Theoretical Computer Science 605 (2015) 62–79

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Closure properties and complexity of rational sets of regular languages

Andreas Holzer [a,*,1], Christian Schallhart [b,2], Michael Tautschnig [c], Helmut Veith [a]

[a] Institut für Informationssysteme 184/4, Vienna University of Technology, Favoritenstrasse 9-11, 1040 Wien, Austria
[b] Department of Computer Science, Oxford University, Wolfson Building, Oxford OX1 3QD, UK
[c] School of Electronic Engineering and Computer Science, Queen Mary University of London, Mile End Road, London E1 4NS, UK

ARTICLE INFO

*Article history:*
Received 15 August 2014
Received in revised form 25 August 2015
Accepted 27 August 2015
Available online 3 September 2015
Communicated by D. Perrin

*Keywords:*
Rational sets
Regular languages
Test specification in FQL
Closure properties
Decision problems

ABSTRACT

The test specification language FQL describes relevant test goals as regular expressions over program locations, such that each matching test case has an execution path matching this expression. To specify not only test goals but entire suites, FQL describes families of related test goals by regular expressions over extended alphabets: Herein, each symbol corresponds to a regular expression over program locations, and thus, a word in an FQL expression corresponds to a regular expression describing a single test goal. In this paper we provide a systematic foundation for FQL test specifications, which are in fact rational sets of regular languages (RSRLs). To address practically relevant problems like query optimization, we tackle open questions about RSRLs: We settle 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 deciding whether a regular language is a member of a general or star-free RSRL.



## 1. Introduction

Despite the success of model checking and theorem proving, software testing retains a dominant role in industrial practice. In fact, state-of-the-art development guidelines such as the avionic standard DO-178B [1] 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 [2–7], we have addressed this situation and introduced the Fshell Query Language (FQL). FQL allows to specify and tailor coverage criteria, and has been implemented in Fshell [6] and CPA/Tiger [7], tools to generate matching test suites for ANSI C programs. At the semantic core of FQL, test goals are described as regular expressions the alphabet of which are the edges of the program's 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

* Corresponding author.
*E-mail addresses:* aholzer@cs.toronto.edu (A. Holzer), christian.schallhart@cs.ox.ac.uk (C. Schallhart), michael.tautschnig@qmul.ac.uk (M. Tautschnig), veith@forsyte.at (H. Veith).
[1] Now at University of Toronto.
[2] Now at Google London.

http://dx.doi.org/10.1016/j.tcs.2015.08.035
0304-3975/

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, delimited by 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 yields a regular expression, e.g., $\Sigma^\star\ a\ \Sigma^\star$, that will in turn serve as test goal. Following [8], we call such languages *rational sets of regular languages (RSRLs)*.

The goal of this paper is to initiate a systematic theoretical study of RSRLs, 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 [7].

*Contributions and organization* Our results on RSRLs encompass closure properties for set theoretic operations and variants thereof as well as complexity results on decision problems, justifying the design of FQL, as detailed in Section 3. Our paper is organized as follows:

- We formally introduce RSRLs in Section 2, then, sketch FQL and clarify the questions leading to the presented research in Section 3, and, then, survey related work in Section 4.
- *Closure properties (Section 5).* We consider general and finite RSRLs together with the operators Kleene star, product, complement, union, intersection, set difference, and symmetric difference. We also consider the case of finite RSRLs with a fixed language substitution $\varphi$, as this case is of particular interest for testing applications (cf. Section 3).
- *Complexity results (Section 6).* We discuss the complexity to decide equivalence, inclusion, and membership for Kleene-star free RSRLs. To prove an upper bound on the complexity of the membership problem for general RSRLs, we expand the decidability proof in [8] and give a first complete and explicit algorithm for the problem.
- *Justification of FQL Design (Section 7).* We conclude in discussing how our results reflect back on the design of FQL. In particular, our theoretical results confirm the decision to allow only Kleene-star free RSRLs in FQL.

A preliminary version of this work has been published in [9]; however, this older version was lacking almost all proofs. In contrast, the current paper contains full proof details on all results presented in Sections 5 and 6.

## 2. Rational sets of regular languages

One of the most fundamental concepts in this paper are regular language substitutions, which map symbols in one alphabet to regular languages over another alphabet.

**Definition 1** *(Regular language substitution).* Given a finite alphabet $\Sigma$, let $\textsc{Reg}(\Sigma)$ denote the set of regular languages over $\Sigma$. Then, given alphabets $\Delta$ and $\Sigma$, a *regular language substitution* $\varphi : \Delta \to \textsc{Reg}(\Sigma)$ maps each symbol $\delta \in \Delta$ to a regular language $\varphi(\delta) \in \textsc{Reg}(\Sigma)$. We extend $\varphi$ 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 $L \subseteq \Delta^+$.

Please note that the extension of a regular language substitution to words yields regular languages again. Before we define rational sets of regular languages, we define rational sets of a monoid in general.

**Definition 2** *(Rational sets of a monoid).* 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 [10,11].

**Definition 3** *(Rational Sets of Regular Languages, RSRLs).* (See [8].) Given a finite alphabet $\Sigma$, the *rational sets of regular languages* are the rational sets over the monoid $(\textsc{Reg}(\Sigma), \cdot, \{\varepsilon\})$, where $\varepsilon$ denotes the empty word.

**Definition 4** *(Representation of RSRLs).* (See [8].) We represent an RSRL $\mathcal{R}$ as a tuple $(K, \varphi)$, where $K \subseteq \Delta^+$ is a regular language over a finite alphabet $\Delta$, and $\varphi$ is a regular language substitution $\varphi : \Delta \to \textsc{Reg}(\Sigma)$, such that $\mathcal{R} = \{\varphi(w) \mid w \in K\}$. We say that the RSRL $\mathcal{R}$ is *Kleene-star free*, if $\mathcal{R}$ is finite. That means that there exists a tuple $(K, \varphi)$ such that $(K, \varphi) = \mathcal{R}$ where $K$ is finite (and hence Kleene-star free).

Note that we require a regular language $K \subseteq \Delta^+$ to exclude the empty word as the extension of regular language substitutions from symbols to words is not defined for the empty word. In the next two lemmas, we show that each RSRL can be represented by a tuple $(K, \varphi)$ and that each tuple $(K, \varphi)$ represents an RSRL.

**Lemma 5** *(Representation of RSRLs, part 1). Any RSRL $\mathcal{R}$ can be represented as a tuple $(K, \varphi)$.*