Complexity of finite state Turing machine with other domain

Rajesh Kumar[1], Anju Jain[2], Rakesh Kumar[3]
[1]Department of Computer Science and Applications, Chhaju Ram Memorial Jat College, Hisar, India
[2]Department of Computer Science, Government College Hansi, Hansi, India
[3]Department of Computer Science and Applications, Kurukshetra University, Kurukshetra, India

Article Info

Article history:

Received Sep 21, 2025
Revised Apr 25, 2026
Accepted May 16, 2026

Keywords:

Complexity
Finite state automata
Non-deterministic FSA
Partial function
Turing machine

ABSTRACT

In this paper, the authors investigate and discussed the non-deterministic state complexity of certain operations on finite state Turing machine on other domain which includes partial function and natural function over an alphabet set Σ^*. It is found that in some boolean operations on said domains, the state complexity reaches up to upper bound $O(\sqrt{n!})$. This result is complement for the operation on Kleene star-free unary and recursive languages accepted by the finite state Turing machine.

Corresponding Author:

Rajesh Kumar
Department of Computer Sciences and Applications, Chhaju Ram Memorial Jat College
Hisar, Haryana, India
Email: rajtaya@kuk.ac.in

1. INTRODUCTION

The descriptive complexity issues about operation problem for regular languages of finite state Turing machine have been discussed by the authors in this paper. The operation problem on a language family is defined in term of total states in relation to accepting state by deterministic finite state Turing machine. It is well known that deterministic and non-deterministic finite state machines are equivalent in terms of their computational power [1], [2]. In terms of states, a given n-state NFA, one can always construct an equivalent DFA with at most 2^n state but the space complexity of this newly constructed DFA will be huge. It has been found that in most of the cases when an operation is expensive [3] for NFA, it is very cheap for DFA and vice versa. In this paper, the author gives two examples:

- In the two languages accepted by M- and N-state DFA in Figure 1, after concatenation it has a upper bound of $m * 2^n - t * 2^{n-1}$ states [4], where t is the number of final states it goes $m + n + 1$ when considering NFAs.
- When complement operation applied to a language, a n-state NFA in Figure 2 gives an upper bound of 2^n states [5] whereas by an n-state DFA results in exactly the same number of states in the complemented DFA and vice versa.

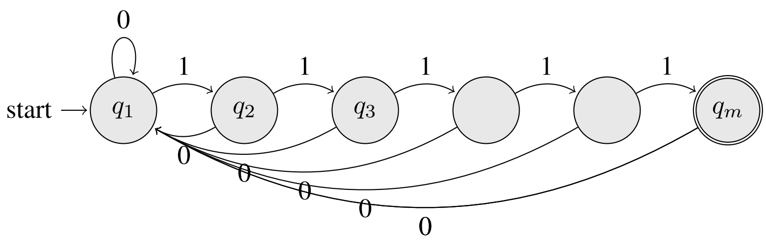

Figure 1. $\mathcal{L}_1$ = $\{(0,1)^*1^m$ | (0,1) ∈ Σ }

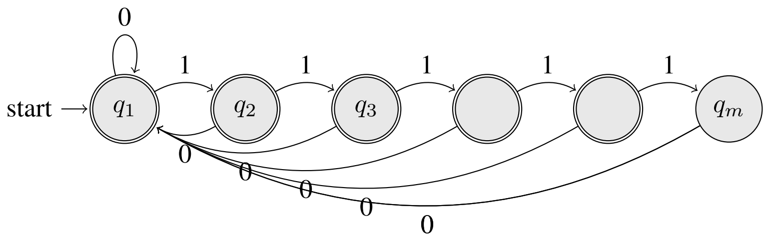

Figure 2. $\mathcal{L}_1$= $\{\Sigma^* - (0,1)^*1^m \mid (0,1) \in \Sigma\}$

2. TURING MACHINE: A STANDARD MODEL

The evolution of machine or automata are as: i) finite memory (encoded in state) —DFAs/ NFAs; ii) unbounded stack —NPDAs/DPDAs; and iii) unbound memory —TMs.

2.1. Definition

A turing machine: T= $\langle\ Q, \underbrace{\Sigma, \tau}_{\Sigma\subseteq\tau-\{\Box\}}, \frac{\delta_B}{\triangle}, Q_o, Q_F \subseteq Q, \underbrace{\Box}_{blanksymbol} \rangle$ where $\frac{\delta_B}{\triangle} \implies \delta_B : Q \times \tau \underbrace{\rightharpoonup}_{partial function} Q \times \tau \times \{L, R\} \triangle \subseteq_f (Q \times \tau) \times (Q \times \tau \times \{L, R\})$. Figure 3 depicts the interaction between these components.

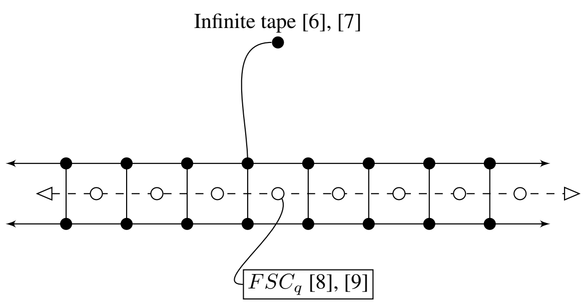

Figure 3. FSC_q- is the read/ write head of finite state control whereas infinite tape- is used for storing input, as scratch pad and also (in the case of Turing machine as transducers) for storing the output

2.1.1. Instantaneous Description (ID)

An instantaneous description [10], [11] of a TM is a sequence of the form xqy where x,y ∈ τ^* such that $x \neq \Box^* \neq y$.

- There is non-blank symbol either to the left of x or to the right of y,
- The first symbol of x and the last symbol of y are both non-blank,
- The first symbol of y is the symbol immediately under the read/write head,
- q is the current state of the FSC.

An ID is also called a configuration. A *move* of the TM from an ID $x_1q_1y_1$ to another ID $x_2q_2y_2$ is denoted $x_1q_1y_1 \vdash x_2q_2y_2$ where $x_1 = a_1....a_k$; $y_1 = b_1....b_l$ is possible provided one of the following holds.

- $\delta(q_1, b_1) = (q_2, c, R)$ and $x_2 = a_1....a_kc; y_2 = b_2....b_l$ or
- $\delta(q_1, b_1) = (q_2, c, L)$ and $x_2 = a_1....a_{k-1}; y_2 = a_kcb_1....b_l$.

The language accepted by a turing machine T is $\mathcal{L}(T)$= $\{x \in \Sigma^* \mid q_0x \vdash^* yq_fz, y, z \in \tau^*, q_f \in F\}$

3. TURING MACHINE AS ACCEPTORS

Example 1: $\mathcal{L} = \{a^nb^n \mid n \geq 1\}$ [12], [13]. Table 1 describes the transition function of the Turing machine for Example 1.

Table 1. Transition table for turing machine which accept the language $\mathcal{L} = \{a^nb^n \mid n \geq 1\}$

$\mathcal{Q}$ \ τ	a	b	$\mathcal{A}$	$\mathcal{B}$	$\square$
q_0	$(q_a, \mathcal{A}, \mathcal{R})$			$(q_2, \mathcal{B}, \mathcal{R})$	
q_a	$(q_a, \mathcal{A}, \mathcal{R})$	$(q_b, \mathcal{B}, \mathcal{R})$		$(q_a, \mathcal{B}, \mathcal{R})$	
q_b	$(q_b, \mathcal{A}, \mathcal{L})$		$(q_0, \mathcal{A}, \mathcal{R})$	$(q_b, \mathcal{B}, \mathcal{L})$	
q_0'				$(q_0', \mathcal{B}, \mathcal{R})$	$(q_f, \square, \mathcal{R})$
q_f					

In any ID the tape contents are of the following invariant property can be used to determine the transition:

$$A^*a^*B^*b^* \tag{1}$$

if $n \geq 0$ then the following transition could also be included:

$$\delta(q_0, \square) = (q_f, \square, R) \tag{2}$$

3.1. Algorithm

Problem: a language $\mathcal{L} = \{xx \mid x \in (a, b)^*\}$. Finding the midpoint of the sub-string and inserting a symbol "C" at the midpoint. Solution schema:

- Starting from the leftmost symbol move to the rightmost symbol and replace the first "$\square$" by "C".
- Change the leftmost lowercase symbol to UPPERCASE and the rightmost lowercase symbol to UPPERCASE.
- Exchange the "C" with the new uppercase symbol immediately preceding it.
- The tape invariant is now something like $\boxed{(A + B)^*.(a + b)^*.C.(A + B)^*}$.
- Repeat steps 2-3 till the tape contents become $\boxed{(A + B)^*C(A + B)^*}$.
- Now change all uppercase letter except C to lowercase (if necessary).

4. RESULTS AND DISCUSSION

Clearly the Turing machine discussed in previous section can only compute functions over strings over a finite alphabet [14], [15]. What about computing functions over arbitrary (computable) domains such as the naturals [16], [17], integers or the rationals or even Cartesian products [18], [19] of such domains or even more generally functions from one domain to another?

4.1. Computability in other domains

Clearly if the elements of such domains can be represented as string over an alphabet, then one can talk about Turing machine computability over such domains [20], [21].

4.1.1. Definition

A set $\mathcal{S}$ is represented by elements of another set $\mathcal{R}$ if there exists a partial surjective function [22] $\boxed{\mathcal{R} \rightharpoonup \mathcal{S}}$ called the interpretation [23], [24] of $\mathcal{R}$ onto $\mathcal{S}$, as in Figure 4, where:

- Partial is not every element in $\mathcal{R}$ need have a meaning in $\mathcal{S}$.
- Surjective is every element of $\mathcal{S}$ should have a representation in $\mathcal{R}$.
- On the other hand two or more elements of $\mathcal{R}$ may be interpreted as representing the same element in $\mathcal{S}$.

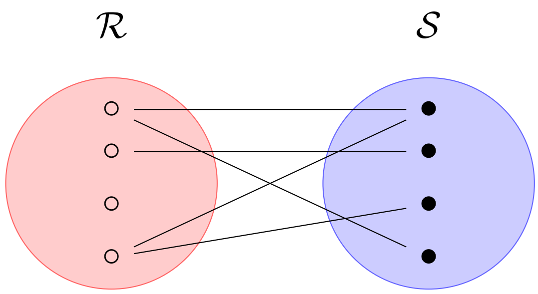

Figure 4. Mapping diagram of partial surjective function $\boxed{\mathcal{R} \rightharpoonup \mathcal{S}}$

Example: consider the unary representation of the naturals as strings of '1_S' terminated by a single '0'. Clearly $\mathcal{I} : \{0, 1\}^* \rightharpoonup \mathbb{N}$ is a partial function because strings such as '00100' have no interpretation in $\mathbb{N}$. $\mathcal{I}$ is also surjective since every natural does have a representation in $\{0, 1\}^*$. If leading 0_S are allowed in the representation, then each natural has more than one possible representation.

Giving two sets $\mathcal{S}_1$ and $\mathcal{S}_2$ represented by $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively through interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ respectively, as in Figure 5.

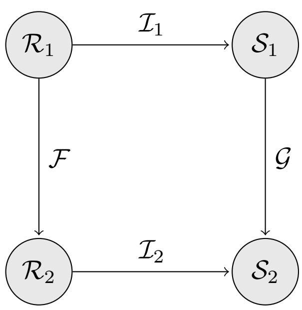

Figure 5. A partial function $\boxed{\mathcal{F} : \mathcal{R}_1 \rightharpoonup \mathcal{R}_2}$ represents a partial function $\boxed{\mathcal{G} : \mathcal{S}_1 \rightharpoonup \mathcal{S}_2}$ if for all $r_1 \in Dom(\mathcal{I}_1)$, $\mathcal{G}(\mathcal{I}_1(r_1)) = \mathcal{I}_2(\mathcal{F}(r_1))$ i.e. the above commutative diagram holds

Alternatively, one may attempt to construct $\mathcal{F}$ as a representation of $\mathcal{G}$ as a total function, rather than expressing the diagram as a partial function.

As in Figure 6, given $\mathcal{I}_1$: $\mathcal{R}_1 \rightharpoonup \mathcal{S}_1$ consider:
$\mathcal{I}_1^{-1}$: $\mathcal{S}_1 \rightarrow \mathbb{N}^R - \{\phi\}$ where
$\mathcal{I}_1^{-1}(s_1)$ = $\{r_1 \in \mathcal{R} | \mathcal{I}_1(r_1) = s_1\}$ i.e. $\mathcal{I}_1^{-1}$ for any $s_1 \in \mathcal{S}_1$ yield the set of all possible representations of s_1.
Then $\mathcal{F} : \mathcal{R}_1 :\rightharpoonup \mathcal{R}_2$ may also be extended to subsets of $\mathcal{R}_1$ as:
$\mathcal{F} : \mathbb{N}^{R_1} \rightharpoonup \mathbb{N}^{R_2}$ is defined for each $\mathcal{X}_1 \subseteq \mathcal{R}_1$ as $f(\mathcal{X}_1) = \{f(r_1) | r_1 \in \mathcal{X}_1\} = \mathcal{X}_1 \subseteq \mathcal{R}_2$. But one should be interested in ensuring that the following diagram holds for any $s_1 \in \mathcal{S}_1$, if $\mathcal{G}(s_1) \in \mathcal{S}_2$, require that:

- There is at least one representation $r_1 \in \mathcal{R}_1$ such that $\mathcal{I}_2(\mathcal{F}(r_1)) = \mathcal{G} \in \mathcal{S}_1$.
- $r_1 \neq r_1^{‘} \curlywedge \mathcal{I}_1(r_1) = \mathcal{I}_1(r_2) = s_1 \curlywedge \mathcal{F}(r_1), \mathcal{F}(r‘_1) \in \mathcal{R}_2 \Rightarrow \mathcal{I}_2(\mathcal{F}(r_1)) = \mathcal{I}_2(\mathcal{F}(r‘_1))$.
- There may be representation of s_1 in $\mathcal{R}_1$ for which $\mathcal{F}(r_1)$ may not be defined.

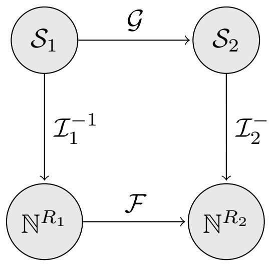

Figure 6. A function $\boxed{\mathcal{F} : \mathbb{N}_1^{R_1} \rightharpoonup \mathbb{N}_2^{R_2}}$ represents the possibility that $\mathcal{F}$ may not be defined for some value in $f(\mathcal{X}_1) = \{f(r_1) | r_1 \in \mathcal{X}_1\} = \mathcal{X}_1 \subseteq \mathcal{R}_2$

4.2. Turing-computable

As in Figure 7, a partial function $\mathcal{G} : \mathcal{S}_1 \rightharpoonup \mathcal{S}_2$ is Turing-computable [25], [26] if there exists a Turing-computable function $\mathcal{F} : \Sigma^* \rightharpoonup \Sigma^*$ on an alphabet Σ and interpretations $\mathcal{I}_1, \mathcal{I}_2$ with $\mathcal{I}_1 : \Sigma^* \rightharpoonup \mathcal{S}_1$ and $\mathcal{I}_2 : \Sigma^* \rightharpoonup \mathcal{S}_2$ such that:

- For each $s_1 \in \mathcal{S}_1$ and $\mathcal{F}(s_1) = s_2 \in \mathcal{S}_2, \exists x_1, x_2 \in \Sigma^*[\mathcal{I}_1(x_1) = s_1 \curlywedge \mathcal{I}_2(x_2) = s_2 \curlywedge \mathcal{F}(x_1) = x_2 \curlywedge \forall_{y_1} \in \Sigma^*[x_1 \neq y_1 \curlywedge \mathcal{I}_1(y_1) = s_1 \Rightarrow \mathcal{F}(y_1) \notin \Sigma^* \curlyvee \mathcal{I}_2(\mathcal{F}(y_1)) = s_2]]$
- For each $s_1 \in \mathcal{S}_1$, such that $\mathcal{G}(s_1) \notin s_2, \forall_{x_1} \in \Sigma^*[\mathcal{I}_1(x_1) = s_1 \Rightarrow \mathcal{F}(x_1) \notin \Sigma^* \curlyvee \mathcal{I}_2(\mathcal{F}(x_1)) \notin s_2]]$

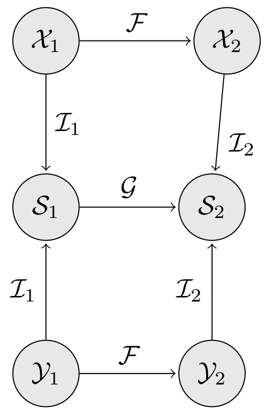

Figure 7. For any partial function $\boxed{\mathcal{G} : \mathcal{S}_1 \rightharpoonup \mathcal{S}_2}$ where $\mathcal{G}(s_1) \notin \mathcal{S}_2$ if and only if $s_1 \notin Dom(\mathcal{G})$

5. CONCLUSION

A consequences of the above definition is that the implementation of any function of any arity is that of a unary function on Σ^* for the chosen alphabet. Facts: with addition to natural function a binary function is considered which is defined as : $\boxed{+ : \mathbb{N} \times \mathbb{N} \Rightarrow \mathbb{N}}$. However for any pair of naturals $m, n \in \mathbb{N}$ it can be encoded over $\Sigma^* = \{0, 1\}^*$ in unary with '0' as a separator between the components of the pair. Hence $\boxed{\Sigma^* \Rightarrow \mathbb{N} \times \mathbb{N}}$ is defined as $\boxed{\mathcal{I}_1(1^m 0 1^n) = (m, n)}$ and all other patterns of strings $x \in \Sigma^* - \mathcal{L}(1^* 0 1^*), \mathcal{I}_1(x)$ is undefined. Similarly $\mathcal{I}_2$ for all the result is simply defined as:

$$\mathcal{I}_2(y) = \begin{cases} P \text{ if y} = 1^p \in 1^* \\ \text{undefined otherwise} \end{cases}$$

With above representation the partial function $\hat{+} : \Sigma^* \rightharpoonup \Sigma*$ that we require is given by:

$$\hat{+}(x) = \begin{cases} 1^{m+n} \text{ if x} = 1^m 0 1^n \text{ for all } m, n \geq 0 \\ \text{undefined otherwise} \end{cases}$$

Any Turing machine $\mathcal{T}_{\hat{+}}$ which implement $\hat{+}$ is a correct implementation of addition on the naturals.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Rajesh Kumar	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Anju Jain		✓				✓		✓	✓	✓	✓	✓		
Rakesh Kumar	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	

C	: **C**onceptualization	I	: **I**nvestigation	Vi	: **Vi**sualization
M	: **M**ethodology	R	: **R**esources	Su	: **Su**pervision
So	: **So**ftware	D	: **D**ata Curation	P	: **P**roject Administration
Va	: **Va**lidation	O	: Writing - **O**riginal Draft	Fu	: **Fu**nding Acquisition
Fo	: **Fo**rmal Analysis	E	: Writing - Review & **E**diting		

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author, [RK], upon reasonable request.

REFERENCES

[1] H. M. Yao and L. Jiang, "Machine-learning-based PML for the FDTD method," *IEEE Antennas and Wireless Propagation Letters*, vol. 18, no. 1, pp. 192–196, 2019, doi: 10.1109/LAWP.2018.2885570.

[2] N. Boullé and A. Townsend, "Learning elliptic partial differential equations with randomized linear algebra," *Foundations of Computational Mathematics*, vol. 23, no. 2, pp. 709–739, Apr. 2023, doi: 10.1007/s10208-022-09556-w.

[3] J. L. Hennessy and D. Patterson, "A New Golden Age for Computer Architecture Innovations like domain-specific hardware, enhanced security, open instruction sets, and agile chip development will lead the way." Cacm.acm.org. Accessed Jun. 21, 2025. [Online]. Available: https://cacm.acm.org/research/a-new-golden-age-for-computer-architecture/

[4] G. Jirásková, A. Szabari, and J. Šebej, "The complexity of languages resulting from the concatenation operation," in *International Conference on Descriptional Complexity of Formal Systems*, 2016, pp. 153–167, doi: 10.1007/978-3-319-41114-9_12.

[5] G. Jirásková and A. Okhotin, "State complexity of unambiguous operations on finite automata," *Theoretical Computer Science*, vol. 798, pp. 52–64, Dec. 2019, doi: 10.1016/j.tcs.2019.04.008.

[6] Y. Oktar, "A computing machinery using a continuous memory tape," *arXiv:*2401.02420, 2023.

[7] M. Bojańczyk and R. Stefański, "Single-use automata and transducers for infinite alphabets," in *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, 2020, pp. 1–14, doi: 10.4230/LIPIcs.ICALP.2020.113.

[8] H. Boche, A. Grigorescu, R. F. Schaefer, and H. V. Poor, "Algorithmic computability of the capacity of additive colored Gaussian noise channels," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, Dec. 2023, pp. 4375–4380, doi: 10.1109/GLOBECOM54140.2023.10436918.

[9] L. Dartois, P. Gastin, L. G. Guizouarn, R. Govind, and S. Krishna, "Reversible transducers over infinite words," *arXiv:*2406.11488, 2024.

[10] M. de Benedetto, "Explication as a three-step procedure: the case of the Church-Turing thesis," *European Journal for Philosophy of Science*, vol. 11, no. 1, p. 21, Mar. 2021, doi: 10.1007/s13194-020-00337-2.

[11] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game, or a completeness theorem for protocols with honest majority," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 218–229, doi: 10.1145/3335741.3335755.

[12] H. B. Axelsen and R. Glück, "On reversible Turing machines and their function universality," *Acta Informatica*, vol. 53, no. 5, pp. 509–543, Aug. 2016, doi: 10.1007/s00236-015-0253-y.

[13] B. Gonçalves, "The Turing test is a thought experiment," *Minds and Machines*, vol. 33, no. 1, pp. 1–31, Mar. 2023, doi: 10.1007/s11023-022-09616-8.

[14] F. Neven, T. Schwentick, and V. Vianu, "Finite state machines for strings over infinite alphabets," *ACM Transactions on Computational Logic (TOCL)*, vol. 5, no. 3, pp. 403–435, Jul. 2004, doi: 10.1145/1013560.1013562.

[15] Y. Forster, F. Kunze, and M. Wuttke, "Verified programming of Turing machines in Coq," in *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, Jan. 2020, pp. 114–128, doi: 10.1145/3372885.3373816.

[16] T. Ha, V. Harizanov, L. Marshall, and H. Walker, "Computability and definability," in *Structure and Randomness in Computability and Set Theory*, D. Cenzer, C. Porter, and J. Zapletal, Eds., Singapore: World Scientific, 2021, pp. 285–355.
[17] K. Mainzer, "Logical thinking becomes automatic," in *Artificial Intelligence - When do Machines Take Over?*, K. Mainzer, Ed., Berlin: Springer Berlin Heidelberg, 2020, pp. 15–45, doi: 10.1007/978-3-662-59717-0_3.
[18] J. You, R. Ying, and J. Leskovec, "Design space for graph neural networks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., New York, NY: Curran Associates, Inc, 2020, pp. 17009–17021.
[19] B. Jacquet, F. Jamet, and J. Baratgin, "On the pragmatics of the Turing test," in *2021 International Conference on Information and Digital Technologies (IDT)*, Jun. 2021, pp. 123–130, doi: 10.1109/IDT52577.2021.9497570.
[20] W. Sieg, "Gödel's philosophical challenge (to Turing)," *Studia Semiotyczne*, vol. 34, no. 1, pp. 57–80, 2020.
[21] V. Brattka and P. Hertling, *Handbook of computability and complexity in analysis*. Cham: Springer International Publishing, 2021, doi: 10.1007/978-3-030-59234-9.
[22] C. Borlido and B. McLean, "Difference–restriction algebras of partial functions with operators: discrete duality and completion," *Journal of Algebra*, vol. 604, pp. 760–789, Aug. 2022, doi: 10.1016/j.jalgebra.2022.03.039.
[23] J. Mayr et al., "Thermal issues in machine tools," *CIRP Annals*, vol. 61, no. 2, pp. 771–791, 2012, doi: 10.1016/j.cirp.2012.05.008.
[24] C. Knapp, "Partial functions and recursion in univalent type theory," *arXiv:*2011.00272, 2020.
[25] Z. Hou, "Turing machines and computability," in *Fundamentals of Logic and Computation: With Practical Automated Reasoning and Verification*, Z. Hou, Ed., Cham: Springer International Publishing, 2021, pp. 163–205, doi: 10.1007/978-3-030-87882-5_5.
[26] H. Petersen, "Some remarks on real-time Turing machines," *arXiv:*1902.00975, 2019.

BIOGRAPHIES OF AUTHORS

Rajesh Kumar obtained his B.Sc. Degree, Master's Degree (Master of Computer Applications) and Ph.D. from Kurukshetra University, Kurukshetra. Currently, he is head of the Department of Computer Science and Applications, Chhaju Ram Memorial Jat College, Hisar, Haryana, India. He is a co-author of three books. His research interests are in genetic algorithm, theory of automata, software engineering, artificial intelligence, design and analysis of algorithm and Linux administration and kernel design. He can be contacted at email: rajtaya@kuk.ac.in.

Anju Jain holds a Ph.D. in Computer Science and Engineering from Guru Jambheshwar University of Science and Technology (GJUS&T), India. She has served in various academic roles including lecturer, teaching associate, and assistant professor across academies, colleges, and universities. A two-time Gold Medalist, Dr. Jain has published several research papers in reputed journals and conference proceedings. She possesses nearly 15 years of teaching experience, with over 7 years in the Computer Science department in higher education in India. Her broad research interests include machine learning, data analytics, data mining, and evolutionary algorithms. She can be contacted at email: anjuaryan2012@gmail.com.

Rakesh Kumar obtained his B.Sc. Degree, Master's Degree – Gold Medalist (Master of Computer Applications) and Ph.D. (Computer Science and Applications) from Kurukshetra University, Kurukshetra. Currently, he is dean academic affairs, professor and chairperson in the Department of Computer Science and Applications, Kurukshetra University, Kurukshetra, Haryana, India. His research interests are in genetic algorithm, software testing, artificial intelligence, and networking. He is a senior member of International Association of Computer Science and Information Technology (IACSIT). He can be contacted at email: rakeshkumar@kuk.ac.in.