Theoretical Computer Science 534 (2014) 24–44

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Adding pebbles to weighted automata: Easy specification & efficient evaluation ☆

Paul Gastin, Benjamin Monmege *

LSV, ENS Cachan, CNRS, Inria, France

ARTICLE INFO

Keywords:
Weighted automata
Weighted expressions
Specification of weighted properties
Kleene–Schützenberger theorem
Reusable pebbles

ABSTRACT

We extend weighted automata and weighted rational expressions with 2-way moves and reusable pebbles. We show with examples from natural language modeling and quantitative model-checking that weighted expressions and automata with pebbles are more expressive and allow much more natural and intuitive specifications than classical ones. We extend Kleene–Schützenberger theorem showing that weighted expressions and automata with pebbles have the same expressive power. We focus on an efficient translation from expressions to automata. We also prove that the evaluation problem for weighted automata can be done very efficiently if the number of reusable pebbles is low.

1. Introduction

Regular expressions are used to specify patterns. They are popular because they propose a concise and intuitive way of denoting such patterns. Therefore, they have a long history in the formal language community. A seminal result, known as Kleene's theorem, establishes that the (denotational) regular expressions have the same expressive power as the (operational) finite state automata. Efficient translation algorithms of regular expressions into finite automata are crucial since expressions are convenient to denote patterns and automata are amenable to efficient algorithms. Regular expressions and finite automata have been extended in several directions, e.g., tree (walking) automata, (regular) XPath, etc.

Nowadays, quantitative models and analysis are intensively studied, resulting in a revision of the foundation of computer science. The essence of the quantitative approach is that yes/no answers from the classical Boolean framework are replaced by quantities such as probability, energy consumption, reliability, cost, etc. In the 1960s, Schützenberger provided a generic way of turning qualitative into quantitative systems, starting the theory of weighted automata [38] (see [21,17,4] for recent books on this theory). Indeed, probabilistic automata and word transducers appear as instances of that framework, which found its way into numerous application areas such as natural language processing [26], speech recognition [32] or digital image compression [1]. Schützenberger proved the equivalence between weighted automata and weighted regular expressions, extending Kleene's theorem. Various translation algorithms can be extended from the Boolean framework to the weighted case, see [34,36] for surveys about these methods, and [28] which obtains Schützenberger's theorem as a corollary of Kleene's theorem.

In Sections 4 and 5, we extend weighted expressions and automata with 2-way moves and pebbles which follow a stack policy. There are several motivations for these extensions. First, as shown in Section 2 for applications in natural language

☆ Supported by LIA INFORMEL.
* Corresponding author.
E-mail addresses: paul.gastin@lsv.ens-cachan.fr (P. Gastin), benjamin.monmege@lsv.ens-cachan.fr (B. Monmege).

http://dx.doi.org/10.1016/j.tcs.2014.02.034
0304-3975/

processing and quantitative model-checking, 2-way moves and pebbles allow more natural and more concise descriptions of the quantitative expressions we need to evaluate. Second, in the weighted case, 2-way and pebbles do increase the expressive power as already observed in [9] in relation with weighted logics or in [33] in the probabilistic setting. This is indeed in contrast with the Boolean case where 2-way and pebbles do not add expressive power over words (see, e.g., [23]) even though they allow more succinct descriptions (see, e.g., [5]). Our work is also inspired by pebble tree-walking automata and in particular their links with powerful logics, XPath formalisms and caterpillar expressions on trees [19,11,7,37,6]. Compared to these works, our pebbles are *weak*, meaning that when a pebble is lifted, the head of the automaton goes back to the position where the pebble was dropped. Whereas in the Boolean case, *strong* pebbles (i.e., pebbles that may be lifted from any position and without moving the head) are equivalent to weak ones, this is still an open question in our weighted case. Moreover, we consider *reusable pebbles*, which may be considered as extensions of the *tree-walking automata with invisible pebbles*, introduced in [20], to the weighted setting and to nested words. This permits to better describe the complexity of our algorithms.

In Sections 6 and 7, we generalize Kleene and Schützenberger theorems to weighted expressions and *layered* automata with 2-way moves and pebbles (the layered hypothesis restricts reusable pebbles to a bounded supply). We establish their expressive power equivalence by providing effective translations in both directions. Showing how to transform an *operational* automaton into an equivalent *denotational* expression is indeed very interesting from a theoretical point of view, but is less useful in practice. On the other hand, we need highly efficient translations from the convenient denotational formalism of expressions to operational automata which, as stated above, are amenable to efficient algorithms. Efficiency is measured both with respect to the size of the resulting automaton, and the space and time complexities of the translation. We show that, Glushkov's [24] or Berry and Sethi [3] translations, which are among the best ones in the Boolean case, can be extended to weighted expressions with 2-way moves and reusable pebbles. The constructions for the rational operations (sum, product, star) can be adapted easily to cope with 2-way moves, even though the correctness proofs are more involved and require new theoretical grounds such as series over a partial monoid as explained in Section 4.1. The main novelty in Sections 6 and 7 is indeed the treatment of pebbles in the translations between expressions and automata.

To complete the picture, we study in Section 8 the evaluation problem of a layered weighted automaton with 2-way moves and reusable pebbles over a given word. The algorithm is polynomial in the size of the word, where the degree is 1 plus the number of reusable pebbles: in particular, this does not depend on the actual number of layers, i.e., the total number of pebbles that the automaton may use. We can even decrease the degree by 1 for *strongly layered* automata. This applies when we only have one reusable pebble, and we obtain an algorithm which is linear in the size of the input word. This is in particular the case for automata derived from weighted LTL.

The paper includes intuitive explanations and examples for a better understanding of weighted expressions with 2-way moves and pebbles, and of the translations between automata and expressions. An extended abstract of this work appeared in [22].

2. Motivations

We give in this section two motivating examples for studying weighted expressions and automata with 2-way moves and pebbles.

2.1. Language modeling

Since decades, weighted automata have been extensively used in Natural Language Processing (see [26]), in particular for automatic translation, speech recognition or transliteration. All these tasks have in common to split the problem into independent parts, certain directly related to the specific task and others related to the knowledge of the current language. For example, in the translation task from French sentences to English sentences, one splits the problem into first knowing translation of single words and then modeling English sentences (knowledge which is independent from the translation task). It has to be noticed that the problem of translation (decoding) is NP-complete, even for a simple translation model, as shown in [25]. We generally compute approximations using statistical methods. The second part, namely to know whether a sequence of words is a *good* English sentence, is known as *language modeling*. Often this knowledge is learned from a large corpus of English texts, and stored into a formal model, e.g., a weighted finite state automaton representing the probability distribution $\mathbb{P}$ of well-formed English sentences. The translation task is then resolved by first generating several English sentences from the original French one (due to ambiguity of the word-by-word translation task), and then choosing among this set of sentences the ones with highest probability.

One broadly used language model is the n-gram model, where the probability of a word in a sentence depends only on the previous $n-1$ words. For example in a 1-gram model, only the individual word frequencies are relevant to generate well-formed English sentences, whereas in a 2-gram model, the probability of a word depends on the very same frequency distribution and also the previous word. To formally describe these models, and further study them, let us define them using regular expressions. Let D denote the dictionary of words in the language. Suppose we are given the conditional probability distributions $\mathbb{P}(u_n \mid u_1, \dots, u_{n-1})$ in the n-gram model (with $u_i \in D$ for all i). The probability of a sentence $(u_i)_{1 \leqslant i \leqslant m} \in D^m$ can be given by the following weighted regular expression in a 1-gram model and a 3-gram model: