A compact proof of decidability for regular expression equivalence

ITP 2012
Princeton, USA

Andrea Asperti

Department of Computer Science
University of Bologna

25/08/2011

Abstract

We introduce the notion of pointed regular expression and use it to get

1. a compact formalization of the relation between regular expressions and deterministic finite automata
2. a formally verified, efficient algorithm for testing regular expression equivalence.

Content

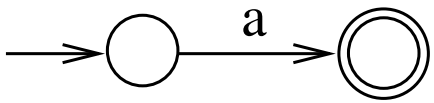
Content

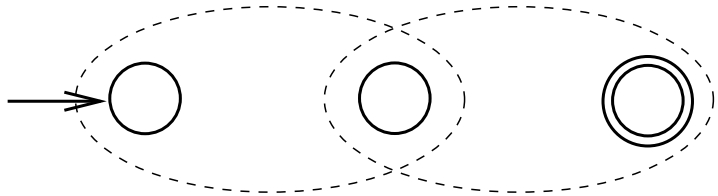
Content

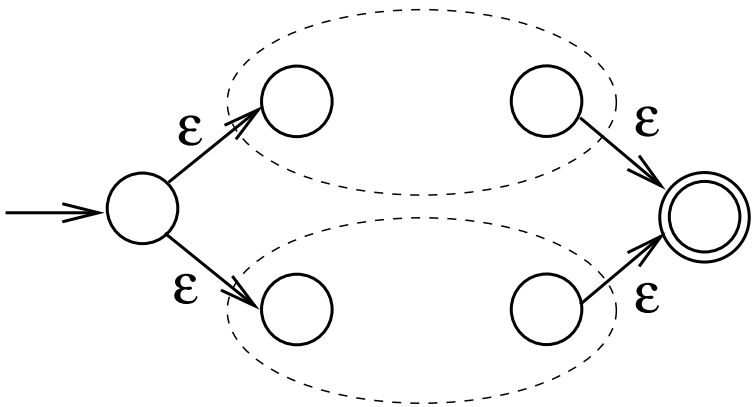
Content

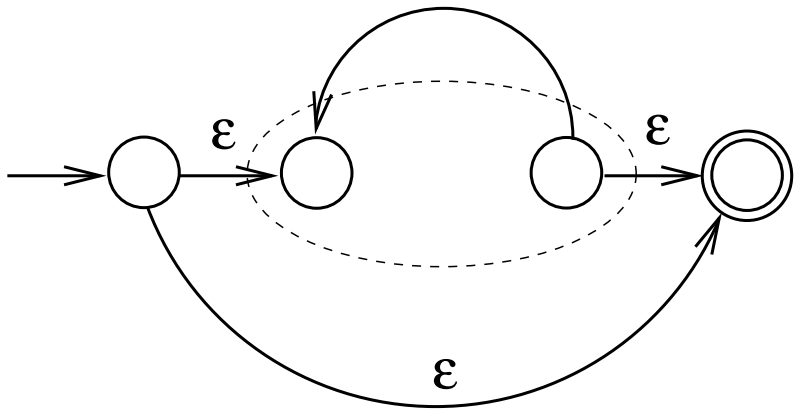
Content

Thompson's algorithm





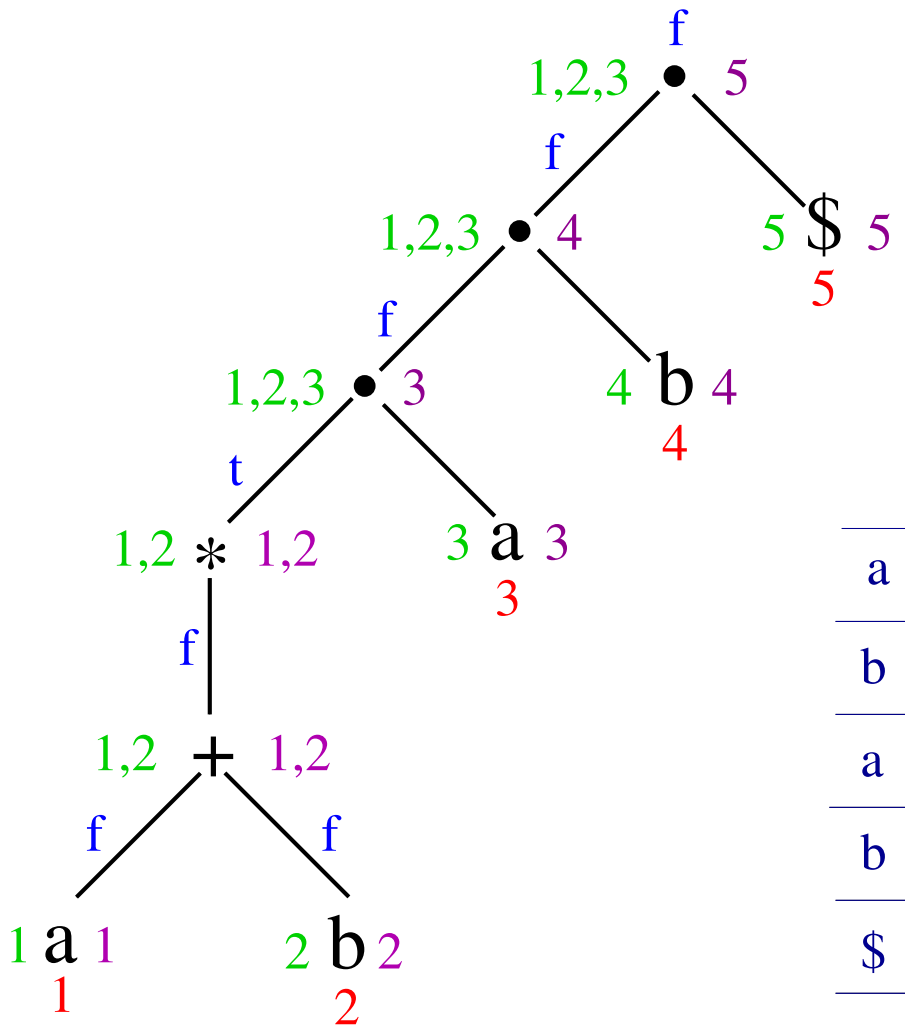




Brzozowski's derivatives $\partial_a(e)$

- $\partial_a(a) = \epsilon$
- $\partial_a(b) = \emptyset$
- $\partial_a(e_1 + e_2) = \partial_a(e_1) + \partial_a(e_2)$
- $\partial_a(e_1 e_2) = \begin{cases} \partial_a(e_1)e2 + \partial_a(e_2) & \text{if nullable } e_1 \\ \partial_a(e_1)e2 & \text{otherwise} \end{cases}$
- $\partial_a(e^*) = \partial_a(e)e^*$

McNaughton and Yamada's algorithm



		followpos
a	1	1,2,3
b	2	1,2,3
a	3	4
b	4	5
$	5	

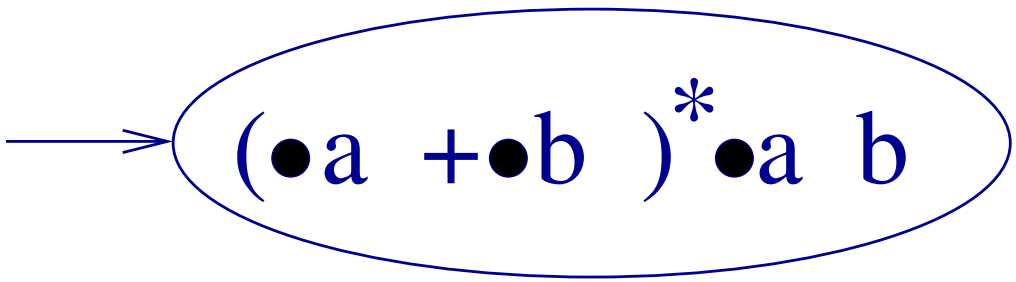
Pointed regular expressions

Intuition: mark the positions inside the regular expression which have been reached after reading some prefix of the input string.

These "pointed" expression are the states of the DFA.

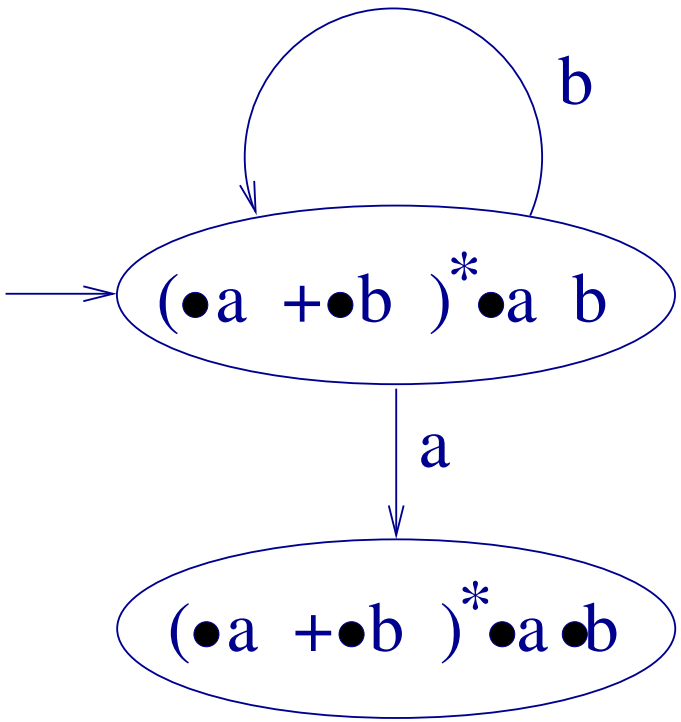
Example: (a+b)*ab

Initial position:

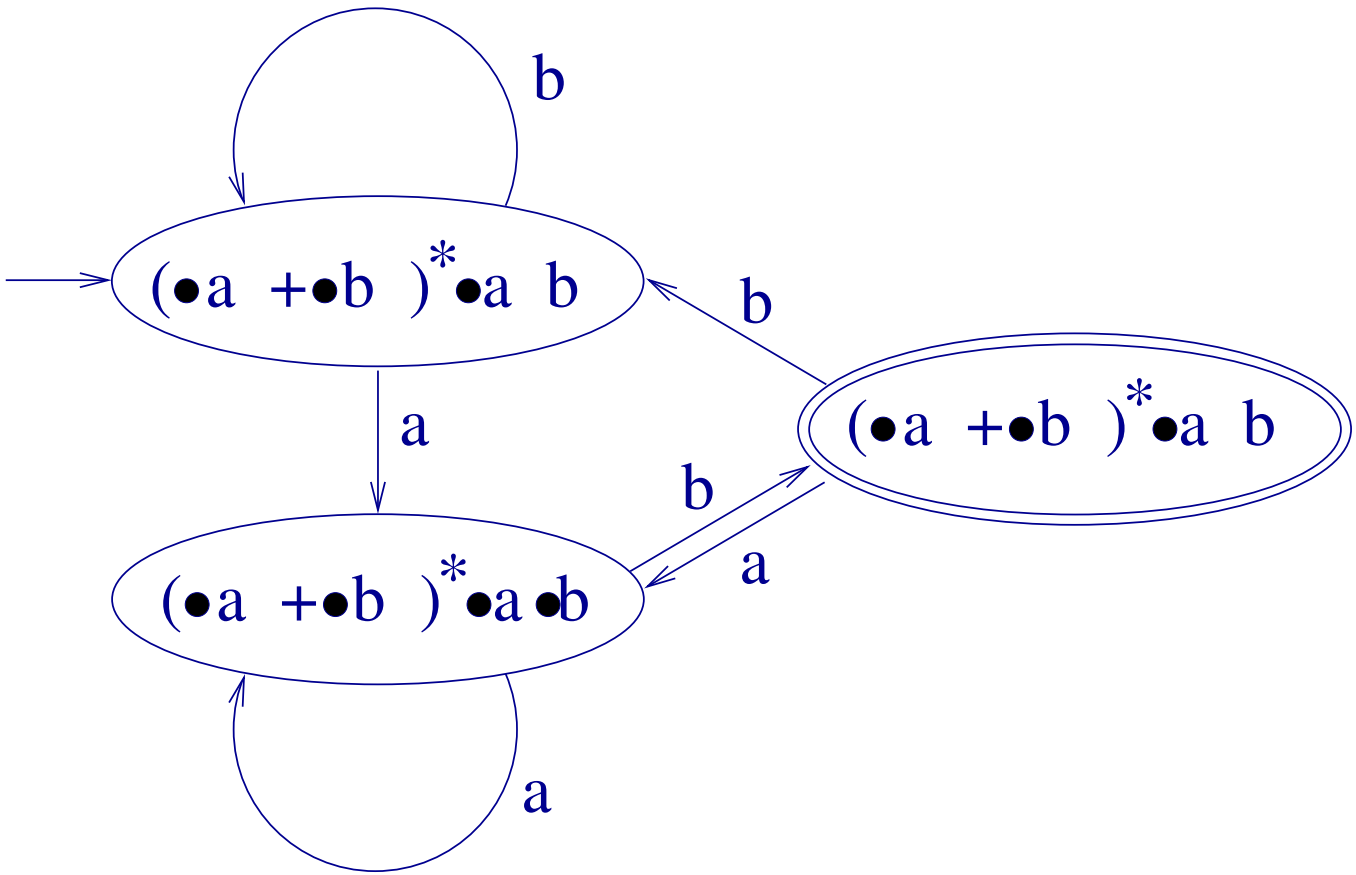


Example: (a+b)*ab

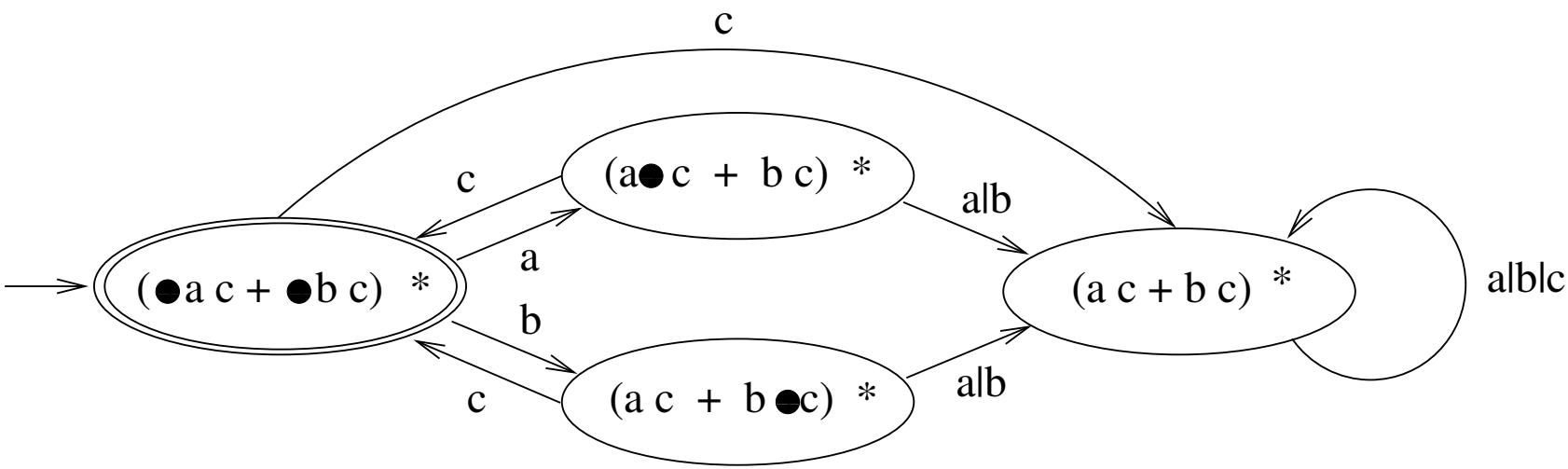
Moves w.r.t. a and b:



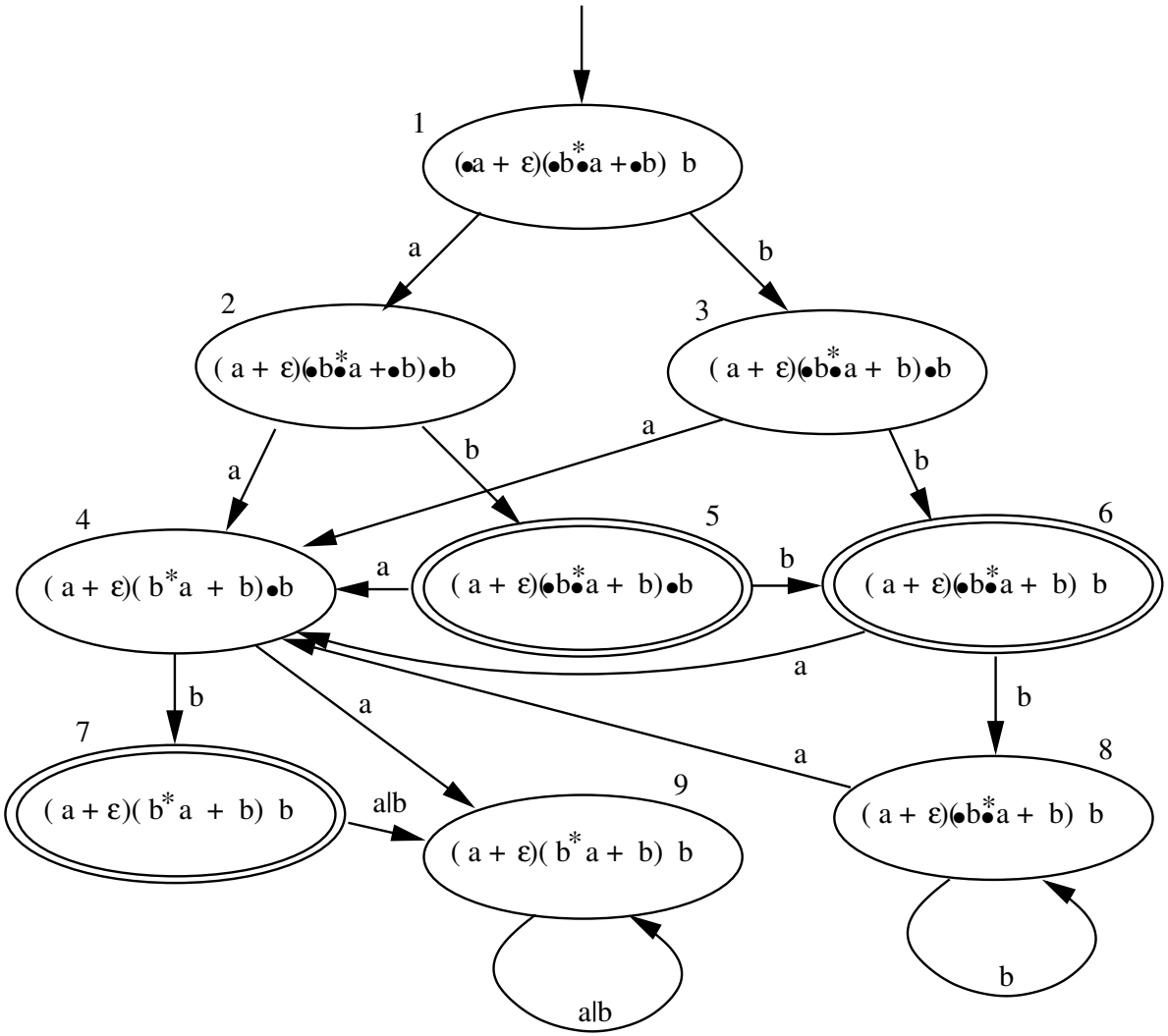
Example: (a+b)*ab



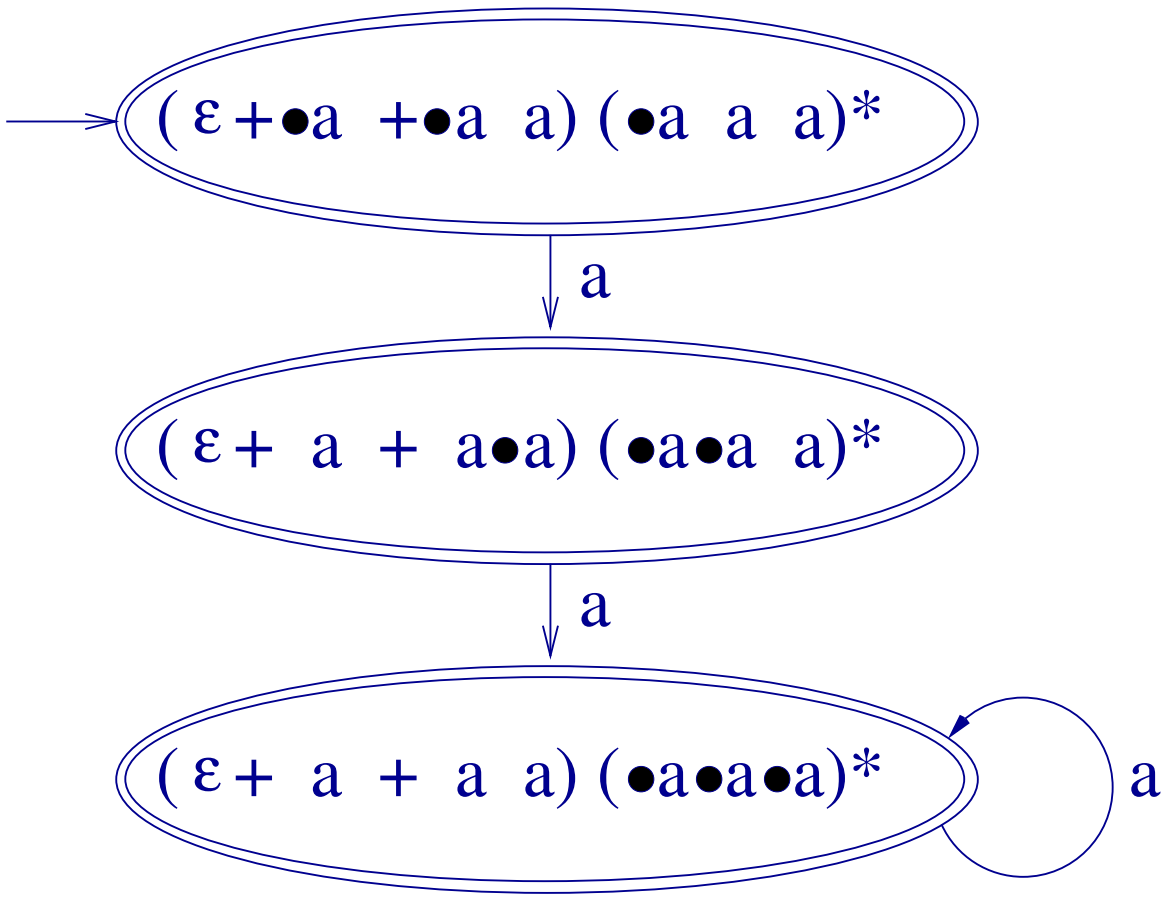
Example: (ac+bc)*



Example:(a+ϵ)(b*a+b)b



Example:$(\epsilon + a + aa)(aaa)^*$



Formal definition

Pointed item:

$$\emptyset, \epsilon, a, \bullet a, i_1 \cdot i_2, i_1 + i_2, i^*$$

Pointed rergular expression (pre):

$$\langle i, b : Bool \rangle$$

b is true if there is a point at the end of the expression.

Semantics

Intuition: Union of all languages starting at the given points.

The carrier $|i|$ of an item i is its underlying r.e.

- $[\![\emptyset]\!] = \emptyset$
- $[\![\epsilon]\!] = \{\epsilon\}$
- $[\![a]\!] = \emptyset$
- $[\![\bullet a]\!] = \{a\}$
- $[\![i_1 + i_2]\!] = [\![i_1]\!] \cup [\![i_2]\!]$
- $[\![i_1 \cdot i_2]\!] = [\![i_1]\!] \cdot [\![|i_2|]\!] \cup [\![i_2]\!]$
- $[\![i^*]\!] = [\![i]\!] \cdot ([\![|i|]\!])^*$

$$[\![\langle i, F\rangle]\!] = [\![i]\!] \qquad [\![\langle i, T\rangle]\!] = [\![i]\!] \cup \{\epsilon\}$$

An important remark

For any i,

$$\epsilon \notin [\![i]\!]$$

hence

$$\epsilon \in [\![\langle i, b \rangle]\!] \Leftrightarrow b = T$$

ϵ-closure

The $\bullet(i)$ operation propagates a point inside an item i.

Remark $\bullet(\_)$ goes from items to pres.

$$\bullet(\emptyset) = \langle \emptyset, F \rangle$$
$$\bullet(\epsilon) = \langle \epsilon, T \rangle$$
$$\bullet(a) = \langle \bullet a, F \rangle$$
$$\bullet(\bullet a) = \langle \bullet a, F \rangle$$
$$\bullet(i_1 + i_2) = \bullet(i_1) \oplus \bullet(i_2)$$
$$\bullet(i_1 \cdot i_2) = \bullet(i_1) \triangleright i_2$$
$$\bullet(i^*) = \langle (fst(\bullet(i)))^*, T \rangle$$

where

$$\langle i_1, b_1 \rangle \oplus \langle i_2, b_2 \rangle = \langle i_1 + i_2, b_1 \vee b_2 \rangle$$

and

$$e_1 \triangleright i_2 =$$
$$\text{let } \langle i_1, b_1 \rangle = e_1 \text{ in}$$
$$\text{if } b_1 \text{ then let } \langle i_2', b_2 \rangle = \bullet(i_2) \text{ in } \langle i_1 \cdot i_2', b_2 \rangle$$
$$\text{else } \langle i_1 \cdot i_2, F \rangle$$

lifted constructions

Similarly to $\oplus$, we can lift concatenation and star from items to pres:

$$
\begin{aligned}
e_1 \odot e_2 \quad = \quad & \text{let } \langle i_2, b_2 \rangle = e_2 \text{ in} \\
& \text{let } \langle i, b \rangle = e_1 \triangleright i_2 \text{ in} \\
& \langle i, b \vee b_2 \rangle
\end{aligned}
$$

$$
\begin{aligned}
e^{\circledast} \quad = \quad & \text{let } \langle i, b \rangle = e \text{ in} \\
& \text{if } b \text{ then } \langle (fst(\bullet(i)))^*, T \rangle \\
& \text{else } \langle i^*, F \rangle
\end{aligned}
$$

Moves

Lifted constructions permit to define moves in a very elegant way:

- $move(\emptyset, a) = \langle emptyset, F \rangle$
- $move(\epsilon, a) = \langle epsilon, F \rangle$
- $move(c, a) = \langle c, F \rangle$
- $move(\bullet c, a) = \langle c, a == c \rangle$
- $move(i_1 + i_2, a) = move(i_1, a) \oplus move(i_2, a)$
- $move(i_1 \cdot i_2, a) = move(i_1, a) \odot move(i_2, a)$
- $move(i^*, a) = move(i, a)^{\circledast}$

Main Results

for all a and w

$$a :: w \in [\![i]\!] \Leftrightarrow w \in [\![move(i, a)]\!]$$

hence

$$w \in [\![i]\!] \Leftrightarrow \epsilon \in [\![move^*(i, w)]\!] = \langle i', b \rangle \Leftrightarrow b = T$$

Related works (theory)

The reference paper for pointed regular expressions is the following report:

- Asperti, Tassi and Sacerdoti Coen. *Regular Expressions, au point*. eprint arXiv:1010.2604, 2010.

A similar notion has been independently introduced in

- Fischer, Huch and Wilke. A play on regular expressions: functional pearl. ICFP 2010, Baltimore, Maryland.

Related works (formalization)

system	approach	reference
COQ	Thompson's algorithm	Braibant and Pous An efficient coq tactic for deciding kleene algebras ITP 2010, LNCS 6172
COQ	partial derivatives	Almeida, Moreira, Pereira and de Sousa Partial Derivative Automata Formalized in Coq IAA 2010, LNCS 6482
Isabelle	Brzozowski's derivatives	Krauss and Nipkow Regular Expression Equivalence and Relation Algebra JAR 2012
Isabelle	partial derivatives	Wu, Zhang, and Urban A formalisation of the myhill-nerode theorem based on regular expressions. ITP 2011, LNCS 6898
SSReflect	Brzozowski's derivatives	Coquand and Siles A decision procedure for regular expression equivalence in type theory. CPP 2011, LNCS 7086

Discussion

- All our proofs have been formalized and checked in Matita
- Due to their algebraic nature, working with pointed expressions at a formal level is a real pleasure.
- Proofs have a strong equational flavor, are short and elegant.
- A self contained snapshot of the Matita library up to the correctness proof of the bisimilarity test takes about 3400 lines; the part concerning regular languages takes less than 1200 lines.

Performance

A couple of examples:

- a version of Bezout's identity

$$\forall n \geq c. \exists x, y. n = xa + yb$$

expressed as the following regular expression problem

$$A(a, b, c) = (0^c)0^* + (0^a + 0^b)^* \simeq (0^a + 0^b)^*$$

- Antimirov's problem, consists in proving the following equality:

$$B(n) = (\epsilon + a + aa + \cdots + a^{n-1})(a^n)^* \simeq a^*$$

Performance

We compare our technique (pres) with that of Coquand&Siles (C&S); execution times have been computed on a machine with a Pentium M Processor 750 1.86GHz and 1GB of RAM.

problem	answer	*pres*	*C&S*
$A(3,5,8)$	yes	0.19	2.09
$A(4,5,11)$	no	0.18	5.26
$A(4,5,12)$	yes	0.24	5.26
$A(5,6,19)$	no	0.30	31.22
$A(5,6,20)$	yes	0.43	31.23
$A(5,7,23)$	no	0.38	70.09
$A(5,7,24)$	yes	0.57	70.19

problem	answer	*pres*	*C&S*
$B(6)$	yes	0.15	0.29
$B(8)$	yes	0.20	1.24
$B(10)$	yes	0.26	3.98
$B(12)$	yes	0.31	10.71
$B(14)$	yes	0.45	25.04
$B(16)$	yes	0.61	53.15
$B(18)$	yes	0.80	104.16

Bibliography


Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman.
Compilers: Principles, Techniques, and Tools.
Pearson Education Inc., 2006.


José Bacelar Almeida, Nelma Moreira, David Pereira, and Simão Melo de Sousa.
Partial derivative automata formalized in coq.
In *Implementation and Application of Automata - 15th International Conference, CIAA 2010, Winnipeg, MB, Canada*, LNCS 6482, pages 59–68. Springer, 2010.


Valentin Antimirov.
Partial derivatives of regular expressions and finite automaton constructions.
Theoretical Computer Science, 155:291–319, 1996.


Andrea Asperti and Jeremy Avigad.
Zen and the art of formalization.
Mathematical Structures in Computer Science, 21(4):679–682, 2011.


Andrea Asperti, Wilmer Ricciotti, Claudio Sacerdoti Coen, and Enrico Tassi.
The Matita interactive theorem prover.
In *Proceedings of the 23rd International Conference on Automated Deduction (CADE-2011), Wroclaw, Poland*, volume 6803 of *LNCS*, 2011.

Bibliography


Andrea Asperti, Wilmer Ricciotti, Claudio Sacerdoti Coen, and Enrico Tassi.
Hints in unification.
In *TPHOLs 2009*, volume 5674 of *LNCS*, pages 84–98. Springer-Verlag, 2009.


Andrea Asperti, Enrico Tassi, and Claudio Sacerdoti Coen.
Regular expressions, au point.
eprint arXiv:1010.2604, 2010.


Gérard Berry and Ravi Sethi.
From regular expressions to deterministic automata.
Theor. Comput. Sci., 48(3):117–126, 1986.


Thomas Braibant and Damien Pous.
An efficient coq tactic for deciding kleene algebras.
In *Proceedings of Interactive Theorem Proving, ITP 2010, Edinburgh, UK*, volume 6172 of *LNCS*, pages 163–178. Springer, 2010.


Anne Brüggemann-Klein.
Regular expressions into finite automata.
Theor. Comput. Sci., 120(2):197–213, 1993.

Bibliography


Chia-Hsiang Chang and Robert Paige.
From regular expressions to dfa's using compressed nfa's.
In *Combinatorial Pattern Matching, Third Annual Symposium, CPM 92, Tucson, Arizona, USA, April 1992, Proceedings*, LNCS 644, pages 90–110. Springer, 1992.


Thierry Coquand and Vincent Siles.
A decision procedure for regular expression equivalence in type theory.
In *Proceedings of Certified Programs and Proofs, CPP 2011, Kenting, Taiwan*, volume 7086 of *Lecture Notes in Computer Science*, pages 119–134. Springer, 2011.


Sebastian Fischer, Frank Huch, and Thomas Wilke.
A play on regular expressions: functional pearl.
In *Proceeding of the 15th ACM SIGPLAN international conference on Functional programming, ICFP 2010, Baltimore, Maryland.*, pages 357–368. ACM, 2010.


Georges Gonthier and Assia Mahboubi.
An introduction to small scale reflection in coq.
Journal of Formalized Reasoning, 3(2):95–152, 2010.


Gérard P. Huet.
Residual theory in lambda-calculus: A formal development.
J. Funct. Program., 4(3):371–394, 1994.

Bibliography


Lucian Ilie and Sheng Yu.
Follow automata.
Inf. Comput., 186(1):140–162, 2003.


Alexander Krauss and Tobias Nipkow.
Proof pearl: Regular expression equivalence and relation algebra.
Journal of Automated Reasoning, published on line, 2011.


R. McNaughton and H. Yamada.
Regular expressions and state graphs for automata.
Ieee Transactions On Electronic Computers, 9(1):39–47, 1960.


Scott Owens, John H. Reppy, and Aaron Turon.
Regular-expression derivatives re-examined.
J. Funct. Program., 19(2):173–190, 2009.


Ken Thompson.
Regular expression search algorithm.
Communications of ACM, 11:419–422, 1968.


Chunhan Wu, Xingyuan Zhang, and Christian Urban.
A formalisation of the myhill-nerode theorem based on regular expressions.
ITP 2011, Berg en Dal, The Netherlands, LNCS 6898, pages 341–356. Springer, 2011.