# Constructive Formalization of Regular Languages

**Jan-Oliver Kaiser**

Advisors: Christian Doczkal, Gert Smolka

Supervisor: Gert Smolka

# Contents

## Motivation

We want to develop an elegant formalization of regular languages in Coq based on finite automata.

There are several reasons for choosing this topic and our specific approach:

- Strong interest in formalizations in this area.
- Few formalizations of regular languages in Coq, most of them very long or incomplete.
- Most formalizations avoid finite automata in favor of regular expressions. Regular expressions (with Brzozowski derivatives) lead to more complex but also more performant algorithms.

# Quick Recap

We use extended **regular expressions** (regexp):

$r,s \ ::= \ \emptyset \mid \varepsilon \mid a \mid rs \mid r+s \mid r\&s \mid r^* \mid \neg r$

- $\mathcal{L}(\emptyset) := \{\}$
- $\mathcal{L}(\varepsilon) := \{\varepsilon\}$
- $\mathcal{L}(a) := \{a\}$
- $\mathcal{L}(rs) := \mathcal{L}(r) \cdot \mathcal{L}(s)$
- $\mathcal{L}(r+s) := \mathcal{L}(r) \cup \mathcal{L}(s)$
- $\mathcal{L}(r\&s) := \mathcal{L}(r) \cap \mathcal{L}(s)$
- $\mathcal{L}(r^*) := \mathcal{L}(r)^*$
- $\mathcal{L}(\neg r) := \overline{\mathcal{L}(r)}$

**Derivatives of Regular Expressions** (1964), *Janusz Brzozowski*:

- der a $\emptyset$= $\emptyset$
- der a $\varepsilon$= $\emptyset$
- der a b = if a = b then $\varepsilon$ else $\emptyset$
- der a (r s) = if $\delta(r)$ then (der a s) + ((der a r) s) else (der a r) s

  with $\delta(r) = true \Leftrightarrow \varepsilon \in \mathcal{L}(r)$. (easily decidable by recursion on r)

  ...

**Theorem 1**: $w \in \mathcal{L}(r)$ if and only if the derivative of r with respect to $w_1 .. w_{|w|}$ accepts $\varepsilon$.

**Theorem 2**: The set of derivatives of r is *closed under derivation* and *finite* up to similarity.

Regular languages are also exactly those languages accepted by **finite automata** (FA).

Our definition of FA over an alphabet $\Sigma$ :

- The finite set of states Q
- The initial state $s_0 \in$ Q
- The (decidable) transition relation $\Delta \in$ (Q, $\Sigma$, Q)

  Deterministic FA: $\Delta$ is functional and **total**.
- The set of finite states F, F $\subseteq$ Q

Let A be a FA:

$$\mathcal{L}(A) := \{w \mid \exists\ s_1, .., s_{|w|} \in Q\ s.t.\ (\forall\ i : 0 < i \leq n \rightarrow (s_{i-1}, w_i, s_i) \in \Delta_A)\ \land\ s_{|w|} \in F_A\ \}$$

Finally, regular languages are also characterized by the Myhill-Nerode theorem (MH).

First, we define an equivalence relation on L (MH relation):

$x\ R_L\ y\ := \forall z,\ x \cdot z\ \in L\ \Leftrightarrow\ y \cdot z\ \in L$

**Myhill-Nerode theorem**: L is regular if and only if $R_L$ divides L into a finite number of equivalence classes.

# Previous work

- **The first constructive formalization of MH**.

  Based on FA.

  Implemented in Nuprl.

  Focus on clear formalization.

  Close to what we want to do.

  (*Constable, Jackson, Naumov, Uribe*, 1997)

- Decision procedure for regexp equivalence.

  Based on Brzozowski's derivatives.

  Only soundness proof, *no proof of termination or completeness*.

  Implemented in Isabelle.

  Focus on simplicity, small regexps.

  (*Krauss, Nipkow*, 2011)

- Decision procedure for regexp equivalence.

  Based on FA, matrices.

  Implemented in Coq.

  Focus on performance. Outperforms every other solution so far.

  (*Braibant, Pous*, 2011)

- **Decision Procedure for regexp equivalence**.

  Based on Brzozowski's derivatives.

  Implemented in Coq.

  Proof of termination given.

  Introduces the notion of *inductively finite sets*.

  (*Coquand, Siles*, 2011)

- **First formalization of MH based on regexp**.

  Based on Brzozowski's derivatives.

  Implemented in Isabelle.

  The first formalization of MH in Isabelle.

  (*Wu, Zhang, Urban*, 2011)

- Decision Procedure for regexp equivalence.

  Based on Brzozowski's derivatives.

  Implemented in Coq.

  Translation of the work done by Krauss and Nipkow to Coq.

  Adds proof of termination.

  (*Moreira, Pereira, de Sousa*, 2011)

## Our Development

- We want to focus on elegance, not performance.
- Our main goals are MH and the decidability of regexp equivalence.
- We use finite automata.
  They are not at all impractical. (Partly thanks to Ssreflect's finType)

# Quick examples

```
Record dfa : Type :=
  dfaI {
    dfa_state :> finType;
    dfa_s0: dfa_state;
    dfa_fin: pred dfa_state;
    dfa_step: dfa_state -> char -> dfa_state
  }.

Fixpoint dfa_accept A (x: A) w :=
match w with
  | [::] => dfa_fin A x
  | a::w => dfa_accept A (dfa_step A x a) w
end.
```

```
Record nfa : Type :=
  nfaI {
    nfa_state :> finType;
    nfa_s0: nfa_state;
    nfa_fin: pred nfa_state;
    nfa_step: nfa_state -> char -> pred nfa_state
  }.

Fixpoint nfa_accept A (x: A) w :=
match w with
  | [::] => nfa_fin A x
  | a::w => existsb y, (nfa_step A x a y) && nfa_accept A y w
end.
```

# 50% of **NFA** $\Rightarrow$ **DFA** (powerset construction)

```
Lemma nfa_to_dfa_correct2 (X: nfa_to_dfa) w:
  dfa_accept nfa_to_dfa X w -> existsb x, (x \in X) && nfa_accept A x w.
Proof. elim: w X => [| a w IHw] X.
  by [].
move/IHw => /existsP [] y /andP [].
rewrite /dfa_step /nfa_to_dfa /= cover_imset.
move/bigcupP => [] x H0 H1 H2.
apply/existsP. exists x. rewrite H0 andTb.
apply/existsP. exists y. move: H1. rewrite in_set => ->.
exact: H2.
Qed.
```

# Roadmap

1. regexp $\Rightarrow$ FA: closure of FA under $\cdot$, $\cup$, $\cap$, $*$, $\neg$. (**Done**)
2. Emptiness test on FA. ( $\mathcal{L}(A) = \emptyset$ )
3. FA $\Rightarrow$ regexp.
4. Dedicedability of regexp equivalence:

$$\mathcal{L}(r) = \mathcal{L}(s) \iff \mathcal{L}(\mathcal{A}(r) \cap \overline{\mathcal{A}(s)}) = \emptyset \wedge \mathcal{L}(\overline{\mathcal{A}(r)} \cap \mathcal{A}(s)) = \emptyset$$

5. Finally, we want to prove the Myhill-Nerode theorem.

   Constable et al. establish a direct equivalence between MH and FA.

   This requires proof of:

   - FA induce an equivalence relation on words
   - This relation is invariant under extension.
   - This relation is a refinement of the MH relation.
   - A finite number of equivalence classes under the MH relation induce a set of states for a FA which accepts exactly the union of these equivalence classes.

# Thank you for your attention.

# References

Constructively formalizing automata theory (1997)

Robert L. Constable, Paul B. Jackson, Pavel Naumov, Juan C. Uribe

Proof Pearl: Regular Expression Equivalence and Relation Algebra (2011)

Alexander Krauss, Tobias Nipkow

Deciding Kleene Algebras in Coq (2011)

Thomas Braibant, Damien Pous

A Decision Procedure for Regular Expression Equivalence in Type Theory (2011)

Thierry Coquand, Vincent Siles

A Formalisation of the Myhill-Nerode Theorem based on Regular Expressions (Proof Pearl) (2011)

Chunhan Wu, Xingyuan Zhang, Christian Urban

Deciding Regular Expressions (In-)Equivalence in Coq (2011)

Nelma Moreira, David Pereira, Simão Melo de Sousa

# Extras

With **Theorem 2**, we can formulate a system of equations:

$$r_0 = \sum_{i=0}^{n} a_{0,i}\ r_{0,i}$$

$$r_1 = \sum_{i=0}^{n} a_{1,i}\ r_{1,i}$$

...

$$r_n = \sum_{i=0}^{n} a_{n,i}\ r_{n,i}$$

where

$r_{j,i} = \emptyset \ \vee\ r_{j,i} = r_i$,

$\{r_k | 0 < k \leq n\}$ is the set of derivatives of $r_0$

and

$r_j = \ldots\ +\ a_{j,i}\ r_i\ + \ldots\ \Leftrightarrow\ der\ r_j\ a_{j,i}\ =\ r_i$.