International Journal of Research in Circuits, Devices and Systems

E-ISSN: 2708-454X
P-ISSN: 2708-4531
IJRCDS 2022; 3(1): 28-33

www.circuitsjournal.com
Received: 07-11-2021
Accepted: 16-12-2021

S Shruthi
School of Computing Science and Engineering, Vellore Institute of Technology, Chennai, Tamil Nadu, India

Correspondence
S Shruthi
School of Computing Science and Engineering, Vellore Institute of Technology, Chennai, Tamil Nadu, India

Detailed study on Turing machines and quantum Turing machines

S Shruthi

Abstract
Turing machines formed the simple model of all existing modern computers today. The effects of a modern quantum computer are modelled by using the Quantum Turing Machine. Turing machines provide a basic limit and extent of what all can be computed. A Turing Machine helps to solve mathematical questions just by following rules or a set of algorithms carefully. For this he had to come up with a generic way of how machines or programs could work. The Turing Machine is a device that accepts Recursive Enumerable Language. Generally, type 0 grammar is used. The quantum Turing machine is the quantum analogy of a Turing Machine. Quantum Turing machines are abstract models which are seen equivalent to the quantum circuit. It achieves all features of a quantum computer that too without entanglement. In this all bits of a Turing tape are converted into quantum bits also called "qubits". This paper highlights how significantly the contributions of Alan Turing affected the way we view computation and computer science in the modern world today.

Keywords: Turing machine, algorithms, computation, grammar, automata

1. Introduction
Computational science is the third leg of science in addition to theoretical and experimental science. It forms the intersection of Computer Science, Mathematics and Science. Computational science is made possible by the advent of powerful computers. Increase in computational power has enabled us to design and conduct experiments and model systems that are too big, expensive or dangerous to experiment with in the real world. What-if scenarios can be run very quickly using computational power. Large amounts of data can be collected and analysed that are produced by these models. It is also to be noted that computational models cannot replace traditional field experimentation approaches. Different approaches are appropriate for different situations.
Mathematical models integrated with scientific computational methods can be used to solve real world problems. Computational science is that interdisciplinary field which aims at creating mathematical models to solve practical problems. Theory of Automata is one such model which is used in designing several hardware and software applications. Lower-level models such as the NFA (Non-Deterministic Finite Automata), DFA (Deterministic Finite Automata), ε-NFA (Epsilon-NFA) etc. are not as useful as real computers. Thus, a need for a theoretical model which can be equivalent to all standard computers comes up.
One might assume that if a machine is given enough memory, time and innovation from the programming end any problem can be solved. But even now there exist many problems that are unsolvable by a computer machine. No matter what amount of time and expertise is spent, some problems cannot be solved by developing algorithms and models. Let's say we need to write an automated test to check if a program will eventually finish running and stop or if it will continue to run forever. But the automated test that we write never actually runs the program. It just reads the lines of the code. So, it is not possible to test the code without actually executing it. This is an essential problem in Computer Science. It is called the 'Halting Problem'. In the theory of computation hence we prove that such problems may or may not be solved by using models of computation. A model of computation is an abstract representation of a machine. It isn't a kind of physical device. Nevertheless, like the machines it takes input and computes an output. These models are used to think about and solve problems in computing. So, given the limitations of computation, it is essential to analyse what kind of problems a computer can solve. String and Language are the key concepts to help solve this problem.
Turing intended to find what kind of problems are computable or solvable. He derived that a task can be computed if a sequence of instructions can be specified which completes the task at hand when used in a machine.

These instructions that are written to solve a problem are called effective procedures or algorithms.
In the mathematical models, the finite state machine lies in the lowest layer, which accepts ‘regular languages. Moving one step ahead we have Pushdown Automata, which accepts Context Free Language. Turing Machines can be considered to be the standard theoretical model for all computers. Alan Turning first came up with these machines in 1936.
Turing machines come in the highest of the computational model hierarchy. Turing machines are more powerful than both finite state machines and Pushdown automata. Turing computers recognise the 'Recursively Enumerable Language' category of languages. It is basically an abstract machine which uses a set of rules to manipulate symbols on a tape. The tape consists of a sequence of infinite symbols. The current control is indicated by using a ‘tape head’. Depending on the kind of computation required the tape head either moves toward the left or right. Input symbols are placed in the cells of the tape and the remaining cells of the tape are filled with ‘blank’ symbols.

2. Working and Description of Turing machine

During the 1930’s ‘Computers’ were referred to Humans especially women who were given a set of instructions and carried out tedious computations by hand. This job required little thought and definitely no mathematical intuition. The term ‘effective procedure’ was too vague to do anything rigorous with, so he decided to define it himself. He based his definition on a human computer. He decided that anything a human computer could do, by mindlessly following instructions using no insight or genius whatsoever was an effective procedure. He thought of what a human computer might do while computing. They basically perform 4 major tasks:

1. Read instructions.
2. Read and write symbols on a paper according to the instructions.
3. Occasionally erase the symbols and replace them with new symbols.
4. Stop, when the computation is done.

Turing realised that all the steps done by the human computer can be replicated by a simple theoretical machine. This is where the notion of Turing Machines comes in. Turing defined these machines as the following:
‘You have a way of writing information in a coded form’.
The machine would replicate all the task performed by a human computer one symbol at a time. It would read, erase and replace one symbol at a time. The piece of paper over which the human works could be replaced by an infinitely long piece of tape. The piece of paper is divided up into squares and in each of the squares there is either a 0 or a 1 or even a blank space. But, how does the machine know what to do?
When the human computer computes calculations their brain tells them what to after each step. This process had to be integrated on the machine. Turing thought of these as states of the mind. Similarly, for the machine he devised the idea of internal states, which tells the machine what to do. The machine is an automatic machine (a-machine), which means that the current state and symbol (called the configuration) being scanned totally determines the machine's behaviour at any given time. This is known as the determinacy condition. These a-machines differ from so-called choice machines, in which the future state is determined by a decision made by an external device or operator.

A Turing machine can do three different actions:

1. Print Si, proceed to state qj by moving one square to the left.
2. Print S_i, move to the right and go to the state q_j.
3. Go to state qj, print Si and don't move.

The machine's elementary or atomic actions are referred to as such. Hundreds of thousands, if not millions, of these atoms may appear in a sophisticated calculation.

2.1 States

The head has a sub device that I'll refer to as the indication. This is a distinct form of working memory from the first. The indicator can be set to any of a number of different 'positions.' In Turing machine language, the position of the indicator refers to the state of the machine at any given time. To demonstrate how the indicator works, consider keeping note of whether the most recent symbol encountered was a '0' or a '1'. The indicator is set to its first position if the value is '0,' and to its second position if the value is '1.'
Commercially available computers are hard-wired to perform elementary operations far more complex than those performed by a Turing machine, such as add, multiply, decrement, store-at-address, branch, and so on. The exact composition of the primitives list differs from one manufacturer to the next. This is a remarkable fact: none of these computers can outperform a Turing machine. The Turing machine, despite its austere simplicity, is capable of calculating anything that any computer on the market is capable of computing. As the head is constantly shuffling backwards and forwards, one square at a time, over an infinite length tape, it is frequently falsely stated that a Turing machine is necessarily slow. However, because a Turing machine is an idealised technology, it has no real-world speed limits.

2.2 The table of instructions

A Turing machine's programme, or ‘instruction table', is a finite set of instructions, each of which requests that specific atomic operations be done if certain criteria are met. Every instruction is in the following format:
Write y on the square beneath the head if the current state is n and the symbol beneath the head is x [y may be similar to x].
'Move left one square, "move right one square,' or 'stop' may be written in place of - - -.
[3] A Turing machine is defined as a 7-tuple (Q, q0, b, F,) where q0 belongs to Q and Q is a finite collection of states with one of them, q0 belongs to Q, being the specified starting state (this is the state the machine starts its operation in). Consists of a limited number of symbols, one of which, b, is a defined starting state (this is the state the machine starts its operation in). Γ is a finite set of symbols with one of them b∈Γ being a designated starting state (this is the state the machine starts its operation in). ∑⊂Γ is a subset of input symbols P⊂Q is a subset of accepting states which finalizes the computation (when the machine reaches F, the computation final state). σ: Q x Γ → Q x Γ x {L, R} is a partial transition function (if the machine reaches a state and input that are not defined for σ, the machine halts). q0∈Q is

the start state, qacceptQ is the accept state, and qrejectQ is the reject state in this transition function, where qaccept ≠ qreject. The core of a Turing machine, as can be seen from the preceding, is its transition, function σ since it tells us how the machine gets on from one configuration to another. The present state of a Turing machine, as well as the tape contents and head location, are frequently used to describe it. The machine at state Q∈q, reads the current symbol γ∈Γ on the tape loading to σ (q, γ) = (q1, γ1, d) where q1∈Q is the next state γ1∈Γ is the output symbol being written by the head on the tape. d∈ {L, R} is the movement of the head (left or right) on the infinite tape. As an example, consider a Turing computer that is designed to process a binary tape to the right of the start point, stopping when the number of ones reaches 10. In this case, we have Q = {q0, q1, …, q10}, q0 is the starting state, F = {q10}, Γ = {0, 1, b}, Σ = {0, 1} and σ defined as follows: σ (qi, 0) = (qi, b, R) and σ (qi, 1) = (qi+1, b, R). The machine reads the input left to right. It writes blanks on the tape and stops after reading 10 1-symbols. It then transitions to the final state, q10, at this moment.

3. Turing test

The Turing Test is an artificial intelligence (AI) research approach for determining if a machine can think like a person. Alan Turing is credited with creating the Turing Test. He was an English computer scientist, cryptanalyst, mathematician, and theoretical biologist. Turing proposed that a computer may be said to have artificial intelligence if it can imitate human behaviour in specific scenarios. The original Turing Test's three terminals are physically isolated from one another. Computers operate two of the terminals, while people operate the other two.

During the test, one of the people serves as the questioner, and the other two humans and the computer serve as the replies. The questioner asks the respondents questions about a certain topic in a specific manner and context. The questioner is then asked to determine which respondent was human and which was a computer after a specified amount of time or a set number of questions. The test is carried out several times. The machine is judged to have artificial intelligence if the questioner makes the correct decision in half of the test runs or less, because the interrogator regards it as "just as human" as the human respondent.

3.1 History

The Turing Test has a long and illustrious history.

Alan Turing, a pioneer of machine learning in the 1940s and 1950s, is honoured with the test's name. While at the University of Manchester, Turing proposed the test in his 1950 article "Computing Machinery and Intelligence".

Turing proposed a new spin on what is known as "The Imitation Game" in his article. The Imitation Game does not use artificial intelligence, but rather three human players in three different rooms. A man, a female and a male or female judge are in each room, which is connected via a screen and keyboard. While the female tries to convince the judge that she is the male, the judge tries to figure out who is who. During the test, one of the humans assumes the role of Turing, transforming the game's idea to include an AI, a human and a human questioner. The task of the questioner thus becomes determining which is the AI and which is the human. Many AI have passed the test because to scientific advancements; one of the earliest was ELIZA, a programme designed by Joseph Weizenbaum.

The Turing Test has been questioned throughout the years, in part because in order for a machine to display human-like intelligence, the nature of the inquiry had to be constrained historically. For many years, a computer could only get a high score if the questioner phrased the questions in such a way that they only had "Yes" or "No" answers or were limited to a specific subject of knowledge. When the queries were open-ended and demanded conversational responses, the computer software had a harder time fooling the questioner.

Furthermore, a programme like ELIZA could pass the Turing Test by manipulating symbols it doesn't completely comprehend. This, according to John Searle, does not determine intelligence.

The subject of whether or not a machine can pass the Turing Test has become meaningless to many researchers. Rather of concentrating on persuading people that they are speaking with a human rather than a computer programme. The actual objective should be to improve the efficiency and intuitiveness of human-machine interactions. Using a conversational interface, for example.

The original Turing Test is still utilised today, despite the fact that its variations are frequently more relevant to our current understanding of AI. For example, since 1990, the Loebner Prize has been given to the most human-like computer programme as determined by a team of experts. The competition follows the Turing Test's regular guidelines. Critics of the award's significance dismiss it as a PR stunt rather than a true test of whether machines can think The University of Reading organised a competition to celebrate the 60th anniversary of Turing's death in 2014, In that a chatbot named Eugene Goostman that simulates a 13-year-old child deceived 33% of the judges, passing the Turing Test in the eyes of some. Those who say that there weren't enough judges, that other robots have performed better in the past, and that the test is illegitimate because it only lasted five minutes has pounced on this so-called first pass. In front of an audience of 7,000 people, Google Duplex arranged an appointment with a hairdresser over the phone in 2018. The receptionist had no idea that they weren't speaking with a real person. Some consider this to be a modern-day Turing Test pass, despite the fact that it does not follow Alan Turing's original test format.

GPT-3, an OpenAI natural language processing model, has the highest chance of winning the exam in its true form of any technology now available, according to some. Despite its advanced text-generation skills, many have criticised the machine, arguing that it might be tricked into answering nonsensical questions and so fail the Turing Test.

Despite significant disagreement over the Turing Test's current relevance and the legitimacy of contests based on it, the test remains a conceptual starting point for debating and developing AI. The Turing Test is still used to define intelligence and serves as a starting point for discussions on what we should anticipate from technology in order for them to cooperate. The Turing Test remains essential for defining AI as we continue to make breakthroughs in AI and better understand and map how the human brain operates. It serves as a starting point for discussion on what we should anticipate from technology in order for them to cooperate.

4. Quantum Turing machine

A Quantum Turing Machine (QTM), also known as a universal quantum computer, is an abstract machine that is

used to simulate quantum computer phenomena. It provides a simple model that encompasses all of quantum computation's power-that is, any quantum algorithm may be formalized as a quantum Turing machine. The computationally equivalent quantum circuit, on the other hand, is a more prevalent model.

However, in retrospect, the quantum Turing machine model appears to be a fairly complicated model, and not a particularly good tool for determining whether quantum algorithms can be effectively implemented on a quantum computer. The quantum circuit model is a more practical option. Deutsch [10] was the first to describe quantum circuits, but Yao [11] suggested and examined the now-standard acyclic variation of the quantum circuit model a few years later. The quantum Turing machine model, on the other hand, looks to be a pretty sophisticated model, and not a particularly suitable tool for deciding whether quantum algorithms can be properly implemented on a quantum computer. A more realistic solution is the quantum circuit model. Although Deutsch [10] was the first to define quantum circuits, Yao [11], a few years later, proposed and investigated the now-standard acyclic version of the quantum circuit model [7].

Quantum Turing computers may be linked to classical and probabilistic Turing machines using a framework based on transition matrices. In other words, a matrix can be specified whose product with a matrix representing a classical or probabilistic machine yields the quantum probability matrix for the quantum machine. Lance Fortnow demonstrated this.

The idea of a Turing machine configuration, which is a traditional representation of the machine's state, tape head location and tape contents, must be defined to explain the global growth of a quantum Turing machine caused by a particular transition function. A Turing machine's state and tape head location are elements of the sets Q and ZZ, respectively, while the contents of a Turing machine tape can be described by a function T:Z→ΓT:Z→Γ, which specifies that the tape symbol $T(j)$ is stored in the tape square indexed by j, for each integer j.

A qubit is a quantum state Q with two ground states 0 and 1. As a result, Q is given by:

$Q=\alpha|0\rangle+\beta|1\rangle$

[4] A quantum Turing computer's basic structure is similar to that of a conventional Turing machine. We'll suppose that the tape is limitless and that a read/write head is present. All other components of the specification may be recast canonically, with the exception of the transition function. Q={qi}iQ={qi}i, finite state set

- Σ={σj}jΣ={σj}j, finite input alphabet
- Γ={γk}kΓ={γk}k, finite tape alphabet
- ⊔∈Γ⊔∈Γ, blank symbol with ⊔∉Σ⊔∉Σ
- q0∈Qq0∈Q, start state
- F⊆QF⊆Q, set of end states.

Lance Fortnow (2003) presented a paradigm that incorporates classical Turing computers, nondeterministic Turing machines, probabilistic Turing machines, and quantum Turing machines. His approach tackles elements of dynamics that cannot be transferred to the quantum level in a canonical way. The problem is that the transition function δδ of a Turing machine has the form δ:Q×Σ→Q×Σ×{L,R,N}δ:Q×Σ→Q×Σ×{L,R,N}, wherein the expression {L,R,N}{L,R,N} does not represent a set of states but a set of head actions. As a way-out, Fortnow proposes a representation of δδ as transition matrix in the space (q,B,i)∈Q×Γ∗×Z=:C(q,B,i)∈Q×Γ∗×Z=:C of configurations. The Turing machine's (local) state, as well as the contents of its tape, make up the real configuration. The Turing machine's global state is determined by the location of its head on the tape. Accordingly, the set C={ci}i∈IC={ci}i∈I of configurations is used for representing the ground states of the physical system.

4.1 Quantum Turing machine observables

After the machine has been stopped, measurements must be taken to determine the symbols at the tape places. For the set Q={q0,…,qr}Q={q0,…,qr} of states and the set Γ={γ1,…,γs}Γ={γ1,…,γs} of tape symbols the observables are

- $q\hat{}=\sum l=0rl|ql\rangle\langle ql|q\hat{}=\sum l=0rl|ql\rangle\langle ql|$ for the actual state.
- $B\hat{}(m)=\sum l=0sl|\gamma l\rangle\langle\gamma l|B\hat{}(m)=\sum l=0sl|\gamma l\rangle\langle\gamma l|$ with m∈Zm∈Z for the actual tape symbol at the tape position mm. We just need to measure the contents of a finite number of m∈Zm∈Z since we're assuming a calculation of finite length.

A classical state is used as the input to a quantum Turing computer, making it repeatable. Furthermore, the dynamics of a QTM is deterministical. As a result, quantum Turing machine computing stages can be replicated as well. On the other, reading out the outcome of the calculation, which is accomplished by performing a measurement, is probabilistic. Given the input xx, the probability of reaching a specific accepting configuration c_{accept} c_{accept} after nn computation steps is equal to $|\langle c_{accept}|U_n|c_{init}(x)\rangle|2|\langle c_{accept}|U_n|c_{init}(x)\rangle|2$. In the situation of PP ('correct' result) $=1/2+\varepsilon=1/2+\varepsilon$, one may regulate the accuracy of the outcome despite randomised measurement results, ε>0ε>0, due to the determinism of the dynamics. 'Wrong' results cannot be excluded, however.

4.2 Probabilistic Turing machines

Random state combinations are used in probabilistic Turing computers. As a consequence, the machine produces a probability distribution of potential outcomes, despite the fact that the calculation that leads to this conclusion involves a certain number of steps. The simulation property of a probabilistic Turing computer (for example, testing universality) is based on the equality of probability distributions.

A quantum Turing computer, unlike other types of Turing machines, can execute a variable amount of computing steps for a given input due to the need to determine the validity of a stopping condition. Despite the random nature of the computation result measurement, the computed output is a single tangible value. The calculation's outcome must be defined using statistical sampling. Though this will be an inconvenient complexity in most situations, it does provide a straightforward implementation alternative for probabilistic algorithms. In the setting of quantum Turing computers, the generation of genuine random numbers becomes very feasible. Unlike probabilistical Turing computers, quantum Turing machines act on states rather than probabilities. Because states hold considerably more information than probabilities, this is essential.

4.3 Conceptual problems of quantum Turing machines

Computations, according to the Church-Turing-Deutsch

thesis, should not only be theoretical but also executable. This has an impact on the concept of 'universality.' How can we simulate a second quantum Turing computer once the first has been completed? Specifically, how is a fresh input provided to a universal quantum Turing machine? One of the issues to be resolved in this regard is how to ensure an empty tape. Keep in mind that the machine is still processing during the 'deletion' procedure. Using reversibility as a solution is one option. But when should you stop rolling back? The outcomes of a quantum Turing machine are also probabilistic, which is a concern. According to its definition, it is only possible to evaluate whether two machines have the same behaviour on a rough scale (using statistical sampling). When the legitimacy of an infinite number of behaviour coincidences needs to be confirmed, this is a difficulty. Let n_x represent the number of computing steps required to arrive at an approximation error for the input x. The number n_x will, of course, vary depending on the x. Is the check of behaviour coincidence still computable if we assume $n_x \rightarrow \infty$ for $|x| \rightarrow \infty|$? Due to these problems, the existence of a universal quantum Turing machine has yet to be established.

4.4 Halt of a Turing machine

A quantum Turing machine's halt is substantially different from a conventional Turing machine's halt. Quantum dynamics are unstoppable. As a result, coming up with an appropriate definition of stopping is a difficult issue that has yet to be addressed satisfactorily. One approach is to think of a stop as a scenario in which the actual configuration does not change. This, however, would be incompatible with the crucial property of reversibility. The observable may be measured to check if the quantum Turing machine has reached the final state qfF. Unfortunately, measurements can influence the machine's real state q(t)Q. When quantum dynamics' determinism is abandoned.

In the last, in [16-23] readers, students, researchers can be found interesting works on emerging topics like Industry 4.0, Society 5.0, etc., and their importance for the smart era.

5. Conclusion

The discoveries and findings of Alan Turing contributed immensely towards how we view Computer Science and basic computing in today's modern world. The Turing Machine is the most extensive, deep, and accessible model of computation now available, and its accompanying theories enable fruitful discussion of many concepts involving "complexity". It has sparked new mathematical explorations by offering a kind of atomic structure for the concept of computation. The classification of distinct problems in terms of their complexity is a development of the last 30 years. It provides a platform-agnostic method of calculating complexity. Nowadays, a computer may be used to replicate the operation of a Turing machine and display the results on a monitor. It can serve as an enumerator or a function computer, among other things. The Universal Turing Machine is a problem-solving machine that can address a wide range of issues.

6. References

1. http://www.researchinventy.com/papers/v4i4/G044053060.pdf
2. http://www.researchinventy.com/papers/v4i4/G044053060.pdf
3. https://encyclopediaofmath.org/wiki/Quantum_Turing_machine
4. https://searchenterpriseai.techtarget.com/definition/Turing-test
5. https://encyclopediaofmath.org/wiki/Quantum_Turing_machine
6. A Lexical Analyzer with Ambiguity Support Luis Quesada, Fernando Berzal and Francisco J. Cortijo Department of Computer Science and Artificial Intelligence, CITIC, University of Granada, 18071 Granada, Spain, https://www.scitepress.org/Papers/2011/34768/34768.pdf
7. Social Media Based Opinion Mining Using Lexical Sentiment Analysis 1 2 3 4 5 Puja Munjal, Aditi Gupta, Mahima Abrol, Hema Banati and Sandeep Kumar 1, 2, 3 Jagannath International Management School, GGSIPU, Vasant Kunj, Delhi 4 Dyal Singh College, University of Delhi, Delhi 5 Department of Computer Science and Engineering, Jagannath University, Jaipur (PDF) Social media Based Opinion Mining Using Lexical Sentiment Analysis (researchgate.net)
8. Luis Quesada, Fernando Berzal, Francisco J. Cortijo Department of Computer Science and Artificial Intelligence, CITIC, University of Granada, 18071 Granada, Spain https://www.scitepress.org/Papers/2011/34768/34768.pdf
9. Sabine Glesnera Simone Forstera Matthias J¨agera a Fakult¨at f¨ur Informatik, Universit¨at Karlsruhe, 76128 Karlsruhe, Germany Email: {glesner|simone|matthias}@ipd.info.uni-karlsruhe.de, https://www.researchgate.net/publication/220370238_A_Program_Result_Checker_for_the_Lexical_Analysis_of_the_GNU_C_Compiler.
10. Ezhilarasu P. Associate Professor Department of Computer Science and Engineering Hindusthan College of Engineering and Technology Coimbatore, India. prof.p.ezhilarasu@gmail.com Krishnaraj N Head of the Department of Information Technology Sree Sastha Institute of Engineering and Technology, Chennai, India drnkrishnaraj@gmail.com. http://www.ijcset.com/docs/IJCSET15-06-05-037.pdf
11. Dan Chalmers, Simon Fleming, Ian Wakeman, Des Watson Informatics, University of Sussex, Brighton, U.K. Email: D.Chalmers@sussex.ac.uk, https://www.researchgate.net/publication/220876256_Rhythms_in_Twitter
12. Praveen Saini, Renu Sharma. Department of Computer Science & Engineering, Amrapali Institute of Technology& Sciences, Haldwani (U.K.), https://www.researchtrend.net/ijet/pdf/87-F-769.pdf.
13. https://www.guru99.com/compiler-design-lexical-analysis.html.
14. Singh J. Mathematical modeling with mixed chemotherapy on tumor cells in two different stages under depression effect. Int. J. Stat. Appl. Math. 2021;6(1):242-248. DOI: 10.22271/maths.2021.v6.i1c.655
15. Latha L, Preethi M, Grahalakshmi R. International Journal of Engineering and Advanced Technology (IJEAT), 2019 Sept, 8(6S3). ISSN: 2249 – 8958, https://www.ijeat.org/wp-content/uploads/papers/v8i6S3/F12050986S319.pdf

16. Nair MM, Tyagi AK, Sreenath N. The Future with Industry 4.0 at the Core of Society 5.0: Open Issues, Future Opportunities and Challenges" 2021 International Conference on Computer Communication and Informatics (ICCCI), 2021, 1-7. Doi: 10.1109/ICCCI50826.2021.9402498.
17. Tyagi AK, Fernandez TF, Mishra S, Kumari S. Intelligent Automation Systems at the Core of Industry 4.0. In: Abraham A, Piuri V, Gandhi N, Siarry P, Kaklauskas A, Madureira A. (eds) Intelligent Systems Design and Applications. ISDA 2020. Advances in Intelligent Systems and Computing, Springer, Cham., 2021, 13-51. https://doi.org/10.1007/978-3-030-71187-01.
18. Goyal Deepti, Tyagi Amit. A Look at Top 35 Problems in the Computer Science Field for the Next Decade, 2020. 10.1201/9781003052098-40.
19. Rekha G, Malik S, Tyagi AK, Nair MM. Intrusion Detection in Cyber Security: Role of Machine Learning and Data Mining in Cyber Security, Advances in Science, Technology and Engineering Systems Journal. 2020; 5(3):72-81.
20. Mishra S, Tyagi AK. Intrusion Detection in Internet of Things (IoTs) Based Applications using Blockchain Technolgy, Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2019, 123-128. Doi: 10.1109/I-SMAC47947.2019.9032557.
21. Tyagi Amit Kumar, Nair Meghna Manoj, Niladhuri Sreenath, Abraham Ajith. Security, Privacy Research issues in Various Computing Platforms: A Survey and the Road Ahead, Journal of Information Assurance & Security. 2020; 15(1):1-16.
22. Madhav AVS, Tyagi AK. The World with Future Technologies (Post-COVID-19): Open Issues, Challenges and the Road Ahead. In: Tyagi AK, Abraham A, Kaklauskas A. (eds) Intelligent Interactive Multimedia Systems for e-Healthcare Applications. Springer, Singapore, 2022. https://doi.org/10.1007/978-981-16-6542-4_22
23. Amit Kumar Tyagi, Aghila G. A Wide Scale Survey on Botnet, International Journal of Computer Applications. 2011 Nov; 34(9):9-22. (ISSN: 0975-8887).
24. https://www.tutorialspoint.com/compiler_design/compiler_design_regular_expressions.htm