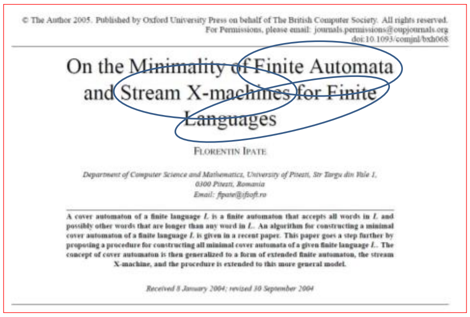
Hvordan referanser kan etablere grunnlaget for en artikkel (og en masteroppgave)

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



doi:10.1093/comjnl/bxh068

On the Minimality of Finite Automata and Stream X-machines for Finite Languages

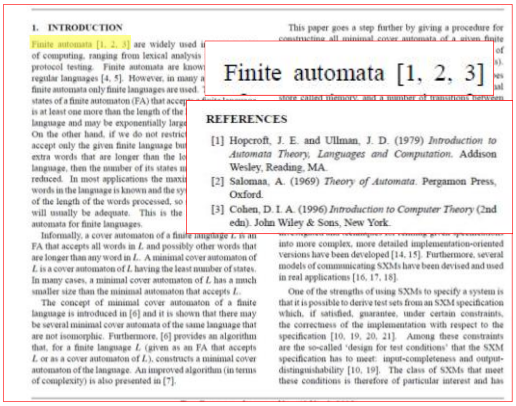
FLORENTIN IPATE

Department of Computer Science and Mathematics, University of Pitesti, Str Targu din Vale 1, 0300 Pitesti, Romania
Email: fipate@ifsoft.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

(The Computer Journal, 48(2), 157-167)



1. INTRODUCTION

Finite automata [1, 2, 3] are widely used i
of computing, ranging from lexical analysis
protocol testing. Finite automata are know
regular languages [4, 5]. However, in many a
finite automata only finite languages are used.
states of a finite automaton (FA) that accep
is at least one more than the length of the
language and may be exponentially large
On the other hand, if we do not restric
accept only the given finite language bu
extra words that are longer than the lo
language, then the number of its states m
reduced. In most applications the maxi
words in the language is known and the sy
of the length of the words processed, so
will usually be adequate. This is the
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

store called memory, and a number of transitions between

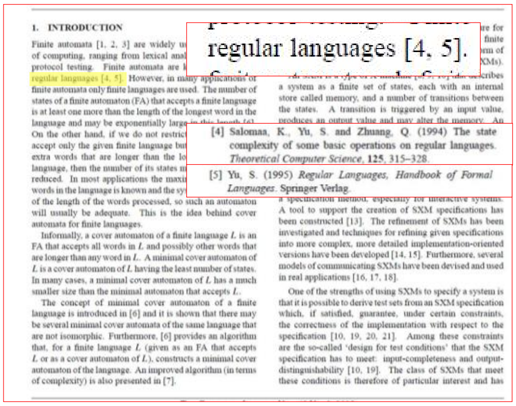
into more complex, more detailed implementation-oriented versions have been developed [14, 15]. Furthermore, several models of communicating SXMs have been devised and used in real applications [16, 17, 18].

One of the strengths of using SXMs 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 SXMs that meet these conditions is therefore of particular interest and has

Finite automata [1, 2, 3]

REFERENCES

[1] Hopcroft, J. E. and Ullman, J. D. (1979) *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, Reading, MA.
[2] Salomaa, A. (1969) *Theory of Automata*. Pergamon Press, Oxford.
[3] Cohen, D. I. A. (1996) *Introduction to Computer Theory* (2nd edn). John Wiley & Sons, New York.



1. INTRODUCTION

Finite automata [1, 2, 3] are widely us
of computing, ranging from lexical anal
protocol testing. Finite automata are k
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
On the other hand, if we do not restric
accept only the given finite language bu
extra words that are longer than the lo
language, then the number of its states m
reduced. In most applications the maxi
words in the language is known and the sy
of the length of the words processed, so such an automaton
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].

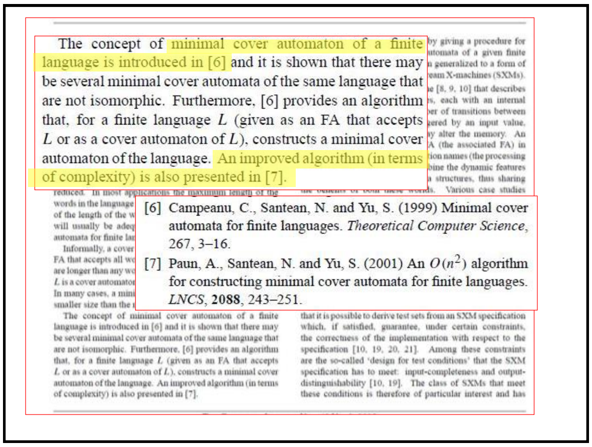
a system as a finite set of states, each with an internal store called memory, and a number of transitions between the states. A transition is triggered by an input value, produces an output value and may alter the memory. An

A tool to support the creation of SXM specifications has been constructed [13]. The refinement of SXMs 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 SXMs have been devised and used in real applications [16, 17, 18].

One of the strengths of using SXMs 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 SXMs that meet these conditions is therefore of particular interest and has

regular languages [4, 5].

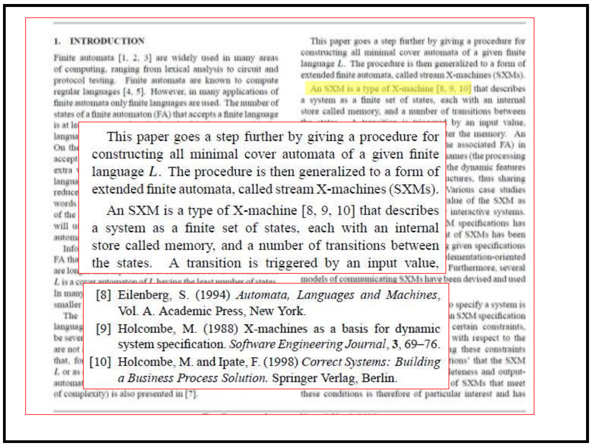
[4] Salomaa, K., Yu, S. and Zhuang, Q. (1994) The state complexity of some basic operations on regular languages. *Theoretical Computer Science*, **125**, 315–328.
[5] Yu, S. (1995) *Regular Languages, Handbook of Formal Languages*. Springer Verlag.



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].

[6] Campeanu, C., Santean, N. and Yu, S. (1999) Minimal cover automata for finite languages. *Theoretical Computer Science*, 267, 3–16.

[7] Paun, A., Santean, N. and Yu, S. (2001) An $O(n^2)$ algorithm for constructing minimal cover automata for finite languages. *LNCS*, **2088**, 243–251.



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,

[8] Eilenberg, S. (1994) *Automata, Languages and Machines*, Vol. A. Academic Press, New York.

[9] Holcombe, M. (1988) X-machines as a basis for dynamic system specification. *Software Engineering Journal*, **3**, 69–76.

[10] Holcombe, M. and Ipate, F. (1998) *Correct Systems: Building a Business Process Solution.* Springer Verlag, Berlin.