Mona Reimplemented: WS1S Logic with Mata

Michal Šedý

Email: xsedym02@stud.fit.vutbr.cz;

Abstract

This paper focuses on the reimplementation of the decision procedure for WS1S logic, a second-order logic that can be decided using finite automata. The well known tool for WS1S logic decision, Mona, employs automata with transitions represented through binary decision diagrams (BDDs). Due to the integration of BDDs in automata operations, tasks like reversal cannot be executed in the conventional manner of reverting individual edges. Instead, the reversal of each BDD must be computed, potentially resulting in an exponential blowup. Motivated by these limitations, Pavel Bednar reimplemented Mona using a pure automata approach with the Mata library. This work optimizes the automata methodology, resulting in a significant speedup, up to ten times faster, in WS1S decision compared to Bednar's original reimplementation.

Keywords: Finite Automata, Binary Decision Diagrams, WS1S, MONA, MATA

1 Introduction

The most well-known decision procedures are SAT and SMT [1], which are widely used in various applications such as verification (e.g., predicate abstraction), test generation, hardware synthesis, minimization, artificial intelligence, etc. The SAT (satisfiability) problem is a decision problem that asks whether a given propositional formula is satisfiable. The SMT (satisfiability modulo theories) problem extends the SAT problem to the satisfiability of first-order formulas with equality and atoms from various first-order theories. There are various higher-order decision procedures such as WS1S, WS2S, WSkS, S1S, etc.

This work focuses on WS1S, the weak monadic second-order theory of the first successor. The term "weak" refers to finite sets, "monadic" indicates unary relations, "second-order" allows the usage of quantifiers over the relations, and "first successor" means that there is only one successor (e.g., the structure is linear). WS1S [2] has an extremely simple syntax and semantics: it is variation of predicate logic with first-order variable that denote natural numbers and second-order variables that denote finite sets of natural numbers, it has a single function symbol, which denotes the successor function and has usual comparison operators such as $\leq$, $=$, $\in$ and $\supseteq$. Richard Büchi presented approach how to decide WS1S using finite automata in [3] The main idea is to recursively transform each subformula of the main WS1S formula into deterministic finite automata (DFA) representing feasible interpretations and simulate boolean operations via the automata operations.

The most commonly used tool for deciding WS1S and WS2S is Mona[1], which employs Büchi's recursion approach for the construction of finite automata with binary

[1] accessible at https://www.brics.dk/mona/index.html

decision diagrams (BDD) to represent all automaton transitions. The use of BDD makes the decision faster, but at the cost of making some automata operations, such as reversion, expensive (potentially resulting in exponential blowup). Despite this limitation, Mona is widely utilized in various fields of program verification, including the verification of programs with complex dynamic data structures [4, 5], string analysis [6], parametrized systems [7], distributed systems [8], automatic synthesis [9], hardware verification [10], and many others.

The previously mentioned problem with hard-to-compute automata operations when using BDDs motivated Bc. Pavel Bednář's master's thesis. He reimplemented Mona's decision of WS1S by using a pure automata-based approach with the Mata automata library[2]. The special type of edge, the *jumping edge* has been introduced. The jumping edge contains information about how many variables can be jumped over. The primary idea behind introducing the jumping edge was to enable jumps not only over inner states but also over automaton states, with no upper limit on the maximal jump. However, despite this innovation, the jumping edge did not yield significant improvements in terms of space or time compression. Furthermore, it appears that jumping edges led to an overcomplication of algorithms.

In our approach, we reimplemented Bednář's solution by enhancing each automaton state with an index corresponding to the variable ID, mirroring the indexing strategy used for each inner node in the ordered BDD employed by Mona. This index information allows us to determine the length of a jump based on the indices of the source and destination states in the automaton's transition. Due to the indexing sequence on states follows a pattern of $0, 1, \ldots, n-1, 0, 1, \ldots, n-1, 0, \ldots$, the longest jump can only reach to the next state with an index of 0. While this might appear to be a step backward from Bednář's approach, the limitation on the jump length simplifies all algorithms. Surprisingly, it results in a significantly faster decision of the input formula, up to 10 times faster, compared to the variant with jumping edges.

The first section introduces basic notation and definitions of finite automata, binary decision diagrams, and WS1S. In the second section, we delve into the background of automata construction from WS1S formulas. The third section provides a detailed description of algorithms for intersection, union, complement, determinization, and minimization of automata with indexes. Moving to the fourth section, we present a comparison of decision times between the Mona tool, automata with jumping edges, and automata with indexes. The experiments are divided into two parts. Initially, automata operations are tested separately on the automata generated during Mona computation. Following that, the comparison is executed on the entire input formula.

References

[1] Vojnar Tomáš, Fiedor Jan, Konečný Filip: Lecture notes in Static Analysis and Verification. BUT - Faculty of Information Technology (2023)

[2] Klarlund, N.: A theory of restrictions for logics and automata. In: Computer Aided Verification, CAV '99. LNCS, vol. 1633

[3] Büchi, J.R.: Weak second-order arithmetic and finite automata. Mathematical Logic Quarterly **6**(1-6), 66–92 (1960) https://doi.org/10.1002/malq.19600060105

[4] Møller, A., Schwartzbach, M.I.: The pointer assertion logic engine. In: Proceedings of the ACM SIGPLAN 2001 Conference on Programming Language Design and Implementation. PLDI '01, pp. 221–231. Association for Computing Machinery, New York, NY, USA (2001). https://doi.org/10.1145/378795.378851 . https://doi.org/10.1145/378795.378851

[5] Madhusudan, P., Parlato, G., Qiu, X.: Decidable logics combining heap structures and data. SIGPLAN Not. **46**(1), 611–622 (2011) https://doi.org/10.1145/

[2] available at: https://github.com/VeriFIT/mata

1925844.1926455

[6] Tateishi, T., Pistoia, M., Tripp, O.: Path- and index-sensitive string analysis based on monadic second-order logic. ACM Trans. Softw. Eng. Methodol. **22**(4) (2013) https://doi.org/10.1145/2522920.2522926

[7] Baukus, K., Bensalem, S., Lakhnech, Y., Stahl, K., Equation, V.: Abstracting ws1s systems to verify parameterized networks. (2001). https://doi.org/10.1007/3-540-46419-0_14

[8] Klarlund, N., Nielsen, M., Sunesen, K.: A case study in verification based on trace abstractions. In: Broy, M., Merz, S., Spies, K. (eds.) Formal Systems Specification, pp. 341–373. Springer, Berlin, Heidelberg (1996)

[9] Sandholm, A., Schwartzbach, M.I.: Distributed safety controllers for web services. In: Astesiano, E. (ed.) Fundamental Approaches to Software Engineering, pp. 270–284. Springer, Berlin, Heidelberg (1998)

[10] Basin, D., Klarlund, N.: Automata based symbolic reasoning in hardware verification. Form. Methods Syst. Des. **13**(3), 255–288 (1998) https://doi.org/10.1023/A:1008644009416