A Brief Survey of Verified Decision Procedures for Equivalence of Regular Expressions

Tobias Nipkow and Maximilian Haslbeck

Institut für Informatik, Technische Universität München

Abstract. We survey a number of decision procedures for the equivalence of regular expressions that have been formalised with the help of interactive proof assistant over the past few years.

1 Introduction

Equivalence of regular expressions is a perennial topic in computer science. Recently it has spawned a number of papers that have formalised and verified various different algorithm for this task in interactive theorem provers. One of the motivations is that such verified decision procedures can help to automate reasoning about binary relations: relation composition corresponds to concatenation, reflexive transitive closure to Kleene star, and $\cup$ to $+$. It can be shown [9] that an equivalence between two relation algebraic expressions holds if the corresponding two regular expressions are equivalent—the other direction holds too, provided the base type of the relations is infinite.

In this brief note we survey the different formalisations that have appeared over the last few years. We have reproduced most of them in the Isabelle proof assistant and compare some of them on this basis.

Braibant and Pous [4] where the first to verify an equivalence checker for regular expressions. The work was carried out in Coq. They followed the classical approach of translating the regular expressions into automata. The resulting theory was quite large and their algorithm efficient. Although they set the trend, the next four verified decision procedures all worked directly on regular expressions. The motivation is simplicity: regular expressions are a free data type which proof assistants and their users love because it means induction, recursion and equational reasoning, the core competence of proof assistants and functional languages.

The outer shell of all the decision procedures that operate directly on regular expressions is the same. Roughly speaking, there is always a function $\delta : regexp \times \Sigma \to regexp$ that extends canonically to words. Starting from some pair (r, s), all the pairs $(\delta(r, w), \delta(s, w))$ are enumerated; the setup guarantees there are only finitely many $\delta(r, w)$ for each r. If for all such pairs (r', s'), r' is final iff s' is final (for a suitable notion of finality), then $r \equiv s$ (and conversely). This is just an incremental computation of the product automaton or a bisimulation.

2 Derivatives of Regular Expressions

Brzozowski [5] had introduced derivatives of regular expressions in 1964 and had shown that modulo ACI of +, there are only finitely many derivatives of a regular expression, which correspond to the states of a DFA for that regular expression. In 2009, Owens *et al.* [13] used derivatives for scanner generators in ML. They wrote

> derivatives have been lost in the sands of time, and few computer scientists are aware of them.

As we shall see, they have certainly become better known by now.

In response to Braibant and Pous, Krauss and Nipkow [9] verified partial correctness an equivalence checker for regular expressions based on derivatives in Isabelle. The formalization is very small and elegant, although not very efficient for larger problems. Coquand and Siles [6] extended this work in Coq. The emphasis of their work is on the finiteness/termination proof in type theory.

Antimirov [1] introduced partial derivatives of regular expressions. They can be viewed as sets of derivatives, thus building in ACI of +. Moreira *et al.* [11] present an equivalence checker for regular expressions based on partial derivatives and show its total correctness in Coq—termination is proved by showing finiteness of the set of partial derivatives of an expression. We have formalized the same proofs in Isabelle. Moreover we have shown termination of a modified version of the equivalence checker by Krauss and Nipkow as follows. Instead of comparing derivatives normalised w.r.t. ACI of +, we convert them into partial derivatives before comparing them. Thus we can reuse finiteness of the set of partial derivatives to prove termination of the algorithm based on derivatives.

3 Marked Regular Expressions

Both McNaughton and Yamada [10] and Glushkov [8] marked the atoms in a regular expression with numbers in order to turn it into an automaton. Fischer *et al.* [7] realize the convenience of working directly with regular expressions in a functional programming setting. They present matching algorithms on regular expression with boolean marks indicating where in the regular expression the matching process has arrived. Independently, Asperti [2] verifies an equivalence checker for regular expressions via marked regular expressions in the Matita proof assistant. We have verified the basic algorithm by Fischer *et al.* and the one by Asperti in Isabelle and shown that they are closely related: the one by Fischer *et al.* marks the atoms that have just occurred, Asperti marks the atoms that can occur next.

4 Related Formalisations

Moreira *et al.* [12] formalise a decision procedure for Kleene Algebra with Tests as an extension of their earlier work [11] and show its application to program verification by encoding Hoare triples algebraically. Traytel and Nipkow [14] present verified decision procedures for monadic second order logics (MSO) on finite words based on derivatives of regular expressions extended with complementation and projection. Outside of the application area of equivalence checking, Wu *et al.* [15] verify thy Myhill-Nerode theorem in Isabelle using regular expressions. Berghofer and Reiter [3] verify a decision procedure for Presburger arithmetic via automata in Isabelle.

References

1. Antimirov, V.: Partial derivatives of regular expressions and finite automaton constructions. Theor. Comput. Sci. 155, 291–319 (1996)
2. Asperti, A.: A compact proof of decidability for regular expression equivalence. In: Beringer, L., Felty, A. (eds.) ITP 2012. LNCS, vol. 7406, pp. 283–298. Springer, Heidelberg (2012)
3. Berghofer, S., Reiter, M.: Formalizing the logic-automaton connection. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 147–163. Springer, Heidelberg (2009)
4. Braibant, T., Pous, D.: An efficient Coq tactic for deciding Kleene algebras. In: Kaufmann, M., Paulson, L. (eds.) ITP 2010. LNCS, vol. 6172, pp. 163–178. Springer, Heidelberg (2010)
5. Brzozowski, J.: Derivatives of regular expressions. J. ACM 11, 481–494 (1964)
6. Coquand, T., Siles, V.: A decision procedure for regular expression equivalence in type theory. In: Jouannaud, J.-P., Shao, Z. (eds.) CPP 2011. LNCS, vol. 7086, pp. 119–134. Springer, Heidelberg (2011)
7. Fischer, S., Huch, F., Wilke, T.: A play on regular expressions. Functional pearl. In: Hudak, P., Weirich, S. (eds.) Proc. Int. Conf. Functional Programming, ICFP 2010, pp. 357–368 (2010)
8. Glushkov, V.M.: The abstract theory of automata. Russian Mathematical Surveys 16, 1–53 (1961)
9. Krauss, A., Nipkow, T.: Proof pearl: Regular expression equivalence and relation algebra. J. Automated Reasoning 49, 95–106 (2012) (published online March 2011)
10. McNaughton, R., Yamada, H.: Regular expressions and finite state graphs for automata. IRE Trans. on Electronic Comput. EC-9, 38–47 (1960)
11. Moreira, N., Pereira, D., de Sousa, S.M.: Deciding regular expressions (in-)equivalence in Coq. In: Kahl, W., Griffin, T.G. (eds.) RAMiCS 2012. LNCS, vol. 7560, pp. 98–113. Springer, Heidelberg (2012)
12. Moreira, N., Pereira, D., de Sousa, S.M.: Mechanization of an algorithm for deciding KAT terms equivalence. Tech. Rep. DCC-2012-04, Universidade do Porto (2012)
13. Owens, S., Reppy, J.H., Turon, A.: Regular-expression derivatives re-examined. J. Functional Programming 19, 173–190 (2009)
14. Traytel, D., Nipkow, T.: A verified decision procedure for MSO on words based on derivatives of regular expressions. In: Proc. Int. Conf. Functional Programming, ICFP 2013 (2013)
15. Wu, C., Zhang, X., Urban, C.: A formalisation of the Myhill-Nerode theorem based on regular expressions (Proof pearl). In: van Eekelen, M., Geuvers, H., Schmaltz, J., Wiedijk, F. (eds.) ITP 2011. LNCS, vol. 6898, pp. 341–356. Springer, Heidelberg (2011)