ICIC Express Letters ICIC International ©2016 ISSN 1881-803X
Volume **10**, Number **11**, November **2016** pp. 2587–2594

SECURE FINITE AUTOMATON: EQUIVALENCE, DETERMINIZATION AND MINIMIZATION

Pinghong Ren, Chu Chen, Yuanke Zhang, Jiguo Yu and Fengyin Li
School of Information Science and Engineering
Qufu Normal University
No. 80, Yantai North Road, Rizhao 276826, P. R. China
phren@qfnu.edu.cn

Received March 2016; accepted June 2016

Abstract. *With the development of new systems, demands for analyzing and processing corresponding properties such as security are growing rapidly. As a kind of powerful tool, traditional automata cannot meet these demands without development. Thus, secure finite automaton is put forward so that security properties can be coped with. Equivalence of secure finite automata is defined based on the secure bisimulation. Algorithms for determinizing and minimizing a secure finite automaton are given with proofs of correctness and analyses of the worst case time complexity. The application of these algorithms shows that this development is very useful.*
Keywords: Secure finite automaton, Equivalence, Determinization, Minimization

1. **Introduction.** As the basis of computer science and technology [1, 2], automata have wide applications such as image analysis [3]. According to determinacy, automata can be roughly divided into two classes: deterministic finite automaton (DFA) and non-deterministic finite automaton (NFA). Traditionally, a DFA is equivalent to an NFA if they accept the same language. However, different views may come from different domains. In [4], Milner gave an example of a coffee/tea vending machine. In this example, one is deterministic and the other is non-deterministic, as shown in Figure 1(a) and Figure 1(b) respectively. They are equivalent in the sense of accepted languages. However, when a purchaser who needs a cup of coffee puts in two pounds (denoted by 2p in Figure 1), the NFA (Figure 1(b)) may change to the state s_2 in which only a cup of tea can be offered and the purchaser has no choice but the tea. This case will never occur in the DFA (Figure 1(a)). Language equivalent automata behave so differently because determinism and non-determinism of internal actions can affect interactions while traditional equivalence has no consideration of this point. Thus, Milner proposed the concepts of bisimulation and weak bisimulation in [4]. Weakly bisimular systems have the same language, not vice versa.

With the development of new systems especially interactive systems of electronic commerce, security is becoming more and more important. In [5], Liu and Jiang found that security policy can affect interacting behaviors and that better security policies make systems securer. Weak bisimulation cannot explain their findings because weak bisimulation does not take into consideration the effect of external actions on systems. Therefore, in the same work, Liu and Jiang proposed secure bisimulation as an extension of weak bisimulation for interactive systems. They modeled systems with a labeled Petri net with insecure places (LPNIP) and used the reachability graph of an LPNIP to define a labeled transition system with insecure states (LTSIS).

In [6], Zhang and Wu proposed an algorithm for minimizing bounded Petri nets. Their algorithm calls traditional algorithms to determinize and minimize a labeled transition system (LTS) of a labeled Petri net (LPN). If the same algorithms are applied to an LPNIP directly, incorrect results will be arrived at. For example, Figure 2 shows two LPNIPs: p_2

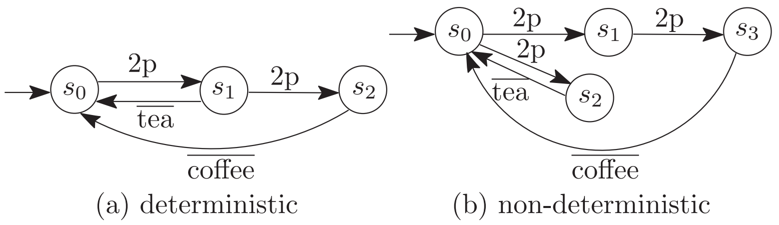

Figure 1. Milner's example

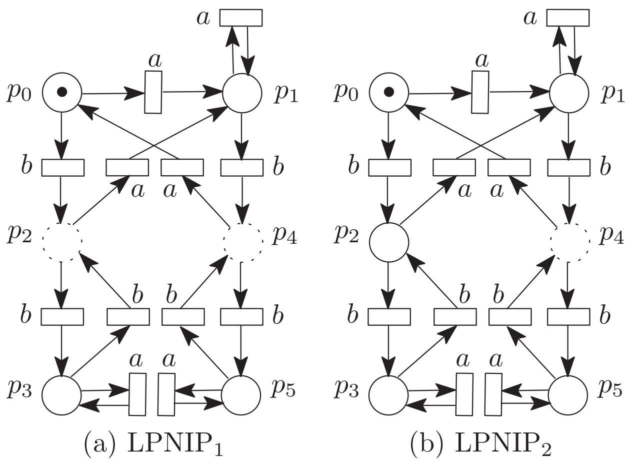

Figure 2. Two LPNIPs

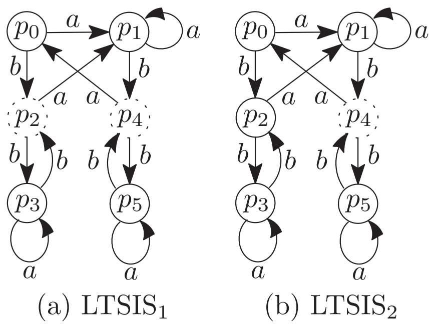

Figure 3. LTSISs

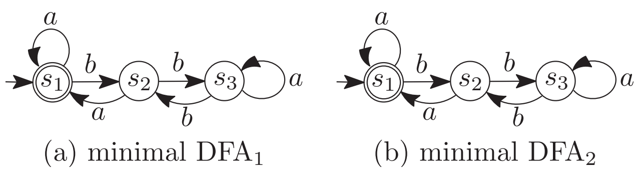

Figure 4. Minimal DFAs

and p_4 are insecure places in LPNIP$_1$ and p_4 is an insecure place in LPNIP$_2$. Obviously these two LPNIPs are not securely bisimular. Figure 3 shows corresponding LTSISs: p_2 and p_4 are insecure states in LTSIS$_1$ and p_4 is an insecure state in LTSIS$_2$. However, the same minimization results (Figure 4) are produced. It means minimal DFAs of the LTSISs are securely bisimular for reflexivity, contrary to the fact that the LTSISs are not securely bisimular. The reason for this incorrectness is that traditional automata and algorithms for determinization and minimization cannot meet the demand of changes. Although new automata and algorithms for minimization are constantly emerging [7, 8], to the best of our knowledge, none of them takes security into account. To address this problem, secure finite automaton, as an extension of traditional automata, is put forward with the equivalence and algorithms for determinization and minimization. With secure finite automaton, systems, especially security, involved can be modeled directly and models such as LPNIP and LTSIS can be minimized correctly in consideration of secure bisimulation. Furthermore, the comparison of security levels [5] among different systems especially using minimal models can arrive at correct results.

2. **Preliminary.**

An NFA is denoted by $\mathcal{N}$ and a DFA is denoted by $\mathcal{D}$. Both NFA and DFA are called finite automaton (FA) denoted by $\mathcal{A}$. The language of $\mathcal{A}$ denoted by $L(\mathcal{A})$ is the set of all sentences recognized by $\mathcal{A}$. Two finite automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are equivalent if $L(\mathcal{A}_1) = L(\mathcal{A}_2)$. Note that the above equivalence is fully based on the languages accepted by automata. Thus, this kind of equivalence is often referred to as language equivalence. The process to convert an NFA to an equivalent DFA is called determinization. One classical determinization algorithm is the subset construction (i.e., the power set construction). A DFA is the minimal DFA among equivalent deterministic finite automata if no other equivalent DFA has less states. The minimal DFA is denoted by $\text{DFA}_{\min}$ or $\mathcal{D}_{\min}$. The process to convert a DFA to a minimal DFA is called minimization. Two kinds of algorithms can be used to minimize a DFA. One is the split algorithm such as Hopcroft's with the worst time complexity $\mathcal{O}(n \log n)$ [9]. The other is Brzozowski and Tamm's double reversal algorithm with the exponential time complexity as the worst case [10]. For more details about FA, please refer to [1, 2, 10].

LPNIP differs from the classical Petri net [11] in that some places of LPNIP are insecure and these insecure ones have no effect on the rules of enabling and firing transitions. LTSIS is based on the reachability graph of an LPNIP and provides the basis of secure bisimulation. Let $(Q_S \cup Q_U, Act, Tr)$ be an LTSIS, where Q_S is the set of secure states and Q_U is the set of insecure states. Binary relation $B \subseteq ((Q_S \cup Q_U) \times (Q_S \cup Q_U))$ is a secure bisimulation if: (1) B is symmetric; (2) if $(q, r) \in B$ and $q \xrightarrow{\epsilon} q'$, then there exists r' such that $r \overset{\epsilon}{\rightsquigarrow} r'$ and $(q', r') \in B$, where ϵ can be omitted and $q_1 \xrightarrow{\epsilon} q_2 \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} q_n$ is denoted by $q_1 \overset{\epsilon}{\rightsquigarrow} q_n$; (3) if $(q, r) \in B$ and $q \xrightarrow{a} q'$, then there exists r' such that $r \overset{a}{\rightsquigarrow} r'$ and $(q', r') \in B$, where $q_1 \overset{\epsilon}{\rightsquigarrow} q_n \xrightarrow{a} q_{n+1}$ is denoted by $q_1 \overset{a}{\rightsquigarrow} q_{n+1}$; (4) if $(q, r) \in B$ and $q \in Q_U$, then $r \in Q_U$. For more details about secure bisimulation, please refer to [5].

3. **Secure Finite Automaton and Equivalence.**

Definition 3.1. *A secure deterministic finite automaton (SDFA, denoted by $\mathcal{D}_S$) is a 7-tuple: $(Q, \Sigma, S, \lambda, \delta, s_0, F)$, where Q is a finite and non-empty set of states, Σ is a finite alphabet, S is either secure denoted by 1 or insecure denoted by 0 (i.e., $S = \{0, 1\}$), λ is a label function: $Q \to S$, δ is a transition function: $Q \times \Sigma \to Q$, s_0 is the only one initial state and $s_0 \in Q$, and F is a finite set of final states and $F \subseteq Q$.*

An SDFA labels its states as secure and insecure. Any state of an SDFA has at most one outgoing arc for any label $a \in \Sigma$. For example, Figure 5 shows two secure deterministic finite automata corresponding to Figure 3. If all states of Q are secure, an SDFA is a traditional DFA in essence. Therefore, traditional DFA can be viewed as a special case of SDFA.

Definition 3.2. *A secure non-deterministic finite automaton (SNFA, denoted by $\mathcal{N}_S$) is a 7-tuple: $(Q, \Sigma, S, \lambda, \delta, I, F)$, where Q is a finite and non-empty set of states, Σ is a finite alphabet, S is either secure denoted by 1 or insecure denoted by 0 (i.e., $S = \{0, 1\}$), λ is a label function: $Q \to S$, δ is a transition function: $Q \times \Sigma \cup \{\epsilon\} \to P(Q)$, I is a set of initial states and $I \subseteq Q$, and F is a finite set of final states and $F \subseteq Q$.*

SDFA and traditional NFA are special cases of SNFA. Figure 7 shows an example of SNFA. Both SDFA and SNFA are called secure finite automaton (SFA) denoted by $\mathcal{A}_S$.

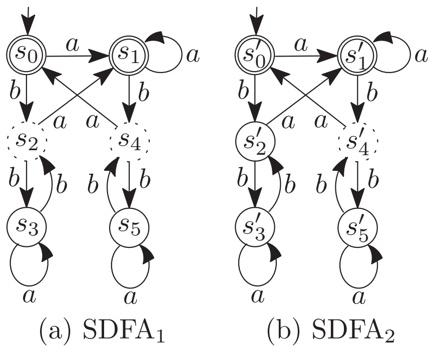

Figure 5. SDFAs

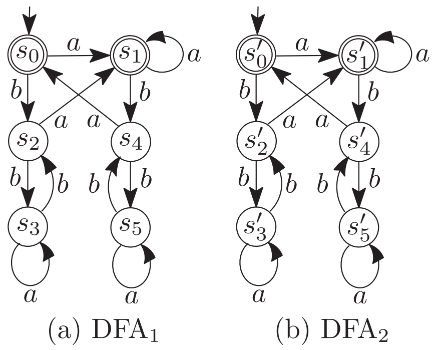

Figure 6. DFAs

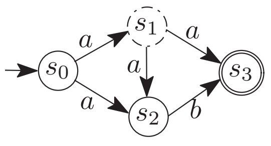

Figure 7. SNFA

$\mathcal{A_S}$ without insecure states (resp. insecure states or arcs labeled with ϵ) can be used to model systems for the purpose of weak bisimulation (resp. bisimulation).

Definition 3.3. *Two secure finite automata $\mathcal{A}_{\mathcal{S}1}$ and $\mathcal{A}_{\mathcal{S}2}$ are security equivalents iff $\mathcal{A}_{\mathcal{S}1}$ and $\mathcal{A}_{\mathcal{S}2}$ are securely bisimular.*

Based on the definition above, similar equivalence can be defined: two secure finite automata $\mathcal{A}_{\mathcal{S}1}$ and $\mathcal{A}_{\mathcal{S}2}$ which have no insecure states (resp. insecure states or ϵ) are equivalent in the sense of weak bisimulation (resp. bisimulation) iff $\mathcal{A}_{\mathcal{S}1}$ and $\mathcal{A}_{\mathcal{S}2}$ are weakly bisimular (resp. bisimular). $\mathcal{A}_{\mathcal{S}1}$ and $\mathcal{A}_{\mathcal{S}2}$ are language equivalents if $\mathcal{A}_{\mathcal{S}1}$ and $\mathcal{A}_{\mathcal{S}2}$ are (securely, weakly) bisimular, not vice versa.

Definition 3.4. *An SFA is called the minimal secure finite automaton (MSFA, denoted by $\mathcal{A}_{\mathcal{S}\min}$) if it has the least number of states regardless of whether secure or insecure among security equivalent secure finite automata.*

Similarly, An SFA without insecure states (resp. insecure states or ϵ) is called the minimal secure finite automaton in the sense of weak bisimulation (resp. bisimulation) if it has the least number of states among equivalent secure finite automata in the sense of weak bisimulation (resp. bisimulation).

4. Determinization and Minimization.

With secure bisimulation in view, an SNFA cannot be determinized in principle because internal deterministic and non-deterministic actions can affect interactions. The elimination of non-determinism cannot retain the (weak, secure) bisimulation relation. For example, if Figure 1(b) is determinized as Figure 1(a), automata before and after determinization (i.e., Figure 1(b) and Figure 1(a)) are not (weakly, securely) bisimular. Even so, the following purpose based algorithm gives a general framework to determinize an SFA including any traditional FA.

Algorithm 1: Determinization of a secure finite automaton

```
Input: a secure finite automaton $\mathcal{A}_\mathcal{S}$ and a purpose
Output: a DFA for a purpose of language equivalence or $\mathcal{A}_\mathcal{S}$ if an SDFA for a
        purpose of bisimulation, otherwise null
1  if (language equivalence purpose) then
2      if (there exist insecure states) then
3          for (each insecure state $p$) do
4              $\lambda(p) = 1$; /* label all insecure states as secure */
5          end
           /* the modified SFA is still denoted by $\mathcal{A}_\mathcal{S}$ */
6      end
7      $\mathcal{A}_\mathcal{S}$ is converted to a language equivalent FA $\mathcal{A} = (Q, \Sigma, \delta, I, F)$;
8      $\mathcal{A}$ is determinized by algorithms such as the subset construction and the result is
       denoted by $\mathcal{D}$;
9      return $\mathcal{D}$;
10 end
11 if ((weak, secure) bisimulation purpose) then
12     if ($\mathcal{A}_\mathcal{S}$ is deterministic) then
13         return $\mathcal{A}_\mathcal{S}$;
14     else
15         return null;
16     end
17 end
```

Proposition 4.1. *Algorithm 1 outputs the correct result of determinization.*

Proof: This algorithm outputs a result according to the purpose for which the result will be used. For the purpose of traditional language equivalence, this algorithm (lines 2-6) changes all insecure states into secure states which coincide with states of traditional finite automata in form. This change does not affect the language equivalence. Then this algorithm (line 7) removes those components which have no relation to traditional algorithms for determinization. This operation does not affect states or arcs and has no effect on the language equivalence. Thus, this algorithm converts an SFA $\mathcal{A}_\mathcal{S}$ to a language equivalent FA $\mathcal{A}$. As a traditional FA, $\mathcal{A}$ can be determinized by traditional algorithms such as the subset construction. The correctness of traditional algorithms for determinization has been proved in [1, 2]. Therefore, Algorithm 1 outputs a correct DFA for the purpose of language equivalence. For the purpose of (weak, secure) bisimulation, Algorithm 1 outputs the original $\mathcal{A}_\mathcal{S}$ if it is deterministic. Otherwise, Algorithm 1 returns null which is right in the sense of (weak, secure) bisimulation because the elimination of non-determinism has effect on (weak, secure) bisimulation.

It follows from the above that Proposition 4.1 holds. □

Proposition 4.2. *The worst case time complexity of Algorithm 1 is $\mathcal{O}\left(2^{|Q|}\right)$.*

Proof: Algorithm 1 can be viewed as a choice of two processes: one is for the purpose of language equivalence and the other is for the purpose of (weak, secure) bisimulation. So the worst case time complexity of Algorithm 1 is determinized by the comparison between these two processes. During the process for the purpose of language equivalence, the change of insecure states to secure ones (lines 2-6) involves $|Q|$ operations at most. The worst case time complexity of the subset construction algorithm for determinization (line 8) is $\mathcal{O}\left(2^{|Q|}\right)$ [12]. Therefore, the worst case time complexity of this process is $\mathcal{O}\left(2^{|Q|}\right)$. The process for the purpose of (weak, secure) bisimulation mainly checks whether $\mathcal{A}_\mathcal{S}$ is deterministic or not and the worst case time complexity is $\mathcal{O}\left(|Q| \cdot |\Sigma|\right)$.

Therefore, the worst case time complexity of Algorithm 1 is $\mathcal{O}\left(2^{|Q|}\right)$. □

Example 4.1. *For the purpose of language equivalence, insecure states in Figure 5(a) and Figure 5(b) are changed into secure ones (lines 2-6 of Algorithm 1) and DFAs (Figure 6(a) and Figure 6(b)) can be obtained (line 7 of Algorithm 1). For the purpose of (weak, secure) bisimulation, Algorithm 1 returns the original SDFA such as Figure 5(a) and Figure 5(b) (lines 12 and 13 of Algorithm 1) and returns null if its input is an SNFA such as Figure 7 (lines 14 and 15 of Algorithm 1). Results of this determinization algorithm are correct.*

It is known that the concept of (weak, secure) bisimulation is stricter than that of language equivalence. Therefore, direct application of traditional algorithms for minimization to SFA cannot guarantee the correctness of results. Algorithm 2 adopts the purpose based division policy similar to that of Algorithm 1 to minimize any SFA.

Algorithm 2: Minimization of a secure finite automaton

```
Input: a secure finite automaton $\mathcal{A}_\mathcal{S}$, a purpose and (weak, secure) bisimulation(s)
       for a purpose of bisimulation
Output: a minimal DFA for a purpose of language equivalence or a minimal SFA
        for a purpose of bisimulation
1  if (language equivalence purpose) then
2      if (there exist insecure states) then
3          for (each insecure state $p$) do
4              $\lambda(p) = 1$; /* label all insecure states as secure */
5          end
           /* the modified SFA is still denoted by $\mathcal{A}_\mathcal{S}$ */
6      end
7      $\mathcal{A}_\mathcal{S}$ is converted to a language equivalent FA $\mathcal{A} = (Q, \Sigma, \delta, I, F)$;
8      if ($\mathcal{A}$ is non-deterministic) then
9          $\mathcal{A}$ is determinized by algorithms such as the subset construction and the
           result is denoted by $\mathcal{D}$;
10     else
11         $\mathcal{D} \leftarrow \mathcal{A}$;
12     end
13     $\mathcal{D}$ is minimized by Hopcroft's algorithm and $\mathcal{D}_{\min}$ is obtained;
14     return $\mathcal{D}_{\min}$;
15 end
16 if (bisimulation purpose) then
17     while ((weak, secure) bisimulation $\mathbf{B}$ exits) do
18         for ($(p, q) \in \mathbf{B}$) do
19             $p$ and $q$ are combined;
20         end
21     end
22     the result is denoted by $\mathcal{A}_{\mathcal{S}\min}$ and $\mathcal{A}_{\mathcal{S}\min}$ is returned;
23 end
```

Proposition 4.3. *Algorithm 2 outputs the correct result of minimization.*

Proposition 4.4. *The worst case time complexity of Algorithm 2 is $\mathcal{O}\left(2^{|Q|}\right)$.*

The proof of Proposition 4.3 (resp. 4.4) is similar in structure to that of Proposition 4.1 (resp. 4.2). Thus, both proofs are omitted owing to the limitation of space.

Example 4.2. *For the purpose of language equivalence, SDFAs in Figure 5(a) and Figure 5(b) are first changed to DFAs in Figure 6(a) and Figure 6(b) (lines 2-12 of Algorithm 2). With Hopcroft's algorithm, these DFAs can be minimized as Figure 4(a) and Figure 4(b) respectively (line 13 of Algorithm 2). For the purpose of (weak, secure) bisimulation, Algorithm 2 minimizes the SDFA$_1$ in Figure 5(a) as an SFA$_{\min}$ in Figure 8 using the secure bisimulation $\{(s_0,s_1),(s_2,s_4),(s_3,s_5)\}$ (lines 17-21 of Algorithm 2). SDFA$_2$ in Figure 5(b) cannot be minimized because there is no secure bisimulation as the input. SFAs in Figure 8 and Figure 5(b) are not securely bisimular and errors mentioned in Section 1 will never be made.*

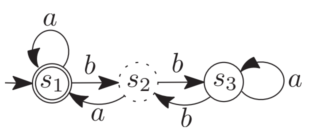

FIGURE 8. SFA$_{\min}$

5. **Conclusion.** The secure finite automaton theory is put forward to meet the demands for analyzing and processing properties especially security. This theory is more general: traditional finite automata are special cases of secure finite automata and traditional algorithms for determinization and minimization are contained in new algorithms respectively. Systems can be modeled and analyzed using the secure finite automaton theory and the correctness of security comparison among minimal models can be guaranteed.

Acknowledgement. This research is supported by the National Natural Science Foundation of China with Grant Nos. 61373027 and 61572283, the Award Foundation Project of Excellent Young Scientists in Shandong Province with Grant No. BS2014DX005, the Project of Shandong Province Higher Educational Science and Technology Program with Grant No. J13LN31, the Scientific Research Foundation of Qufu Normal University with Grant No. XJ201226, and the Program of Science and Technology (the Youth Foundation) of Qufu Normal University with Grant No. xkj201313. The authors also thank the anonymous reviewers for their insightful comments.

REFERENCES

[1] J. E. Hopcroft, R. Motwani and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd Edition, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[2] M. Sipser, *Introduction to the Theory of Computation*, 3rd Edition, Cengage Learning, 20 Channel Center Street, Boston, MA, USA, 2013.

[3] R. W. D. Pedro, F. L. S. Nunes and A. Machado-Lima, Using grammars for pattern recognition in images: A systematic review, *ACM Comput. Surv.*, vol.46, no.2, pp.1-34, 2013.

[4] R. Milner, *Communicating and Mobile Systems: The π-Calculus*, Cambridge University Press, New York, NY, USA, 1999.

[5] G. Liu and C. Jiang, Secure bisimulation for interactive systems, *The 15th International Conference on Algorithms and Architectures for Parallel Processing*, Zhangjiajie, China, pp.625-639, 2015.

[6] J. Zhang and Z. Wu, Minimization of bounded petri net simplification, *Computer Science*, vol.34, no.11, pp.26-28,61, 2007.

[7] L. D'Antoni and M. Veanes, Minimization of symbolic automata, *Proc. of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, New York, NY, USA, pp.541-553, 2014.

[8] L. D'Antoni and M. Veanes, Extended symbolic finite automata and transducers, *Form. Methods Syst. Des.*, vol.47, no.1, pp.93-119, 2015.

[9] J. E. Hopcroft, *An $n \log n$ Algorithm for Minimizing States in a Finite Automaton*, Tech. rep., Stanford, CA, USA, 1971.

[10] J. Brzozowski and H. Tamm, Theory of átomata, *Theoretical Computer Science*, vol.539, pp.13-27, 2014.

[11] T. Murata, Petri nets: Properties, analysis and applications, *Proc. of the IEEE*, vol.77, no.4, pp.541-580, 1989.

[12] F. R. Moore, On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata, *IEEE Trans. Computers*, vol.20, no.10, pp.1211-1214, 1971.