The Application of Decidability Theory to Identify Similar Computer Networks

Article · April 2012

CITATIONS
2

READS
676

3 authors:

Shafi'i Muhammad Abdulhamid
Federal University of Technology Minna
110 PUBLICATIONS **1,503** CITATIONS
SEE PROFILE

Victor Onomza Waziri
Federal University of Technology Minna
36 PUBLICATIONS **81** CITATIONS
SEE PROFILE

Laminu Idris
Federal University of Technology Minna
2 PUBLICATIONS **2** CITATIONS
SEE PROFILE

The Application of Decidability Theory to Identify Similar Computer Networks

Shafi'i M Abdulhamid*, Victor O Waziri** and Laminu Idris***

Recent decades have seen a remarkable development in the use of computer systems and computer networks. With their advances our reliance on hardware and software has amplified, and so has our susceptibility to their malfunction. For fear of any major network failure, it is at all times good to recognize similar networks so that switching can be done to reduce damages. Theoretical computer science aims to model and understand the intricacy of computer systems, and thereby creates the basis for their formal verification: to mathematically prove that a system satisfies its requirement. In this paper, the decidability theory is theoretically applied to identify/decide if two different computer networks are similar (in terms of efficiency, high performance computing and scalability) or not. A finite automata is designed for each network and two different scenarios are considered for demonstration. The results show that the theory can be effectively used to make such comparisons between different computer networks.

Keywords: Finite Automata (FA), Decidability, Computer network, Language, Transition table

Introduction

In logic, the term decidable refers to the existence of an effective method for determining membership in a set of formulas. Logical systems such as propositional logic are decidable if membership in their set of logically valid formulas (or theorems) can be effectively determined. A theory (set of formulas closed under logical consequence) in a fixed logical system is decidable, if there is an effective method for determining whether arbitrary formulas are included in the theory.

Decidability theory is a branch of mathematics. Suppose there is a set, and there is an element. There is also an algorithm. The algorithm will simply check if the element belongs to the set or not. If the algorithm stops (after a limited time) and reaches a decision, whether the element is in the set or not, it is called decidable.

* Lecturer, Department of Cyber Security Science, Federal University of Technology Minna, Niger State, Nigeria; and is the corresponding author. E-mail: shafii.abdulhamid@futminna.edu.ng

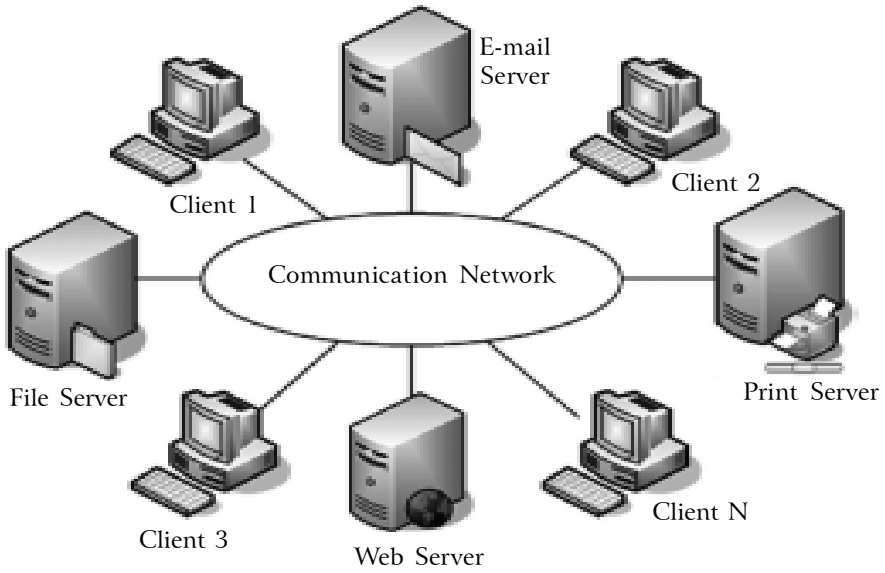
** Senior Lecturer, Department of Cyber Security Science, Federal University of Technology Minna, Niger State, Nigeria. E-mail: onomzavictor@gmail.com

*** Lecturer, Department of Mathematics and Statistics, Federal University of Technology Minna, Niger State, Nigeria. E-mail: lameenoo@yahoo.com

In simple terms, if there is a shopping bag, decidability is that one is able to check whether or not there is some salad in the bag.

Computer networks were developed with the goal to communicate between computers, to increase computing speed and reduce processing time (see Figure 1). However, the development of computer technology and emergence of new kinds of information processing systems essentially extended the functions of computer networks. As a result, solving different problems has become the main goal of computer networks. This involves mostly 'decision mode' of functioning. Decision making is a prime goal of artificial intelligence. At the same time, the appearance of Internet transformed computers into communication devices. Communication involves receiving and sending information. Sending information is realized in 'computing mode', while receiving information demands 'accepting mode'. For example, one of the vital problems for computer security is to make a decision whether to accept a message or to reject it because this message contains a virus.

Figure 1: A Simple Computer Network



Source: Saxena and Arora (2009)

In this paper, we focus on the decision whether the given two different computer networks are similar or not, that is, we check if the two networks are equivalent (equal) in terms of speed of communication, efficiency, high performance computing, and scalability, or not.

2. Related Works

The design of decision procedures for first-order theories and their combinations has been a very active research subject over the last 30 years. It has gained practical importance through the development of Satisfiability Modulo Theories (SMT) solvers. Most results concentrate on combining decision procedures for data structures, such as theories for arrays, bit vectors, fragments of arithmetic, and uninterpreted functions (Fontaine, 2009). In particular, the well-known Nelson-Oppen scheme for the combination of decision procedures requires the signatures to be disjoint and each theory to be stably infinite; every satisfiable set of literals in a stably infinite theory has an infinite model. Ohsaki and Takai (2004) introduced an extension of tree automata framework, called equational tree automata. This theory is useful to deal with unification modulo equational rewriting. They demonstrated how equational tree automata can be applied to several realistic unification examples, including a security problem of network protocols.

Rej (2009) argued that questions of algorithmic decidability, computability and complexity should play a larger role in deciding the 'ultimate' theoretical description of the landscape of string vacua. More specifically, examine the notion of the average rank of the (unification) gauge group in the landscape, the explicit construction of Ricci-flat metrics on Calabi-Yau manifolds as well as the computability of fundamental periods to show that undecidability questions are far more pervasive than that described in the work of Denef and Douglas.

Usually to study properties of computers and to develop more efficient applications, mathematical models are used. There is a variety of such models: turing machines of different kinds (with one tape and one head, with several tapes, with several heads, with *n*-dimensional tapes, non-deterministic, probabilistic, and alternating turing machines, turing machines that take advice, turing machines with oracle, etc.), post productions, partial recursive functions, neural networks, finite automata of different kinds (automata without memory, autonomous automata, accepting automata, probabilistic automata, etc.), minsky machines, normal Markov algorithms, Kolmogorov algorithms, formal grammars of different kinds (regular, context-free, context-sensitive, phrase-structure, etc.), storage modification machines or simply, shönhage machines, Random Access Machines (RAM), petri nets, which like turing machines have several forms (ordinary, regular, free, colored, self-modifying, etc.), and so on (Burgin, 2002).

Petri nets are useful for modeling and analysis of computer networks, distributed computation, and communication processes (Peterson, 1981). Finite automata reflect computer models. Neural networks reflect properties of the brain. Abstract vector and array machines model vector and array computers (Pratt *et al.*, 1974).

There are two fundamental problems concerning equivalence relations in concurrency. First: For which system classes is a given equivalence decidable? Second: When do two equivalences coincide? Two well-known equivalences are history preserving bisimilarity (hpb) and hereditary history preserving bisimilarity (hhpb). These are both 'independence' equivalences. They reflect causal dependencies between events. hhpb is obtained from hpb by adding a 'back-tracking' requirement. This seemingly small change makes hhpb computationally much harder. hpb is well-known to be decidable for finite-state systems, whereas the decidability of hhpb has been a renowned open problem for several years. Only recently it has been shown undecidable. The main aim of Fröschle's (2002) paper was to gain insights into the decidability problem for hhpb, and to analyze when it coincides with hpb. Speaking less technically, its aim was to analyze the power of the interplay between concurrency, causality, and conflict.

3. Decidability Theorem

There is an effective procedure to decide whether:

Theorem 1: A given Finite Automata (FA) is empty or not.

Proof: Given an FA, it defines an empty language (L) if:

- The FA has no final state, or
- The final state is not reachable from the start state
  - Mark start state
  - Mark any state reachable from already marked state
  - Repeat until all states are exhausted or when there are no more reachable states
  - If the marked states include a final state, $L \neq \varnothing$
  - If the marked states do not include a final state, $L = \varnothing$

Theorem 2: Two FAs are equivalent or not.

Proof: Given two FAs or languages, $FA_1 = L_1$ and $FA_2 = L_2$

If $L_1 = L_2$ (as sets)

- $L_1 \cap L_2' = \varnothing$ and $L_2 \cap L_1' = \varnothing$

This implies:

$$(L_1 \cap L_2') \cup (L_2 \cap L_1') = \varnothing$$

i.e.,

$L_1 = L_2$ if

$$(L_1 \cap L_2') \cup (L_2 \cap L_1') = \varnothing \quad \ldots(1)$$

By De Morgan's Rule, Equation (1) becomes:

$L_1 = L_2$ if

$$(L_1' \cup L_2)' \cup (L_1 \cup L_2')' = \varnothing \quad \ldots(2)$$

4. Materials and Methods

Computer network is used to link two or more computers. Network users are able to share files, printers and other resources, send electronic messages, and run programs on other computers. A network has three layers of components: application software, network software, and network hardware.

Application software consists of computer programs that interface with network users and permit the sharing of information, such as files, graphics, and video, and resources, such as printers and disks. One type of application software is called client-server. Client computers send requests for information or requests to use resources to other computers, called servers, that control data and applications. Another type of application software is called peer-to-peer. In a peer-to-peer network, computers send messages and requests directly to one another without a server intermediary.

By interpretation, the symbols used in the FAs below represent:

a $\rightarrow$ send message

b $\rightarrow$ request for message

$\lambda \rightarrow$ no action

state $\rightarrow$ node

final state $\rightarrow$ server

4.1 First Scenario

In the first scenario, we take two different networks (N_1 and N_2) into consideration. The two networks are represented by the FAs presented in Figure 2.

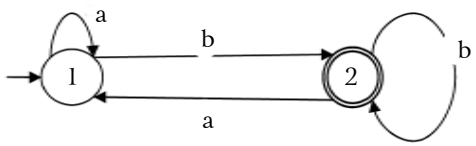
If the two networks work similarly, then Equation (2) must hold.

$$(L_1' \cup L_2)' \cup (L_2' \cup L_1)' = \varnothing$$

Figure 2: Two FAs Representing Two Different Networks

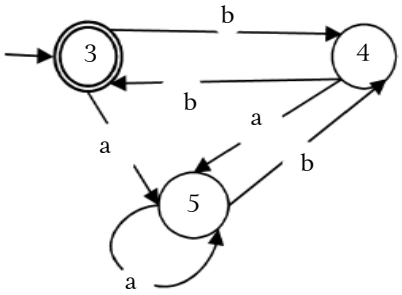
N_1 = Network Setup 1

$N_1 = L_1$ = Language Accepted by FA_1



N_2 = Network Setup 2

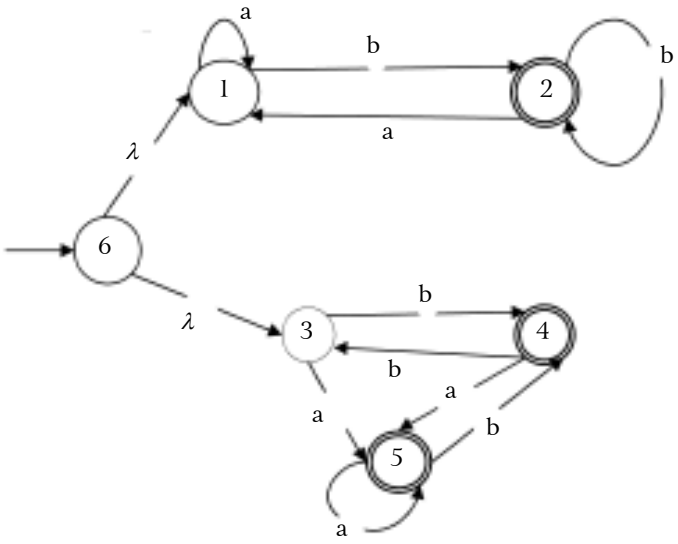
$N_2 = L_2$ = Language Accepted by FA_2



So, if we take the first part of the equation $\left(L_1 \cup L_2{}'\right)$, the networks are represented in Figure 3.

The transition table for $\left(L_1 \cup L_2'\right)$ is given by Table 1, which is obtained from Figure 3.

Figure 3: Two FAs Representing Two Different Networks Connected Together



From the transition table below it is observed that, not all the states are final; state 613 is non-final.

Table 1: Transition Table for Figure 3				
State	Nature	Transition Symbols		
		a	b	λ
613	Initial	15	24	613
15	Final	15	24	–
24	Final	15	23	–
23	Final	15	24	–

That is, $(L_1 \cup L_2') \neq$ final

Therefore, $(L_1 \cup L_2')' \neq$ non-final

and $(L_1' \cup L_2)' \neq \varnothing$

Therefore, $(L_1' \cup L_2)' \cup (L_2' \cup L_1)' \neq \varnothing$

Hence, the languages L_1 and L_2 are not the same or equal, implying that the two networks are not similar.

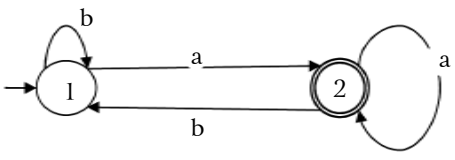
4.2 Second Scenario

In the second scenario, we rearrange the two different networks (N_1 and N_2) and also change the 'send and request message' privileges of the two networks (as represented by the FAs given in Figure 4).

Figure 4: Two FAs Representing Two Different Networks

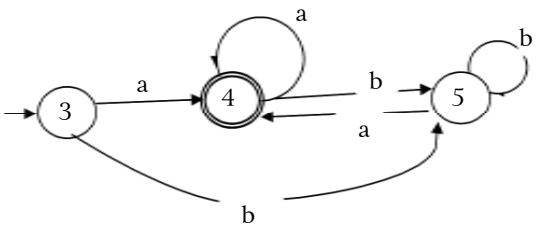
N_1 = Network Setup 1

$N_1 = L_1$ = Language Accepted by FA_1



N_2 = Network Setup 2

$N_2 = L_2$ = Language Accepted by FA_2

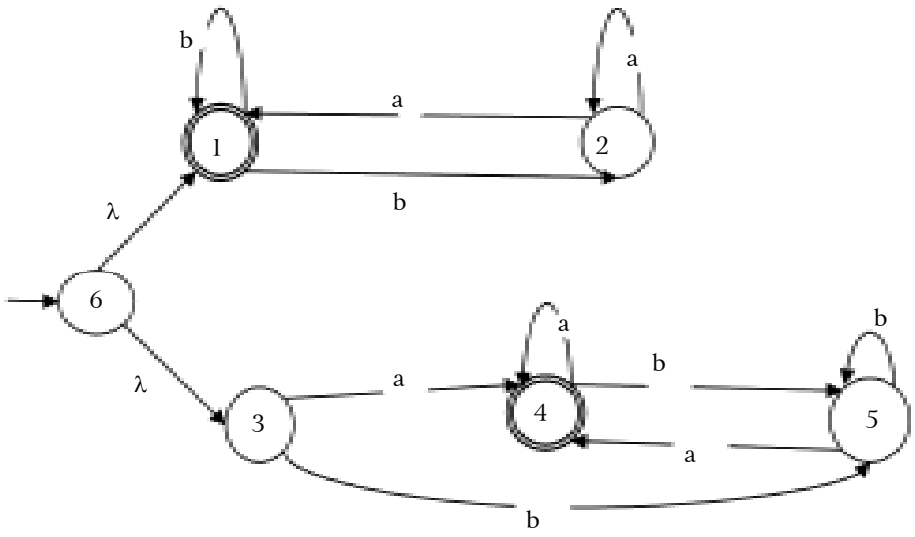


If the two languages are the same, then Equation (2) must hold.

$(L_1' \cup L_2)' \cup (L_2' \cup L_1)' = \varnothing$

So, if we take the first part of the equation $(L_1' \cup L_2)$, the networks are represented in Figure 5.

Figure 5: Two FAs Representing Two Different Networks Connected Together



The transition table for $(L_1' \cup L_2)$ is given by Table 2, which is obtained from Figure 5.

From the transition table below it is observed that, all the states are final.

That is, $(L_1' \cup L_2)$ = final

Therefore, $(L_1' \cup L_2)'$ = non-final

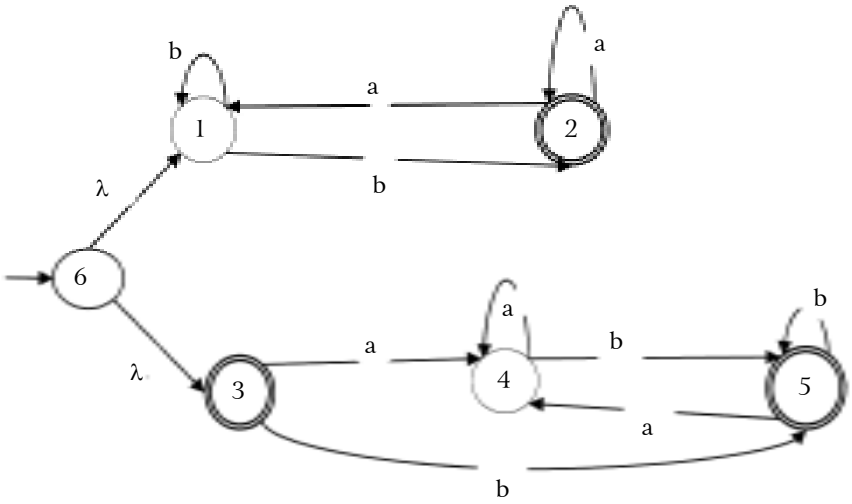
and $(L_1' \cup L_2)' = \varnothing$

Table 2: Transition Table for Figure 5

State	Nature	Transition Symbols		
		a	b	λ
613	Final	24	15	613
24	Final	24	15	–
15	Final	24	15	–

If we take the second part of the equation $(L_2' \cup L_1)$, the networks are represented in Figure 6.

Figure 6: Two FAs Representing Two Different Networks Connected Together



The transition table $(L_2' \cup L_1)$ is given by Table 3, which is obtained from Figure 6.

From the transition table below it is observed that, all the states are final.

That is, $(L_2' \cup L_1)$ = final

Therefore, $(L_2' \cup L_1)'$ = non-final.

and $(L_2' \cup L_1) = \varnothing$

Therefore, $(L_1' \cup L_2)' \cup (L_2' \cup L_1)' = \varnothing$

Table 3: Transition Table for Figure 6

State	Nature	Transition Symbols		
		a	b	λ
613	Final	24	15	613
24	Final	24	15	–
15	Final	24	15	–

Hence, the languages L_1 and L_2 are the same or equal, implying that the two networks are working at the same rate in terms of speed of communication, efficiency, high performance computing and scalability.

Conclusion

In this paper, we put forward some areas through which questions regarding decidability of similar computer networks can be addressed theoretically. We show that it is possible to have two different networks with different number of nodes and different topological arrangements, but working in a very similar manner in terms of efficiency, high performance computing and scalability. It was observed that the topological arrangement of the networks that will give similarity cannot be found with this method, but can be used to ascertain if the arrangement gives similarity or not. Even with this shortcoming, the method has been found to be theoretically working and useful.

The application of decidability theory to identify equal or similar networks was theoretically done in this paper, and the experimental approach would be the next step or focus of the authors in this direction. ✌

References

1. Burgin Mark (2002), "Axiomatic Theory of Algorithms: Computability and Decidability in Algorithmic Classes", Department of Mathematics, University of California, Los Angeles.

2. Denef Fredrik and Douglas Michael R (2007), "Computational Complexity of the Landscape I", *Annals. Phys*., Vol. 322, No. 4, pp. 1096-1142.

3. Fontaine Pascal (2009), "Combinations of Theories for Decidable Fragments of First-Order Logic", Universite de Nancy, Loria Nancy, France, available at Pascal.Fontaine@loria.fr

4. Fröschle Sibylle (2002), "Decidability and Coincidence of Equivalences for Concurrency Do", pp. 3-33, Doctor of Philosophy Thesis, Laboratory for Foundations of Computer Science School of Informatics, University of Edinburgh.

5. Ohsaki Hitoshi and Takai Toshinori (2004), "Equational Tree Automata: Towards Automated Verification of Network Protocols", National Institute of Advanced Industrial Science and Technology (AIST) Nakoji 3-11-46, Amagasaki 661-0974, Japan.

6. Peterson J L (1981), *Petri Net Theory and the Modeling of Systems,* Prentice-Hall, Inc., Englewood Cliffs, NJ.

7. Pratt V, Rabin M and Stockmeyer L J (1974), "A Characterization of the Power of Vector Machines", in 6th ACM Symposium on the Theory of Computing, pp. 122-134.

8. Rej Abhijnan (2009), "Turing's Landscape: Decidability, Computability and Complexity in String Theory", Prepared for the 2nd FQXi Essay Contest What is Ultimately Possible in Physics, The Center for Contemporary Studies, Indian Institute of Science, Bengaluru, available at abhijnan.rej@gmail.com

9. Saxena Vipin and Arora Deepak (2009), "UML Modeling of Network Topologies for Distributed Computer System", *Journal of Computing and Information Technology*, Vol. 17, No. 4, pp. 327-334.

Reference # 61J-2011-06-02-01