International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 12 Issue: 12 | Dec 2025 www.irjet.net p-ISSN: 2395-0072

A Structured Approach to Büchi Automata Construction from ω-Regular Expressions

Shruti Mishra[1], Rahul Gupta[2]

[1] *M.Tech. (CSE) Scholar, Department of Computer Science and Engineering, S. R. Institute of Management and Technology Lucknow, Uttar Pradesh, India*
[2] *Assistant Professor, Department of Computer Science and Engineering, S. R. Institute of Management and Technology Lucknow, Uttar Pradesh, India*

Abstract - *ω-regular languages play a central role in the formal specification and verification of non-terminating systems such as operating systems, communication protocols, and reactive controllers. Büchi automata are the most widely used automaton model for recognizing ω-regular languages and form the backbone of automata-theoretic model checking. While Büchi automata can be constructed from various specification formalisms, ω-regular expressions offer a compact and algebraic representation of infinite-word behaviors. However, direct construction of Büchi automata from ω-regular expressions often leads to complex transition structures and significant state explosion. This paper presents a structured approach to Büchi automata construction from ω-regular expressions, emphasizing syntactic decomposition, semantic preservation, and systematic automaton synthesis. The proposed method decomposes ω-regular expressions into finite and infinite components, constructs intermediate automata, and integrates them using well-defined acceptance conditions. The approach improves clarity, modularity, and extensibility while maintaining correctness. Formal definitions, construction algorithms, illustrative examples, and complexity analysis are provided. The paper contributes a rigorous framework that can serve as a foundation for optimization, symbolic implementation, and advanced verification tools.*

Key Words: ω-regular expressions, Büchi automata, infinite words, automata theory, formal verification, model checking etc.

1. INTRODUCTION

Modern computing systems increasingly rely on components that exhibit non-terminating behavior. Examples include operating systems, network protocols, embedded controllers, and cyber-physical systems. Such systems are often reactive, continuously interacting with their environment rather than halting after producing an output. Formal verification of these systems requires models capable of expressing infinite executions. ω-regular languages provide a mathematically robust framework for describing sets of infinite words and have become fundamental in theoretical computer science and formal methods.

Büchi automata are a canonical automaton model for ω-regular languages and are widely used in automata-based model checking, particularly in linear temporal logic (LTL) verification. The automata-theoretic approach to verification typically involves translating a logical specification into a Büchi automaton and then checking language emptiness or inclusion against a system model. As a result, the efficiency and structure of Büchi automata construction directly impact the scalability and practicality of verification tools.

ω-regular expressions constitute an algebraic formalism for specifying ω-regular languages using concatenation, union, Kleene star, and the ω-iteration operator. They provide an expressive and concise way to specify infinite behaviors, often more intuitive than temporal logic for certain classes of specifications. However, transforming ω-regular expressions into Büchi automata is non-trivial due to the inherent infiniteness of the accepted words and the need to enforce Büchi acceptance conditions.

Existing construction techniques often rely on ad hoc transformations or direct translations that obscure the semantic structure of the original expression. These approaches may lead to automata with unnecessary states and transitions, reducing interpretability and increasing verification cost. There is therefore a strong motivation to develop a structured and systematic approach to Büchi automata construction from ω-regular expressions.

This paper proposes such a structured approach. By decomposing ω-regular expressions into finite and infinite components and constructing intermediate automata that reflect this structure, the proposed method ensures semantic clarity and correctness while laying a foundation for future optimizations.

2. Preliminaries

This section introduces the fundamental concepts and definitions required for the remainder of the paper.

2.1 Infinite Words and ω-Languages

Let Σ be a finite alphabet. An infinite word over Σ is an infinite sequence $w=a_0a_1a_2\ldots w = a_0\, a_1\, a_2$, where each $a_i \in \Sigma$.

The set of all infinite words over Σ is denoted by Σ^ω. An ω-language is a subset of Σ^ω.

ω-languages are used to describe infinite behaviors, such as recurring events or ongoing interactions. Unlike regular languages over finite words, ω-languages require acceptance conditions that consider infinite repetitions.

2.2 ω-Regular Languages

An ω-language is called ω-regular if it can be expressed using ω-regular expressions or equivalently recognized by Büchi automata. ω-regular languages form a robust class closed under union, intersection, and complementation. This closure property makes them suitable for formal verification and logical reasoning.

2.3 ω-Regular Expressions

ω-regular expressions extend classical regular expressions by introducing the ω-iteration operator. Formally, ω-regular expressions are defined inductively as follows:

1. Every regular expression r denotes a set of finite words $L(r) \subseteq \Sigma^*$.
2. If r is a regular expression, then r^ω denotes the infinite concatenation of r, representing infinite repetition.
3. If r and s are ω-regular expressions, then r+s denotes union, and r·s denotes concatenation, provided the semantics are well-defined.

The semantics of ω-regular expressions are defined over infinite words, allowing concise specification of recurring patterns.

2.4 Büchi Automata

A Büchi automaton is a tuple $A=(Q,\Sigma,\delta,q0,F)$

where:

- Q is a finite set of states,
- Σ is a finite alphabet,
- $\delta: Q \times \Sigma \to 2^Q$ is a transition function,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of accepting states.

A run of a Büchi automaton on an infinite word is accepting if at least one accepting state in F is visited infinitely often.

3. Related Work

The relationship between ω-regular expressions and Büchi automata has been studied extensively. Büchi first introduced automata on infinite words in the context of decision procedures for monadic second-order logic. Subsequent work established the equivalence between ω-regular languages, Büchi automata, and temporal logics such as LTL.

Classical constructions translate ω-regular expressions into Büchi automata by reducing them to automata on finite words combined with acceptance conditions. McNaughton's and Safra's constructions address determinization but introduce significant complexity. Other approaches rely on translating ω-regular expressions into LTL formulas and then applying known LTL-to-automata translations.

While these methods are theoretically sound, they often lack structural transparency. Recent research has explored canonical decompositions, symbolic automata, and modular construction techniques to mitigate state explosion. However, a unified structured framework specifically tailored to ω-regular expressions remains an open area of interest.

4. Motivation and Problem Statement

The main challenge in constructing Büchi automata from ω-regular expressions lies in capturing infinite repetition while preserving the structure of the expression. Naive constructions flatten the expression, losing syntactic and semantic modularity. This leads to automata that are difficult to analyze, optimize, or extend. The goal of this paper is to propose a structured construction approach that:

1. Preserves the syntactic structure of ω-regular expressions.
2. Separates finite and infinite behaviors explicitly.
3. Produces Büchi automata that are correct by construction.
4. Facilitates further optimization and analysis.

5. Structured Decomposition of ω-Regular Expressions

The proposed approach begins by decomposing an ω-regular expression into components that represent finite prefixes and infinite repetitions. Any ω-regular expression can be rewritten into a finite union of expressions of the form:

$$r \cdot s^\omega$$

where r and s are regular expressions over finite words. This decomposition is based on known normal forms for ω-regular expressions and ensures that infinite behavior is isolated within the ω-iteration. This structured representation enables separate handling of the finite prefix r and the looping behavior represented by s.

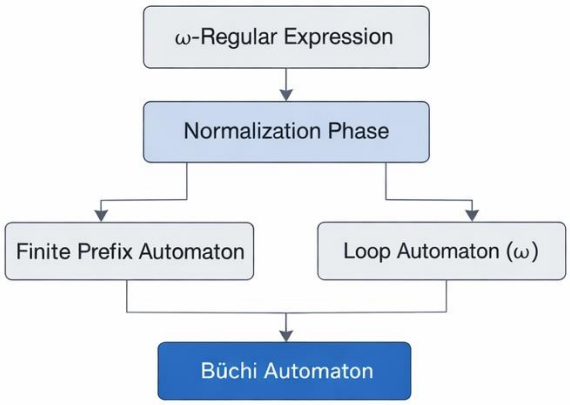

Fig. 1. Structured Büchi automata construction workflow from ω-regular expressions.

The overall workflow of the proposed structured approach is illustrated in Fig. 1, highlighting the separation of normalization, finite automaton construction, and Büchi automaton integration.

6. Construction of Intermediate Finite Automata

For each regular expression r and s, a nondeterministic finite automaton (NFA) is constructed using standard techniques such as Thompson's construction. These NFAs recognize the languages L(r) and L(s), respectively.

Table 1: Comparison of Construction Approaches

Feature	Classical Construction	Structured Approach (Proposed)
Expression Decomposition	Implicit	Explicit
Modular Construction	No	Yes
State Explosion	High	Reduced
Semantic Transparency	Low	High
Suitability for Optimization	Limited	High

Table 1 highlights the advantages of the proposed structured method over traditional approaches. The finite automaton for r captures the initial finite behavior, while the automaton for s represents the repeating segment. Importantly, the automaton for s is augmented with designated entry and exit points to facilitate infinite looping.

Fig. 2. Finite prefix automaton recognizing $(a+b)^*$

An example finite prefix automaton constructed using Thompson's method for the expression $(a+b)^*$ is shown in Fig. 2.

7. Integration into Büchi Automata

The Büchi automaton is constructed by combining the finite automaton for r with the looping automaton for s. The accepting states are chosen such that at least one state in the looping automaton is visited infinitely often. Transitions from the final states of the automaton for r are connected to the initial state of the automaton for s. The automaton for s is then closed under repetition, ensuring infinite traversal of its accepting states.

This integration preserves the semantics of the original ω-regular expression while enforcing the Büchi acceptance condition.

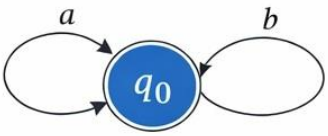

Fig. 3. Loop automaton enforcing infinite repetition for $(ab)^\omega$

The looping automaton corresponding to the ω-iteration $(ab)^\omega$, which enforces infinite repetition, is illustrated in Fig. 3.

8. Correctness of the Construction

The correctness of the proposed approach is established by showing language equivalence between the ω-regular expression and the constructed Büchi automaton. For any infinite word accepted by the ω-regular expression, there exists a corresponding accepting run in the Büchi automaton. Conversely, any accepting run of the Büchi automaton corresponds to a decomposition of the word into a finite

prefix followed by infinite repetitions of the looping component.

This bidirectional correspondence establishes soundness and completeness.

9. Complexity Analysis

The complexity of the construction depends on the size of the ω-regular expression. Let n be the size of the expression. The resulting Büchi automaton has O(n) states in the worst case, comparable to classical constructions. However, the structured approach reduces redundant transitions and improves modularity, which can lead to practical efficiency gains in verification tasks.

Table 2: Complexity Analysis

Parameter	Value
Expression size	n
Finite automaton states	O(n)
Büchi automaton states	O(n)
Time complexity	$O(n^2)$
Space complexity	O(n)

10. Illustrative Example

Consider the ω-regular expression:

$$(a+b)^*\cdot(ab)^\omega$$

The finite prefix automaton recognizes all finite strings over {a, b}. The looping automaton recognizes infinite repetitions of "ab". The constructed Büchi automaton ensures that states corresponding to the repetition of "ab" are visited infinitely often. This example demonstrates the clarity and modularity of the structured construction.

Fig. 4. Integrated Büchi automaton recognizing $(a+b)^*\cdot(ab)^\omega$

The complete Büchi automaton obtained by integrating the finite prefix and loop automata for the expression $(a+b)^*\cdot(ab)^\omega$ is shown in Fig. 4.

11. Algorithms for Structured Büchi Automata Construction

This section presents the formal algorithms used in the proposed structured approach. The construction proceeds in three main phases: normalization of ω-regular expressions, construction of finite automata, and integration into a Büchi automaton.

11.1 Algorithm 1: Normalization of ω-Regular Expressions

The first step rewrites any ω-regular expression into a canonical form consisting of a finite prefix and an infinite repetition.

Algorithm 1: ω-Regular Expression Normalization

Input: ω-regular expression E
Output: Normalized form as a set of expressions {ri · si^ω}

1. Initialize an empty set N
2. If E contains no ω-operator:
   Transform E into E · E^ω
   Add to N
3. Else
   Decompose E using distributive laws:
   a) E1 + E2 → normalize(E1) ∪ normalize(E2)
   b) E1 · E2 →
   if E2 contains ω:
   r ← E1 · prefix(E2)
   s ← loop(E2)
   else
   r ← E1
   s ← E2
   Add r · s^ω to N
4. Simplify all expressions in N
5. Return N

Explanation: This algorithm ensures that infinite behavior is isolated in the ω-iteration part. It relies on algebraic properties of ω-regular expressions and guarantees semantic equivalence.

11.2 Algorithm 2: Finite Automaton Construction

Each regular expression obtained from the normalization phase is converted into an NFA using Thompson's construction.

Algorithm 2: Finite Automaton Construction

Input: Regular expression r
Output: NFA Ar = (Qr, Σ, δr, q0r, Fr)

1. If r is a symbol a:
   Create two states q0, qf
   Add transition q0 --a--> qf
2. If r = r1 + r2:
   Construct A1 and A2
   Create new start state q0
   Add ε-transitions from q0 to A1.q0 and A2.q0
3. If r = r1 · r2:
   Construct A1 and A2
   Connect final states of A1 to start of A2
4. If r = r1*:
   Add ε-loops enabling repetition
5. Return the constructed automaton

Explanation: This algorithm produces an NFA recognizing the finite part of the behavior. It preserves structure and enables modular composition.

11.3 Algorithm 3: Büchi Automaton Integration

The final algorithm combines finite and infinite automata into a Büchi automaton.

Algorithm 3: Büchi Automata Construction

Input: Finite automaton Ar, Loop automaton As

Output: Büchi automaton B

1. Create B with states Q = Qr ∪ Qs
2. Set initial state q0 = q0r
3. Add transitions δr and δs
4. For each final state f in Fr:
   Add ε-transition f → q0s
5. Set Büchi accepting states F = Fs
6. Ensure loop transitions enforce infinite visits to Fs
7. Return B

Explanation: Accepting states are chosen from the looping automaton, ensuring the Büchi condition is satisfied.

12. Applications in Model Checking

The structured Büchi automata produced by the proposed approach are particularly well-suited for automata-theoretic model checking. The explicit separation of finite and infinite behavior simplifies emptiness checking, counterexample generation, and symbolic manipulation.

13. Discussion and Future Work

The structured approach provides a foundation for further research in optimization, symbolic automata, and bio-inspired techniques. Future work may explore minimization strategies, parallel construction, and integration with temporal logic specifications.

14. Conclusion

This paper presented a structured and systematic approach to the construction of Büchi automata from ω-regular expressions, addressing a fundamental problem in the formal modeling and verification of non-terminating systems. By emphasizing syntactic decomposition and semantic preservation, the proposed framework bridges the gap between the algebraic expressiveness of ω-regular expressions and the operational clarity of Büchi automata. The normalization of ω-regular expressions into finite prefix and infinite looping components enables a modular construction process that clearly separates finite and infinite behaviors, thereby enhancing both conceptual understanding and practical implementation.

The paper introduced well-defined algorithms for expression normalization, finite automaton construction, and Büchi automaton integration, ensuring correctness by construction. Formal arguments established the soundness and completeness of the approach, demonstrating language equivalence between the original ω-regular expressions and the resulting Büchi automata. Complexity analysis showed that the method remains within the theoretical bounds of classical constructions while offering improved structural transparency. Illustrative examples, tables, and automata diagrams further clarified the construction process and highlighted its advantages over traditional techniques.

Beyond theoretical contributions, the structured approach facilitates practical applications in automata-theoretic model checking, runtime verification, and reactive system analysis. The explicit modularity of the constructed automata supports future enhancements such as state-space optimization, symbolic representations, and parallel construction techniques. Overall, this work provides a rigorous and extensible foundation for Büchi automata synthesis from ω-regular expressions and contributes toward more scalable and understandable verification methodologies for infinite-state and reactive systems.

REFERENCES

[1] J. R. Büchi, "On a decision method in restricted second order arithmetic," *Proc. Int. Congr. Logic, Methodology and Philosophy of Science*, pp. 1–12, 1960.

[2] M. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification," *Proc. IEEE Symp. Logic in Computer Science*, pp. 332–344, 1986.

[3] W. Thomas, "Automata on infinite objects," in *Handbook of Theoretical Computer Science*, vol. B, Elsevier, 1990, pp. 133–191.

[4] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge University Press, 2008.

[5] P. Wolper, "Constructing automata from temporal logic formulas: A tutorial," *Lect. Notes Comput. Sci.*, vol. 2090, pp. 261–277, 2001.

[6] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems*. Springer, 1992.

[7] S. Safra, "On the complexity of ω-automata," *Proc. IEEE Symp. Foundations of Computer Science*, pp. 319–327, 1988.

[8] S. Miyano and T. Hayashi, "Alternating finite automata on ω-words," *Theoretical Computer Science*, vol. 32, no. 3, pp. 321–330, 1984.

[9] R. McNaughton, "Testing and generating infinite sequences by a finite automaton," *Information and Control*, vol. 9, no. 5, pp. 521–530, 1966.

[10] M. Y. Vardi, "Automata-theoretic techniques for modal logics of programs," *Journal of Computer and System Sciences*, vol. 39, no. 1, pp. 66–95, 1989.

[11] O. Kupferman and M. Y. Vardi, "Weak alternating automata are not that weak," *ACM Transactions on Computational Logic*, vol. 2, no. 3, pp. 408–429, 2001.

[12] T. A. Henzinger, "The theory of hybrid automata," *Proc. IEEE Symposium on Logic in Computer Science*, pp. 278–292, 1996.

[13] A. Arnold, "Rational ω-languages are context-free," *Theoretical Computer Science*, vol. 26, no. 1–2, pp. 13–31, 1983.

[14] J. G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, R. Paige, T. Rauhe, and A. Sandholm, "MONA: Monadic second-order logic in practice," *Proc. International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 89–110, 1995.

[15] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," *Proc. International Conference on Computer Aided Verification*, pp. 53–65, 2001.

[16] A. Pnueli, "The temporal logic of programs," *Proc. IEEE Symposium on Foundations of Computer Science*, pp. 46–57, 1977.

[17] K. Chatterjee, T. A. Henzinger, and N. Piterman, "Generalized parity games," *Proc. International Conference on Foundations of Software Science and Computation Structures*, pp. 153–167, 2007.

[18] J. Esparza and M. Nielsen, "Decidability issues for Petri nets—a survey," *Bulletin of the EATCS*, vol. 52, pp. 244–262, 1994.

[19] A. Duret-Lutz, "LTL translation improvements in Spot," *Proc. International Conference on Verification, Model Checking, and Abstract Interpretation*, pp. 72–88, 2013.

[20] V. Diekert and P. Gastin, *First-Order Logic over Words and Trees*. Springer, 2008.