2 \dots N_m S_m^\star N_{m+1}$ and a bound B to unfold the Kleene-star expressions of L until the precondition of Proposition 16 is satisfied or $\mathsf{minlen}(\varphi(L)) > B$.

More specifically, unfold exploits a rewriting, based on the following terms: Given a union free language S_h, let $E = S_h \cap \Delta_\varepsilon^\star$ with $\Delta_\varepsilon = \{\delta \in \Delta \mid \varepsilon \in \varphi(\delta)\}$ denote all words w in S_h with $\varepsilon \in \varphi(w)$ and let $\bar{E} = S_h \setminus E$. Since $\bar{E}$ is in general not union free, we need to split $\bar{E}$ further. To this end, we define $\mathsf{ufs}(S_h, p)$ recursively for an integer sequence $p = \langle p_H \mid p_T \rangle$ with head element p_H and tail sequence p_T. Intuitively, a sequence p identifies a subexpression in S_h by recursively selecting a nested Kleene star expression; $\mathsf{ufs}(S_h, p)$ unfolds S_h such that this selected expression is instantiated at least once. Formally, for $S_h = \alpha_1 \beta_1^\star \alpha_2 \dots \alpha_n \beta_n^\star \alpha_{n+1}$ we set $\mathsf{ufs}(S_h, \varepsilon) = S_h$ and $\mathsf{ufs}(S_h, p) = \alpha_1 \dots \alpha_{p_H} \beta_{p_H}^\star \mathsf{ufs}(\beta_{p_H}, p_T) \beta_{p_H}^\star \alpha_{p_H+1} \dots \alpha_{n+1}$. Consider $S_h = A^\star (B^\star C^\star)^\star D^\star$ (with all $\alpha_i = \varepsilon$ for brevity), then we obtain

$$\begin{array}{lclllllll}
\mathsf{ufs}(S_h, \langle 2,1 \rangle) & = & A^\star & (B^\star C^\star)^\star & & \mathsf{ufs}(B^\star C^\star, \langle 1 \rangle) & & & (B^\star C^\star)^\star \quad D^\star \\
& = & A^\star & (B^\star C^\star)^\star & (B^\star & \mathsf{ufs}(B, \varepsilon) & B^\star & C^\star) & (B^\star C^\star)^\star \quad D^\star \\
& = & A^\star & (B^\star C^\star)^\star & (B^\star & (B) & B^\star & C^\star) & (B^\star C^\star)^\star \quad D^\star
\end{array}$$

instantiating B at position $\langle 2, 1 \rangle$ at least once. Let $\mathsf{critical}(S_h)$ be integer sequences which identify a subexpression of S_h which directly contain a symbol δ with $\varepsilon \notin \varphi(\delta)$ (and not only via another Kleene-star expression). Then, we write $\bar{E} = \bigcup_{p \in \mathsf{critical}(S_h)} \bar{E}_p$, with $\bar{E}_p = \mathsf{ufs}(S_h, p)$. This discussion leads to the following rewriting:

▶ **Proposition 17** (Rewriting for 1-Word Summaries). *For every union free language $S_h^\star$, we have $S_h^\star = E^\star \cup \bigcup_{p \in \mathsf{critical}(S_h)} E^\star \bar{E}_p S_h^\star$. All languages in the rewriting, i.e., $E^\star$ and $E^\star \bar{E}_p S_h^\star$, are union free, $E^\star$ has 1-word summaries, and $\mathsf{minlen}(S_h^\star) < \mathsf{minlen}(E^\star \bar{E}_p S_h^\star)$ holds for all $p \in \mathsf{critical}(S_h)$.*

If L already satisfies the precondition imposed by Proposition 16, Algorithm 4 **yield**-s L and terminates (Line 1). Otherwise, it fixes an arbitrary S_h violating this precondition and rewrites L recursively with Proposition 17 (Lines 3-7). **(1)** *Termination:* In each recursive call, unfold either eliminates in L_0 an occurrence of a subexpression S_h violating the precondition of Proposition 16 (Line 6), or increases the minimum length in L_p, eventually running into the upper bound B (Line 7). **(2)** *Correctness:* Setting $B = \infty$, unfold **yield**-s a possibly infinite sequence of union free languages which have 1-word summaries such that their union equals the original language L: As the generation of these languages is based on the equality of Proposition 17 each rewriting step is sound and complete, leading to an infinite recursion tree whose leaves **yield** the languages in the sequence. The upper bound on minimum length only cuts off languages L_p producing words of minimum length beyond B, i.e., $L_p \cap L[B, \varphi] = \emptyset$, and in consequence, it is safe to drop L_p, since we only need to construct $\mathsf{rep}(L, B, \varphi)$ with $\mathsf{rep}(L, B, \varphi) \supseteq L[B, \varphi]$.

4.4 Upper Bound of the Complexity

The proof of Theorem 13 is based on the size of the maximum rewriting $M = M_\varphi(R) \cap K$ of $||K|| 2^{2^{(||R||+||\varphi||)^l}}$ for some constant l, shown in [7], and unfold's complexity: In Proposition 18, we show an upper bound on the space complexity of unfold, leading to the complexity of enumerate in Lemma 19 and the desired proof of Theorem 13.

▶ **Proposition 18** ($\mathsf{unfold}(L, \varphi, B)$ runs in $\mathrm{DSPACE}(B^2 ||L||^4 + ||\varphi||)$).

▶ **Lemma 19** ($\mathsf{enumerate}(R, K, \varphi)$ runs in $\mathrm{DSPACE}\left(||K||^4 2^{2^{(||R||+||\varphi||)^k}}\right)$).

Proof of Theorem 13. The enumeration runs in $\mathrm{DSPACE}\left(||K||^4 2^{2^{(||R||+||\varphi||)^k}}\right)$, producing expressions for basiccheck at most of the same size (Lemma 19). Since basiccheck is in PSPACE (Lemma 15), we obtain the overall complexity $\mathrm{DSPACE}\left(||K||^r 2^{2^{(||R||+||\varphi||)^s}}\right) \subseteq \mathrm{2EXPSPACE}$ for some constants r and s. ◀

4.5 Differences to Afonin and Khazova [1]

Afonin and Khazova show that the membership problem is decidable. In determining an upper bound for the complexity of membership problem, we had to expand their approach significantly: In general, we follow a top-down approach to describe the overall algorithm, whereas Afonin and Khazova go bottom-up, focusing on the building blocks enabling the decision procedure. More specifically, basiccheck is described in [1], while enumerate is omitted, as [1] deals with decidability only, deeming the bound on the enumeration size irrelevant. Hence Algorithms 3 and 4 are new, as well as the construction in Section 4.3, leading to Proposition 17. Based on the new algorithms, we contribute Theorem 13, together with Proposition 18, and Lemma 19. Moreover, in [1], the overall algorithm and the proof for Theorem 12 are only described in a brief paragraph. Finally, Section 4.1, albeit technically not new, provides a much more conceptual and hopefully accessible description of the algorithm.

5 Conclusion

Motivated by applications in test case specifications with FQL, we have studied general and finite RSRLs. While we showed that general RSRLs are not closed under most common operators, *finite* RSRLs are closed under all operators except Kleene stars and complementation (Theorem 4). This shows that our restriction to Kleene star free and hence finite RSRLs in FQL results in a natural framework with good closure properties. Likewise, the proven PSPACE-completeness results for Kleene star free RSRLs provide a starting point to develop practical reasoning procedures for Kleene star free RSRLs and FQL. Experience with LTL model checking shows that PSPACE-completeness often leads to algorithms which are feasible in practice. In contrast, for general and possibly infinite RSRLs, we have described a 2EXPSPACE membership checking algorithm – leaving the question for matching lower bounds open. Nevertheless, reasoning on general RSRLs seems to be rather infeasible.

Last but not least, RSRLs give rise to new and interesting research questions, for instance the decidability of inclusion and equivalence for general RSRLs, and the closure properties left open in this paper. In our future work, we want to generalize RSRLs to other base formalisms. For example, we want φ to substitute symbols by context-free expressions, thus enabling FQL test patterns to recognize e.g. matching of parentheses or emptiness of a stack.

Acknowledgements. This work received funding in part by the Austrian National Research Network S11403-N23 (RiSE) of the Austrian Science Fund (FWF), by the Vienna Science and Technology Fund (WWTF) grant PROSEED, and by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement DIADEM no. 246858.

References

1 S. Afonin and E. Hazova. Membership and Finiteness Problems for Rational Sets of Regular Languages. In *DLT*, pages 88–99, 2005.
2 G. A. Agasandyan. Variable-Structure Automata. *Soviet Physics Doklady*, 1967.
3 P. Barceló, D. Figueira, and L. Libkin. Graph Logics with Rational Relations and the Generalized Intersection Problem. In *LICS*, pages 115–124, 2012.
4 P. Barceló, J. L. Reutter, and L. Libkin. Parameterized Regular Expressions and their Languages. *Theor. Comput. Sci.*, 474:21–45, 2013.
5 J. Berstel. *Transductions and Context-Free Languages*. Teubner Studienbücher, Stuttgart, 1979.

6 D. Beyer, A. Holzer, M. Tautschnig, and H. Veith. Information Reuse for Multi-goal Reachability Analyses. In *ESOP*, pages 472–491, 2013.
7 D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Rewriting of Regular Expressions and Regular Path Queries. *JCSS*, 64:443–465, 2002.
8 T. Colcombet and L. Daviaud. Approximate Comparison of Distance Automata. In *STACS*, pages 574–585, 2013.
9 S. Eilenberg and M. P. Schützenberger. Rational Sets in Commutative Monoids. *J. Algebra*, 13:173–191, 1969.
10 K. Hashiguchi. Limitedness Theorem on Finite Automata with Distance Functions. *J. Comput. Syst. Sci.*, 24(2):233–244, 1982.
11 K. Hashiguchi. Representation Theorems on Regular Languages. *J. Comput. Syst. Sci.*, 27(1):101–115, 1983.
12 A. Holzer, V. Januzaj, S. Kugele, B. Langer, C. Schallhart, M. Tautschnig, and H. Veith. Seamless Testing for Models and Code. In *FASE'11*, pages 278–293, 2011.
13 A. Holzer, D. Kroening, C. Schallhart, M. Tautschnig, and H. Veith. Proving Reachability using FShell (Competition Contribution). In *TACAS*, pages 538–541, 2012.
14 A. Holzer, C. Schallhart, M. Tautschnig, and H. Veith. How did You Specify Your Test Suite? In *ASE*, pages 407–416, 2010.
15 A. Holzer, M. Tautschnig, C. Schallhart, and H. Veith. FSHELL: Systematic Test Case Generation for Dynamic Analysis and Measurement. In *CAV*, pages 209–213, 2008.
16 A. Holzer, M. Tautschnig, C. Schallhart, and H. Veith. Query-Driven Program Testing. In *VMCAI*, pages 151–166, 2009.
17 A. Holzer, M. Tautschnig, C. Schallhart, and H. Veith. An Introduction to Test Specification in FQL. In *HVC*, pages 9–22, 2010.
18 Andreas Holzer, Christian Schallhart, Michael Tautschnig, and Helmut Veith. On the Structure and Complexity of Rational Sets of Regular Languages. *CoRR*, abs/1305.6074, 2013.
19 J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
20 D. Kirsten. Distance Desert Automata and the Star Height One Problem. In *FoSSaCS*, pages 257–272, 2004.
21 D. Kirsten. Distance Desert Automata and the Star Height Problem. *ITA*, 39(3):455–509, 2005.
22 S. C. Kleene. Representation of Events in Nerve Nets and Finite Automata. *RAND Corporation Memorandum*, 1951.
23 D. Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Intl. Journal of Algebra and Computation*, 4(3):405–425, 1994.
24 O. Kupferman and Y. Lustig. Lattice Automata. In *VMCAI*, pages 199–213. 2007.
25 H. Leung and V. Podolskiy. The limitedness problem on distance automata: Hashiguchi's method revisited. *TCS*, 310(1–3):147–158, 2004.
26 J.-E. Pin. Mathematical Foundations of Automata Theory. Lecture Notes, 2011.
27 RTCA DO-178B. Software Considerations in Airborne Systems and Equipment Certification, 1992.
28 A. Salomaa. On Finite Automata with a Time-Variant Structure. *Information and Control*, 13(2):85 – 98, 1968.