How references may establish a sound foundation of an article (and maybe a theses)

Hvordan referanser kan etablere grunnlaget for en artikkel (og en masteroppgave)
On the Minimality of Finite Automata and Stream X-machines for Finite Languages

Florentin Ipate

Department of Computer Science and Mathematics, University of Pitești, Str Targu din Vale 1, 0300 Pitești, Romania
Email: fpate@iissoft.ro

A cover automaton of a finite language $L$ is a finite automaton that accepts all words in $L$ and possibly other words that are longer than any word in $L$. An algorithm for constructing a minimal cover automaton of a finite language $L$ is given in a recent paper. This paper goes a step further by proposing a procedure for constructing all minimal cover automata of a given finite language $L$. The concept of cover automaton is then generalized to a form of extended finite automaton, the stream X-machine, and the procedure is extended to this more general model.

Received 8 January 2004; revised 30 September 2004
1. INTRODUCTION

Finite automata [1, 2, 3] are widely used in various aspects of computing, ranging from lexical analysis to protocol testing. Finite automata are known to accept regular languages [4, 5]. However, in many applications, finite automata only finite languages are used. Specifically, the number of states of a finite automaton (FA) that accepts a given finite language is at least one more than the length of the longest word in the language and may be exponentially large. On the other hand, if we do not restrict the FA to accept only the given finite language but allow it to accept extra words that are longer than the longest word in the language, then the number of its states may be greatly reduced. In most applications the maximum number of words in the language is known and the system is capable of accepting words of the length of the words processed, so this approximation will usually be adequate. This is the intuition behind automata for finite languages.

Informally, a cover automaton of a finite language $L$ is an FA that accepts all words in $L$ and possibly other words that are longer than any word in $L$. A minimal cover automaton of $L$ is a cover automaton of $L$ having the least number of states. In many cases, a minimal cover automaton of $L$ has a much smaller size than the minimal automaton that accepts $L$.

The concept of minimal cover automaton of a finite language is introduced in [6] and it is shown that there may be several minimal cover automata of the same language that are not isomorphic. Furthermore, [6] provides an algorithm that, for a finite language $L$ (given as an FA that accepts $L$ or as a cover automaton of $L$), constructs a minimal cover automaton of the language. An improved algorithm (in terms of complexity) is also presented in [7].

This paper goes a step further by giving a procedure for constructing all minimal cover automata of a given finite language. The procedure is not restricted to specific applications of the automata and may be used to construct automata for other purposes.

REFERENCES


Investigated techniques for refining given specifications into more complex, more detailed implementation-oriented versions have been developed [14, 15]. Furthermore, several models of communicating SXM, mones have been devised and used in real applications [16, 17, 18].

One of the strengths of using SXM to specify a system is that it is possible to derive test sets from an SXM specification which, if satisfied, guarantee, under certain constraints, the correctness of the implementation with respect to the specification [10, 19, 20, 21]. Among these constraints are the so-called ‘design for test conditions’ that the SXM specification has to meet: input-completeness and output-distinguishability [10, 19]. The class of SXM's that meet these conditions is therefore of particular interest and has
1. INTRODUCTION

Finite automata [1, 2, 3] are widely used in various areas of computing, ranging from lexical analysis in compilers to protocol testing. Finite automata are known to accept regular languages [4, 5]. However, in many applications of finite automata only finite languages are used. The number of states of a finite automaton (FA) that accepts a finite language is at least one more than the length of the longest word in the language and may be exponentially large in this length [6]. On the other hand, if we do not restrict the automata to accept only the given finite language but may also accept extra words that are longer than the longest word in the language, then the number of its states may be reduced. In most applications the maximum number of words in the language is known and the system will usually be adequate. This is the idea behind cover automata for finite languages.

Informally, a cover automaton of a finite language \( L \) is an FA that accepts all words in \( L \) and possibly other words that are longer than any word in \( L \). A minimal cover automaton of \( L \) is a cover automaton of \( L \) having the least number of states. In many cases, a minimal cover automaton of \( L \) has a much smaller size than the minimal automaton that accepts \( L \).

The concept of minimal cover automaton of a finite language is introduced in [6] and it is shown that there may be several minimal cover automata of the same language that are not isomorphic. Furthermore, [6] provides an algorithm that, for a finite language \( L \) (given as an FA that accepts \( L \) or as a cover automaton of \( L \)), constructs a minimal cover automaton of the language. An improved algorithm (in terms of complexity) is also presented in [7].

---


---

A specification method, especially for interactive systems, A tool to support the creation of SXM specifications has been constructed [13]. The refinement of SXM has been investigated and techniques for refining given specifications into more complex, more detailed implementation-oriented versions have been developed [14, 15]. Furthermore, several models of communicating SXM have been devised and used in real applications [16, 17, 18].

One of the strengths of using SXM to specify a system is that it is possible to derive test sets from an SXM specification which, if satisfied, guarantee, under certain constraints, the correctness of the implementation with respect to the specification [10, 19, 20, 21]. Among these constraints are the so-called ‘design for test conditions’ that the SXM specification has to meet: input-completeness and output-distinguishability [10, 19]. The class of SXM that meet these conditions is therefore of particular interest and has
The concept of minimal cover automaton of a finite language is introduced in [6] and it is shown that there may be several minimal cover automata of the same language that are not isomorphic. Furthermore, [6] provides an algorithm that, for a finite language $L$ (given as an FA that accepts $L$ or as a cover automaton of $L$), constructs a minimal cover automaton of the language. An improved algorithm (in terms of complexity) is also presented in [7].

In most applications the maximum length of the words in the language will usually be adequate. The length of the word in a language will be an upper bound of the length of the words in the language.

Informally, a cover automaton of $L$ is a FA that accepts all words in $L$. Cover automata of $L$ are called $L$-cover automata.

In many cases, a minimal cover automaton has a smaller size than the maximum number of states in all cover automata of $L$.

The concept of minimal cover automaton of a finite language is introduced in [6] and it is shown that there may be several minimal cover automata of the same language that are not isomorphic. Furthermore, [6] provides an algorithm that, for a finite language $L$ (given as an FA that accepts $L$ or as a cover automaton of $L$), constructs a minimal cover automaton of the language. An improved algorithm (in terms of complexity) is also presented in [7].


1. INTRODUCTION

Finite automata [1, 2, 3] are widely used in many areas of computing, ranging from lexical analysis to circuit and protocol testing. Finite automata are known to compute regular languages [4, 5]. However, in many applications of finite automata only finite languages are used. The number of states of a finite automaton (FA) that accepts a finite language is at least equal to the length of the language. On the other hand, a finite automaton accepts all regular languages extra time. Therefore, the number of states of the automaton will be the state complexity of the language. The number of states of an FA that accepts a finite language is equal to the number of states of an automaton that accepts all regular languages extra time. In many cases, a smaller number of states is needed.

The language $L$ is a cover automaton of $L$ having the least number of states. In many cases, a smaller number of states is needed.

This paper goes a step further by giving a procedure for constructing all minimal cover automata of a given finite language $L$. The procedure is then generalized to a form of extended finite automata, called stream X-machines (SXMs).

An SXM is a type of X-machine [8, 9, 10] that describes a system as a finite set of states, each with an internal store called memory, and a number of transitions between the states. A transition is triggered by an input value.


These conditions is therefore of particular interest and has