Article

Addressing the Limitation of JFLAP Mealy Machine for Binary Increments

Kardan Journal of Engineering and Technology
2 (1) 67–75
©2020 Kardan University
Kardan Publications
Kabul, Afghanistan
DOI: 10.31841/KJET.2021.13
https://kardan.edu.af/Research/Kardan_journal_of_engineering_and_technology.aspx#

Amjad Khan
Muhammad Haleem

Abstract

JFLAP stands for java formal language and automata package has developed by Susan H. Rodger and Thomas W. Finley at Duke University for graduate and undergraduate students to simulate and test different automatons using its graphical interface. Using JFLAP students can develop the generation and recognition devices not only for regular languages but also for non-regular languages as well. Moore and Mealy are two machines which will produce output on any given input. But in some cases when a user develops valid Mealy machine (MM) and enter the valid input string, JFLAP produces wrong output. In this article, the discrepancy of JFLAP in term results produces by Mealy machine (MM) is identified and solution has been proposed using java language. It has been observed from that the solution proposed in this article totally eliminate the problem of producing wrong results especially in case of developing an incremental Mealy machine (MM).

Keywords: *Moore machine, Mealy machine, Finite automata*

Mr. Amjad Khan and Mr. Muhammad Haleem, *are Assistant Professors at Faculty of Engineering and Technology, Kardan University - Kabul, Afghanistan. <amjad@aup.edu.pk>,<mu_haleem@yahoo.com>*

Introduction

Students at undergraduate and graduate level study an important course by the name of Theory of Computation. Usually students find this course boring, tedious and not easy as unlike other computer science courses, this course do not involve practical and they cannot check the correctness of their work immediately. Usually students designed different recognition devices/models manually which is not only time consuming job but also can not verify their results instantly. Thanks to Susan H. Rodger and her team for providing a simulation tool in the form of JFLAP where students can design different automatons and check their functionalities by a single click. This not only reduced design time of automatons but also increase students interest in the course of computational theory [1].

Susan H. Rodger [2] stated in "JFLAP 7.0 License" that this simulation tools evolved since 1993 and was originally developed at Duke University with the support of National Science Foundation. This simulation tool is free available online and can run both 32/64 bits operating systems. Java 1.4, 1.5 or latest version must install on PC to run JFLAP simulations. Most of its recent version source code is available online but no modification is allowed. Different version of this simulation tools available with 7.1 version released on July 27, 2018. With some modification JFLAP 8.0 beta version is available but not stable and completed [5].

P. Chakraborty et-al [4] ranked JFLAP as most cultured and refined tool for simulating automat in their work "Fifty years of automata simulation: a review". They described this tool well documented, easy to use and possess state of the art graphics.

D. Caugherty et-al [3] described JFLAP history in their work, "NPDA: A Tool for Visualizing and Simulating Nondeterministic Pushdown Automata" and stated that JFLAP development started in the early 90's with limited capabilities. With the passage of time more features were added to provide a best platform for students to simulate the design of their automatons.

Using the latest version of JFLAP, students can simulate all generating and recognition devices for formal languages ranging from type-3 to type-0. The automatons for type-03 and type-02 languages are used just as recognition devices and they cannot produce any output. At the same time Mealy machine (MM) is an FSA (Finite State Automaton) that can produce output on any given string. But if we design the correct MM for incrementing binary digits and run the simulation, JFLAP will produce wrong results. In this work the cause of problem is identified and solution has been provided so that if incorporate in the present version of JFLAP will remove discrepancy related to MM.

2. Related Work

Modification applied to JFLAP that let the students write programs using java to aid more features to the existing JFLAP. This improvement allow student' to add-up more capabilities to the existing JFLAP they need [6].

This is insisted that the data stored in tools that can be used online can help other new in subjects or junior instructed to solve the problems they are facing in that domain. This argument has been tested in two webs available several tools. Most experiments done on JFLAP and Logic-ITA.[7]

The overall look and feel of the JFLAP tool have been changed in-order to make it more user friendly. So that "easy to work" property of the tool will motivate the students to take more interest in the field of problem automation [8].

Process communicative model has been proposed to add with JFLAP to show the flow chart like layout of written script. This approach can precisely describe the mechanism behind implementation to the readers [9].

The surveys regarding JFLAP shows that student can easily learn the key concepts about the theories of automation with the help of JFLAP. So, more engagement exists of automata theory with JFLAP. [10]

Work has been carried out on JFLAP to present extensions that could enhance the weight of JFLAP as a learning for students and key guide for instructors these enhancements to the tool capable this in finite automata. Here the authors contributed for instructors to have the see the activity log recorded by JFLAP. To know about students' trials and then assist the students in the problems of automata construction. Research contributed for student to have a simple and native means to show the correction tips while they are constructing FAs [11].

Automata theory covers several machines. Among those Turing machines are working as dominant computational machines. These have similarity to algorithms; these machines are bases for real computers. Building Turing Machines to cover wide range of numerical and non-numerical problems, is a remarkable job. Thus, a Universal Turing Machine (UTM) has been introduced [12].

3. Existing Problem with JFLAP

The decades of work have been carried out over JFLAP. The literature review shows that JFLAP is tightly coupled automata and will be available as the most interactive tool for many years to come. As mealy machine has its own significance in automata theory. This research identified an active problem with mealy machine in JFLAP. The incrementing mealy machine does not return the correct result as it starts the incrementing process from the left. The actual increment process needs to be from the right. The above given literature review and the best of knowledge regarding JFLAP confirms that the problem has not been addressed so far. The problem has been shown in the given figure1, figure 2 and figure 3. The problem is clearly demonstrated her by inspecting the given JFLAP outputs for the given inputs. It can be easily pointed out that the JFLAP output are not to the point as incrimination has been performed from the left which is incorrect. The basic arithmetic principals of binary numbers allow the increment from the right side only.

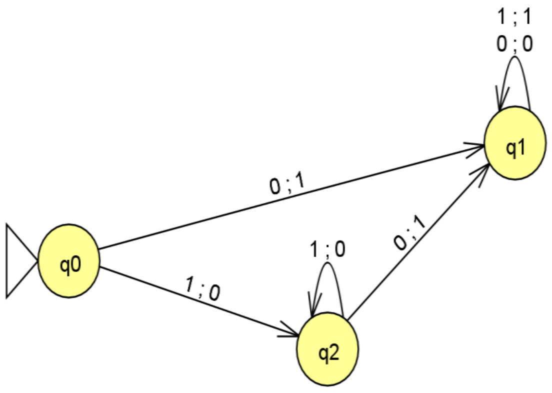
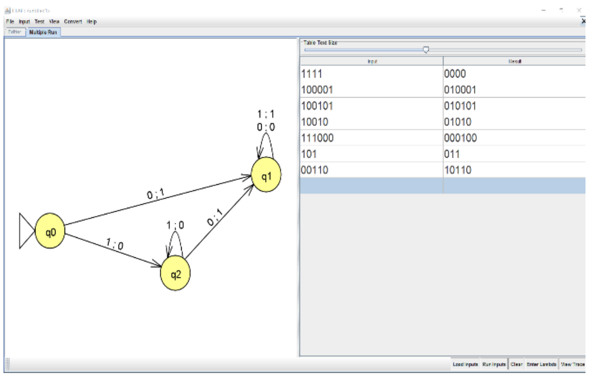


Figure 1:



Input	Result
1111	0000
100001	010001
100101	010101
10010	01010
111000	000100
101	011
00110	10110

Figure 2:

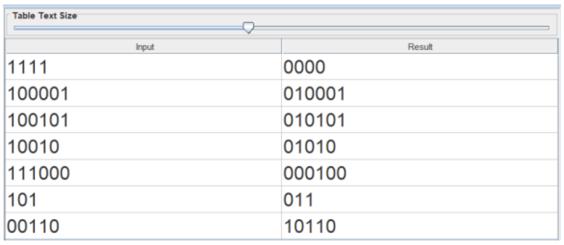


Table Text Size

Input	Result
1111	0000
100001	010001
100101	010101
10010	01010
111000	000100
101	011
00110	10110

Figure 3:

4. Proposed Solution for AMI Mealy Machine

The proposed algorithm is designed to the address the problem of incrementing mealy machine. This solution work will for unlimited input string. This technique provides the validation mechanism as well. This research contributes the solution to overflow bit as well. For example, the mealy machine converts the input of all 1's to all 0's. But our solution cover-up this active problem significantly. According to procedure of this solution. It first read the user input. Validate the input for the validity of input binary number. If the input is not in proper binary form the user will be prompted for. In case of no issue with input string the string will parsed to a character array to handle the individual bits.

After the input string comes ready in array the increment process will be initiated from right most bit. The bit access pointer will start traversing the array from the position (n-1). The bits will be accessed one by one toward left. If the pointed bit is 0, it will be changed to 1 and the rest of the bits will be unchanged and so the result. In case the current bit is 1 then it will be changed to 0 with a carry of one. The carry will be counted with the adjacent bit to the left of the input string. And the procedure will continue in this manner.

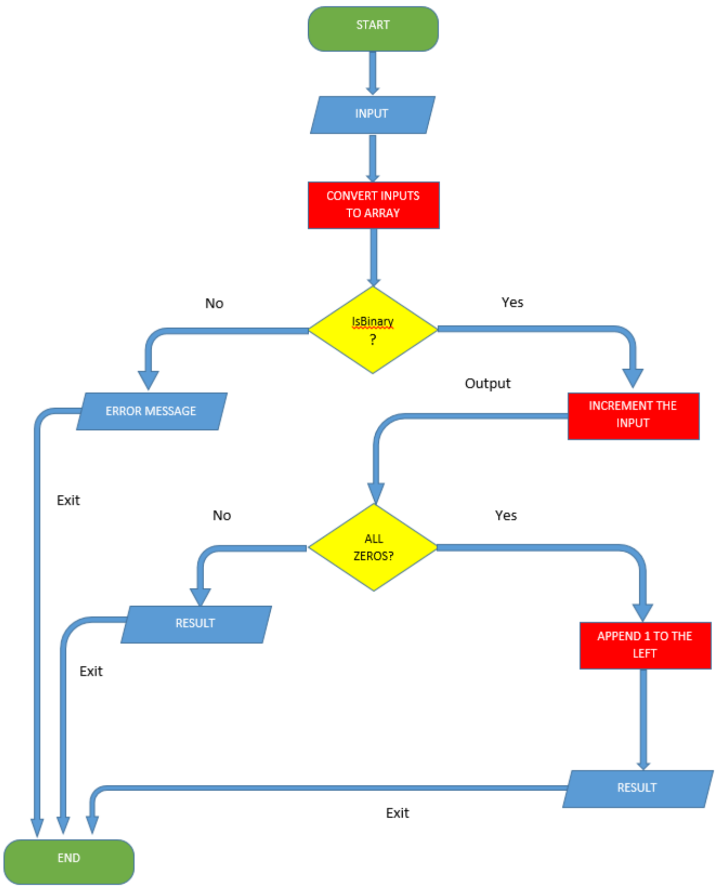
In case all zero's encountered in the result after bits traversal process, then "1" will be appended to the left of the resultant string.

5. Algorithm AMI Mealy Machine

```
1.  BEGIN
2.  [INPUT]
    INPUT_STRING=INPUT_VALUE
3.  [CONVERT INPUT_STRING TO ARRAY]
4.  INPUT_ARRAY [] =INPUTSTRING
5.  [CHECK THE VALIDITY OF STRING FOR BINARY]
     IF (VALID BINARY) THEN
            IS_BINARY = TRUE;
    ELSE
      IS_BINARY = FALSE;
6.  [INCREMENT]
7.  IF (IS_BINARY = = TRUE) THEN
              OUTPUT=INPUT_ARRAY+1
```

```
                IF (OUTPUT= "ALL ZEROS") THEN
                [APPEND 1 TO THE LEFT OF OUTPUT]
                OUTPUT=CONCAT (1, OUTPUT)
        ELSE
                PRINT ERROR_MEESAGE
                GOTO STEP 9
8.      [DISPLAY RESULT]
        PRINT OUTPUT
9.      END
```

6. System's Flowchart



7. Implementation Detail

As the JFLAP is developed using JAVA. So, in order to avoid possible issue of cross-platform compatibility, the technique is implemented in java. Exception handling has been used in the development to avoid the unexpected failure of the system. The unexpected failure can be cause from incorrect user inputs or some internal runtime interrupt. So, robustness added to the system by utilizing the exception handling feature.

Regarding the GUI setup the Javax.Swing and AWT has been applied. The JFrame class has been extended for adding the components to the application form. To make the GUI interactive, the action listener interface implemented here. Static bounds used to manage the components positions.

The below given test bed has been used for implementation.

IDE: JCreator 4.0

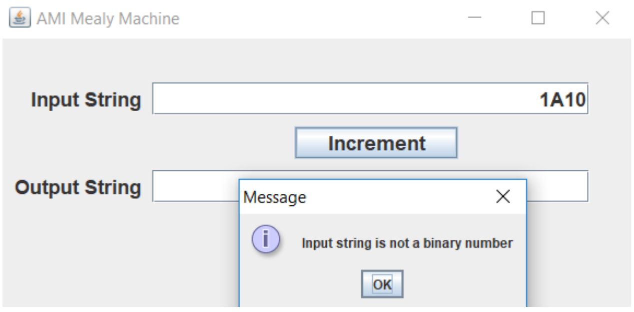
Compiler: J2SDK 9.0

Operating System: windows 2010

System: Core i7

RAM: 8GB.

Validation rules are applied over the interface input text field. It will accept proper binary number only. So, the incorrect entry as shown here in figure 4 has been rejected and the user is prompted with a diagnostic informative message.

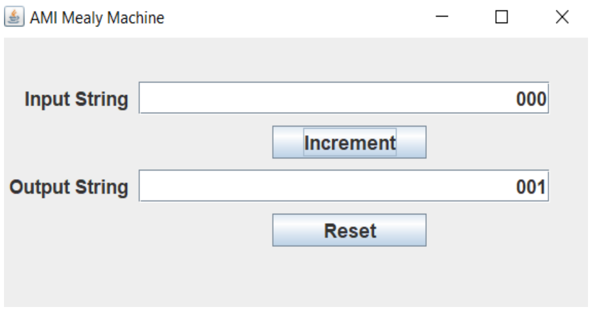


1A10

Input string is not a binary number

Figure 4: *AMI Mealy Machine*

The valid binary number can be incremented accurately without any issue. Here the accurate result delivered by our implemented technique. As this research identified the problem with existing JFLAP's Mealy machine. The problem is that it increments the input from left side. Means for input '000' it produces '100', which totally incorrect result. The correct one needs to be 001. This scenario is demonstrated by *figure* 5 given below.

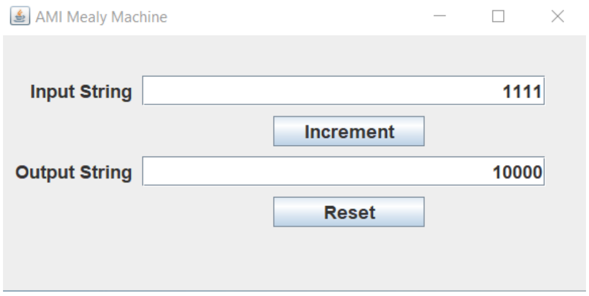


000

001

Figure 5: *AMI Mealy Machine*

When the system gets an input of all 1's of length *in* the produced final result passes from two phases. First all 1's will be converted to all zero's as per mealy machine pre-define procedure. In order to have the correct expected result, in second phase 1 will be appended to the left of the first phase result. Thus, resultant output will have a length of *n+1.* This point has been demonstrated by *figure* 6. Where the input '1111' all 1's of length *n*=4 given the result '10000' of length *n+1*=5.

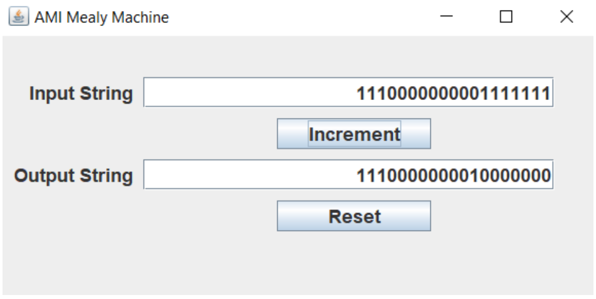

AMI Mealy Machine

Input String 1111

Output String 10000

Figure 6: *AMI Mealy Machine*

The implemented system operates will for any kind and limit of input binary strings. Another snapshot of the system has been provided in *figure 7.*


AMI Mealy Machine

Input String 1110000000001111111

Output String 1110000000010000000

Figure 7: *AMI Mealy Machine*

8. Conclusion and Future Work

After analyzing results of new accuracy measures introduced in this work, it has been observed that incrementing Mealy machine produced not only accurate results but also there is no restriction on the length of strings to be added.

Furthermore, it has been observed during this work that the problem of incrementing the strings by incrementing Mealy machine in JFALP version 7.0 was because of coding. Programmer not considered addition of bits from the right of string, rather it was adding bits from left of string, that produced wrong results. This problem has been overcome in this work.

Researcher should also check and analyze Turing Machine while simulating them for the solution of real-world problems.

References

[1] Susan H. Rodger; Eric Wiebe; Kyung Min Lee; Chris Morgan; Kareem Omar; Jonathan Su, "Increasing Engagement in Automata Theory with JFLAP". Fortieth SIGCSE Technical Symposium on Computer Science Education: 403–407, 2009.

[2] Susan H. Rodger. "JFLAP 7.0 LICENSE". Retrieved 2 October 2016.

[3] D. Caugherty; S. H. Rodger, "NPDA: A Tool for Visualizing and Simulating Nondeterministic Pushdown Automata". DIMACS Workshop March 12–14, 1992: 365–377, 1992.

[4] P. Chakraborty; P.C. Saxena; C. P. Katti, "Fifty years of automata simulation: a review". ACM Inroads. 2 (4): 59–70, 2011.

[5] Paul J. Using jFlap to engage students and improve learning of computer science theory: tutorial presentation. *Journal of Computing Sciences in Colleges*. 1;31(2):145-8, 2015.

[6] Verma, Rakesh, "A visual and interactive automata theory course emphasizing breadth of automata". ACM Sigcse Bulletin, 2005.

[7] Merceron, Agathe & Yacef, Kalina, "Web-based learning tools: storing usage data makes a difference", 104-109, 2007.

[8] M. Lucas, Joan & Jarvis, Jonathan, "Incorporating transformations into JFLAP for enhanced understanding of automata", 14-18, 2008.

[9] C. M. Baeten, J & J. L. Cuijpers, P & Luttik, Bas & Van Tilburg, Paul, "A Process-Theoretic Look at Automata" 1-33, 2010.

[10] Rodger, Susan & Wiebe, Eric & Min Lee, Kyung & Morgan, Chris & Omar, Kareem & Su, Jonathan, "Increasing engagement in automata theory with JFLAP" SIGCSE'09 - Proceedings of the 40th ACM Technical Symposium on Computer Science Education. 41. 403-407, 2009.

[11] S, Vinay & Prabhu, Akshata & Puranik, Kavitha & Antin, Lusi & Kumar, Viraj. (2015). JFLAP Extensions for Instructors and Students. Proceedings - IEEE 6th International Conference on Technology for Education, T4E 2014. 140-143. 10.1109/T4E.2014.22.

[12] Pradhan, Tribikram, "Enhancement of Turing Machine to Universal Turing Machine to Halt for Recursive Enumerable Language and its JFLAP Simulation", *International Journal of Hybrid Information Technology*, 8. 193-202, 2015.