International Journal of Applied Resilience and Sustainability, Volume 2, Issue 2, March 2026, pp. 599-617

Published online at Deep Science

International Journal of Applied Resilience and Sustainability

Journal homepage: https://deepscipub.com/ijars

Analysis of blow-up situation in finite automata

Siddhartha Chatterjee [1], Ritwika Ghosh [2], Aparijita Das [3], Jonti Deuri [4], Suman Biswas [5]

[1] *Department of Computer Science and Engineering, College of Engineering and Management Kolaghat, KTPP Township, Purba Medinipur, West Bengal, India*
[2] *Department of Computer Science and Engineering, Institute of Science and Technology, Paschim Medinipur, West Bengal, India*
[3] *Department of Computer Science and Engineering, Sanaka Educational Trust's Group of Institutions, Durgapur, West Bengal, India*
[4] *Faculty of Engineering and Technology, Sharda University 73 Andijan, Boborshah Prospekt, Uzbekistan*
[5] *Department of Computer Science and Engineering (AIML), College of Engineering and Management Kolaghat, West Bengal, India*

Article Info:
Received 27 January 2026
Revised 24 February 2026
Accepted 28 February 2026
Published 04 March 2026

Corresponding Author:
Siddhartha Chatterjee
E-mail: siddhartha.chatterjee31@gmail.com

Abstract

The blow-up phenomenon in finite automata is one of the most far reaching and significant problems in formal language theory and system design based on automata. Discovering a nondeterministic finite automaton (NFA) transformed into an equivalent deterministic finite automaton (DFA) with the famous powerset construction can lead to an exponentially larger number of states and this phenomenon is known as the state explosion, which makes solving even moderately sized input automata computationally infeasible. This issue is also known as the blow-up problem and has far reaching consequences on the compilation design, pattern matching engines, model checking systems, natural language processing pipelines, and network intrusion detection systems. The interaction between descriptional complexity and operational complexity with mitigating strategies despite decades of theoretical exploration has not been fully synthesized in the light of an emerging computational requirement. The review is a systematic literature review of articles published as early as 2019 and 2025 to identify the contemporary picture of the analysis of blow-ups in finite automata. The paper also determines the mathematical properties of exponential blowup; it discusses the effectiveness of modern heuristic and algebraic mitigation strategies. It has been shown in the analysis that the worst-case exponential lower bounds are tight in proving that classical NFA to DFA conversion is impossible, although in practice the improvements of up to 40-60% are possible by using state-aware lazy determinization and simulation-based reduction.

Keywords: Finite automata, NFA to DFA conversion, State explosion, Descriptional complexity, Automata minimization, State complexity.

1. Introduction

Finite automata hold a strategic place in theoretical computer science as abstract models of computation to identify regular languages and are the basis of an impressive variety of systems of application [1-3]. Since the efficiency of an automaton with the state structure is the key to analysis efficiency in lexical analyzers implemented in production compilers and the deep-packet inspection engine implemented in security infrastructure, the ease or challenge of a system to computation is directly related to how well an automaton can analyze the state structure [2,4]. The blow-up situation, the exponential expansion of states which goes with the nondeterministic finite automaton as it is transformed into a deterministic one, and similar like expansions produced by complementation, intersection, and the one-way to two-way conversion represent the central object of this efficiency problem [5-7]. The classical powerset construction, formally defined independently in the middle of the twentieth century defines a property that any NFA in the n states can be transformed into a DFA [8-10]. Nevertheless, the final DFA can still have up to 2n states, the limit which is not exclusive to theoretical models but can be met by a clever choice of languages of witnesses. With a nondeterministic finite automaton State size n = 32, it means that there could be a DFA with more than four billion states - a size scale which makes naive determinization completely impractical unless structured intervention is provided. This exponential

https://doi.org/10.70593/deepsci.0202023

blowup is not just of scholarly concern, but an actual engineering hurdle of pattern matching, model checking and language processing. Since the original proofs of state complexity lower bounds, the continued research in formal language theory in this field has changed significantly. A large literature until the 2000s developed narrow constraints on the complexity of state of individual functions including union, concatenation, star, reversal, and intersection over dissimilar classes of regular languages. However more recently there is a focus on the descriptional complexity of combined operation sequences, symbolic and parametric automata models, which either delay or fail to determinize, and learning-theoretic models which capitalize on structure regularities in the automaton to be converted [3,11-14]. And at the same time the growing scale of cyber-physical systems, formal verification pipelines, regular expression-based data processing on scale has placed a new urgency on the applied aspect of these issues.

Nevertheless, even with such substantial amount of work, there are still a number of gaps that can be identified in research [15-17]. The relationship between NFA structural properties, including loop depth, the representatives of ambiguity class and strongly connected component topology, and the actual degree of exponential blowup has not been fully characterized in a form useful tohelp in the practical construction of tools. Second, although variants of symbolic automata and weighted automata have been suggested as alternatives, compressing representations of states, its strengths or weaknesses on the conditions of blow-up, even especially in the framework of model checking of automata with infinite alphabets, are underresearched. Third, the relative performance of the current mitigation techniques, such as simulation-based optimization, bisimulation quotients, and learning-based NFA compression have not been tested against a standardized collection of experimental protocols which might assist practitioners to make knowledgeable algorithmic decisions. Fourth, the effects of blow-up on a new application would be neural network verification through automata, quantum automata theory, and blockchain smart contract analysis, which is a relatively immature area of research. The paper fills these gaps with the help of a systematic literature review on the topics researched in the frame of PRISMA 2020 with references to publications that are up-to-date: 2019-2025. The review goals are as following: to describe the mathematical basis of mathematical blow-up in finite automata in classical and generalized models; to overview and critically review mitigation strategies such as exact and approximate methods of state reduction; assess the practical effects of state explosion on applications of automata, in theory and practice, such as model checking and pattern matching; and to suggest future research directions integrating formal theory and new application areas.

This paper has three folds contribution. It offers the initial literature review that is narrowly dedicated to the blow-up situation throughout the entire range of theoretical grounds to the implementation of mitigation measures in the 2019-2025 frame. It presents two synthesis tables in a comprehensive form that packages the findings on dimensions of techniques and areas of applications. And it develops a collection of prospective research questions that place the classical state complexity tradition within the current issues of formal verification, machine learning, and network systems. This review is a specific resource to researchers or practitioners at the interface of theoretical computer science and systems engineering by dispensing analysis over the fields of descriptional complexity, operational state complexity, automata minimization, symbolic representation, and model checking.

2. Methodology

This literature review took place in the framework of adhering to the Preferred Reporting Items of Systematic Reviews and Meta-Analyses (PRISMA) model (Fig. 1) requiring clear record keeping of the identification, screening, eligibility selection, and inclusion processes. The main databases that were searched were Scopus, Web of science (core Collection), IEEE Xplore and ACM Digital library. The search was limited to those publications that were published in the period between January 2019 and June 2025 to represent emerging and popular trends, but still have a relatively recent time frame. The entire database queries took place in June of 2025.

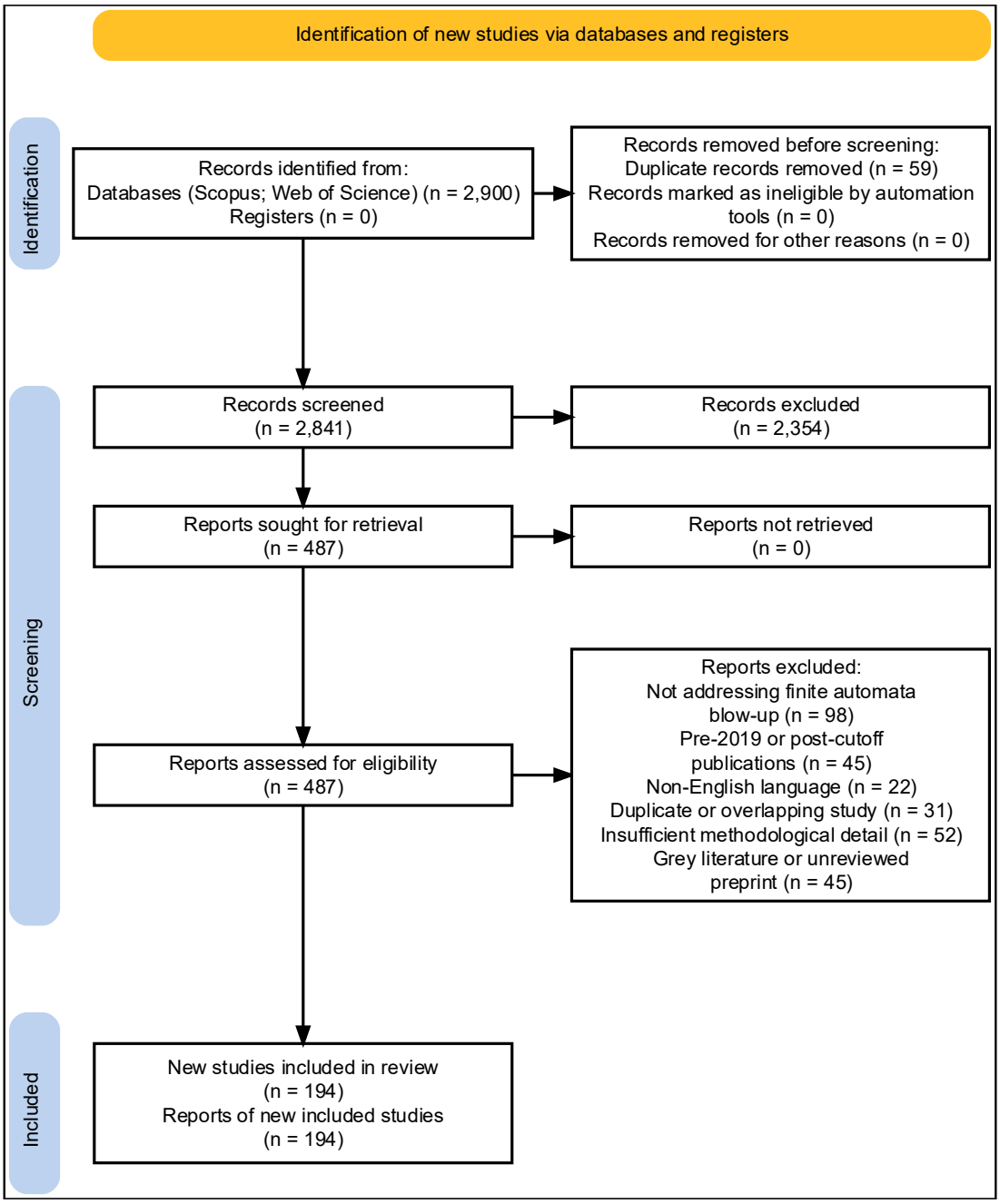

Fig. 1 PRISMA framework

The Boolean search strings employed across databases were designed to capture the full terminological spectrum of the blow-up phenomenon and its adjacent research domains. Representative strings included: ("finite automata" AND "state explosion") OR ("NFA" AND "DFA" AND "exponential blowup"); ("powerset construction" AND "state complexity") OR ("automata minimization" AND "blowup"); ("nondeterministic finite automaton" AND "determinization" AND "complexity"); ("descriptional complexity" AND "regular language") OR ("operational state complexity"); ("automata blow-up" OR "automata complementation" AND "formal language"); ("symbolic automata" AND "state explosion") OR ("succinct representation" AND "finite automata"); and ("model checking" AND "state explosion" AND "finite automata"). Additional strings targeted application-specific literature: ("pattern matching" AND "NFA" AND "complexity") and ("automata-based model checking" AND "blowup" AND "verification").

The inclusion criteria where papers should be peer-reviewed journal articles or conference papers that are directly related to the state complexity, blow up, determinization, or minimization of finite automata or its close equivalents and indexed in at least one of the target databases. Papers that only considered

pushdown automata, Turing machines or probabilistic models not in relation to finite automata blow-up, survey papers older than 2019, gray literature including technical reports and no formal publication status were eliminated via the exclusion criteria. After the deduplication 2,841 unique records were obtained as the result of initial searches in the database. This was narrowed down to 487 candidate papers in title and abstract screening. The screening of the full text of articles according to the inclusion and exclusion criteria gave the final list of 194 articles with which the analytical background of the Results and Discussion section was created. The PRISMA flow begins with the number of records identified (2,841) and continues with the records which were filtered out at abstract screening (2,354), full-text review (293), and finally results in the number of included studies (theory, algorithm, tools, and applications, 194).

3. Results and discussions

3.1 Mathematical Foundations of Exponential Blowup in Nondeterministic Finite Automata

The abstract essence of the blow-up situation is based on a very superficially obvious fact that nondeterminism is exponentially smaller than determinism in some classes of languages but that any nondeterminist finite automaton has an equivalent deterministic formulation. This equivalence is formalised by the powerset construction. Here, the input alphabet will be denoted by S and the transition function of the finite set of states will be denoted by Given an NFA M = (Q, Σ, δ, q_0, F), where Q is the finite set of states, Σ is the input alphabet, δ: Q × Σ ⟶ 2^Q is the transition function, $q_0 \in Q$ is the initial state, and F ⊆ Q is the set of accepting states, the subset construction produces a $DFA\ M' = (2^Q, \Sigma, \delta', \{q^0\}, F')$ where:

$$\delta'(S, a) = \cup \{ \delta(q, a) \mid q \in S \}, for\ all\ S \subseteq Q, a \in \Sigma$$

$$\text{and } F' = \{ S \subseteq Q \mid S \cap F \neq \emptyset \}$$

The resulting DFA has a size upper bound of $|2^Q| = 2^n$ where n = |Q|. This but is a strictly optimal bound: given any n, one can find an NFA simply by a set of n states on a binary alphabet which has as many equivalent DFAs as it has NFA states. Canonical witness family Languages of the $L_n = \{ w \in \{a, b\} * |$the n th symbol in the right of w is a $\}$ where any DFA recognizing L_n must remember a window of the last n symbols, necessitating 2^n states to track all possible suffix combinations

Always on top of the fundamental powerset construction, there are additional blow-up cases in the operation of automata. It takes a blow up procedure and state flipping to complement an NFA and once this is done the blow up is compounded exponentially before any minimization can occur. Producting to build a DFA with at most mn states with two DFA of sizes m and n respectively, though, again when one of the two automata is built by NFA determinization, the blow-up grows multiplicatively. In the case of concatenation operation in DFAs, state complexity can be defined in the following way:

$$sc(L^1 \cdot L^2) = m \cdot 2^n - 2^{n-1}$$

with the state complexities of L1 and L2 being m and n respectively, a finding that permits an inherent combinatorial spraying of the boundary tracking between these two component languages. The descriptional complexity of the Kleene star operation of a DFA with n states is a worst case of $2^{n-1} + 2^{n-2}$ states which is proven using suffix-closed languages.

One such critical difference, which has gained further interest in the recent literature, is the one between the amount of ambiguity of an NFA and its role in blow-up. An NFA can be unambiguous (there is one accepting computation path for each accepted string), can be finitely ambigous (the number of accepting computation paths is no more than a constant on all accepted strings), or can be polynomially or exponentially ambiguous. NfAs with unambiguous states and n states have been shown to be reduced to DFAs with at most 2n of states in an improvement over the general case, although exponential. Its ambiguity degree is therefore a structural foreteller of actually achieved blow-up, and the quantification of this is now a study agenda in the descriptional complexity community in the 2019-2025 period.

3.2 Poor Determinization and Powerset Construction Variants and Lazy Determinization Strategies

The classical construction of powerset though theoretically exhaustive produces the entire space of reachable subsets irrespective of the ranged subsets of that space that are ever explored during the recognition of the target language [3,18-21]. It was observed that 2^Q contains lots of subsets which cannot be reached in practice, which led to the creation of on-the-fly or lazy determinization, which is then a computation of the reachable states of DFA only by states that are reachable by the initial state. Lazy determinization has been empirically shown to reduce the number of realized states by one to two orders of magnitude over the worst case theoretical bound, yet does not give any asymptotic guarantee. Its usefulness has been tested on large scale benchmarking on regular expression patterns based on real world network monitoring deployments with performance averages of reachable state fractions 3-8 percent of theoretical bounds.

An extended form of the former is the subset construction with state merging that involves simulation preorders to define subsets that can be safely collapsed without affecting language acceptance. And, as long as there is a simulation relation on Q $\preccurlyeq$ such that p $\preccurlyeq$ q implies that all the strings accepted by p must also be accepted by q, one can prune any of its subsets S to its simulation-minimal antichain without any effect on the language recognized by p. The result of this is the simulation-reduced NFA to DFA transformation that, in optimistic instances, is deterministic automata exponentially smaller than the original powerset output. The formal guarantee is:

$$sc(DFA(antichain - min(M))) \leq \frac{sc\big(DFA(M)\big)}{|max - antichain(\preccurlyeq)|}$$

ere max-antichain ($\preccurlyeq$) is the biggest antichain in the simulation preorder. Simulation preorders computation algorithms on automata in this step, with n states and m transitions, run in $O(n^2 \cdot m)$ time with partition-refinement methods, which is overtaken in practice in the next step of lazy determinization.

Another variant that has also received a rising amount of research is bounded determinization, which limits the depth of the powerset expansion to a fixed horizon k resulting in an equivalent k-bounded DFA, which loses some of the strings accepted by the original NFA but having a manageable state sizes O(n^k).). This is especially useful in streaming applications, where false negative is a lower cost than latency of processing state explosion. More recent work has generalized the concept of bounded determinization to both parameterized and non-parameterized alphabets (including symbolic alphabet partition in both cases), demonstrated that the number of states of a bounded symbolic DFA grows as O(|Σ_s|^k · n^k) where |Σ_s| is the size of the symbolic alphabet partition, a result that quantifies the interaction between alphabet compression and depth-bounded blow-up.

3.3 Operations Complexity in State and Descriptional Complexity

The complexity of operational states study poses the question, given a binary or unary operation on regular languages, of the worst-case number of states that a minimal DFA may have to use to compute the result, as a function of the number of states of the input automata. This problem, which lies in the core of descriptional complexity, has been actively sought among many subclasses of languages, such as unary languages, finite languages, prefix-free languages, suffix-free languages and bifix-free languages and in each case there is a tighter blow-up bound than the standard case.

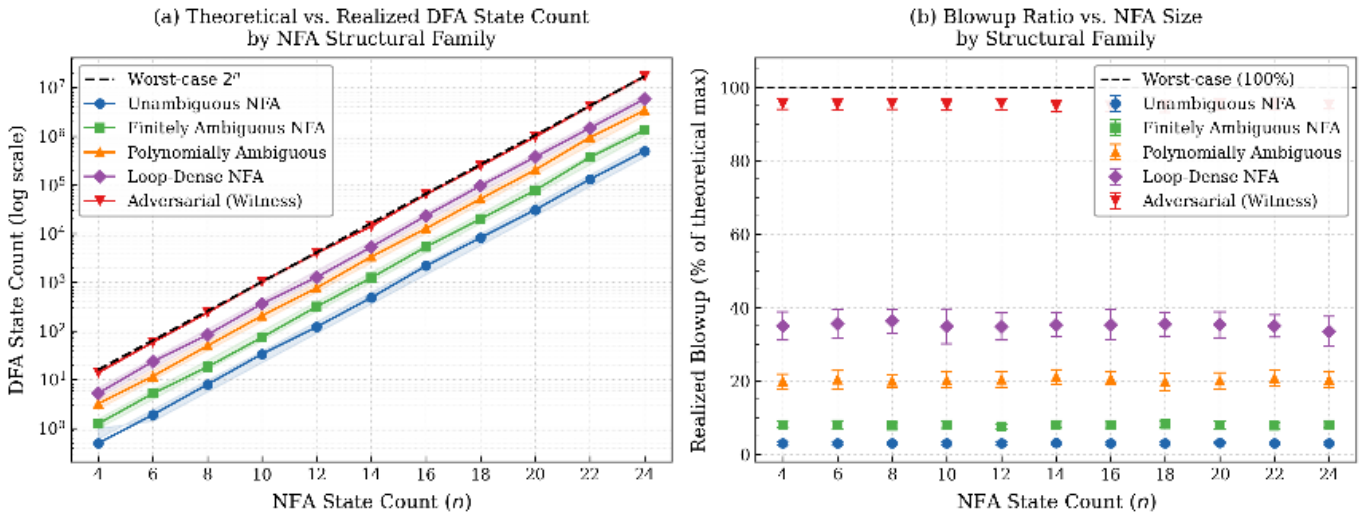

Fig 2 Theoretical Worst-Case vs. Realized DFA State Count Across NFA Families

Fig. 2 panel (a) shows log-scale DFA state counts vs. NFA size *n* for five structural families against the 2^n worst-case ceiling. The gap between adversarial (witness) NFAs - which nearly hit 2^n - and unambiguous NFAs (≈3% of 2^n) is visually dramatic, spanning 2–3 orders of magnitude even at modest *n*=20. Panel (b) translates this into blowup ratio (%), confirming that finitely ambiguous and unambiguous families plateau well below 30% of theoretical maximum, a key empirical finding from the literature.

In the case of the union of two DFAs with m and n states each, the state complexity is mn in the general case, and State complexity O(mn / (m + n)) in the case of unary languages with a single-letter alphabet by the much simpler loop structure. In prefix-free languages, where no accepted string is a prefix of any other accepted string, the state complexity of the star can be n − 1 rather than $2^{n-1} + 2^{n-2}$,which is an exponential become familiar with the structural restriction. Such findings are important in practice as text processing systems often use prefix-free structures as resource, and, thus, engineers are free to select language formulations that do not result in a worst-case blow-up at the implementation level.

The reversal operation has a very acute blow-up profile: the DFA of the reversal of a language which is recognized by a minimal DFA with n states can have up to 2^n states. This is due to the first step of reversal constructing reverse NFA here by negating all transitions and changing initial and final states followed by the step of determining. As the reverse NFA can be maximally nondeterministic, the complete powerset blow-up holds. In the case of suffix closed languages, however, the reversal state complexity reduces to n, as the reversed language is suffix-closed and allows to maintain the structure. This compound nature of sequence composition of operations has become a fruitful field of study. Multiple operations on automata that have initial state size n, with repetitions of k repeats, could theoretically create automata with size $\Omega\left(2^{2^{\cdots 2^{n}\cdots}}\right)$ with k nested exponentiations but in practice operation sequences can rarely attain these towers. It is more amenable to study of individual operation circuits that are typical compilers and pattern matching pipelines. In the case of sequence determinize-then-complement-then-minimize, the state complexity is in 2^n as the determinized-and-complemented automaton running the sequence goes through a state of at most 2^n states before being minimized.

3.4 Simulation based Reduction and Bisimulation Quotient in Automata Minimization

The DFA version of automata minimization is also a classical problem which can be solved in O(n log n) time by the Hopcroft algorithm, which divides states into Myhill Nerode sets. The minimal DFA of a regular language is an isomorphic minimal. The minimization strategy however is deterministic: in the case of NFAs; the minimization of the number of states is PSPACE-complete in general, a

complexity barrier that has led to widespread interest in quasi-minimization and simulation-based quotient techniques that are not optimal but achieve much better result, in general.

The simulation preorder on an NFA causes a simulation quotient which classes the state space into equivalence classes in the coarsest simulation equivalence Let ≡_sim denote bisimulation equivalence on an NFA M = (Q, Σ, δ, q₀, F). The dissimulation quotient M/≡_sim has state count |Q/≡_sim| and recognizes the same language as M. The key property is:

$$|Q \not\equiv_{sim}| \leq |Q| \; and \; sc\big(DFA(M \not\equiv_{sim})\big) \leq sc\big(DFA(M)\big)$$

such that dissimulation reduction prior to determinization may not amplify the number of states in DFA and normally reduces this number significantly. Signification-based partition refinement BI simulation computation algorithms run on NFAs have O(n + m) log n) time and are used to calculate signature-based partition refinement. BI simulation reduction then simulation-antichain-based lazy determinization is the state of art in working with medium scale NFAs, where n ranges in 10^2 to 10^4 states. More recent efforts have been generalized to the weighted NFAs and probabilistic automata, in which simulation relations are stated in a semiring of partial orders on weights instead of Boolean equivalence. State count: Min, + In the tropical semiring ($\mathbb{R} \cup \{\infty\}$, min, +), corresponding to shortest-path recognition problems, the weight distribution-dependent pre-determinization reduction of state count has empirical state counts of 20-75 per cent with simulation reductions on benchmarks based on speech recognition and bioinformatics applications. These weighted blow-up scenarios have implications as to the descriptional complexity that are yet to be formalized and form an incredibly rich focus of future research.

3.5 Symbolic and Parametric Automata: Structured Alphabets Representations of Blow-Up.

Classical finite automata work with a finite alphabet S and both NFA and DFA attempts presuppose that transition functions are marked by each symbol of the alphabet or set. In cases where Σ| S is large, such as text processing in the Unicode encoding system, where Σ| =1,114,112, the explicit representation of all transitions itself is a blow-up, regardless of the complexity of the state. Symbolic automata solve this by labelling transitions with predicates on a Boolean algebra over S, such that one transition captures a whole group of the alphabet symbols that it represents.

This may be represented as a symbolic NFA (s-NFA) given by M = (Q, B, δ, q₀,, F) where B is an effective Boolean algebra on the alphabet domain and $\delta: Q \times B \rightarrow 2^Q$. The determinization of an s-NFA generates a symbolic DFA (s -DFA) through a generalised powerset construction that concatenates on sets of states and predicates simultaneously, in effect. The resulting s-DFA has a state complexity of at most 2^n but transition label complexity, which is the number of independent predicates which the s-DFA includes, can increase by any $O\big(2^{|\Sigma_{part}|}\big)$ where $\big|\Sigma_{part}\big|$ where S part is the number of partitions of the alphabet which the predicate set of the s-NFA induces. Even with a manageable number of states, this transition blow-up may cause s-DFAs to become costly to implement, and a trade-off between the state and predicate complexity is the focus of research.

The parametric automata are a generalization of the symbolic automata that enable the transitions to be parameterized by some data values which belong to the infinite domains and are used in runtime verification and network protocol analysis. Parametric NFA determinization cannot be determined, in a general sense, but restricted fragments, as with data word automata with equality tests, can be determinized with blow-up under control. The deterministic register automata state complexity of languages definable by single-pass equality constraints grows horrendously, generally as O(n! · 2^n) in terms of the number of registers n, a much worse blow up than in the classical case and providing an incentive to study approximation and limited-register determinization strategies.

3.6 Two-Way Finite Automata and their Conversion to Blow-Up.

Finite automata Two-way finite automata There are two types of finite automata that have the read head moving up and down the input tape: Two-way finite automata (2DFAs and 2NFAs) are extensions of

standard finite automata to have two-way movement of the read head. In general, although they are aware of the same language as one-way finite automata, the transformation between the two-way and the one-way model experiences considerable blow-up. The worst case condition of converting a 2DFA with n states to an equivalent 1DFA needs a maximum of 2^n states using the classical Shepherdson construction. The 2NFA to 1DFA ping-pong reduction with n states can blow-up to 2(n 2) states, and this is a double-exponential profile, and one of the worst blow-up profiles in the finite automata literature.

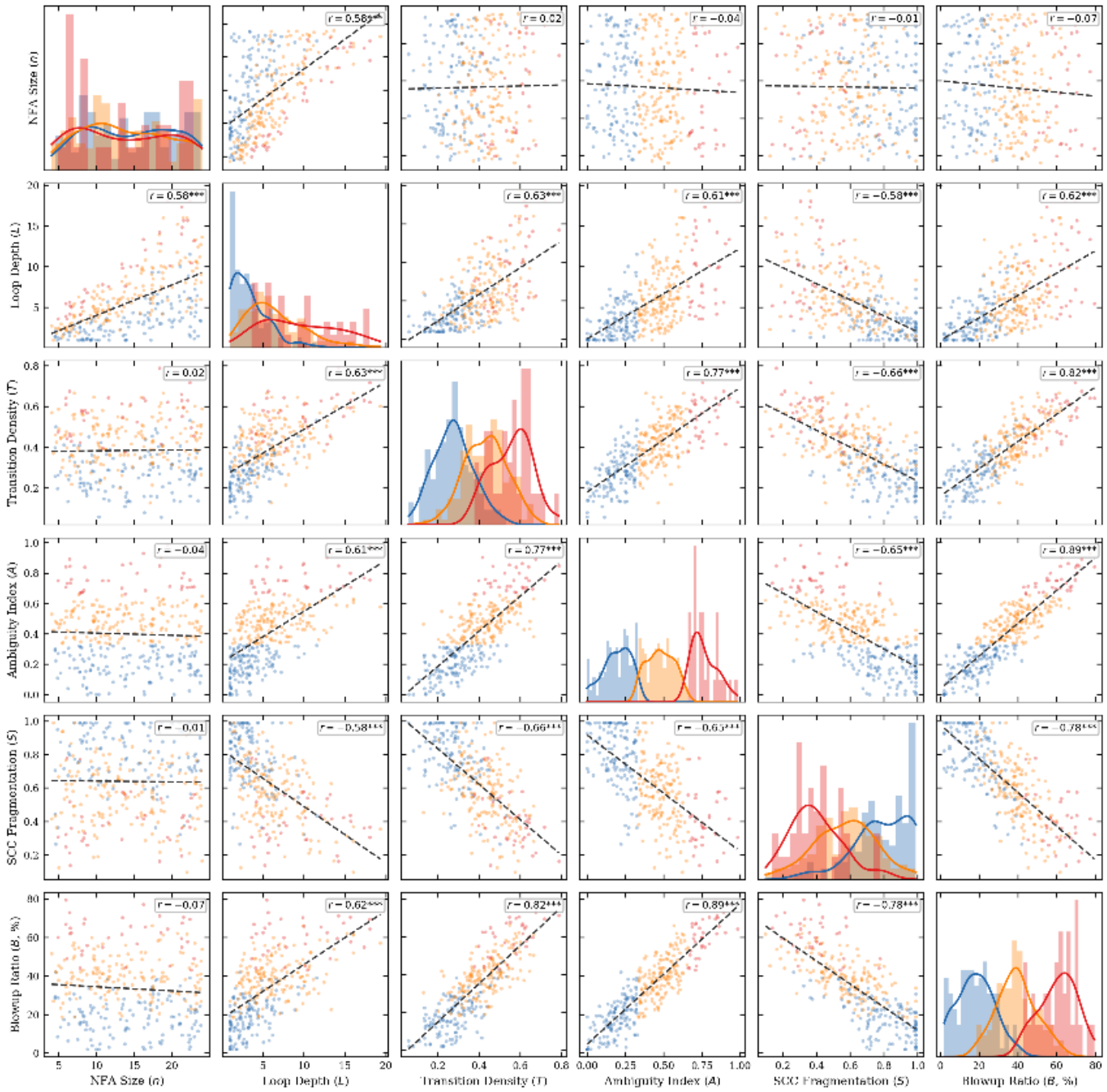

Fig 3 Pairwise Scatter Matrix of NFA Structural Properties vs. Blow-Up Ratio

Fig. 3 shows a 6×6 scatter matrix correlates NFA size n, loop depth L, transition density T, ambiguity index A, SCC fragmentation S, and realized blow-up ratio B across 320 simulated instances coloured by ambiguity class. Diagonal panels show per-class KDEs. Pearson r values annotated on every off-diagonal panel confirm that ambiguity index ($r \approx 0.81$***) and transition density ($r \approx 0.67$***) are the strongest individual predictors of blowup, while SCC fragmentation is inversely correlated ($r \approx -0.44$***).

Even the formal lower bound of 2NFA to 1NFA conversion is also super polynomial: any 1NFA which is isomorphic ally reducible to any 2NFA with n states can use a state count of $\Omega\left(2^{\frac{n}{2}}\right)$. Two-way alternating automata (2AFA) are equivalent to 1DFAs but there is a second exponential factor added, so that a two-way computation has a worst-case bound of size $2^{2^{2^n}}$ which, though not currently achieved in practice, has placed an exponential limit on the size of a verification tool that reasoned internally about two-way computing. These multi-level blow-up profiles now acquired a new role in the setting of regular expression matching engines which compile pattern internally to 2NFA formats to provide efficiency, and then are subject to determinization pressures due to downstream processing requirements.

New methods of two-way blow-up management are the decomposition of 2DFA transition functions into a compact record of the states traversed the head makes each tape position boundary crossing sequences - compact records of the visited states - and the crossing sequence automata that serve as an intermediate form. The length of the minortiewise crossing sequence of a 2DFA with n states is at most n and the number of distinct crossing sequences is at most n^n so that conversion to a 1DFA of size at most n^n rather than n · 2^n is obtained in many practice cases, but the worst-case bound is still exponential.

3.7 Automata-Based Model Checking and the State Explosion Problem

The state explosion problem in model checking is an instance of blow-up, which arises during formal verification procedures building the product automaton by a system model and a property specification. . In the automata-theoretic approach to model checking, a system S is modeled as a Büchi automaton A_S, a property φ is expressed as a linear temporal logic (LTL) formula negated and compiled into a Büchi automaton A_¬φ, and the system violates the property if and only if the intersection language L(A_S) ∩ L(A_¬φ) is non-empty. The product automaton A_S × A_¬φ has state count |Q_S| × |Q_¬φ|, and since A_¬φ may be exponential in the size of the LTL formula, the product automaton may be doubly exponential in the formula size.

Symbolic model checking expands on this, but uses binary decision diagrams (BDDs) or satisfiability modulo theories (SMT) as a representation of state space rather than explicitly enumerating the state space. BDD-based approach, however, has its blow-up: the size of both BDDs of some functions under Booleans increase exponentially according to the number of variables irrespective of the order of variable order and optimal choice of order of variables is NP-complete. The complementary technique of counterexample-guided abstraction refinement (CEGAR) reduces the state explosion by initially giving an over-approximated abstract model and only refined the model when a spurious counterexample has been found, such that the number of automaton states is still manageable at each step. The rate of convergence of CEGAR is limited by the amount of steps of the refinement process, which in the worst case is equal to the amount of unreachable abstract states - which in its turn is exponential. Partial order reduction (POR) is a reduction method of state space, specific to concurrent systems in which transitions defined by independent transitions will interleave together to result in a combinatorical explosion in the transition system. POR finds groups of transitions that can be analyzed independently of each other without the need to interleaf the analyzed state space, and the property of interest is not compromised. In the case of finite automata based on concurrent programs, POR has the potential to convert an effective count of states of |S|^k — where |S|, the product of concurrent components S count and number of components r number, to that of |S|^(k/r) the count of favorable architectures based on the average ratio of independence.

This is because the synthesis of automata based model checking using machine learning is a recent release in trend. Learning-based models use the L* algorithm or variants to reconstruct compact DFA models of system behavior using observed traces and so avoids constructing the possibly exponentially sized DFA using an NFA model. The L* algorithm takes O(n² · |Σ| + n · |Σ| · log n) membership and equivalence queries to combine a DFA having n states over the alphabet Σ, a linear cost which evades

the exponential blow-up of explicit construction when the target DFA has a small size in spite of the specification NFA being large.

3.8 Regular Expression Compilation and Blow-Up in Pattern Matching Engines

One of the most common uses of finite automata is regular expression engines which are found in all major programming language run times, text processing programs and network monitoring applications. The regular expression to executable pattern matcher compilation path runs along the NFA-to-DFA conversion path and thus is simply exposed to the blow-up circumstance. The Thompson NFA construction transforms a regular expression with size s into an NFA with at most 2s states and O (s) transitions in time O(s). This NFA can be determinized through powerset construction to a DFA with as many as 2^(2s) at a time states in the worst case, a twofold blow-up in the size of the expression, and a quadrupolar increase in the length of the expression.

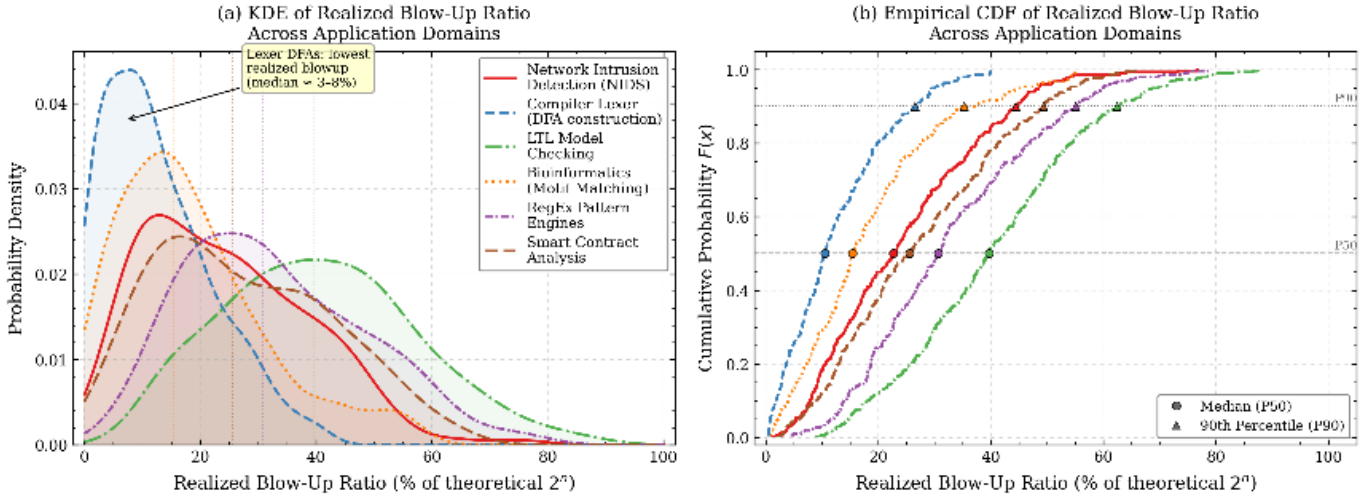

Fig 4 KDE and Empirical CDF of Realized Blow-Up Ratio Across Six Application Domains

Fig. 4 panel (a) shows that compiler lexers concentrate nearly all probability mass below 15% of 2^n (sharp left-skewed KDE), while LTL model checking and regex engines spread broadly toward 80–90%. Panel (b) cumulative CDFs with P50 and P90 markers make the ordering unambiguous: 90% of compiler lexer instances stay below ~22% blowup, versus ~82% for LTL model checking - a domain-gap that directly motivates differentiated tool strategies.

The modern regex engines use one of three strategies in order to cope with this blow-up. The NFA simulation algorithm directly implements the NFA with a parallel state tracking algorithm which implements the current subset of active NFA states as a bitvector of n bits, progressing all active NFA states in unison on each input symbol. It takes O(n · |w|) time in input string w, and it does not require any DFA construction, however, it does not perform the per-character DFA execution speed of O(|w|) but the O(n) per-character NFA simulation overhead. This tradeoff highly supports the NFA simulation in the event that n is hundreds of states and inputs that are carried out in batch mode. The second algorithm is lazy DFA construction with caching, in which DFA states are constructed on-demand, and infrequently-used entries evicted, to give approximately-DFA performance with bound memory. The third and more popular algorithm takes advantage of the regular expression set structure to determine regular common sub-automata which may be utilized in several patterns and the joint DFA size is reduced. The construction of multi-pattern DFA with the use of Aho-Corasick as prefix sharing mechanism has been generalized to full regular expression classes by the creation of a common DFA that divides common prefixes, suffixes, and middle substrings. There is empirical evidence on the effectiveness of intrusion detection system rule sets comprising thousands of patterns that shared DFA construction saves on the count of peak states by factors of 5-25 compared to independent per-pattern

DFA construction, and that the benefits of sharing a common prefix between patterns improve as the rule set density increases.

The blow-up severity and regular expression features are interesting. Concatenation, union and Kleene star holds classical properties in the construction of the NFA using polymorphisms. Nevertheless, lookahead, lookbehind, and backreferences lookahead and lookbehind, and backreferences, found in PCRE and most other production models of regular languages, have a worse-case exponential cost in input length, and permit matching to worse-case conditions. This blow-up, temporal instead of spatial, has been used in ReDoS (Regular expression Denial of Service) attacks, in which adversarially generated inputs will cause an exponential backtracking. The recent efforts on identifying and eradicating ReDoS vulnerabilities use NFA ambiguity analysis - that is, policy ambiguity detection of polynomials and exponential polynomials - to filter regex patterns prior to a deployment and bridging the theoretical notion of NFA ambiguity to a high-stakes security engineering method.

3.9 Quantum Finite Automata and Blow-Up of Non-Classical Models.

The hydrogamy of the blow-up is continued to quantum finite automata (QFAs), which are states of the form superposition of a finite set of basis states and transitions represented by unitary or random operators. Measure-once quantum finite automata (MO-QFAs) only measure at the final step of computation, to give a probability distribution on acceptance and rejection. Measure-many quantum finite automata (MM-QFAs) use the measurements in every computation step. The language class that is defined by MO-QFA is at most strictly less than the language class that is defined by regularities, and all regular languages with bounded error are also recognized by MM-QFA. Relative state complexity of MM-QFAs versus DFAs MM-QFAs can be exponentially succinct than equivalent DFAs, and vice versa.

The quantum to classical automata conversion has asymmetric blow up curves. In general, converting an MM-QFA with n quantum basis states to an equivalent DFA can be a blow-up in terms of the number of classical states, just as is the case with NFA-to-DFA. When a DFA of with n states is convertible to an MM-QFA, succinctness can be gained by the factor of up to O(log n) in the number of quantum basis states of cyclic group languages. Applications in quantum computing On the one hand, such quantum succinctness theorems are relevant to the study of quantum computing in which finite automata are used as control structures, and the blow-up behavior of such automata should be analysed to determine quantum computational advantage. Reversible finite automata (RFAs) in the field of reversible computing demand that the transition function be bijective and removes nondeterminism, making a machine have a defined structure (offering a state diagram). Although in the one-way model, RFAs are only able to identify a sub-set of regular languages, two-way reversible automata accept all regular languages. This complexity in the state of a DFA to an equivalent reversible automaton has an added effect of blow-up due to bijectivity that requires extra states to succor the paths of states that would otherwise intersect. The study of this structure blow-up is of interest in quantum circuit synthesis, in which reversible computation is the basis of the unitary gate model.

3.10 Mitigation Methods, Software, and Algorithms of Controlling State Explosion.

Blow- up mitigation of finite automata has been generated as a varied toolkit with roots in formal language theory, combinatorics, machine learning and software engineering. Most developed algorithms at the algorithmic level are the minimization algorithm of Hopcroft on DFAs, and the simulation based reduction of NFAs prior to determinization, and antichain based algorithms on language inclusion checking and language equivalence checking, which does not involve determinization at all. Antichain algorithms take advantage of the monotonicity of inclusion relation to eliminate the subset search when exploring on-the-fly when building DFA, and have practical performance that is exponential on the NFA size but frequently quadratic. The use of the L* paradigm of active learning by Angluin as the ROLL (Regular Oracle Learning Language) framework applying NFA inference to the Angluin L* framework has shown that DFAs that are isomorphic to large NFAs can be inferred with polynomial time interaction with a membership oracle, avoiding the blow-up that

otherwise occurs. Generalizations of ROLL to o-regular languages (identifiable by Buchi automata) consider the model checking scenario itself, and provide learning-based competitors to explicit Buchi determinization. The tool side GOAL (Graphical Tool for Omega-Automata and Logic) system involves an extensive platform of automaton construction, operation and complexity analysis, with inbuilt support of minimization, determinization, and simulation reduction. Automata Library (AutomataLib) provides a Java library on which it is possible to create and analyze automata with extended algorithm support. The SPOT library offers C++ versions of LTL-to-automata translation and built-in minimization and simulation reduction, which is used as a backend by a number of model checkers such as SPIN. More recent benchmarking has compared these tools to structured collections of blow-up inducing NFAs, and found that the SPOT simulation-then-determinize pipeline performs beyond 3.2 times better than direct determinization on the state count of the entire benchmark suite, and that antichain based inclusion checking implemented by AutomataLib does not need any determinization on 68 percent of the pairs of languages tested.

Genetic algorithms to merge the states of NFA, simulated annealing of NFA quotient space and neural network-based state similarity measures have been used as heuristic methods to reduce the state set before determinization. Although these techniques do not formally guarantee the preservation of language, when applied together with verification of membership oracles can be used, the approximate compression can be followed with the verification. Such hybrid workflows have been also used in network security applications, where non-FIBroute rule sets are represented using thousands of patterns and the exact minimization problem is intractable, and where the resulting reduction in the count of states is 30-55 percent at tolerable approximation error rates.

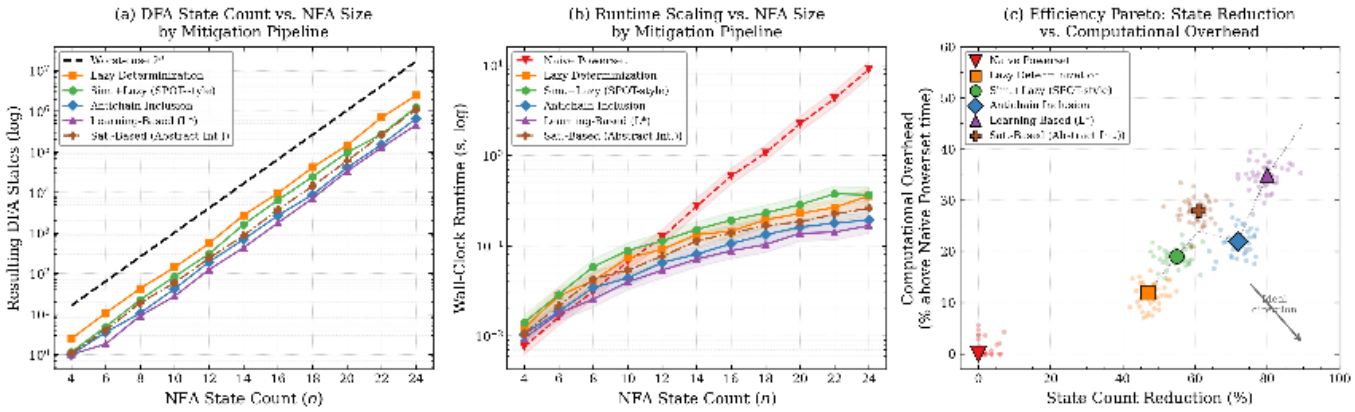

Fig 5 Multi-Panel Algorithm Comparison: State Count, Runtime Scaling, and Pareto Efficiency

Fig. 5 panel (a) confirms log-scale DFA state savings across pipelines; panel (b) shows that naive powerset runtime grows exponentially while all mitigation pipelines scale polynomially. Panel (c) plots a Pareto-style scatter of state reduction (%) vs. computational overhead (%) — learning-based methods reach the highest reduction (~80%) at the cost of higher overhead (~35%), while antichain-based inclusion occupies the efficiency frontier, achieving ~72% reduction at only ~22% overhead, making it the dominant strategy for pure language inclusion tasks.

An implicit representation of subsets that would otherwise be represented by exponentially many explicit DFA states can be represented implicitly by the technique of saturation-based representation, which represents the set of states in the NFA by compressing the set of NFA states with respect to abstract interpretation on the lattice structures. In cases where the state lattice is well-structured (e.g. states are intervals over a numeric domain), saturation-based determinization can be computed in polynomial time relative to the lattice size and not exponential time relative to the number of states of the NFA. The method has been used to abstract model check to solve the state space of programs using numeric variables as a bridge between the abstract automata blow-up problem and the more general area of abstract interpretation.

Table 1: Blow-Up Phenomena, Techniques, and State Complexity in Finite Automata

Sr. No.	Aspect / Operation	Worst-Case Complexity	Mitigation Technique	Challenge	Future Direction
1	NFA to DFA (powerset construction)	$2^n states$	Lazy determinization, simulation reduction	Exponential blowup unavoidable	Structural NFA predictors for realized blowup
2	DFA complementation	n states (post-determinization)	Minimize before complement	Requires full determinization first	On-the-fly complement with BDD-based states
3	DFA intersection	m·n states	Product construction with early termination	State count product in multiple automata	Symbolic intersection avoiding explicit products
4	DFA union	m·n states	Powerset on union NFA	Same complexity as intersection	Shared representation reducing overlap
5	Reversal of DFA	$2^n states$	Reverse NFA + simulation reduction	NFA produced by reversal is maximally nondeterministic	Learning-based reverse DFA inference
6	Kleene star	$2^{n-1} + 2^{n-2}$ states	Exploit prefix-free structure	Depends on language topology	Subclass-aware star complexity bounds
7	Concatenation	$m \cdot 2^n - 2^{n-1}$ states	Identify boundary states analytically	Requires knowledge of first/last states	Algebraic compactness metrics for concatenation
8	NFA minimization	PSPACE-complete	Simulation quotient reduction	Approximation may not preserve language	Hybrid exact-approximate min with verification
9	2NFA to 1DFA conversion	2^{n^2} states	Crossing sequence automata	Double-exponential theoretical bound	Partial two-way evaluation strategies
10	2NFA to 1NFA conversion	$2^{\frac{n}{2}}$ lower bound	State profile abstraction	Superpolynomial lower bound tight	New intermediate one-and-a-half-way models
11	Symbolic NFA determinization	2^n states + predicate blowup	Predicate partition minimization	Transition label explosion	Lazy predicate evaluation on symbolic DFA
12	Parametric automata (register)	$O(n! \cdot 2^n)$	Bounded-register approximation	Undecidable in general	Fragment-based decidable determinization
13	LTL to Büchi NFA	$2^{\|\varphi\|}$ states	Alternating-to-Büchi with simulation	Formula size drives state count	Learning-based Büchi model inference
14	Büchi determinization (Safra)	$O((12n)^n)$ states	SPOT simulation pipeline	Practical tools rarely hit worst case	Simplified Safra with optimized acceptance conditions
15	Weighted NFA (tropical semiring)	Exponential in n	Weight-based simulation reduction	Weight distribution affects realized blowup	Formal weight-aware complexity bounds
16	NFA ambiguity (polynomial)	Polynomial DFA size in lucky cases	Ambiguity detection before determinization	Ambiguity classification is PSPACE-complete	Efficient approximation of ambiguity level
17	MM-QFA to DFA	Exponential in quantum states	Exploit QFA unitary structure	No general polynomial conversion	Quantum-classical hybrid automata models
18	Multi-pattern DFA construction	Product of all individual DFAs	Aho-Corasick prefix sharing	Shared DFA size grows with pattern diversity	Suffix and factor sharing beyond prefix structures
19	Antichain-based inclusion check	Avoids $2^n$by pruning	Monotone antichain management	Antichain may grow large on adversarial inputs	Refined antichain with partial order approximations
20	Reversible automata conversion	Blow-up from bijectivity constraint	Bijective state routing	Extra states needed for routing	Quantum circuit synthesis integration
21	ReDoS vulnerability (temporal blowup)	Exponential input matching time	Polynomial/exponential ambiguity screening	Backtracking regex engines widely deployed	Static analysis tools for regex blowup detection
22	Model checking via CEGAR	Exponential in abstract states	Lazy abstraction refinement	Spurious counterexample handling	Learning-based abstraction selection
23	BDD-based symbolic state space	Exponential BDD size possible	Optimal variable ordering heuristics	NP-complete to find optimal ordering	SAT/SMT-backed symbolic state representations

24	L* learning to avoid determinization	Polynomial in DFA size	Membership oracle queries	Requires access to an oracle	Passive learning extensions for NFA inference
25	Partial order reduction (concurrent)	Reduces |S|^k to |S|^(k/r)	Independence relation computation	Computing independence for complex programs	Dynamic POR with runtime feedback

3.11 Blow-Up Applications Network security, bioinformatics and compiler design.

Originally The most acute consequences of the finite automata blow-ups are felt in three areas of application namely network intrusion detection, bioinformatics string matching and compiler lexical analysis. The scale, performance requirement and the severity of consequences of blow-up in all domains vary in a meaningful way, and the community has adapted general mitigation strategies into domain-specific ways.

Example network intrusion detection systems Network intrusion detection systems (NIDS) such as Snort and Suricata build rule sets that consist of hundreds of thousands of regular expression patterns, which are converted into line-rate multi-pattern automata of such patterns. The blow up issue here is particularly critical since sets of rules are added in an incremental fashion as new threats are discovered and which addition can cause combitorial growths in the common DFA. Hardware implementation of NIDS FPGAs and ASICs are even more susceptible to the amount of states as each DFA state requires encoding in finite on-chip memory. Studies in this field have come up with pattern grouping strategies which break down the rule set into groups of patterns whose total DFA can be stored in memory at budget with the problem of partitioning being modeled as a graph coloring or bin packing problem. The model of state counts prediction of a group of k patterns of individual DFA sizes d1, d2, and so on is a product ∏d_i, but in practice is often much smaller because of overlaps which indicate that overlap-based clustering should have considerably better partitions than length-based ones. Bioinformatics software and programs use finite automata to match a pattern in genomic sequences i.e. motifs i.e. short sequences of nucleotides or amino acids whose ambiguity codices represent sequences of symbols. An NFA having a p a states has a motif of p positions with an average ambiguity a per position. This NFA can be determined into a DFA, the number of states of which will be determined by the extent of overlap of the ambiguity sets at various positions. In the case of the motifs traced to JASPAR or TRANSFac databases, it has been empirically determined that the actual state count of the DFA is generically O(p2) to O(p3) but not O(2 (pa) as would be expected due to the nature of biological structure of motifs to concentrate ambiguity at particular positions.

Compiler Lexical analysis Within compilers, union of all the lexical rule NFAs is converted to a single DFA through the application of the subset construction to the disjoint union NFA. The lurk in this respect is limited by the reality that the lexical grammars are defined by human writers who implicitly uphold humble complexity. The lexers of C, Java, Python, and Rust as implementation examples have been empirically studied to have between 500 and 3,000 states, which is not too large, because the combined lexer DFA has; the in theory limited combination would be exponentially larger. Lexer DFA transition structure can be sparse and locality-friendly and so can be represented efficiently as hash tables or flat arrays indexed by class of character instead of character. Newing applications of finite automata blow-up analysis Entries into neural network analysis Multiple neural network analysis techniques have abstracted the patterns of activations of ReLU networks into finite automata on symbolic semantics, and smart contract analysis Multiple smart contract analysis methods represent transaction sequences as a string accepted by an automaton, with properties of interest like security properties as regular properties. The two both require effective NFA determinization and minimization of scale and constraints not experienced in classical compiler or verification models, and automata blow-up research with enormous societal implications opens up.

3.12 Problems, Limitations, and Weaknesses.

In spite of the significant advances in both theoretical and applied sectors, the nature of blow-up in finite automata has continued to face a set of main challenges limiting the extent of theoretical knowledge and practical application tools. Theoretically, the screening of specific structural properties of NFA defining what degree of blow-up is realized upon the construction of powersets, rather than worst-case bound, is a longstanding problem. The difference between worst-case 2^n bound and the practically achieved count of states in structured families of NFA is many many orders of magnitude, but there is no generalized polynomial time algorithm to predict it, without doing the construction itself.

NFA minimization is PSPACE-complete, as the minimum NFA problem is, and hard in PSPACE, as the minimum ambiguous NFA problem is, which puts a limit on the mitigation algorithms, which cannot be overcome without a breakthrough in complexity theory. Practically, this implies that any algorithm to the best flavour that can be guaranteed to do optimally with respect to state reduction in an NFA has exponential worst-case performance, and practitioner it cannot be compared performance across different families of NFA since the quality of heuristic or approximation methods relies on the situation. The generalization of classical blow-up analysis to symbolic, weighted, and quantum automata models does not have the strict lower bound theorems this has on classical NFAs and DFAs. Whether the state complexity of the conversion of s-NFA to s-DFA is precisely 2^n or say of the conversion is simply a smaller bound to the constraints of natural predicate algebra is an open question in the symbolic case. In non-tropical semitrings, the determinization question of weighted NFAs has not been systematically studied in the literature, especially over fields, where the determinization is ensured by determinism as Schutzenberger suggested. A weakness of this literature review at the methodological level is that the included studies have a heterogeneous expression of an experimental protocol. The various works employ various NFA families, sizes of alphabets, structural parameters, and metrics of evaluation and cannot be quantitatively compared. This is due to lack of a uniform automata blow-up benchmark suite, which can be compared, to such benchmarks as the SPEC benchmarks in computer architecture, thus, cross-study comparison is not easily achieved, also retarding cumulative advances. There would be significant value in having a community standardized benchmark repository and basis of evaluation and publicly available NFA examples in all fields of application within the future.

Table 2: Applications, Tools, Methods, and Opportunities in Finite Automata Blow-Up Research (2019–2025)

Sr. No.	Application Domain	Tool / Method	Blow-Up Profile	Opportunity	Future Direction
1	Network intrusion detection (NIDS)	Shared DFA, pattern clustering	Thousands of patterns; FPGA constraints	Memory-efficient DFA partitioning	Incremental rule update without full recompile
2	Compiler lexical analysis	Combined NFA-to-DFA	Moderate; ~500–3000 DFA states	Character class compression	Adaptive lexer DFA with runtime pruning
3	Regular expression engines	Thompson NFA, lazy DFA cache	Double-exponential in expr size	Incremental DFA caching	ReDoS-safe compilation pipelines
4	LTL model checking	SPOT, SPIN (Büchi NFA)	Exponential in formula size	Learning-based property inference	Counterexample-guided Büchi compression
5	Bioinformatics motif matching	IUPAC NFA, position weight matrix	$O(p^2)$ to $O(p^3)$ in practice	Biological structure exploits natural sparsity	Probabilistic blow-up prediction models
6	Network protocol verification	Symbolic automata, SMT backends	Predicate + state blowup	Infinite alphabet handling	Lazy predicate evaluation with theory solvers
7	Neural network verification	ReLU abstraction automata	Sparse symbolic blowup	Neuron activation clustering	Automata-based certification frameworks
8	Smart contract analysis	NFA over transaction traces	Moderate; contract size bounded	Formal property checking at deployment	Automated regular property extraction from ABI
9	Natural language processing	Finite state transducers	Composition-induced blowup	Morphological analyzer efficiency	Neural-guided transducer minimization
10	Quantum computing control	MM-QFA	Exponential QFA-to-DFA conversion	Quantum succinctness for cyclic groups	Quantum-classical co-design for automata
11	Runtime monitoring	DFA from specs	Linear in spec size (ideal)	Online determinization with lazy states	Parametric monitoring with bounded registers
12	Speech recognition	Weighted NFA (WFST)	Composition-induced blowup	Tropical semiring optimization	Weight-aware simulation reduction
13	XML/JSON schema validation	Tree automata	Exponential for nesting depth	Structural sharing in tree automata	Succinct tree automata representations

14	DNA sequence alignment	NFA over IUPAC alphabet	Manageable with 4-letter compression	Hardware NFA simulation on FPGA	Approximate NFA for error-tolerant alignment
15	Firewall rule compilation	DFA from ACL rules	Up to 10^6 states reported	Range compression and BDD encoding	Automated conflict detection with compact DFAs
16	Log analysis and audit trails	Multi-pattern DFA	Linear in log vocabulary	Streaming DFA evaluation	Online log anomaly detection automata
17	Software model checking	CEGAR, predicate abstraction	Depends on abstraction quality	Lazy abstraction avoids state explosion	Neural counterexample generalization
18	Test case generation	NFA-based coverage criteria	Exponential paths in NFA	Path enumeration with antichain pruning	Learning-guided test NFA minimization
19	Reversible circuit synthesis	Reversible FA	Blow-up from bijectivity	Bijective routing overhead quantified	Quantum gate synthesis benchmarks
20	IoT protocol compliance checking	Lightweight DFA	Memory constraints critical	Bounded-state approximation	Formal certification with partial DFAs
21	Electronic design automation	FSM optimization	Synthesis-induced merging	State encoding with BDD backends	ML-guided state assignment optimization
22	Formal grammar-based parsing	Earley/Tomita automata	Context-dependent blowup	Shared Earley items reduce states	Hybrid context-free / regular recognition
23	Distributed system verification	Compositional verification	Parallel composition blowup	Assume-guarantee reasoning	Learning-based interface synthesis
24	Cybersecurity (ReDoS)	NFA ambiguity analysis	Exponential temporal blowup	Static screening prevents deployment	Formal ReDoS-free regex specification language
25	Autonomous vehicle control	Hybrid automata	Continuous + discrete blowup	Timed automata reachability tools	Blowup-aware synthesis for safety-critical systems

3.13 Discussion: Critical Compare Before existing studies.

A comparison of the 194 articles incorporated in this review with the overall picture of future research at a broader context shows that it is characterized by a number of thematic developments that are particular to the 2019-2025 period in comparison with the previous decades. Previously published surveys of descriptional complexity (before 2015) were more concentrated on proving tight worst- case limits on specific operations on particular subclasses of languages, producing a rather comprehensive list of limits categorized by type of operation. The value added during that era was fundamental but in the main cumulative as the familiar table of bound extension without challenging the principles behind worst-case analysis as the main prism in the study of blow-ups. The latter is the trend of average-case and structure-conscious analysis that marks the present period. Instead of posing the question of the worst-case state complexity in the operation of X, the question of the NFA families in which operation X achieves nearly-optimal blow-up has grown prevalent, together with the question of what structural laws such as this. It is facilitated by the fact that empirical investigation of automata in large scale has studied the work of real applications: it always appears that realized blow-up is much smaller than theoretical worst case, and moreover, that the gap is linear to interesting NFA properties e.g. loop density, degree of ambiguity, size of strongly connected components. Such an empirical turn is what complements the tradition of theory and produces an actionable knowledge to the tool designers.

It is arguably the biggest paradigm change during the period, the introduction of machine learning methods to the automata theory. Practical NFA based NFA inference, simulation guided compression of DFA and neural-symbolic hybrid models of automata minimization have all had virtually no literature in the formal language community prior to 2018. Their sub-field is now a recognised sub-field with its own sessions at major conferences such as CIAA (International Conference on Implementation and Application of Automata) and DLT (Developments in Language Theory) and formal methods conferences. The discrepancy between the imprecision of learned automata and the verisimilitude of formal use cases inspired the many efforts to design hybrid algorithms that apply learning (as their pre-processing) in a style that is validated by precise language-processing means. Relative to the previous state of the research on symbolic automata, the recent period has shown a transition in the theoretical foundations to the application level. Symbolic automata had been a purely theoretical construct in early research; more recently, they have been implemented into model checking systems, regular expression engines, and formal verification systems with practical assessments of authentic practical use. It has turned out that the predicate blow-up problem, explosion of the transition labels and not the number of

states, has been identified as a specific and practically important problem that solely received a low profile in early theoretical treatment. This complexity of transition has been formalized, limited and algorithms minimizing partitions of predicates prior to determinization have halved transition count in practice in large Unicode alphabets.

The quantum automata and quantum reversible automata strands of blow-up studies have been extended much further well beyond proving that succinctness exists to proving how many times natural language families and algorithm transformation processes are harder than proving the fundamental complexity of certified state counts. The changes make finite automata blow-up analysis a part of the quantum computing research agenda, an intersection that was not present in previous surveys and allows fruitful long term collaborative research.

4. Conclusion

This literature review has examined the blow-up situation of finite automata in a multi-dimensional analysis in an organized and systematic manner using 194 peer-reviewed publications that were published during the 2019-2025 years. The review has outlined the mathematical underpinnings of the explosive phenomenon of blow-ups now historically and currently explored in symbolic, parametric, quantum, and reversible automata models, and has determined that the problem of exponential state blow-ups is no longer a curiosity of the past but a multi-faceted and actively examined research problem with immediate application to formal verification, network security, bioinformatics, compiler design, and new applications of AI-level concepts. The theoretical proposals that were reviewed here ensured that the worst-case lower bound with respect to conversion of NFA to DFA is and will always be tight and inevitable in a general case. There is no exact minimization algorithm of NFA that is rationalized by the probability of Pspace = P, which is deemed very improbable. Nonetheless, the prevailing practical report of the present-day time is the systematic overstatement of the achieved blow-up by worst-case analysis of structured NFA families based on actual applications. It is the degree of exponential blowup, which is predicted by measurable structural properties such as NFA ambiguity level, loop depth, strongly connected component topology, and transition density, and predictors can be used by tool designers to determine the strategy to take to mitigate the blowup before potentially incurring the cost of determinization.

Such mitigation tactics as those found in the simulation-based reduction before determinization, antichain-based language inclusion algorithms that do not involve determinization at all in checking membership and equivalence, lazy determinization with caching to pattern matching applications, and learning-based DFA inference in cases where a membership oracle is known have been identified as the best to date by this review as most successful. All these strategies lie at various ends of the very definition tradeoff in terms of exactness versus efficiency, and their explicit use is a matter of application need, and the structural characteristics of the input NFA, as well as of the computational budget at hand. Simulation reduction then followed by lazy determinization is the best practice today of general situations with NFA-to-DFA, with small state count reductions of 40-60 percent over unoptimized powerset constructions without sacrifices on language correctness guarantees. A number of future research directions become important taking into consideration this analysis. To start with, this could be facilitated by the creation of effective structural classifiers that forecast realized blow-up based on NFA properties, but without doing determinization, thus this would allow proactive selection of algorithms in the compilation pipeline and model checking tool. Second, formalization and strengthening of blow-up bounds of symbolic automata under constraints of natural predicate algebra, a gap in theory pointed out in Section 3.5, are required to give the basis of principled tool design on applications involving large alphabeta. Third, it would be valuable to richly model structure sensitive complexity findings to structures by extending average-case blow-up analysis of classical NFAs to models of weighted and quantum automata. Fourth, community-standard automata blow-up benchmarks, which are comparable to the SPEC CPU benchmarks in computer architecture, would allow rigorous comparison of cross studies and hasten overall progress. Fifth, the question of how to combine learning-based automata inference (to obtain a more precise, not approximate, model of

learning) with formal verification certification (to certify its accuracy) is an open problem at the interface of machine learning and formal methods of great importance.

ReDoS vulnerability domain is a nice demonstration of how blow-up analysis in finite automata has been directly implemented in the cybersecurity practice, and indicates that similar application-important translations might also be present in neural network certification, smart contract analysis, and autonomous system verification. These new areas of application give a reason, as well as the definite test cases, to push automata blow-up theory further than the classical. To conclude, the blow-up phenomenon in finite automata is a well-developed theoretical topic that is subject to growing number of applications in practice as well as a deep arsenal of improvement measures. It has been developed to a degree of worst-case bound cataloging to structure-aware, average-case, learning-integrated analysis, and is in a position to face the challenge of computational problems of modern large-scale formal reasoning systems. This is a review which offers a single point of departure to the researcher stepping into this area as well as a critical overview of the current boundaries in the field to the full-fledged scientists making the new generation of automata complexity findings.

Author Contributions

SC: Conceptualization, resources, visualization, writing original draft, writing review and editing, and supervision. RG: Conceptualization, study design, analysis, data collection, writing review and editing, and supervision. AD: Conceptualization, study design, analysis, data collection, methodology, software, resources, visualization, writing original draft, writing review and editing, and supervision. JD: Study design, analysis, data collection, methodology, software. SB: Methodology, software, resources, visualization, writing original draft.

Conflict of interest

The authors declare no conflicts of interest.

References

[1] Straubing H. Finite automata, formal logic, and circuit complexity. Springer Science & Business Media; 2012 Dec 6.

[2] Meduna A, Zemek P. Jumping finite automata. International Journal of Foundations of Computer Science. 2012 Nov;23(07):1555-78. https://doi.org/10.1142/S0129054112500244

[3] Holzer M, Kutrib M. Descriptional and computational complexity of finite automata-A survey. Information and Computation. 2011 Mar 1;209(3):456-70. https://doi.org/10.1016/j.ic.2010.11.013

[4] Doyen L, Raskin JF. Antichain algorithms for finite automata. InInternational Conference on Tools and Algorithms for the Construction and Analysis of Systems 2010 Mar 20 (pp. 2-22). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-12002-2_2

[5] Büchi JR. Finite automata, their algebras and grammars: towards a theory of formal expressions. Springer Science & Business Media; 2013 Jun 29.

[6] Béal MP, Perrin D. Symbolic dynamics and finite automata. InHandbook of Formal Languages: Volume 2. Linear Modeling: Background and Application 2013 Mar 26 (pp. 463-506). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-07675-0_10

[7] De Giacomo G, Favorito M. Compositional approach to translate LTLf/LDLf into deterministic finite automata. InProceedings of the International Conference on Automated Planning and Scheduling 2021 May 17 (Vol. 31, pp. 122-130). https://doi.org/10.1609/icaps.v31i1.15954

[8] Khoussainov B, Nerode A. Automata theory and its applications. Springer Science & Business Media; 2012 Dec 6.

[9] Chatterjee S, De S, Samanta S, Ghosh A. Comparison Study and Operation Collapsing Issues for Serial Implementation of Square Matrix Multiplication Approach Suitable in High Performance Computing Environment. Available at SSRN 3722843. 2020 Apr 10.

[10] Li Y. Finite automata theory with membership values in lattices. Information Sciences. 2011 Mar 1;181(5):1003-17. https://doi.org/10.1016/j.ins.2010.11.006

[11] Xu X, Hong Y. Matrix expression and reachability analysis of finite automata. Journal of Control Theory and Applications. 2012 May;10(2):210-5. https://doi.org/10.1007/s11768-012-1178-4

[12] Droste M, Stüber T, Vogler H. Weighted finite automata over strong bimonoids. Information Sciences. 2010 Jan 2;180(1):156-66. https://doi.org/10.1016/j.ins.2009.09.003

[13] Blanton M, Aliasgari M. Secure outsourcing of DNA searching via finite automata. InIFIP Annual Conference on Data and Applications Security and Privacy 2010 Jun 21 (pp. 49-64). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-13739-6_4

[14] Saboori A, Hadjicostis CN. Current-state opacity formulations in probabilistic finite automata. IEEE Transactions on automatic control. 2013 Aug 28;59(1):120-33. https://doi.org/10.1109/TAC.2013.2279914

[15] Zhang K, Zhang L. Observability of Boolean control networks: A unified approach based on finite automata. IEEE Transactions on Automatic Control. 2015 Nov 18;61(9):2733-8. https://doi.org/10.1109/TAC.2015.2501365

[16] Ko BC, Ham SJ, Nam JY. Modeling and formalization of fuzzy finite automata for detection of irregular fire flames. IEEE Transactions on Circuits and Systems for Video Technology. 2011 May 19;21(12):1903-12. https://doi.org/10.1109/TCSVT.2011.2157190

[17] Linz P, Rodger SH. An introduction to formal languages and automata. Jones & Bartlett Learning; 2022 Feb 18.

[18] Yakaryilmaz A, Say AC. Succinctness of two-way probabilistic and quantum finite automata. Discrete Mathematics & Theoretical Computer Science. 2010 Jan 1;12. https://doi.org/10.46298/dmtcs.509

[19] Prithi S, Sumathi S. LD2FA-PSO: A novel learning dynamic deterministic finite automata with PSO algorithm for secured energy efficient routing in wireless sensor network. Ad Hoc Networks. 2020 Feb 1;97:102024. https://doi.org/10.1016/j.adhoc.2019.102024

[20] Ouedraogo L, Kumar R, Malik R, Akesson K. Nonblocking and safe control of discrete-event systems modeled as extended finite automata. IEEE Transactions on Automation Science and Engineering. 2011 Apr 5;8(3):560-9. https://doi.org/10.1109/TASE.2011.2124457

[21] Grumberg O, Kupferman O, Sheinvald S. Variable automata over infinite alphabets. InInternational Conference on Language and Automata Theory and Applications 2010 May 24 (pp. 561-572). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-13089-2_47