Toward Determining NFA Equivalence via QBFs (Student Abstract)

Hannah Miller and David E. Narváez
Golisano College of Computing and Information Sciences
Rochester Institute of Technology, Rochester, NY 14623
den9562@rit.edu

Abstract

Equivalence of deterministic finite automata (DFAs) has been researched for several decades, but equivalence of nondeterministic finite automata (NFAs) is not as studied. Equivalence of two NFAs is a PSPACE-complete problem. NFA equivalence is a challenging theoretical problem with practical applications such as lexical analysis. Quantified boolean formulas (QBFs) naturally encode PSPACE-complete problems, and we share our preliminary work towards determining NFA equivalence via QBFs.

Motivation

Equivalence of two finite automata is a classic computational problem. For *deterministic finite automata* (DFAs), Hopcroft's algorithm (Hopcroft 1971) runs in near-linear time. For *nondeterministic finite automata* (NFAs), the equivalence problem is PSPACE-complete, and we are interested in studying NFA equivalence. In addition to being a challenging theoretical problem, practical uses of NFAs include lexical analysis in compilers (Aho et al. 2006).

Background

A *finite automaton* has a finite alphabet of symbols Σ, a finite set of states Q, a start state $q_0 \in Q$, a transition function δ to move between the states as the automaton reads an input string w, and a set of final states $F \subseteq Q$ where the string w is accepted if the automaton ends on a final state after reading w. The *language* L of a finite automaton is the set of all strings which the automaton accepts, and two automata are *equivalent* if they accept the same language L. For two inequivalent automata, a *witness string* is a string that one automaton accepts, but the other automaton rejects.

The key difference between a *deterministic finite automaton* (DFA) and a *nondeterministic finite automaton* (NFA) is the transition function δ to move between states. For DFAs, $\delta : Q \times \Sigma \rightarrow Q$ indicates the next state after reading a symbol σ at a state q, but for NFAs, $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ indicates the set of possible next states after reading a symbol σ at a state q.[1] DFAs and NFAs accept the same class of languages, and an NFA can be transformed to a DFA that accepts the same language. This transformation may incur an exponential blow-up.

Boolean *satisfiability* (SAT) is the classic NP-complete problem of determining whether a Boolean formula is satisfiable. In the encoding below, we describe our Boolean formulas using conventional operations like $\wedge$ (logical and), $\vee$ (logical or), and $\rightarrow$ (implication). In practice, our implementation is based on the representation of Boolean formulas as combinational circuits. The inputs of a circuit representing the Boolean formula φ are the variables of φ. A *satisfying assignment* is an assignment of values to the inputs of the circuit such that the output of the circuit is True. If we include the quantifiers *for all* (written as $\forall$) and *there exists* (written as $\exists$), we get a *quantified boolean formula* (QBF).

The problem of determining whether a QBF is true (TQBF) is complete for PSPACE, a problem class that is even harder than the NP-complete problem class (Stockmeyer and Meyer 1973). Determining if two NFAs are equivalent is a PSPACE problem, so QBFs can be used to encode PSPACE problems. Recent work by Bonchi and Pous (2013; 2015) uses bisimulation and coinduction to show the equivalence of two NFAs. We are interested in comparing the bisimulation approach to the TQBF approach for different classes of NFAs, starting with randomly generated NFAs.

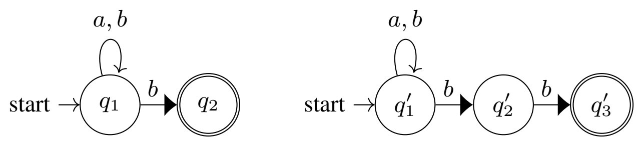

Figure 1: NFAs N (left) and N' (right). The alphabet Σ is $\{a, b\}$, and final states are drawn with a double circle. These NFAs are inequivalent since N accepts the witness string ab, but N' does not.

QBF Encoding

Consider two NFAs N and N' with transition functions δ and δ', respectively. For S and T two sets of states of N, S' and T' two sets of states of N', and k an integer, we will

[1]Formally, NFAs include so-called "ϵ transitions," but due to space constraints, we do not deal with ϵ transitions here. Our experimental implementation does support ϵ transitions.

construct a QBF $\varphi_k(S, S', T, T')$ that is true if and only if there exists a string w of length $|w| \leq k$ such that if N is possibly in one of the states in the set S and N' is possibly in one of the states in the set S', then after reading w, N will possibly be in one of the states in the set T and N' will possibly be in one of the states in the set T'.

For k a power of 2, we define φ_k recursively as follows. For the case $k > 1$, we know $\varphi_k(S, S', T, T')$ is true if and only if there exist intermediate sets of states R, R' such that for all sets of states X, X', Y, Y', we have[2]

$$\begin{aligned}\exists R, R'. \forall X, X', Y, Y'. [((S, S', R, R') = (X, X', Y, Y') \\ \vee (X, X', Y, Y') = (R, R', T, T')) \\ \rightarrow \varphi_{k/2}(X, X', Y, Y')]. \qquad (1)\end{aligned}$$

For the base case of $k = 1$, we have

$$\begin{aligned}\varphi_1(S, S', T, T') = \left[S = T \wedge S' = T'\right] \\ \vee \bigvee_{\sigma \in \Sigma} \left[\delta(S, \sigma) = T \wedge \delta'(S', \sigma) = T'\right], \qquad (2)\end{aligned}$$

where δ and δ' are naturally extended to take sets as parameters.

Intuitively, Equation 2 encodes that either the sets S, T (resp. S', T') are equal or when N (resp. N') is possibly in one of the states in the set S (resp. S') and reads one symbol σ, then N (resp. N') will possibly be in one of the states in the set T (resp. T').

N and N' are inequivalent if and only if there is a witness string w of length at most k that is accepted by one NFA and not by the other. This is to say that N and N' are inequivalent iff

$$\begin{aligned}\exists T. \exists T'. [\varphi_k(\{q_0\}, \{q'_0\}, T, T') \\ \wedge (T \text{ contains a final state of } N \text{ iff} \\ T' \text{ does not contain a final state of } N')], \qquad (3)\end{aligned}$$

where q_0 and q'_0 are the start states of N and N', respectively. An upper bound for k is $2^{n+n'}$, where n and n' are the number of states in N and N', respectively (Hopcroft, Motwani, and Ullman 2006). We keep our QBF length polynomial with the same technique used to prove that QBF is PSPACE-hard (Stockmeyer and Meyer 1973). The size of Equation 1 is polynomial in the size of the input NFAs because there is only one recursive call per recursion level, the recursion depth is $O(n + n')$, and each call has one child.

Preliminary Experimental Results

For our preliminary work, we have generated random NFAs inspired by the random method from Tabakov and Vardi (2005). To create a random NFA, we fix the alphabet Σ to be $\{a, b\}$ and the start state as the single state q_0. We define the total number of states n, the probability of choosing a state to be a final state p_f, and the probability of drawing a transition arrow between two states p_δ. For every possible directed pair of states $q, \tilde{q} \in Q$ and every possible input $\sigma \in \Sigma$, we draw a transition arrow between q and $\tilde{q}$ with probability p_δ.

We encoded the equivalence problem between pairs of these random NFAs in the QCIR-14 format (Jordan, Klieber, and Seidl 2016). From the list of winners of the last QBFEVAL competition, we selected the solvers that were compatible with our output format and runtime environment; in the future, we plan to expand our runtime environment to include more solvers. We also used a conversion script[3] to turn these encodings into the QDIMACS format supported by many QSAT solvers. The supplemental material[4] includes preliminary running times for these instances when using popular QSAT solvers for the respective formats.

Future Work

We will extend the output format of our conversion tool to directly support outputting formulas in the QDIMACS format. We will compare our QBF technique to Hopcroft's algorithm and the bisimulation approach of Bonchi and Pous. To normalize the results among the different approaches, we will develop and use the same merit function[5] for each approach so that the programming language used (e.g., Bonchi and Pous used OCaml) does not change the results. We will investigate producing a short witness string from the certificate of satisfiability of our QBFs.

Acknowledgments

We thank Edith Hemaspaandra for her helpful comments. Research supported in part by NSF grant DUE-1819546.

References

Aho, A. V.; Lam, M. S.; Sethi, R.; and Ullman, J. D. 2006. *Compilers: Principles, Techniques, and Tools*. Pearson Education, Inc., 2nd edition.

Bonchi, F.; and Pous, D. 2013. Checking NFA equivalence with bisimulations up to congruence. In *POPL '13*, 457–468. ACM.

Bonchi, F.; and Pous, D. 2015. Hacking Nondeterminism with Induction and Coinduction. *CACM* 58(2): 87—95.

Hopcroft, J. 1971. An $n \log n$ algorithm for minimizing states in a finite automaton. Technical Report Technical Report STAN-CS-71-190, Stanford University.

Hopcroft, J. E.; Motwani, R.; and Ullman, J. D. 2006. *Introduction to Automata Theory, Languages, and Computation*. Pearson Education, Inc., 3rd edition.

Jordan, C.; Klieber, W.; and Seidl, M. 2016. Non-CNF QBF Solving with QCIR. In *AAAI Workshop: Beyond NP*, AAAI Press.

Stockmeyer, L. J.; and Meyer, A. R. 1973. Word Problems Requiring Exponential Time (Preliminary Report). In *STOC '73*, 1–9.

Tabakov, D.; and Vardi, M. Y. 2005. Experimental Evaluation of Classical Automata Constructions. In Sutcliffe, G.; and Voronkov, A., eds., *LPAR 2005*, 396–411. Springer.

[2]When we write $A = B$ for sets A and B, we mean that the standard representation of A and B as arrays of Boolean values is equal, i.e., $A = B \equiv \bigwedge(a_i \leftrightarrow b_i)$.

[3]https://github.com/gogforce/qcir14_to_pcnf

[4]https://doi.org/10.5281/zenodo.4279874

[5]e.g., Bonchi and Pous track the number of processed pairs in their experimental work