Computation Problems and Decidability: Finite State Halting Problems in Turning Machines

M. Muzammul*, M. Awais, M. Yahya Saeed, Arfa Zaidi, Anaiza Akram

Department of Software Engineering, Government College University, Faisalabad, Pakistan

Abstract
In this paper we are going to discuss different type of model based computation languages like Finite automata, turning machine, Pushdown automata. The main reason behind this research is that, understanding the decidability of halting problems. It accomplish a proof of stated argument that some computational languages are decidable but some questions cannot decidable. Limited automata and standard dialects have been helpful in a wide assortment of issues in calculation, correspondence and control incorporate formal demonstrating and confirmation. This paper provide the decidable problem of finite automata (NFA, DFA), pushdown and turning machine. We summarize the discussion with some problems and provide the theorems as well.

Key words
Decidability of halting problems, decidability computation, finite automata, turning machine decidability

1. Introduction

Understanding whether the program eventually end is of extreme significance in a few functional field of PC building. For example if there should be an occurrence of the safety- critical continuous it is even imperative that one can limited the time break from program begin to its end. In this way, we verify the program ends that is also said halting problem. In the turning machine the halting problems are undecidable. But while the halting problem in all finite states system are decidable? There are many theorems with proof of availability of halting problem. The section 2 is provided the halting problem with pseudo code calculation.

2. Halting Problem

The halting or hang problem present the program whether we mention in the fig 1 where the given value of program p is eventually end on its inputs of i. In the program if the program halt in its inputs statement then it will be return true otherwise it will be return false [1].

```
void halt (p,i)
{
    if(the program p halt on the input i)
        {
            return true; // it means that p halt
        }
    else
        {
            return false; // it means that p doesn't halt
        }
}
```

Fig. 1 Pseudocode of the Halting Function

3. Decidability

In decidability we discuss the problem with the answer of No and Yes. Let the string w and the problem p accept and halt the input string which is provide in the Turing machine are the recursive or decidable. Every Turing-Acceptable that are decidable language.

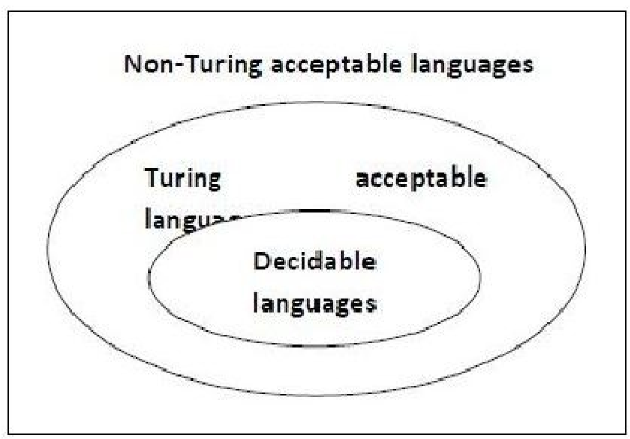

Fig. 2 The overview of decidable language

The problem of the P is the decidable if the language accept in the all cases of the Turing machine (TM).
For the decidable language, for all the input strings that are acceptable and halt in the Turing machine (TM) or reject the states. Which is as shown in the fig.

Manuscript received August 5, 2018
Manuscript revised August 20, 2018

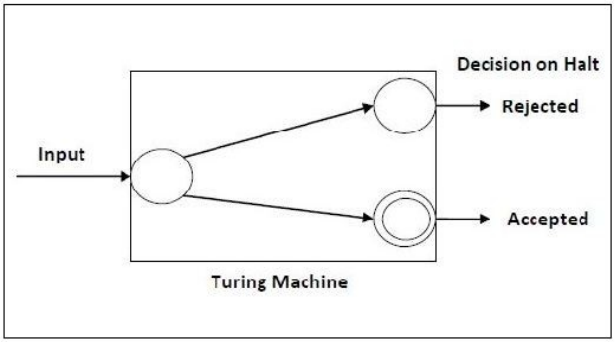

Fig. 3 The overview of turning machine [3]

Example 1: We take the example of prime number m whether to check the problem decidability which is decidable or not.Prime numbers = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29}
Divide with prime numbers with all the number of between
2 and under square of prime number the start from the 2.
If the number produce the leftover portion zero, then it will be goes to rejected states else it will be goes to the acknowledged state. So we can says that if the state is rejected then it will be mention with No otherwise it will be mention with Yes. Hence, we can decidable such that problem.
Example 2: We take another example of the natural number n whether to check the problem decidability which is decidable or not.
Natural numbers = {1, 2, 3, 4, 5....}
Take the modulus of the natural number n between 2 check the output for this problem.
If the number produce the leftover portion zero, then it will be goes to rejected states else it will be goes to the acknowledged state. So we can says that if the state is rejected then it will be mention with No otherwise it will be mention with Yes. Hence, we can decidable such that problem.
The machine that decided if the answer will be yes then the halt with yes state. If the answer will be No then the halt with No state [6].

4. Halting problem in turning machines

Let the string w of input in the Turing Machine (TM). But the problem is that the machine it will be halt which any state of point whether it is the final state with accepting the all inputs Yes or reject it with any state No. The Hang Machine produce the Yes or No in the limited measure of time. If hang machine ends in the limited measure of the time yield will be goes to Yes otherwise goes to No. We can mention in the figure are as follows:

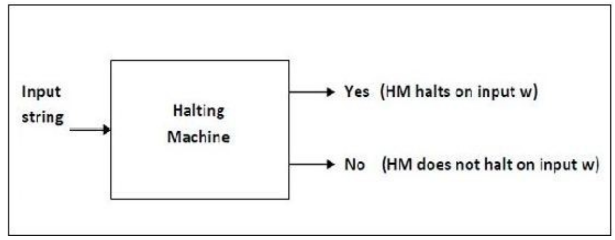

Fig. 4 The overview of halting machine

In this figure the input string given and if the input string accepted then His return with Yes and halt. Otherwise His return with No and halt.
In general we can't always know, the best we can do is run the program or solve the problem and check whether it halt with yes state or no state. We can mention in figure it always halt or it may sometimes loop.
In this figure the input string given and if the input string accepted then His return with Yes and loop. Otherwise His return with No and halt [4].

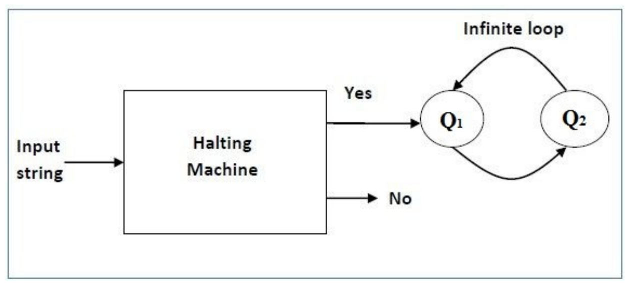

Fig. 5 The overview of halting machine

5. Decidable Problems

In decidable problem the regular languages are generally the decidable.

A. Acceptance problem for deterministic finite automata (DFAs)

In DFA we check the input string w which is given in the deterministic finite automata that accepts the following language:
BDFA = {(E, w) : EistheDF Athatacceptswiththeinputstringw}
Theorem:
BDFA is the decidable and accept the language (L) and with the string w, so we can check that if w is belongs to L.
Proof:
In this figure show that the DFA accept the language L with input string w is accepted W L then it will be goes to final state (yes) otherwise it will be goes to reject state (No) [2] .

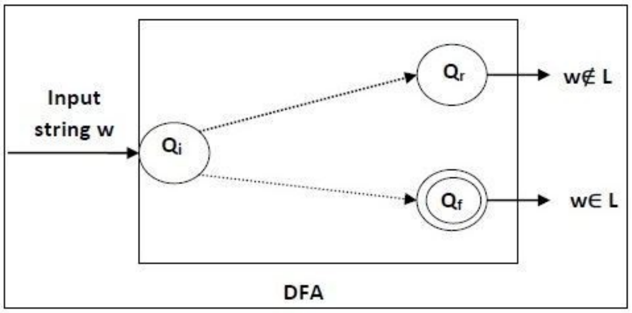

Fig. 6 The overview of decidable problem

B. Acceptance problem for NFA

In NFAs we check the input string w which is given in the non-deterministic finite automata that accepts the following language:
BNFA = {(E, w) : EistheN FAthatacceptswiththeinputstringW
Theorem:
BNFA is the decidable languages.
Proof:Construct the Turing machine (TM) that takes as input the representation of a BNFA and with the string w. we take the two approaches to make the TM the first approach is to simulate the NFA on w. The other approach is convert NFA to DFA. If machine accept the language with all input string then it will be goes to accept with final state otherwise rejected.

C. Empty problem for Deterministic finite automata

In emptiness problem we check the input string w which is given in the deterministic finite is the empty or not.
BDFA = {(E): E is the DFA and L (B) = Empty Set}
Theorem:
BDFA is the decidable languages.
Proof:
Construct the Turing machine the input E where E is the DFA, so we mark the state on which is a start state. We can repeat it until the no new states are marked. If the accepted states are marked then it will be goes to reject states else it will be goes to acknowledged states [5].

D. Acceptance problem of the regular expression

Consider the regular expression check the input string is decidable.
BREX = (R, w): R is the regular expressions and w belongs to L(R) that generate with the input strings w.
Theorem:
BREX is the decidable languages.
Proof:
In regular language do not matter which we use like DFA, NFA or regular expression the actual problem is that the string w of the L is the decidable or not. In this problem we chose the two approaches like the first approach to simulate NFA on w and other approach is that the language R convert NFA to a B. If the machine accept the string then it will be goes to acceptance states else goes to reject states.

E. Equivalence language problem for DFA

In equivalence problem we check that whether two DFA which is given in the language are recognize the same language.
EDFA = (A, B): A and B the DFA and L (A) = L (B).
Theorem:
EDFA is decidable languages.
Proof:
We consider that the language L (X) and L (Y) are the regular languages. So we know that the regular languages are closed under the union. Consider the DFAs produce the A and B to DFA E to accept that:

$$L(E) = L(X) \cap L(Y) \cup L(X) \cap L(Y) \quad (1)$$

If the machine accept the string then it will be goes to Acceptance states else goes to reject states.

6. Conclusion

Inside this paper we tended to the ending issue on limited state programs. As we provide the different types of the proofs examples of halting problems. The proof shows the finite states of halting problems whether it will be accepted or rejected states in the Turing Machine (TM) as well. Different proof and examples shows the different results whether it will be halt with accepted states (Yes) else it will be halt with rejected states (No). The consequences of this paper are expected to give a more instinctive comprehension of the ending issue.

References

[1] Kirner Raimund, On the Halting Problem of Finite-State Programs. Timmendorfer Strand, Germany (2007).
[2] Rabin and D. Scott, ”Finite Automata and their Decision Problems”, IBM Journal of Research and Development, (1959)
[3] Wason Saksham and Kumar Piyush, ”Research Paper on Automata ”, IJIRT Journal of Research and Development, (2014).
[4] Ylikoski, A.: The Halting Problem on finite and infinite computers. Journal of Advanced Computing Machinery 36 (2005.
[5] ”www.tutorialspoint.com/automata t heory30 − 01 − 2017”
[6] www.iis.sinica.edu.tw/bywang/courses/theory-comp/chapter-4.pdf 30-01-2017

Anaiza Akram got her (BSSE) degree from Sargodha university women campus Faisalabad, Pakistan. From session 2013-17.And worked as lecturer one year at Pakistan scholars Academy Faisalabad, Pakistan. She started her MS Software Engineering degree from Government college university Faisalabad, Pakistan. Session 2017-19.Her fanatical interested area of research is new technologies re-engineering with Machine learning problem in computation and automatic data summarization in Machine learning and data mining.

Muhammad Muzammul received his BSIT (2011-2013) from Govt. College University, Faisalabad (GCUF)-Pak. He worked as Lecturer 1.5 year in department of Software engineering GCUF. He worked as websoft trainer in Pro websoft org. He joined Fatima Jinnah college, Pak as lecturer and due to excellent performance he become Head of college in 2017. He started his MS (Software engineering) session 2017-2019 (GCUF), and he is top most student of class with CR ship and excellent in research with area of Automatic Machines learning to artificial intelligence with latest technologies trends and re-engineering.

Muhammad Awais received his Master Degree in Computer Science from the University of Agriculture, Faisalabad, Pakistan, in 2001, MPhil degree in Computer Science in 2004 from the University of Agriculture, MPhil degree in applied information Science in 2008 from the Albert-Ludwigs University, Freiburg, Germany and Ph.D. degree in Robotics from University of Bayreuth, Germany in 2013. In 2005, he joined the Government college university Faisalabad as Lecturer and became Assistant Professor in 2016. His current research interests include Human robot interaction, intention estimation and machine learning.

Arfa Zaidi completed her BS (Software Engineering) in 2017 from University of Sargodha Women Campus Faisalabad, Pakistan. She is student of MS in software engineering from Government College University of Faisalabad, Pakistan from session 2017-2019. Her area of interest of research is Automata state problems with Reengineering to Quality Assurence.

Muhammad Yahya Saeed received the MSc degree from UAAR, Rawalpindi, in Computer Science in 2003 and M.S degrees in Computer Science from UAF,Faisalabad in 2007. He has also done MSc from UAF, Faisalabad in Statistics in 2000.Currently he is doing PhD from GCUF,Faisalabad in Computer Science since 2015.He is faculty member in GCUF in software Engineering Department.