Formalizing Symbolic Decision Procedures for Regular Languages

in Isabelle HOL

Dmitriy Traytel

Technische Universität München
Representations of Regular Languages

\[
\begin{align*}
  a(a^* + b^*) & \quad \neg (b(\neg \emptyset) + a(\neg(a^*) \cap \neg(b^*))) \\
  \exists x. (\forall y. x \leq y) \land x \in X
\end{align*}
\]

RE \quad ERE \quad \text{WMSO}
Representations of Regular Languages

effective

DFA

NFA

a(a^* + b^*) \implies (\neg(b(\neg\emptyset) + a(\neg(a^*) \cap \neg(b^*)))) \exists x. (\forall y. x \leq y) \land x \in X

RE
ERE
WMSO

symbolic
Theme equivalence problem of symbolic representations

$L(a^*) = L(aa^* + \varepsilon)$? $L(\exists X. \forall y. y \in X) = L F$?
Theme equivalence problem of symbolic representations

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

Setting in a proof assistant
My Thesis

Theme  equivalence problem of symbolic representations

$L\left(a^*\right) = L\left(aa^* + \varepsilon\right)$?

$L\left(\exists X. \forall y. y \in X\right) = L\ F$?

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

\[ \mathcal{L}(a^*) = \mathcal{L}(aa^* + \varepsilon) \]

\[ \mathcal{L}(\exists X. \forall y. y \in X) = \mathcal{L} F \]

Setting  in a proof assistant

Catch  without resorting to explicit representations

unlike traditional methods

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

⇒ use variations of Brzozowski derivatives
$a^* \equiv \epsilon + a \cdot a^*$ for $\Sigma = \{a, b\}$

\[
\begin{array}{c|c}
   & a^* \\
\hline
   & \epsilon + a \cdot a^* \\
\end{array}
\]

Brzozowski 1964
Ginzburg 1967
$a^* \equiv \varepsilon + a \cdot a^*$ for $\Sigma = \{a, b\}$

Brzozowski 1964
Ginzburg 1967

Brzozowski derivative

$d: \text{letter} \rightarrow \text{regex} \rightarrow \text{regex}$

$\mathcal{L}(d a r) = \{w \mid aw \in \mathcal{L}(r)\}$
\[ a^* \equiv \varepsilon + a \cdot a^* \text{ for } \Sigma = \{a, b\} \]

Brzozowski 1964
Ginzburg 1967
\[ a^* \equiv \varepsilon + a \cdot a^* \text{ for } \Sigma = \{a, b\} \]

Brzozowski 1964
Ginzburg 1967
$a^* \equiv \varepsilon + a \cdot a^*$ for $\Sigma = \{a, b\}$

Brzozowski 1964
Ginzburg 1967
$a^* \equiv \varepsilon + a \cdot a^*$ for $\Sigma = \{a, b\}$
\[ a^* \equiv \varepsilon + a \cdot a^* \text{ for } \Sigma = \{a, b\} \]

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 Languages 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, Petričan, and Silva
*CSL 2015*  T.
Derivatives in Literature

Theoretical groundwork

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

Programming Languages 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.
Unified Decision Procedures for Regular Expression Equivalence
Nipkow & T., ITP 2014

<table>
<thead>
<tr>
<th>Conference</th>
<th>Authors</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICFP 2010</td>
<td>Fischer, Huch, &amp; Wilke</td>
<td>Mark after</td>
</tr>
<tr>
<td>JAR 2011</td>
<td>Krauss &amp; Nipkow</td>
<td>Brzozowski</td>
</tr>
<tr>
<td>CPP 2011</td>
<td>Coquand &amp; Siles</td>
<td>Brzozowski</td>
</tr>
<tr>
<td>RAMiCS 2012</td>
<td>Moreira, Pereira, &amp; de Sousa</td>
<td>Antimirov</td>
</tr>
<tr>
<td>ITP 2012</td>
<td>Asperti</td>
<td>Mark before</td>
</tr>
</tbody>
</table>
Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2010  Fischer, Huch, & Wilke

JAR 2011  Krauss, Brzozowski

CPP 2011  Moreira, Pereira, & de Sousa

RAMiCS 2012  Antimirov

ITP 2012  Asperti

\[ a \circ (a \circ^* + b \circ^*) \]

Our contribution

• Abstract bisimulation computation
• Insantiations with different derivatives
• \( \sum \circ \partial \) = \( \text{pnorm} \circ d \)

• "Mark before" yields smaller bisimulations than "Mark after" (proof due H. Seidl)
• Empirical comparison
Unified Decision Procedures for Regular Expression Equivalence
Nipkow & T., ITP 2014

ICFP 2010  Fischer, Huch, & Wilke
JAR 2011  Krauss & Brzozowski
CPP 2011  Moreira, Pereira, & de Sousa
RAMiCS 2012  Antimirov
ITP 2012  Asperti

Our contribution
• Abstract bisimulation computation
• Insatiations with different derivatives

Mark after
Mark before

\(\text{read } a\)
Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

ICFP 2010  Fischer, Huch, & Wilke  Mark after

JAR 2011  Krauss  Brzozowski

CPP 2011  Moreira, Pereira, & de Sousa  Antimirov

RAMiCS 2012  

ITP 2012  Asperti  Mark before

\[ \circ a \circ (a \bullet^* + b \bullet^*) \]

read \( aa \)
Unified Decision Procedures for Regular Expression Equivalence
Nipkow & T., ITP 2014

ICFP 2010 Fischer, Huch, & Wilke

JAR 2011 Krauss Brzozowski

CPP 2011 Coquand Siles Brzozowski

RAMiCS 2012 Moreira Pereira de Sousa Antimirov

ITP 2012 Asperti

Our contribution
• Abstract bisimulation computation
• Insantiations with different derivatives
• \( \sum \circ \partial \alpha = \text{pnorm} \circ d \alpha \)
• "Mark before" yields smaller bisimulations than "Mark after" (proof due H. Seidl)
• Empirical comparison

\[ \circ \ a \circ(a \circ^* + b \circ^*) \]

read aab
### Unified Decision Procedures for Regular Expression Equivalence

*Nipkow & T., ITP 2014*

<table>
<thead>
<tr>
<th>Conference</th>
<th>Authors</th>
<th>Marking Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICFP 2010</td>
<td>Fischer, Huch, &amp; Wilke</td>
<td>Mark after</td>
</tr>
<tr>
<td>JAR 2011</td>
<td>Krauss &amp; Nipkow</td>
<td>Brzozowski</td>
</tr>
<tr>
<td>CPP 2011</td>
<td>Coquand &amp; Siles</td>
<td>Brzozowski</td>
</tr>
<tr>
<td>RAMiCS 2012</td>
<td>Moreira, Pereira, &amp; de Sousa</td>
<td>Antimirov</td>
</tr>
<tr>
<td>ITP 2012</td>
<td>Asperti</td>
<td>Mark before</td>
</tr>
</tbody>
</table>
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  Pereira, & de Sousa Antimirov

ITP 2012  Asperti  Mark before

\[ d : \text{letter} \rightarrow \text{regex} \rightarrow \text{regex} + \text{ACI} \]
Unified Decision Procedures for Regular Expression Equivalence

Nipkow & T., ITP 2014

\[ \partial : \text{letter} \rightarrow \text{regex} \rightarrow \text{regex set} \]

ICFP 2010     Mark after
JAR 2011       Brzozowski
CPP 2011       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 2012 Moreira, Pereira, & de 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 & Nipkow                Brzozowski

CPP 2011          Coquand & Siles                Brzozowski

RAMiCS 2012       Moreira, Pereira, & de Sousa   Antimirov

ITP 2012          Asperti                        Mark before

\( \diamond a(a^* + b^*) \)

\( \text{read } a \)
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 Coquand & Siles Brzozowski

RAMiCS 2012 Moreira, Pereira, & de Sousa Antimirov

ITP 2012 Asperti Mark before

$$\alpha \cdot a^* + \beta \cdot b^*$$

read $aa$

\[
d: \text{letter} \rightarrow \text{regex} \rightarrow \text{regex}
\]
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

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

read \(aab\)
Our contribution

- Abstract bisimulation computation
- Insantiations with different derivatives
- $\Sigma \circ \partial a = \text{pnorm} \circ d a$
- “Mark before” yields smaller bisimulations than “Mark after” (proof due H. Seidl)
- Empirical comparison
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 semantic 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.
| T | F | $x \in X$ | $x < y$ | $\varphi \lor \psi$ | $\neg \varphi$ | $\exists x. \varphi$ | $\exists X. \varphi$ |
Verified Decision Procedures for MSO on Words
Based on Derivatives of Regular Expressions
T. & Nipkow, ICFP 2013 & JFP 2015

\[
\begin{align*}
T & | F & x \in X & | x < y & \varphi \lor \psi & | \neg \varphi & | \text{FO } x & | \exists X. \varphi
\end{align*}
\]
Verified Decision Procedures for MSO on Words
Based on Derivatives of Regular Expressions
T. & Nipkow, ICFP 2013 & JFP 2015
Verified Decision Procedures for MSO on Words
Based on Derivatives of Regular Expressions

T. & Nipkow, ICFP 2013 & JFP 2015

\[
\begin{align*}
\text{mkRE}(T) &= \neg \emptyset \\
\text{mkRE}(F) &= \emptyset \\
\text{mkRE}(x < y) &= \neg \emptyset \cdot \text{ANth}_x T \cdot \neg \emptyset \cdot \text{ANth}_y T \cdot \neg \emptyset \\
\text{mkRE}(x \in X) &= \neg \emptyset \cdot \text{ANth}_2 x X \cdot \neg \emptyset \\
\text{mkRE}(\text{FO } x) &= (\text{ANth } x F)^* \cdot \text{ANth } x T \cdot (\text{ANth } x F)^* \\
\text{mkRE}(\varphi \lor \psi) &= \text{mkRE}(\varphi) + \text{mkRE}(\psi) \\
\text{mkRE}(\neg \varphi) &= \neg \text{mkRE}(\varphi) \\
\text{mkRE}(\exists \varphi) &= \prod (\text{mkRE}(\varphi)) 
\end{align*}
\]
Verified Decision Procedures for MSO on Words
Based on Derivatives of Regular Expressions
T. & Nipkow, ICFP 2013 & JFP 2015

\[
\begin{align*}
\text{mkRE}(T) &= \neg \varnothing \\
\text{mkRE}(F) &= \varnothing \\
\text{mkRE}(x < y) &= \neg \varnothing \cdot \text{ANth} \ x \ T \cdot \neg \varnothing \cdot \text{ANth} \ y \ T \cdot \neg \varnothing \\
\text{mkRE}(x \in X) &= \neg \varnothing \cdot \text{ANth}_2 \ x \ X \cdot \neg \varnothing \\
\text{mkRE}(\text{FO} \ x) &= \left( \text{ANth} \ x \ F \right)^* \cdot \text{ANth} \ x \ T \cdot \left( \text{ANth} \ x \ F \right)^* \\
\text{mkRE}(\varphi \lor \psi) &= \text{mkRE}(\varphi) + \text{mkRE}(\psi) \\
\text{mkRE}(\neg \varphi) &= \neg \text{mkRE}(\varphi) \\
\text{mkRE}(\exists \varphi) &= \Pi \left( \text{mkRE}(\varphi) \right)
\end{align*}
\]

preserves semantics
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 con-
nection, 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

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
true – 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).
Key ingredients: derivative + $\varepsilon$-acceptance test
Key ingredients: derivative + $\varepsilon$-acceptance test
Key ingredients: derivative + \( \varepsilon \)-acceptance test
A Coalgebraic Decision Procedure for WS1S

T., CSL 2015

$(\exists X. x \in X) \equiv (\neg x < x)$ for $\Sigma = \{(0), (1)\}$
$(\exists X. x \in X) \equiv (\neg x < x)$ for $\Sigma = \{(0), (1)\}$
A Coalgebraic Decision Procedure for WS1S

T., CSL 2015

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

\[\exists X. x \in X \equiv (\neg x < x)\]

Benefits

- Implementation
- Formalization
- Presentation

Efficiency
A Coalgebraic Decision Procedure for WS1S

T., CSL 2015

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

Benefits
- Simplicity
  - Implementation
  - Formalization
  - Presentation

Efficiency
- vs. MONA
  - MonaCo (Pous & T.)
Conclusion
DERIVE AND CONQUER

Thanks for your attention!

Questions?
DERIVE AND CONQUER
DERIVE
AND
CONQUER

Thanks for your attention!
Questions?
Formalizing Symbolic Decision Procedures for Regular Languages

Dmitriy Traytel