1 Introduction

Our goal is to give a concise formalization of the theory of regular languages. We include several different characterizations of regular languages and prove them equivalent. These include regular expressions, finite automata and the characterizations usually combined in the Myhill-Nerode theorem. All our proofs are constructive and, thus, constitute procedures to convert between these characterizations. Our formalization also includes decision procedures for the equivalence of finite automata and regular expressions.

Regular languages are a well-studied class of formal languages. In their current form, they were first studied by Kleene [23], who introduced regular expressions. The concept of deterministic finite automata was introduced before Kleene's invention of regular expressions by Huffman [21] and Moore [27]. Rabin and Scott later introduced the concept of non-deterministic finite automata [29], for which they were given the Turing award [4].

We take classical proofs from [24] and translate them to our constructive setting. We employ the proof assistant Coq [26] for our formalization. Additionally, we make use of the SSReflect plugin [18]. SSReflect offers an extended scripting language and provides support for finite types, which we use for formalizing finite automata. It also comes with a lot of general infrastructure useful for our purpose. For every lemma and theorem proven in this thesis, we provide corresponding statements from the Coq development. Our development does not depend on axioms.

One of the most interesting parts of the formalization was to find a suitable representation of quotient types in Coq, which has no notion of quotient types. Our approach seems to work very well for our purpose.

1.1 Related work

There have been many publications on formalizations of the theory of regular languages in recent years. Most of them investigate decidability of equivalence of regular expressions, often with a focus on automatically deciding Kleene algebras.

Coquand and Siles develop a decision procedure for equivalence of regular expressions [16] on the basis of Brzozowski derivatives [14] in Coq (using SSReflect) with the goal of providing a practically executable tactic on top of the decision procedure. Their development weighs in at 7,500 lines of code, 700 of which serve as the basis of our formalization.

Krauss and Nipkow give a decision procedure for equivalence of regular expressions in Isabelle/HOL [25]. Their development is very concise with just over 1,000 lines of code. Being interested only in a correct (and efficient) tactic for deciding equivalences, they did not prove completeness and termination.

Another decision procedure for equivalence of regular expressions is developed by Braibant and Pous [12], with the goal of deciding Kleene algebras in Coq. Their formalization is based on matrices and comprises 19,000 lines of code. It encompasses finite automata, regular expressions and the Myhill-Nerode theorem.

Moreira, Pereira and Sousa give a decision procedure for equivalence of regular expressions in Coq [28]. Their development is based on Antimirov's partial derivatives of regular expressions [2] and contains a refutation step to speed up inequality checking. It consists of 19,000 lines of code.

Asperti formalizes a decision procedure for equivalence of regular expressions [5] based on the notion of pointed regular expressions [6]. This development was done in the Matita proof assistant [7]. It weighs in at 3,400 lines of code.

There is also a paper by Wu, Zhang and Urban on formalizing the Myhill-Nerode theorem using only regular expressions and not, as is commonly done, finite automata [31]. The authors state that this unusual choice stems, at least partly, from the restrictions of Isabelle/HOL (and similar HOL-based theorem provers). In particular, the fact that Isabelle/HOL does not allow for quantification over types prevents straight-forward formalizations of finite automata. Their development consists of roughly 2,000 lines of code.

1.2 Contributions

Our formalization is done in constructive type theory. Thus, all our proofs are algorithms that are, in theory, executable. However, our focus is solely on clarity and simplicity. As a result, the algorithms and procedures given in this thesis are very close to the mathematical definitions in [24], but not executable in practice.

Our development shows that Coq (particularly with SSReflect) is well suited for this kind of formalization. Furthermore, we have also developed a new characterization derived from the Nerode relation and proven it equivalent to all other characterizations. Our development weighs in at about 2,700 lines of code.

1.3 Outline

In Chapter 2 we give a brief introduction to Coq and SSReflect and introduce concepts that are relevant to our formalization.

In Chapter 3 we give basic definitions (words, languages, etc.). We also introduce decidable languages, regular languages, and regular expressions. Furthermore, we prove the decidability of regular languages.

In Chapter 4 we introduce finite automata. We prove the equivalence of deterministic and non-deterministic finite automata. We also give a procedure to remove unreachable states from deterministic finite automata. Furthermore, we prove decidability of emptiness and equivalence of finite automata. Finally, we prove that regular expressions and finite automata are equally expressive.

In Chapter 5 we introduce the Myhill-Nerode theorem. We give three different characterizations of regular languages based on the Myhill-Nerode theorem and prove them all equally expressive to finite automata.