Verified Decision Procedures for Equivalence of Regular Expressions

Tobias Nipkow & Dmitriy Traytel

Fakultät für Informatik
Technische Universität München

Background

Recent series of papers presenting such decision procedures verified in Coq, Isabelle or Matita:

Background

Recent series of papers presenting such decision procedures verified in Coq, Isabelle or Matita:

Braibant & Pous 2010, Krauss & Nipkow 2011, Coquand & Siles 2011, Asperti 2012, Moreira *et al.* 2013

Background

Recent series of papers presenting such decision procedures verified in Coq, Isabelle or Matita:

Braibant & Pous 2010, Krauss & Nipkow 2011, Coquand & Siles 2011, Asperti 2012, Moreira *et al.* 2013

They all operate on regular expressions, not automata

Background

Recent series of papers presenting such decision procedures verified in Coq, Isabelle or Matita:

Braibant & Pous 2010, Krauss & Nipkow 2011, Coquand & Siles 2011, Asperti 2012, Moreira *et al.* 2013

They all operate on regular expressions, not automata

They all look different but related . . .

This talk

- Unified framework

This talk

- Unified framework
- Derivation of all previous procedures as instances

This talk

- Unified framework
- Derivation of all previous procedures as instances
- Verification in Isabelle

Regular expressions

$$\textbf{datatype}\ \alpha\ rexp = \begin{array}{l} 0 \mid \\ 1 \mid \\ Atom\ \alpha \mid \\ \alpha\ rexp + \alpha\ rexp \mid \\ \alpha\ rexp \cdot \alpha\ rexp \mid \\ \alpha\ rexp^{*} \end{array}$$

Regular expressions

$$
\textbf{datatype}\ \alpha\ rexp = \begin{array}{l} 0 \mid \\ 1 \mid \\ Atom\ \alpha \mid \\ \alpha\ rexp + \alpha\ rexp \mid \\ \alpha\ rexp \cdot \alpha\ rexp \mid \\ \alpha\ rexp\ ^{*} \end{array}
$$

Semantics: $L :: \alpha\ rexp \to \alpha\ lang$
where $\alpha\ lang = \alpha\ list\ set$

How to prove $r \equiv s$

How to prove $r \equiv s$

1. Translate to DFAs A and B

How to prove $r \equiv s$

1. Translate to DFAs A and B
2. Compare A and B

How to prove $r \equiv s$

1. Translate to DFAs A and B
2. Compare A and B
   - Standard algorithm:
     Minimize A and B, check isomorphism.

How to prove $r \equiv s$

1. Translate to DFAs A and B
2. Compare A and B
   - Standard algorithm:
     Minimize A and B, check isomorphism.
   - Easy alternative:
     Check for all reachable states (p, q) of $A \times B$ that p is final iff q is final.

Framework parameters

Framework parameters

Type σ

Framework parameters

Type	σ
Init	$init :: \alpha\ rexp \rightarrow \sigma$

Framework parameters

Type	σ
Init	$init :: \alpha\ rexp \to \sigma$
Transition	$\delta :: \alpha \to \sigma \to \sigma$

Framework parameters

Type	σ
Init	$init :: \alpha\ rexp \to \sigma$
Transition	$\delta :: \alpha \to \sigma \to \sigma$
Final	$fin :: \sigma \to bool$

Framework parameters

Type	σ
Init	$init :: \alpha\ rexp \to \sigma$
Transition	$\delta :: \alpha \to \sigma \to \sigma$
Final	$fin :: \sigma \to bool$
Language	$\mathcal{L} :: \sigma \to \alpha\ lang$

Framework parameters

Type	σ
Init	$init :: \alpha\ rexp \to \sigma$
Transition	$\delta :: \alpha \to \sigma \to \sigma$
Final	$fin :: \sigma \to bool$
Language	$\mathcal{L} :: \sigma \to \alpha\ lang$

Assumptions:

$$\mathcal{L}(init(r)) = L(r)$$

Framework parameters

Type	σ
Init	$init :: \alpha\ rexp \to \sigma$
Transition	$\delta :: \alpha \to \sigma \to \sigma$
Final	$fin :: \sigma \to bool$
Language	$\mathcal{L} :: \sigma \to \alpha\ lang$

Assumptions:

$$\mathcal{L}(init(r)) = L(r)$$

$$\mathcal{L}(\delta\ x\ s) = \{w \mid xw \in \mathcal{L}(s)\}$$

Framework parameters

Type	σ
Init	$init :: \alpha\ rexp \to \sigma$
Transition	$\delta :: \alpha \to \sigma \to \sigma$
Final	$fin :: \sigma \to bool$
Language	$\mathcal{L} :: \sigma \to \alpha\ lang$

Assumptions:

$$\mathcal{L}(init(r)) = L(r)$$

$$\mathcal{L}(\delta\ x\ s) = \{w \mid xw \in \mathcal{L}(s)\}$$

$$fin(s) \Leftrightarrow [] \in \mathcal{L}(s)$$

Equivalence checker

Equivalence checker

$eqv :: \alpha\ rexp \rightarrow \alpha\ rexp \rightarrow bool$

Equivalence checker

$$
\begin{aligned}
&eqv :: \alpha\ rexp \rightarrow \alpha\ rexp \rightarrow bool\\
&eqv\ r\ s = \textbf{case}\ closure\ (init(r),\ init(s))\ \textbf{of}\\
&\qquad\qquad\quad Some([],\ \_) \Rightarrow True\\
&\qquad\qquad\ \mid\ \_ \Rightarrow False
\end{aligned}
$$

Equivalence checker

$$
\begin{aligned}
&eqv :: \alpha\ rexp \to \alpha\ rexp \to bool\\
&eqv\ r\ s = \textbf{case}\ closure\ (init(r),\ init(s))\ \textbf{of}\\
&\qquad\qquad\quad Some([],\ \_) \Rightarrow True\\
&\qquad\qquad \mid\ \_ \Rightarrow False
\end{aligned}
$$

Theorem

$eqv\ r\ s \Longrightarrow L(r) = L(s)$

Equivalence checker

$$
\begin{aligned}
&eqv :: \alpha\ rexp \to \alpha\ rexp \to bool\\
&eqv\ r\ s = \textbf{case}\ closure\ (init(r),\ init(s))\ \textbf{of}\\
&\qquad\qquad\quad Some([],\ \_) \Rightarrow True\\
&\qquad\qquad\ \mid \_ \Rightarrow False
\end{aligned}
$$

Theorem

$eqv\ r\ s \Longrightarrow L(r) = L(s)$

If the set of reachable states is finite:

Theorem

$L(r) = L(s) \Longrightarrow eqv\ r\ s$

Derivatives (Brzozowski 1964)

$$d :: \alpha \to \alpha\ rexp \to \alpha\ rexp$$

Derivatives (Brzozowski 1964)

$$d :: \alpha \rightarrow \alpha\ rexp \rightarrow \alpha\ rexp$$

- $d\ x\ r$ is the derivative of r wrt x

Derivatives (Brzozowski 1964)

$$d :: \alpha \to \alpha\ rexp \to \alpha\ rexp$$

- $d\ x\ r$ is the derivative of r wrt x
- $d\ x\ r =$ "what is left after x has been read"

Derivatives (Brzozowski 1964)

$$d :: \alpha \to \alpha\ rexp \to \alpha\ rexp$$

- $d\ x\ r$ is the derivative of r wrt x
- $d\ x\ r =$ "what is left after x has been read"
- Example: $d\ a\ (Atom(a) \cdot r) = 1 \cdot r$

Derivatives (Brzozowski 1964)

$$d :: \alpha \to \alpha\ rexp \to \alpha\ rexp$$

- $d\ x\ r$ is the derivative of r wrt x
- $d\ x\ r =$ "what is left after x has been read"
- Example: $d\ a\ (Atom(a) \cdot r) = 1 \cdot r$
- Semantics is left-quotient:
  $L(d\ x\ r) = \{w \mid xw \in L(r)\}$

$$d\ x\ 0 \;=\; 0$$

$$
\begin{aligned}
d\,x\,0 &= 0 \\
d\,x\,1 &= 0
\end{aligned}
$$

$$
\begin{aligned}
d\;x\;0 &= 0 \\
d\;x\;1 &= 0 \\
d\;x\;(Atom\;y) &= \textsf{if } x = y \textsf{ then } 1 \textsf{ else } 0
\end{aligned}
$$

$$
\begin{aligned}
d\ x\ 0 &= 0 \\
d\ x\ 1 &= 0 \\
d\ x\ (Atom\ y) &= \textsf{if}\ x = y\ \textsf{then}\ 1\ \textsf{else}\ 0 \\
d\ x\ (r + s) &= d\ x\ r + d\ x\ s
\end{aligned}
$$

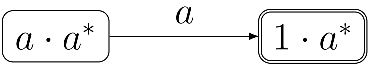
$$
\begin{aligned}
d\ x\ 0 &= 0 \\
d\ x\ 1 &= 0 \\
d\ x\ (Atom\ y) &= \textsf{if } x = y \textsf{ then } 1 \textsf{ else } 0 \\
d\ x\ (r + s) &= d\ x\ r + d\ x\ s \\
d\ x\ (r \cdot s) &= \textsf{if } \varepsilon(r) \textsf{ then } d\ x\ r \cdot s + d\ x\ s \\
&\quad \textsf{else } d\ x\ r \cdot s
\end{aligned}
$$

$$
\begin{aligned}
d\;x\;0 &= 0 \\
d\;x\;1 &= 0 \\
d\;x\;(Atom\;y) &= \textsf{if } x = y \textsf{ then } 1 \textsf{ else } 0 \\
d\;x\;(r + s) &= d\;x\;r + d\;x\;s \\
d\;x\;(r \cdot s) &= \textsf{if } \varepsilon(r) \textsf{ then } d\;x\;r \cdot s + d\;x\;s \\
&\quad\; \textsf{else } d\;x\;r \cdot s \\
d\;x\;(r^*) &= d\;x\;r \cdot r^*
\end{aligned}
$$

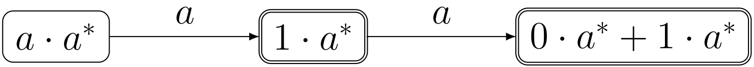
Regular Expression $\rightsquigarrow$ DFA

$a \cdot a^*$

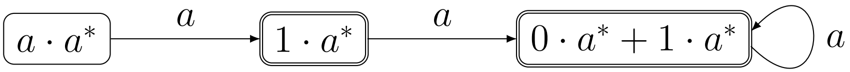
Regular Expression $\rightsquigarrow$ DFA



Regular Expression $\rightsquigarrow$ DFA



Regular Expression $\rightsquigarrow$ DFA



Finiteness

Finiteness

Let $\equiv_{ACI}$ be the equivalence induced by ACI of $+$

Finiteness

Let $\equiv_{ACI}$ be the equivalence induced by ACI of $+$

Theorem (Brzozowski 1964)
The set $\{fold\ d\ w\ r \mid w \in \Sigma^*\}/{\equiv_{ACI}}$ is finite.

Finiteness

Let $\equiv_{ACI}$ be the equivalence induced by ACI of $+$

Theorem (Brzozowski 1964)
The set $\{fold\ d\ w\ r \mid w \in \Sigma^*\}/{\equiv_{ACI}}$ is finite.

How large?

Finiteness

Let $\equiv_{ACI}$ be the equivalence induced by ACI of $+$

Theorem (Brzozowski 1964)
The set $\{fold\ d\ w\ r \mid w \in \Sigma^*\}/{\equiv_{ACI}}$ is finite.

How large? Brzozowski's proof yields $O(2^{\cdot^{\cdot^{\cdot^{2^n}}}})$

Instantiation of framework

Instantiation of framework

$$\sigma \ = \ \alpha \, rexp$$

Instantiation of framework

$$\begin{aligned} \sigma &= \alpha\, rexp \\ init(r) &= r \end{aligned}$$

Instantiation of framework

$$
\begin{aligned}
\sigma &= \alpha\, rexp \\
init(r) &= r \\
\delta\; x\; r &= norm_{ACI}(d\; x\; r)
\end{aligned}
$$

Instantiation of framework

$$
\begin{aligned}
\sigma &= \alpha\ rexp \\
init(r) &= r \\
\delta\ x\ r &= norm_{ACI}(d\ x\ r) \\
fin &= \varepsilon
\end{aligned}
$$

Instantiation of framework

$$
\begin{aligned}
\sigma &= \alpha\ rexp \\
init(r) &= r \\
\delta\ x\ r &= norm_{ACI}(d\ x\ r) \\
fin &= \varepsilon \\
\mathcal{L} &= L
\end{aligned}
$$

Instantiation of framework

$$\begin{aligned}
\sigma &= \alpha\ rexp \\
init(r) &= r \\
\delta\ x\ r &= norm_{ACI}(d\ x\ r) \\
fin &= \varepsilon \\
\mathcal{L} &= L
\end{aligned}$$

Finiteness:

Instantiation of framework

$$\begin{array}{rcl} \sigma & = & \alpha\ rexp \\ init(r) & = & r \\ \delta\ x\ r & = & norm_{ACI}(d\ x\ r) \\ fin & = & \varepsilon \\ \mathcal{L} & = & L \end{array}$$

Finiteness:

- Not immediate from Brzozowski's theorem

Instantiation of framework

$$
\begin{aligned}
\sigma &= \alpha\ rexp \\
init(r) &= r \\
\delta\ x\ r &= norm_{ACI}(d\ x\ r) \\
fin &= \varepsilon \\
\mathcal{L} &= L
\end{aligned}
$$

Finiteness:

- Not immediate from Brzozowski's theorem
- Open for stronger normalization functions

Antimirov 1996

Antimirov 1996

Idea: build some of $\equiv$ into the data structure set:

Antimirov 1996

Idea: build some of $\equiv$ into the data structure set:

$$d : \alpha \to \alpha\ rexp \to \alpha\ rexp$$

Antimirov 1996

Idea: build some of $\equiv$ into the data structure set:

$$D : \alpha \rightarrow \alpha\ rexp \rightarrow \alpha\ rexp\ set$$

Antimirov 1996

Idea: build some of $\equiv$ into the data structure set:

$$D : \alpha \to \alpha\ rexp \to \alpha\ rexp\ set$$

$$d\ x\ (r + s) \ = \ d\ x\ r + d\ x\ s$$

Antimirov 1996

Idea: build some of $\equiv$ into the data structure set:

$$D : \alpha \to \alpha\ rexp \to \alpha\ rexp\ set$$

$$D\ x\ (r + s) \ = \ D\ x\ r \cup D\ x\ s$$

Antimirov 1996

Idea: build some of $\equiv$ into the data structure set:

$$D : \alpha \to \alpha\ rexp \to \alpha\ rexp\ set$$

$$\begin{array}{rcl} D\ x\ (r + s) & = & D\ x\ r \cup D\ x\ s \\ d\ x\ (r \cdot s) & = & \text{if } \varepsilon(r) \text{ then } d\ x\ r \cdot s + d\ x\ s \\ & & \text{else } d\ x\ r \cdot s \end{array}$$

Antimirov 1996

Idea: build some of $\equiv$ into the data structure set:

$$D : \alpha \to \alpha\ rexp \to \alpha\ rexp\ set$$

$$\begin{aligned} D\ x\ (r + s) &= D\ x\ r \cup D\ x\ s \\ D\ x\ (r \cdot s) &= \text{if } \varepsilon(r) \text{ then } D\ x\ r \odot s \cup D\ x\ s \\ &\quad \text{else } D\ x\ r \odot s \end{aligned}$$

Antimirov 1996

Idea: build some of $\equiv$ into the data structure set:

$$D : \alpha \rightarrow \alpha\ rexp \rightarrow \alpha\ rexp\ set$$

$$\begin{array}{rcl} D\ x\ (r + s) & = & D\ x\ r \cup D\ x\ s \\ D\ x\ (r \cdot s) & = & \text{if } \varepsilon(r) \text{ then } D\ x\ r \odot s \cup D\ x\ s \\ & & \text{else } D\ x\ r \odot s \end{array}$$

where $\{r_1, \ldots, r_n\} \odot s = \{r_1 \cdot s, \ldots, r_n \cdot s\}$

Antimirov 1996

Idea: build some of $\equiv$ into the data structure set:

$$D : \alpha \to \alpha\ rexp \to \alpha\ rexp\ set$$

$$\begin{array}{rcl} D\ x\ (r + s) & = & D\ x\ r \cup D\ x\ s \\ D\ x\ (r \cdot s) & = & \text{if } \varepsilon(r) \text{ then } D\ x\ r \odot s \cup D\ x\ s \\ & & \text{else } D\ x\ r \odot s \\ & \vdots & \end{array}$$

where $\{r_1, \ldots, r_n\} \odot s = \{r_1 \cdot s, \ldots, r_n \cdot s\}$

Instantiation of framework

Instantiation of framework

$$\sigma \;=\; \alpha\, rexp\, set$$

Instantiation of framework

$$\begin{aligned} \sigma &= \alpha\ rexp\ set \\ init(r) &= \{r\} \end{aligned}$$

Instantiation of framework

$$\begin{aligned}
\sigma &= \alpha\ rexp\ set \\
init(r) &= \{r\} \\
\delta\ x\ R &= \bigcup_{r \in R} D\ x\ r
\end{aligned}$$

Instantiation of framework

$$\begin{aligned}
\sigma &= \alpha\ rexp\ set \\
init(r) &= \{r\} \\
\delta\ x\ R &= \bigcup_{r \in R} D\ x\ r \\
fin(R) &= \exists r \in R.\ \varepsilon(r)
\end{aligned}$$

Instantiation of framework

$$
\begin{aligned}
\sigma &= \alpha\ rexp\ set \\
init(r) &= \{r\} \\
\delta\ x\ R &= \bigcup_{r \in R} D\ x\ r \\
fin(R) &= \exists r \in R.\ \varepsilon(r) \\
\mathcal{L}(R) &= \bigcup_{r \in R} L(r)
\end{aligned}
$$

Finiteness

Theorem (Antimirov 1996)
Starting from a regular expression r

Finiteness

Theorem (Antimirov 1996)
Starting from a regular expression r
at most $|r|_{at} + 1$ regular expressions are reachable

Finiteness

Theorem (Antimirov 1996)
Starting from a regular expression r
at most $|r|_{at} + 1$ regular expressions are reachable
where $|r|_{at}$ is the number of occurrences of atoms in r.

Finiteness

Theorem (Antimirov 1996)
Starting from a regular expression r
at most $|r|_{at} + 1$ regular expressions are reachable
where $|r|_{at}$ is the number of occurrences of atoms in r.

$\Longrightarrow 2^{|r|_{at}+1}$ sets of regular expressions reachable

History

McNaughton & Yamada 1960, Glushkov 1961:

History

McNaughton & Yamada 1960, Glushkov 1961:

- Translation of regular expression to N/DFA

History

McNaughton & Yamada 1960, Glushkov 1961:

- Translation of regular expression to N/DFA
- Atoms in regular expression are indexed, eg $a_1 \cdot a_2 + a_3 \cdot b_1$

History

McNaughton & Yamada 1960, Glushkov 1961:

- Translation of regular expression to N/DFA
- Atoms in regular expression are indexed, eg $a_1 \cdot a_2 + a_3 \cdot b_1$
- States are (sets of) indexed atoms, eg $\{a_1, a_3\}$

History

McNaughton & Yamada 1960, Glushkov 1961:

- Translation of regular expression to N/DFA
- Atoms in regular expression are indexed, eg $a_1 \cdot a_2 + a_3 \cdot b_1$
- States are (sets of) indexed atoms, eg $\{a_1, a_3\}$

Functional implementation by
Fischer, Huch & Wilke [ICFP 2009]:

- Replace sets of positions
  by marked regular expressions: $Atom(bool, \alpha)$

History

McNaughton & Yamada 1960, Glushkov 1961:

- Translation of regular expression to N/DFA
- Atoms in regular expression are indexed, eg $a_1 \cdot a_2 + a_3 \cdot b_1$
- States are (sets of) indexed atoms, eg $\{a_1, a_3\}$

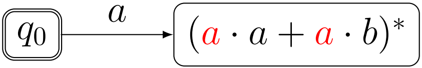
Functional implementation by
Fischer, Huch & Wilke [ICFP 2009]:

- Replace sets of positions by marked regular expressions: $Atom(bool, \alpha)$
- Only matching, not $\equiv$, no proofs

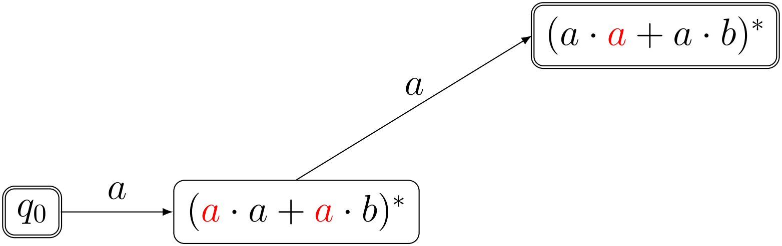
Example: $(a \cdot a + a \cdot b)^*$

Example: $(a \cdot a + a \cdot b)^*$

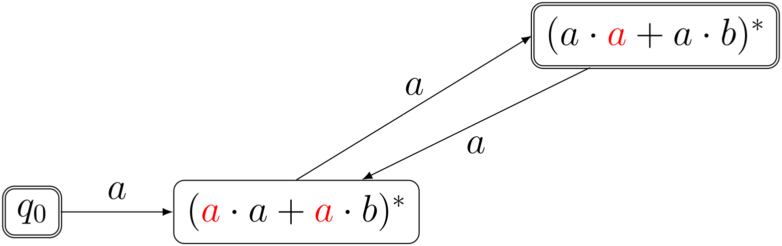
Example: $(a \cdot a + a \cdot b)^*$



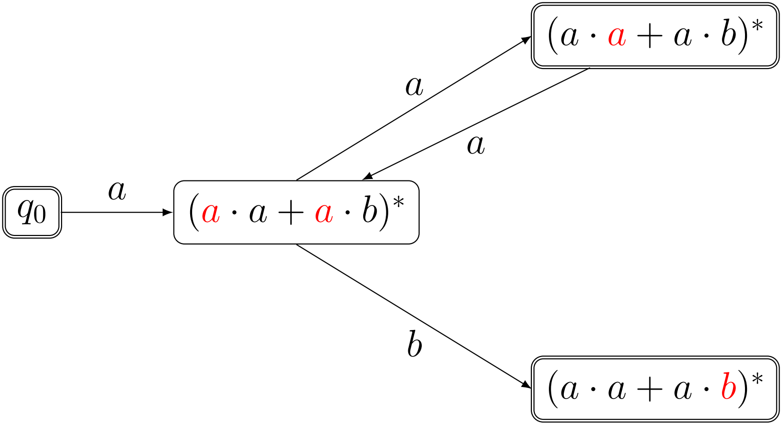
Example: $(a \cdot a + a \cdot b)^*$



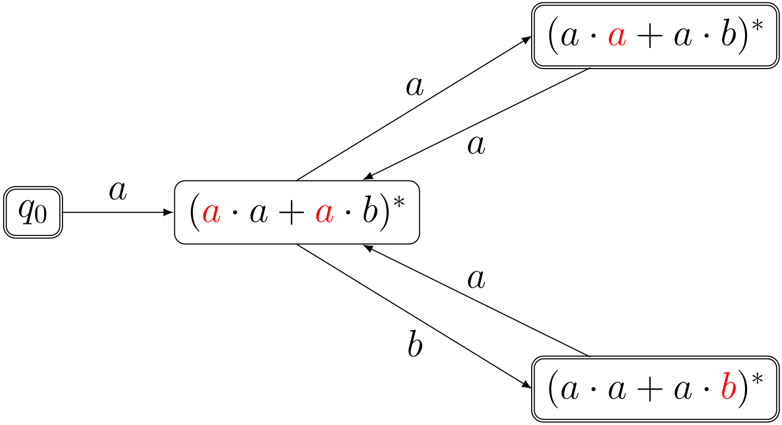
Example: $(a \cdot a + a \cdot b)^*$



Example: $(a \cdot a + a \cdot b)^*$



Example: $(a \cdot a + a \cdot b)^*$



Instantiation of framework

Instantiation of framework

$$\sigma \;=\; bool \times (bool \times \alpha)\, rexp$$

Instantiation of framework

$$\begin{aligned} \sigma &= bool \times (bool \times \alpha)\, rexp \\ init(r) &= (True,\ map\,(\lambda a.\,(False, a))\, r) \end{aligned}$$

Instantiation of framework

$$\begin{aligned} \sigma &= \mathit{bool} \times (\mathit{bool} \times \alpha)\ \mathit{rexp} \\ \mathit{init}(r) &= (\mathit{True},\ \mathit{map}\ (\lambda a.\ (\mathit{False}, a))\ r) \\ \delta\ x\ (m, r) &= (\mathit{False},\ \mathit{read}\ x\ (\mathit{follow}\ m\ r)) \end{aligned}$$

Instantiation of framework

$$
\begin{aligned}
\sigma &= \mathit{bool} \times (\mathit{bool} \times \alpha)\ \mathit{rexp} \\
\mathit{init}(r) &= (\mathit{True},\ \mathit{map}\ (\lambda a.\ (\mathit{False}, a))\ r) \\
\delta\ x\ (m, r) &= (\mathit{False},\ \mathit{read}\ x\ (\mathit{follow}\ m\ r)) \\
\mathit{fin}(m, r) &= \ldots \\
\mathcal{L}(m, r) &= \ldots
\end{aligned}
$$

Conceptually,
the marks in McNaugton/Glushkov/Fisher
are *after* the atoms

Marked regular expressions II

Asperti [ITP 2012]:

- Verified $\equiv$-checker via marked $rexp$ in Matita

Marked regular expressions II

Asperti [ITP 2012]:

- Verified $\equiv$-checker via marked $rexp$ in Matita
- Says he has formalised McNaughton & Yamada

Marked regular expressions II

Asperti [ITP 2012]:

- Verified $\equiv$-checker via marked $rexp$ in Matita
- Says he has formalised McNaughton & Yamada
- ... but he invented his own variation:

Marked regular expressions II

Asperti [ITP 2012]:

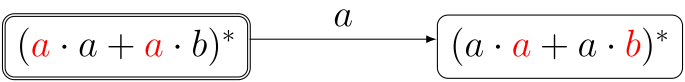
- Verified $\equiv$-checker via marked $rexp$ in Matita
- Says he has formalised McNaughton & Yamada
- ...but he invented his own variation:
  Puts the mark *before* the atom

Example: $(a \cdot a + a \cdot b)^*$

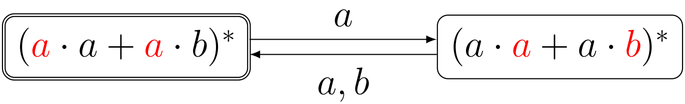
Example: $(a \cdot a + a \cdot b)^*$

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

Example: $(a \cdot a + a \cdot b)^*$



Example: $(a \cdot a + a \cdot b)^*$



Instantiation of framework

Similar but a bit more complicated

Before vs After

Before vs After

Transitions can be decomposed into two steps:

- Before: $read; follow$

Before vs After

Transitions can be decomposed into two steps:

- Before: $read; follow$
- After: $follow; read$

Before vs After

Transitions can be decomposed into two steps:

- Before: $read; follow$
- After: $follow; read$

Theorem

The before-automaton is a homorphic image of the after-automaton.

Before vs After

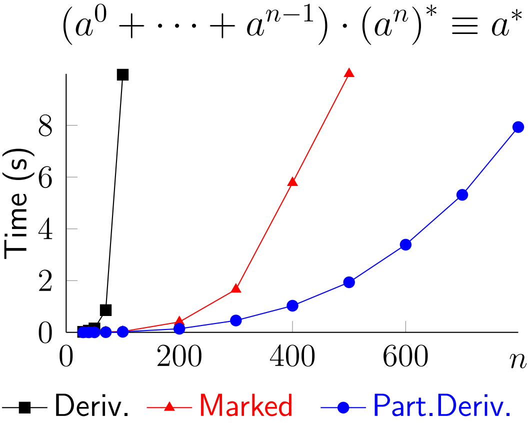
Transitions can be decomposed into two steps:

- Before: $read; follow$
- After: $follow; read$

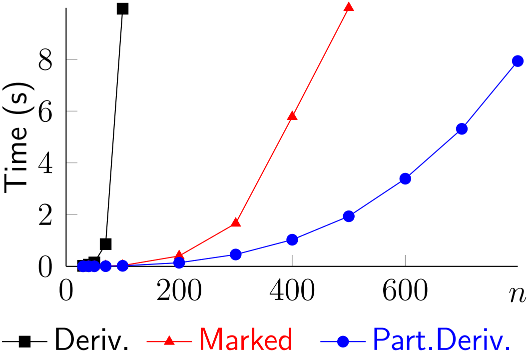
Theorem

The before-automaton is a homorphic image of the after-automaton.

Proof idea due to Helmut Seidl.


$(a^0 + \cdots + a^{n-1}) \cdot (a^n)^* \equiv a^*$
Time (s)
0
2
4
6
8
0
200
400
600
n
Deriv.
Marked
Part.Deriv.

$$(a^0 + \cdots + a^{n-1}) \cdot (a^n)^* \equiv a^*$$



For randomly generated examples:

Deriv. $\gg$ Part.Deriv. $\gg$ Fischer, Asperti

Extended regular expressions

Extended regular expressions

Complement and intersection:

Extended regular expressions

Complement and intersection:

- Trivial for derivatives (Brzozowski)

Extended regular expressions

Complement and intersection:

- Trivial for derivatives (Brzozowski)
- Harder for partial derivatives (Champarnaud and Mignot)

Extended regular expressions

Complement and intersection:

- Trivial for derivatives (Brzozowski)
- Harder for partial derivatives (Champarnaud and Mignot)
- Unclear for marked regular expressions

Extended regular expressions

Complement and intersection:

- Trivial for derivatives (Brzozowski)
- Harder for partial derivatives (Champarnaud and Mignot)
- Unclear for marked regular expressions

. . . and projection:

Extended regular expressions

Complement and intersection:

- Trivial for derivatives (Brzozowski)
- Harder for partial derivatives (Champarnaud and Mignot)
- Unclear for marked regular expressions

. . . and projection:

- Traytel & N. [ICFP 13] extend derivatives

Extended regular expressions

Complement and intersection:

- Trivial for derivatives (Brzozowski)
- Harder for partial derivatives (Champarnaud and Mignot)
- Unclear for marked regular expressions

... and projection:

- Traytel & N. [ICFP 13] extend derivatives
  $\leadsto$ decision procedure for MSO on finite strings

Summary

Equivalence-checkers for regular expressions
can be defined purely functionally

Summary

Equivalence-checkers for regular expressions
can be defined purely functionally
via (partial) derivatives
or marked regular expressions

Summary

Equivalence-checkers for regular expressions
can be defined purely functionally
via (partial) derivatives
or marked regular expressions

Perfect proof assistant fodder ☺