$\underbrace{\textbf{Formalizing}}_{\text{in Isabelle HOL}}$ Symbolic Decision Procedures

for

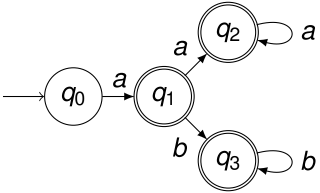
Regular Languages

Dmitriy Traytel

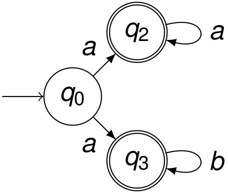
Technische Universität München

Representations of Regular Languages

DFA



NFA

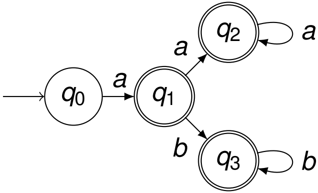


$a(a^* + b^*)$	$\neg(b(\neg\varnothing) + a(\neg(a^*) \cap \neg(b^*)))$	$\exists x.\, (\forall y.\, x \leq y) \wedge x \in X$
RE	ERE	WMSO

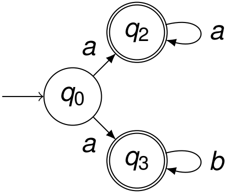
Representations of Regular Languages

explicit

DFA



NFA



$a(a^* + b^*)$	$\neg(b(\neg\varnothing) + a(\neg(a^*) \cap \neg(b^*)))$	$\exists x. (\forall y. x \leq y) \wedge x \in X$
RE	ERE	WMSO

symbolic

My Thesis

Theme equivalence problem of symbolic representations

$\mathscr{L}(a^*) = \mathscr{L}(aa^* + \varepsilon)$? $\qquad$ $\mathscr{L}(\exists X.\, \forall y.\, y \in X) = \mathscr{L}\,\mathsf{F}$?

My Thesis

Theme equivalence problem of symbolic representations

$\mathscr{L}\ (a^*) = \mathscr{L}\ (aa^* + \varepsilon)$? $\mathscr{L}\ (\exists X.\ \forall y.\ y \in X) = \mathscr{L}$ F?

Setting in a proof assistant

My Thesis

Theme equivalence problem of symbolic representations

$\mathscr{L}\ (a^*) = \mathscr{L}\ (aa^* + \varepsilon)$? $\mathscr{L}\ (\exists X.\ \forall y.\ y \in X) = \mathscr{L}$ F?

Setting in a proof assistant

Catch without resorting to explicit representations

unlike traditional methods

Thompson, McNaughton–Yamada, Glushkov, Büchi, Elgot, Trakhtenbrot

My Thesis

Theme equivalence problem of symbolic representations

$\mathscr{L}\,(a^*) = \mathscr{L}\,(aa^* + \varepsilon)$? $\qquad$ $\mathscr{L}\,(\exists X.\, \forall y.\, y \in X) = \mathscr{L}\,\mathsf{F}$?

Setting in a proof assistant

Catch without resorting to explicit representations

unlike traditional methods

Thompson, McNaughton–Yamada, Glushkov, Büchi, Elgot, Trakhtenbrot

$\Rightarrow$ use variations of Brzozowski derivatives

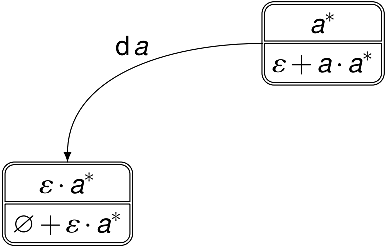
$a^* \stackrel{?}{=} \varepsilon + a \cdot a^*$ for $\Sigma = \{a, b\}$

Brzozowski 1964
Ginzburg 1967

a^*
$\varepsilon + a \cdot a^*$

$a^* \stackrel{?}{\equiv} \varepsilon + a \cdot a^*$ for $\Sigma = \{a, b\}$

Brzozowski 1964
Ginzburg 1967



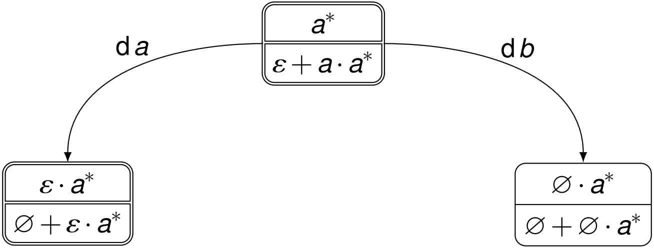
Brzozowski derivative

$$\mathrm{d} : \text{letter} \to \text{regex} \to \text{regex}$$

$$\mathscr{L}(\mathrm{d}\, a\, r) = \{w \mid aw \in \mathscr{L}(r)\}$$

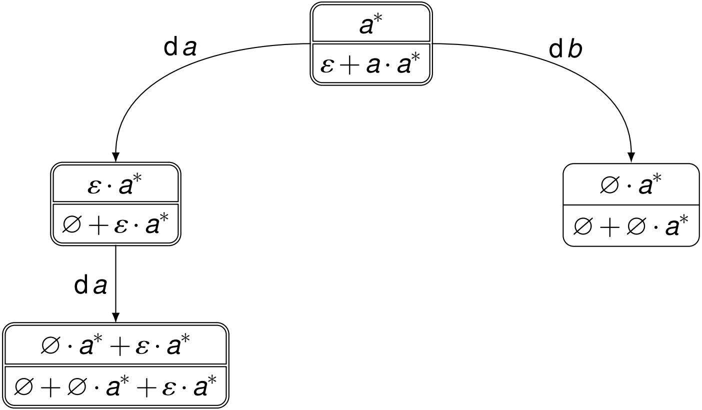
$a^* \stackrel{?}{\equiv} \varepsilon + a \cdot a^*$ for $\Sigma = \{a, b\}$

Brzozowski 1964
Ginzburg 1967



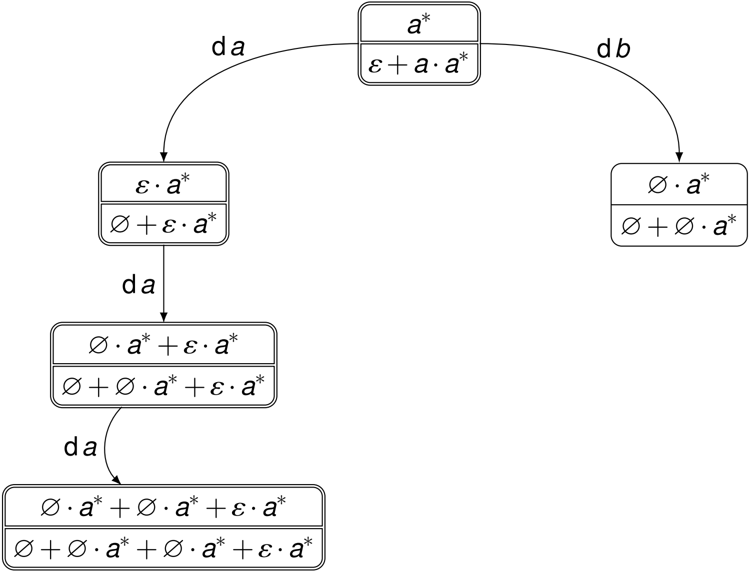
$a^* \stackrel{?}{\equiv} \varepsilon + a \cdot a^*$ for $\Sigma = \{a, b\}$

Brzozowski 1964
Ginzburg 1967



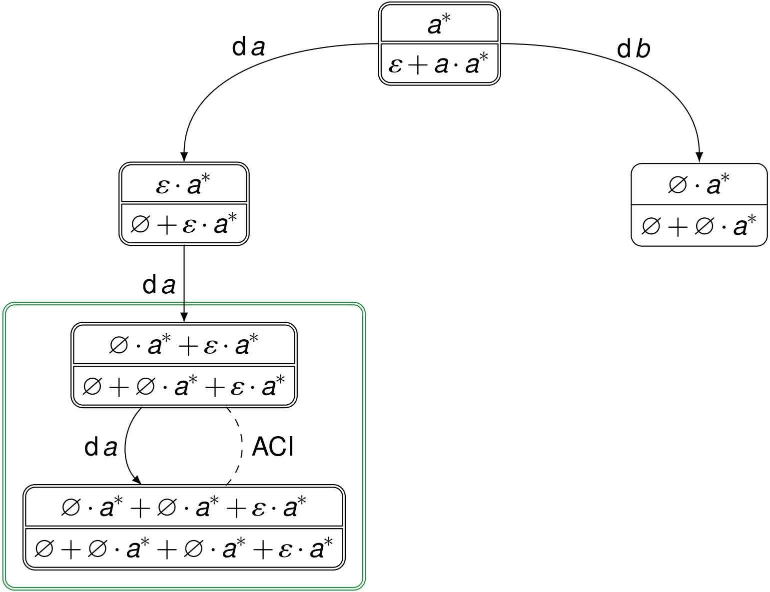
$a^* \stackrel{?}{\equiv} \varepsilon + a \cdot a^*$ for $\Sigma = \{a, b\}$

Brzozowski 1964
Ginzburg 1967



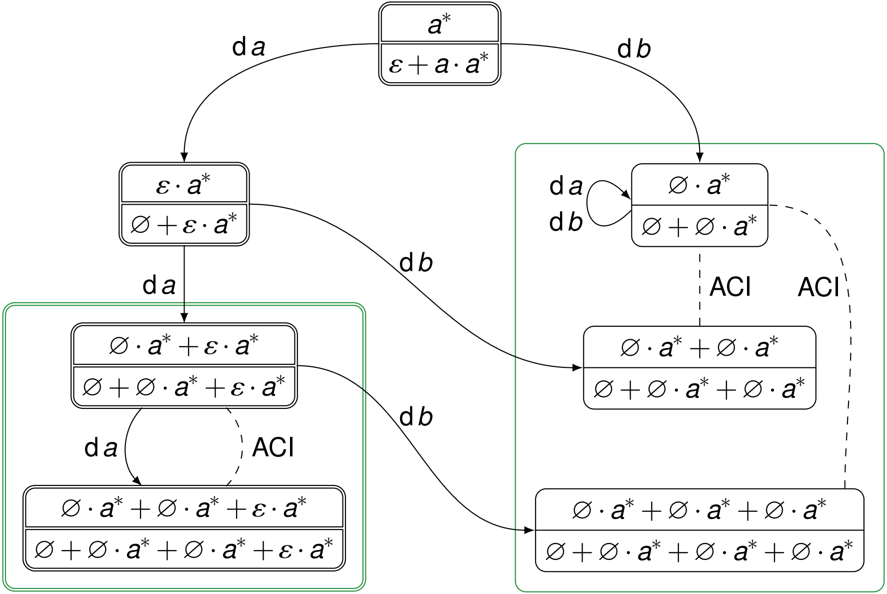
$a^* \stackrel{?}{\equiv} \varepsilon + a \cdot a^*$ for $\Sigma = \{a, b\}$

Brzozowski 1964
Ginzburg 1967



$a^* \stackrel{?}{\equiv} \varepsilon + a \cdot a^*$ for $\Sigma = \{a, b\}$

Brzozowski 1964
Ginzburg 1967



Derivatives in Literature

Theoretical groundwork

JACM 1964	Brzozowski
JACM 1967	Ginzburg
TCS 1996	Antimirov
CONCUR 1998	Rutten

Derivatives in Literature

Theoretical groundwork

JACM 1964	Brzozowski
JACM 1967	Ginzburg
TCS 1996	Antimirov
CONCUR 1998	Rutten

Programming Lanugages community

JFP 2009	Owens, Reppy, and Turon
ICFP 2010	Fischer, Huch, and Wilke
ICFP 2010	Danielsson
ICFP 2011	Might, Darais, and Spiewak
ICFP 2013	T. and Nipkow
POPL 2015	Pous
POPL 2015	Foster, Kozen, Milano, Silva, and Thompson
JFP 2015	T. and Nipkow

Interactive Theorem Proving community

JAR 2011	Krauss and Nipkow
CPP 2011	Coquand and Siles
ITP 2012	Asperti
RAMiCS 2012	Moreira, Pereira, and de Sousa
ITP 2013	Pous
ITP 2014	Nipkow and T.

Logic in Computer Science community

ICALP 2015	Kozen, Mamouras, Petrişan, and Silva
CSL 2015	T.

Derivatives in Literature

Theoretical groundwork

JACM 1964	Brzozowski
JACM 1967	Ginzburg
TCS 1996	Antimirov
CONCUR 1998	Rutten

Programming Lanugages community

JFP 2009	Owens, Reppy, and Turon
ICFP 2010	Fischer, Huch, and Wilke
ICFP 2010	Danielsson
ICFP 2011	Might, Darais, and Spiewak
ICFP 2013	T. and Nipkow
POPL 2015	Pous
POPL 2015	Foster, Kozen, Milano, Silva, and Thompson
JFP 2015	T. and Nipkow

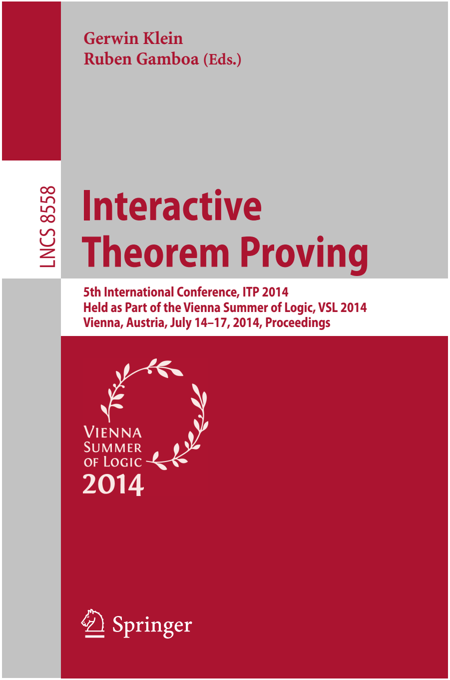
Interactive Theorem Proving community

JAR 2011	Krauss and Nipkow
CPP 2011	Coquand and Siles
ITP 2012	Asperti
RAMiCS 2012	Moreira, Pereira, and de Sousa
ITP 2013	Pous
ITP 2014	Nipkow and T.

Logic in Computer Science community

ICALP 2015	Kozen, Mamouras, Petrişan, and Silva
CSL 2015	T.

this thesis



Unified Decision Procedures for Regular Expression Equivalence

Tobias Nipkow and Dmitriy Traytel

Fakultät für Informatik, Technische Universität München, Germany

Abstract. We formalize a unified framework for verified decision procedures for regular expression equivalence. Five recently published formalizations of such decision procedures (three based on derivatives, two on marked regular expressions) can be obtained as instances of the framework. We discover that the two approaches based on marked regular expressions, which were previously thought to be the same, are different, and we prove a quotient relation between the automata produced by them. The common framework makes it possible to compare the performance of the different decision procedures in a meaningful way.

1 Introduction

Equivalence of regular expressions is a perennial topic in computer science. Recently it has spawned a number of formalized and verified decision procedures for this task in interactive theorem provers [3, 6, 10, 19, 21]. Except for the formalization by Braibant and Pous [6], all these decision procedures operate directly on variations of regular expressions. Although they (implicitly) build automata, the states of the automata are labeled with regular expressions, and there is no global transition table but the next-state function is computable from the regular expressions. The motivation for working with regular expressions is simplicity: regular expressions are a free datatype which proof assistants and their users love because it means induction, recursion and equational reasoning—the core competence of proof assistants and functional programming languages. Yet all these decision procedures based on regular expressions look very different. Of course, the next-state functions all differ, but so do the actual decision procedures and their correctness, completeness and termination proofs. The contributions of our paper are the following:

– A unified framework (Sect. 3) that we instantiate with all the above approaches (Sects. 4 and 5). The framework is a simple reflexive transitive closure computation that enumerates the states of a product automaton.
– Proofs of correctness, completeness and termination that are performed once and for all for the framework based on a few properties of the next-state function.
– A new perspective on partial derivatives that recasts them as Brzozowski derivatives followed by some rewriting (Sect. 4).
– The discovery that Asperti's algorithm is not the one by McNaughton-Yamada [20], as stated by Asperti [3], but a dual construction which apparently had not been considered in the literature and which produces smaller automata (Sect. 5).
– An empirical comparison of the performance of the different approaches (Sect. 6).

The discussion of related work is distributed over the relevant sections of the paper.

G. Klein and R. Gamboa (Eds.): ITP 2014, LNAI 8558, pp. 450–466, 2014.

Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2010	Fischer, Huch, & Wilke	Mark after
JAR 2011	Krauss & Nipkow	Brzozowski
CPP 2011	Coquand & Siles	Brzozowski
RAMiCS 2012	Moreira, Pereira, & de Sousa	Antimirov
ITP 2012	Asperti	Mark before

Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2010	Fischer, Huch, & Wilke	Mark after
JAR 2011	Krauss	Brzozowski
CPP 2011		zowski
RAMiCS 2012	More	Antimirov
ITP 2012	Asperti	Mark before

$$\bullet\quad a\circ(a\circ^* + b\circ^*)$$

Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2010	Fischer, Huch, & Wilke	Mark after
JAR 2011	Krauss	Brzozowski
CPP 2011		zowski
RAMiCS 2012	More	Antimirov
ITP 2012	Asperti	Mark before

$$\circ \quad a{\bullet}(a{\circ}^{*} + b{\circ}^{*})$$

read *a*

Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2010	Fischer, Huch, & Wilke	Mark after
JAR 2011	Krauss	Brzozowski
CPP 2011		zowski
RAMiCS 2012	More	Antimirov
ITP 2012	Asperti	Mark before

$$\circ \quad a{\circ}(a{\bullet}^* + b{\circ}^*)$$

read *aa*

Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2010	Fischer, Huch, & Wilke	Mark after
JAR 2011	Krauss	Brzozowski
CPP 2011		zowski
RAMiCS 2012	Morei	Antimirov
ITP 2012	Asperti	Mark before

$$\circ \;\; a\circ(a\circ^* + b\circ^*)$$

read *aab*

Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2010	Fischer, Huch, & Wilke	Mark after
JAR 2011	Krauss & Nipkow	Brzozowski
CPP 2011	Coquand & Siles	Brzozowski
RAMiCS 2012	Moreira, Pereira, & de Sousa	Antimirov
ITP 2012	Asperti	Mark before

Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2010	Fischer, Huch, & Wilke	Mark after	
JAR 2011	Krauss & Nipkow	Brzozowski	
CPP 2011	Coquand & Siles	Brzozowski	
RAMiCS 201	a	Antimirov	
ITP 2012		Mark before	

$\mathrm{d} : \text{letter} \to \text{regex} \to \text{regex}$

$+$ ACI

Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

∂ : letter $\rightarrow$ regex $\rightarrow$ regex set

ICFP 2010		Mark after	
JAR 201		Brzozowski	
CPP 2011		Brzozowski	
RAMiCS 2012	Moreira, Pereira, & de Sousa	Antimirov	
ITP 2012	Asperti	Mark before	matita

Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2010	Fischer, Huch, & Wilke	Mark after
JAR 2011	Krauss & N	Brzozowski
CPP 2011	C	zozowski
RAMiCS 2012	Moreira, Per , & Sousa	Antimirov
ITP 2012	Asperti	Mark before

$$\bullet a(\circ a^* + \circ b^*)$$

Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2010	Fischer, Huch, & Wilke	Mark after
JAR 2011	Krauss & N	Brzozowski
CPP 2011	C	zozowski
RAMiCS 2012	Moreira, Pere, & Sousa	Antimirov
ITP 2012	Asperti	Mark before

$$\circ a(\bullet a^* + \bullet b^*)$$

read *a*

Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2010	Fischer, Huch, & Wilke	Mark after
JAR 2011	Krauss & Nipkow	Brzozowski
CPP 2011	C	zozowski
RAMiCS 2012	Moreira, Pere , & Sousa	Antimirov
ITP 2012	Asperti	Mark before

$$\circ a(\bullet a^* + \circ b^*)$$

read *aa*

Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2010	Fischer, Huch, & Wilke	Mark after
JAR 2011	Krauss & N	Brzozowski
CPP 2011	C	zozowski
RAMiCS 2012	Moreira, Per , & Sousa	Antimirov
ITP 2012	Asperti	Mark before

$$\circ a(\circ a^* + \circ b^*)$$

read *aab*

Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2

J

C

RA

ITP 2012

Our contribution

- Abstract bisimulation computation
- Insantiations with different derivatives
- $\sum \circ \partial a = \text{pnorm} \circ \mathrm{d}\, a$
- "Mark before" yields smaller bisimulations than "Mark after" (proof due H. Seidl)
- Empirical comparison

September 25–27, 2013
Boston, Massachusetts, USA

Association for Computing Machinery

Advancing Computing as a Science & Profession

ICFP'13

Proceedings of the 2013 ACM SIGPLAN
International Conference on Functional Programming

Sponsored by:

ACM SIGPLAN

Supported by:

Jane Street, FPComplete, Microsoft Research, NSF, ORACLE LABS, Standard Chartered, Credit Suisse, Erlang Solutions, Galois, Google, INRIA, Twitter, Northwestern University, IntelliFactory, & QuviQ

Verified Decision Procedures for MSO on Words Based on Derivatives of Regular Expressions

Dmitriy Traytel Tobias Nipkow
Technische Universität München, Germany
traytel@in.tum.de www.in.tum.de/~nipkow

Abstract

Monadic second-order logic on finite words (MSO) is a decidable yet expressive logic into which many decision problems can be encoded. Since MSO formulas correspond to regular languages, equivalence of MSO formulas can be reduced to the equivalence of some regular structures (e.g. automata). This paper presents a verified functional decision procedure for MSO formulas that is not based on automata but on regular expressions. Functional languages are ideally suited for this task: regular expressions are data types and functions on them are defined by pattern matching and recursion and are verified by structural induction.

Decision procedures for regular expression equivalence have been formalized before, usually based on Brzozowski derivatives. Yet, for a straightforward embedding of MSO formulas into regular expressions an extension of regular expressions with a projection operation is required. We prove total correctness and completeness of an equivalence checker for regular expressions extended in that way. We also define a language-preserving translation of formulas into regular expressions with respect to two different semantics of MSO. Our results have been formalized and verified in the theorem prover Isabelle. Using Isabelle's code generation facility, this yields purely functional, formally verified programs that decide equivalence of MSO formulas.

Categories and Subject Descriptors F.4.3 [*Mathematical Logic And Formal Languages*]: Formal Languages—Decision problems; F.3.1 [*Mathematical Logic And Formal Languages*]: Specifying and Verifying and Reasoning about Programs

General Terms Algorithms, Theory, Verification

Keywords MSO; WS1S; decision procedure; regular expressions; Brzozowski derivatives; interactive theorem proving; Isabelle

1. Introduction

Many decision procedures for logical theories are based on the famous logic-automaton connection. That is, they reduce the decision problem for some logical theory to a decidable question about some class of automata. Automata are usually implemented with the help of imperative data structures for efficiency reasons.

In functional languages, automata are not an ideal abstraction because they are graphs rather than trees. In contrast, regular expressions are perfect for functional languages and they are equally expressive. In fact, Brzozowski [8] showed how automata-based algorithms can be recast as recursive algebraic manipulations of regular expressions. His *derivatives* can be seen as a way of simulating automaton states with regular expressions and computing the next-state function symbolically.

Recently Brzozowski's derivatives were discovered by functional programmers and theorem provers. Owens *et al.* [23] realized that regular expressions and their derivatives fit perfectly with data types and recursive functions. Their paper explores regular expression matching based directly on regular expressions rather than automata. Fischer *et al.* [13] also explore regular expression matching, but by means of marked regular expressions rather than derivatives. Slightly later, the interactive theorem proving community woke up to the beauty of derivatives, too. This resulted in four papers about verified decision procedures for the equivalence of regular expressions based on derivatives and on marked regular expressions (see Related Work below). In one of these four papers, Coquand and Siles [10] state that "A more ambitious project will be to use this work for writing a decision procedure for WS1S", a monadic second-order logic. Our paper does just that (and more).

Monadic second-order logic on finite words (MSO) is a decidable yet expressive logic into which many decision problems can be encoded [26]. MSO allows only monadic predicates but quantification both over numbers and finite sets of numbers. Two closely related but subtly different semantics can be found in the literature. One of the two, WS1S—the Weak monadic Second-order logic of 1 Successor, is based on arithmetic. The other, M2L(Str) [16], is more closely related to formal languages. There seems to be some disagreement as to which semantics is the more appropriate one for verification purposes [3, 17]. Hence we cover both.

Essentially, MSO formulas describe regular languages. Therefore MSO formulas can be decided by translating them into automata. This is the basis of the highly successful MONA tool [12] for deciding WS1S. MONA's success is due to its (in practical terms) highly efficient implementation and to the ease with which very different verification problems can be encoded in monadic second-order logic, for example Presburger arithmetic and Hoare logic for pointer programs.

The contribution of this paper is the presentation of the first purely functional decision procedures for two interpretations of MSO based on derivatives of regular expressions. These decision procedures have been verified in Isabelle/HOL and we sketch their correctness proofs. We are not aware of any previous decision procedure for MSO based on regular expressions (as opposed to automata), let alone a verified program.

It is instructive to compare our decision procedure for WS1S with MONA. MONA is a highly tuned implementation using

ICFP '13, September 25–27, 2013, Boston, MA, USA.

ACM 978-1-4503-2326-0/13/09…$15.00.
http://dx.doi.org/10.1145/2500365.2500612

Verified Decision Procedures for MSO on Words Based on Derivatives of Regular Expressions

T. & Nipkow, ICFP 2013 & *JFP* 2015

$$\mathsf{T} \mid \mathsf{F} \mid x \in X \mid x < y \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x.\, \varphi \mid \exists X.\, \varphi$$

Verified Decision Procedures for MSO on Words Based on Derivatives of Regular Expressions

T. & Nipkow, ICFP 2013 & *JFP* 2015

$$\mathsf{T} \mid \mathsf{F} \mid x \in X \mid x < y \mid \varphi \vee \psi \mid \neg\varphi \mid \mathsf{FO}\ x \mid \exists X.\ \varphi$$

Verified Decision Procedures for MSO on Words Based on Derivatives of Regular Expressions

T. & Nipkow, ICFP 2013 & *JFP* 2015

$$\mathsf{T} \mid \mathsf{F} \mid x \in X \mid x < y \mid \varphi \vee \psi \mid \neg\varphi \mid \mathsf{FO}\ x \mid \exists\varphi$$

Verified Decision Procedures for MSO on Words Based on Derivatives of Regular Expressions

T. & Nipkow, ICFP 2013 & *JFP* 2015

$$
\begin{array}{lcl}
\mathrm{mkRE}(\mathrm{T}) & = & \neg\,\varnothing \\
\mathrm{mkRE}(\mathrm{F}) & = & \varnothing \\
\mathrm{mkRE}(x < y) & = & \neg\,\varnothing \cdot \mathrm{ANth}\ x\ \mathrm{T} \cdot \neg\,\varnothing \cdot \mathrm{ANth}\ y\ \mathrm{T} \cdot \neg\,\varnothing \\
\mathrm{mkRE}(x \in X) & = & \neg\,\varnothing \cdot \mathrm{ANth}_2\ x\ X \cdot \neg\,\varnothing \\
\mathrm{mkRE}(\mathrm{FO}\ x) & = & (\mathrm{ANth}\ x\ \mathrm{F})^* \cdot \mathrm{ANth}\ x\ \mathrm{T} \cdot (\mathrm{ANth}\ x\ \mathrm{F})^* \\
\mathrm{mkRE}(\varphi \vee \psi) & = & \mathrm{mkRE}(\varphi) + \mathrm{mkRE}(\psi) \\
\mathrm{mkRE}(\neg\varphi) & = & \neg\,\mathrm{mkRE}(\varphi) \\
\mathrm{mkRE}(\exists\varphi) & = & \Pi\ (\mathrm{mkRE}(\varphi))
\end{array}
$$

Verified Decision Procedures for MSO on Words Based on Derivatives of Regular Expressions

T. & Nipkow, ICFP 2013 & *JFP* 2015

$$
\begin{aligned}
\mathsf{mkRE}(\mathsf{T}) &= \neg\,\varnothing \\
\mathsf{mkRE}(\mathsf{F}) &= \varnothing \\
\mathsf{mkRE}(x < y) &= \neg\,\varnothing \cdot \mathsf{ANth}\ x\ \mathsf{T} \cdot \neg\,\varnothing \cdot \mathsf{ANth}\ y\ \mathsf{T} \cdot \neg\,\varnothing \\
\mathsf{mkRE}(x \in X) &= \neg\,\varnothing \cdot \mathsf{ANth}_2\ x\ X \cdot \neg\,\varnothing \\
\mathsf{mkRE}(\mathsf{FO}\ x) &= (\mathsf{ANth}\ x\ \mathsf{F})^* \cdot \mathsf{ANth}\ x\ \mathsf{T} \cdot (\mathsf{ANth}\ x\ \mathsf{F})^* \\
\mathsf{mkRE}(\varphi \vee \psi) &= \mathsf{mkRE}(\varphi) + \mathsf{mkRE}(\psi) \\
\mathsf{mkRE}(\neg\varphi) &= \neg\,\mathsf{mkRE}(\varphi) \\
\mathsf{mkRE}(\exists\varphi) &= \Pi\,(\mathsf{mkRE}(\varphi))
\end{aligned}
$$

preserves semantics

24th EACSL Annual Conference on Computer Science Logic

CSL 2015, September 7–10, 2015, Berlin, Germany

Edited by

Stephan Kreutzer

EUROPEAN ASSOCIATION FOR COMPUTER SCIENCE LOGIC

LIPIcs – Vol. 41 – CSL 2015 www.dagstuhl.de/lipics

A Coalgebraic Decision Procedure for WS1S

Dmitriy Traytel

Fakultät für Informatik, Technische Universität München, Germany
traytel@in.tum.de

Abstract

Weak monadic second-order logic of one successor (WS1S) is a simple and natural formalism to specify regular properties. WS1S is decidable, although the decision procedure's complexity is non-elementary. Typically, decision procedures for WS1S exploit the logic–automaton connection, i.e., they escape the simple and natural formalism by translating formulas into equally expressive regular structures such as finite automata, regular expressions, or games. In this work, we devise a coalgebraic decision procedure for WS1S that stays within the logical world by directly operating on formulas. The key operation is the derivative of a formula, modeled after Brzozowski's derivatives of regular expressions. The presented decision procedure has been formalized and proved correct in the interactive proof assistant Isabelle.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases WS1S, decision procedure, coalgebra, Brzozowski derivatives, Isabelle

Digital Object Identifier 10.4230/LIPIcs.CSL.2015.487

1 Introduction

In his seminal work [8], Büchi envisioned weak monadic second-order logic of one successor (WS1S) to become a "more conventional formalism [that] can be used in place of regular expressions [. . .] for formalizing conditions on the behavior of automata". This vision became truth – WS1S has been used to encode decision problems in hardware verification [3], program verification [22], network verification [4], synthesis [19], as well as many others.

WS1S is a logic that supports first-order quantification over natural numbers and second-order quantification over finite (therefore "Weak") sets of natural numbers, and beyond this has few additional special predicates, such as $<$ to compare first-order variables. Equivalence of WS1S formulas is decidable, although the complexity for deciding it is non-elementary [27]. Nevertheless, the MONA tool [20] shows that the daunting theoretical complexity can often be overcome in practice by employing a multitude of smart optimizations. Similarly to Büchi, MONA's manual [24] calls WS1S a "simple and natural notation" for regular languages.

Traditionally[1], decision procedures for WS1S do not try to benefit from the conventional, simple, and natural logical notation. Instead, by exploiting the logic–automaton connection, formulas are translated into finite automata which are then minimized. During the translation all the rich algebraic formula structure including binders and high-level constructs is lost. On the other hand, the subsequent minimization might have benefited from some simplifications on the formula level.

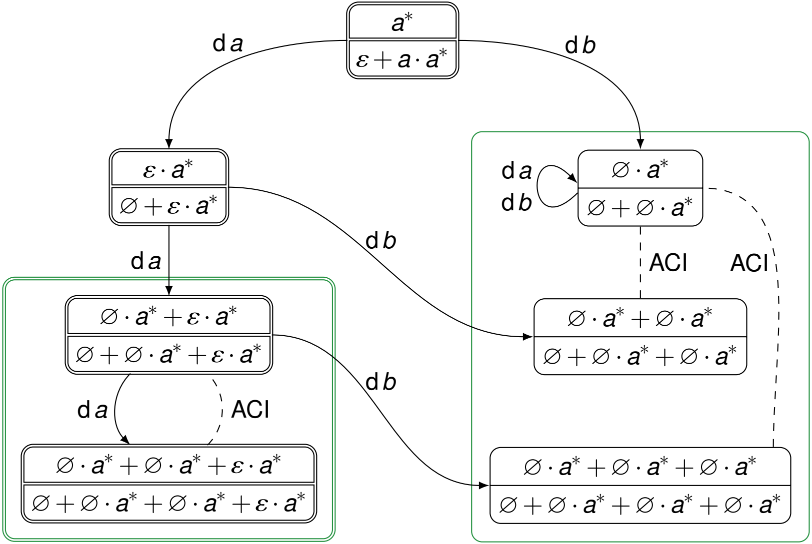
Concerning the algebraic structure, regular expressions are situated somewhere in between WS1S formulas and automata. In earlier work [40, 39], we propose a semantics-preserving translation of WS1S formulas into regular expressions. Thereby, equivalence of formulas is

[1] The only notable exception, we are aware of, is the decision procedure implemented in the Toss tool [17] (Sect. 7).

24th EACSL Annual Conference on Computer Science Logic (CSL 2015).
Editor: Stephan Kreutzer; pp. 487–503
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A Coalgebraic Decision Procedure for WS1S

T., CSL 2015



A Coalgebraic Decision Procedure for WS1S

T., CSL 2015

Key ingredients: derivative + ε-acceptance test

A Coalgebraic Decision Procedure for WS1S

T., CSL 2015

Key ingredients: $\underbrace{\text{derivative} + \varepsilon\text{-acceptance test}}_{\text{coalgebra}}$

A Coalgebraic Decision Procedure for WS1S

T., CSL 2015

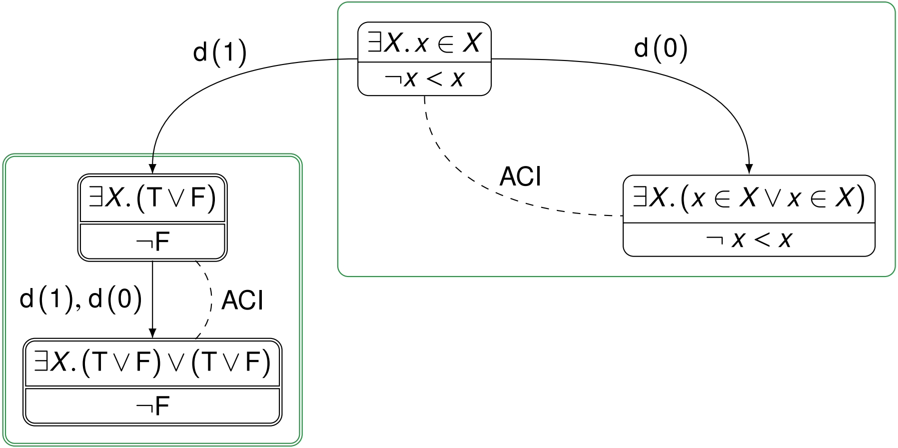
Key ingredients: $\underbrace{\text{derivative} + \varepsilon\text{-acceptance test}}_{\text{coalgebra}}$

Let's define them on WS1S formulas directly!

A Coalgebraic Decision Procedure for WS1S

T., CSL 2015

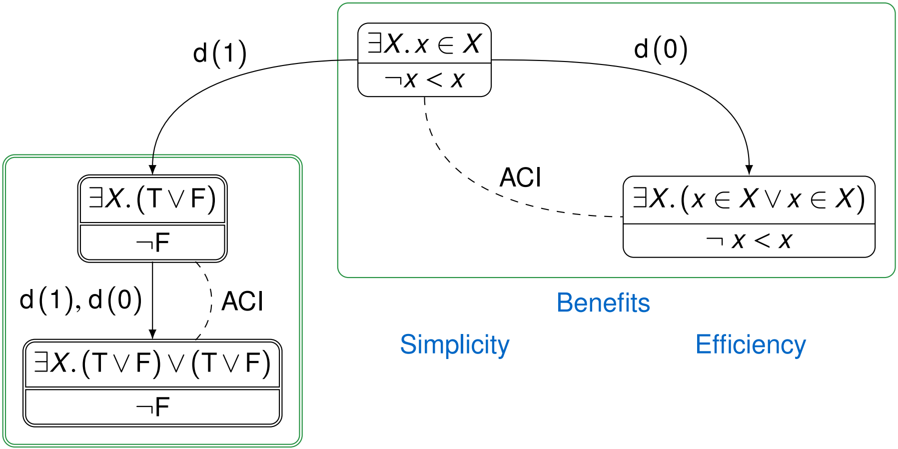
$(\exists X.\, x \in X) \stackrel{?}{\equiv} (\neg x < x)$ for $\Sigma = \{(0), (1)\}$



A Coalgebraic Decision Procedure for WS1S

T., CSL 2015

$(\exists X.\, x \in X) \stackrel{?}{\equiv} (\neg x < x)$ for $\Sigma = \{(0),(1)\}$



Benefits

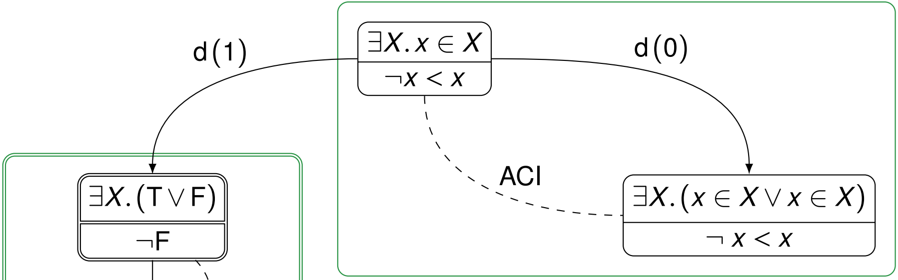
Simplicity

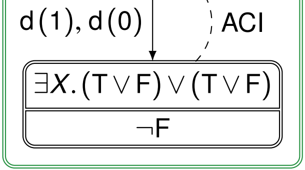
Efficiency

A Coalgebraic Decision Procedure for WS1S

T., CSL 2015

$(\exists X.\, x \in X) \stackrel{?}{\equiv} (\neg x < x)$ for $\Sigma = \{(0), (1)\}$





Benefits

Simplicity

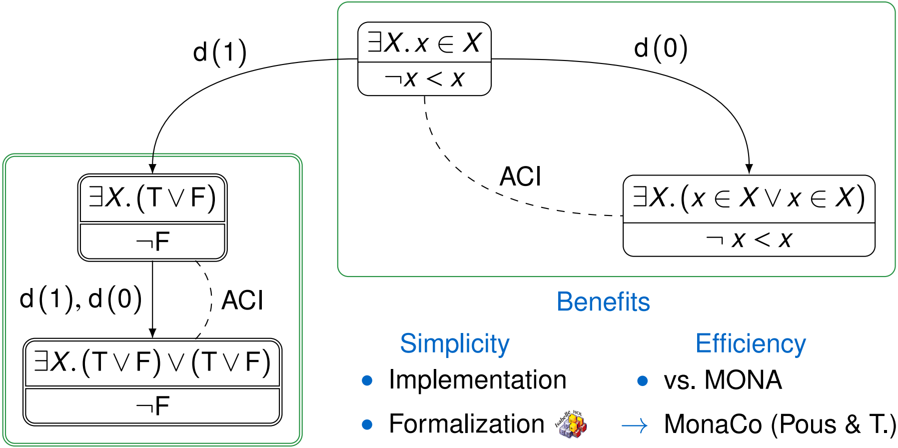
- Implementation
- Formalization
- Presentation

Efficiency

A Coalgebraic Decision Procedure for WS1S

T., CSL 2015

$(\exists X.\, x \in X) \stackrel{?}{\equiv} (\neg x < x)$ for $\Sigma = \{(0), (1)\}$



Benefits

Simplicity

- Implementation
- Formalization
- Presentation

Efficiency

- vs. MONA

$\rightarrow$ MonaCo (Pous & T.)

Conclusion

DERIVE

AND

CONQUER

DERIVE

AND

CONQUER

Thanks for your attention!
Questions?

Formalizing Symbolic Decision Procedures

in Isabelle HOL

for

Regular Languages

Dmitriy Traytel

Technische Universität München