On the Power of Automata Minimization in Reactive Synthesis

Shufang Zhu[1], Lucas M. Tabajara[2], Geguang Pu[3] and Moshe Vardi[2]

[1]Sapienza University of Rome
[2]Rice University
[3]East China Normal University

Temporal Logic

- Temporal logic, properties over a sequence of successive state
- Linear Temporal Logic (LTL), infinite sequence [Pnueli, 1977]
- LTL over Finite traces (LTL_f) [De Giacomo & Vardi, 2013]
    - Natural to capture finite-horizon tasks
    - Numerous applications
        - Markov Decision Processes (MDPs) with non-Markovian rewards, MDPs policy synthesis, program synthesis etc.

LTL_f and Finite-word Automata

LTL_f: same syntax as LTL, interpreted on finite (unbounded) traces

$$\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid X\phi \mid \phi_1 U \phi_2 \mid F\phi$$

- LTL_f formula, corresponding DFA that accepts the same language
- Reasoning about LTL_f, compiling to DFA
- LTL_f synthesis, reduction to an adversarial reachability game on DFA

LTL_f and Minimized DFA

- Doubly-exponential blowup, bottleneck of LTL_f synthesis [Zhu et al., 2017]
- DFA can be fully minimized
- The best way of constructing a minimal DFA from LTL_f?

Automata Minimization

- Two minimization algorithms
  - Hopcroft algorithm
  - Brzozowski algorithm
- Prior work, starting from randomly-generated NFA [Tabakov, Rozier & Vardi, 2012]
  - Neither algorithm dominates
  - No specific application scenario

Automata Minimization in LTL$_f$ Synthesis

- Hopcroft vs. Brzozowski, starting from LTL$_f$ formulas
- In the context of program synthesis
- Symbolic synthesis framework
  - Symbolic DFA representation

Semi-Symbolic DFA

A tuple $\mathcal{A} = (\mathcal{P}, \mathcal{S}, s_0, H, Acc)$

- $\mathcal{P}$ a set of propositions
- $\mathcal{S}$ is a set of states
- $s_0 \in \mathcal{S}$ is the initial state
- $H : \mathcal{S} \times \Lambda \times \mathcal{S}$, where Λ is a set of propositional formulas over $\mathcal{P}$. For example, $(s, a \wedge b, s') \in H$
- Acc is a set of final states

Symbolic DFA

Given a semi-symbolic DFA $\mathcal{A} = (\mathcal{P}, \mathcal{S}, s_0, H, Acc)$, its symbolic representation is a tuple $\mathcal{D} = (\mathcal{P}, \mathcal{Z}, I, \delta, f)$

- $\mathcal{P}$ a set of propositions
- $\mathcal{Z}$ is a set of state variables, every state is an assignment over $\mathcal{Z}$
- $I \in 2^{\mathcal{Z}}$ is the initial assignment
- $\delta : 2^{\mathcal{Z}} \times 2^{\mathcal{P}} \to 2^{\mathcal{Z}}$ is the transition function, an indexed family $\{\delta_z : 2^{\mathcal{Z}} \times 2^{\mathcal{P}} \to \{0, 1\} \mid z \in \mathcal{Z}\}$
- f is a propositional formula over $\mathcal{Z}$, the set of accepting states

Hopcroft vs. Brzozowski from LTL_f

- Hopcroft's algorithm
  - Construct non-deterministic automata $\mathcal{N}$
  - $\mathcal{A} = [\textit{equivalence} \circ \textit{determinize}](\mathcal{N})$
  - MONA, temporal specification to minimized DFA [Henriksen, 1995]
- Brzozowski's algorithm
  - $\mathcal{A} = [\textit{reachable} \circ \textit{determinize} \circ \textit{reverse}]^2(\varphi)$
  - No existing implementation

Brzozowski's Algorithm from LTL_f

$\mathcal{A} = [\mathit{reachable} \circ \mathit{determinize} \circ \mathit{reverse}]^2(\varphi)$

1. Reverse DFA construction, DFA accepting the **reverse** language of φ
   - $[\mathit{reachable} \circ \mathit{determinize} \circ \mathit{reverse}](\varphi)$
2. Reversal into a co-DFA
   - $[\mathit{reverse}]$
3. Determinization and pruning
   - $[\mathit{reachable} \circ \mathit{determinize}]$

Brzozowski's Algorithm from LTL_f

1. Reverse DFA construction
   - Minimal DFA of φ^R [Zhu et al., 2019]
2. Reversal into a co-DFA, [*reverse*]
   - Swapping initial and final states, reversing transitions
3. Determinization and pruning, [$\mathit{reachable} \circ \mathit{determinize}$]
   - Symbolic or Explicit

Symbolic Subset Construction

Given NFA $\mathcal{N} = (\mathcal{P}, \mathcal{S}, \mathcal{S}_0, H, Acc)$, the symbolic DFA $\mathcal{D} = (\mathcal{P}, \mathcal{S}, I, \delta, f)$ can be obtained by

- $\mathcal{S}$ is the set of state variables
- $I \in 2^{\mathcal{S}}$ is such that $I(s) = 1$ iff $s \in \mathcal{S}_0$
- $f = \bigvee_{s \in Acc} s$
- $\delta_s : 2^{\mathcal{S}} \times 2^{\mathcal{P}} \to \{0, 1\}$ is such that $\delta_s(S, \sigma) = 1$ iff $(d, \lambda, s) \in H$ for some d such that $S(d) = 1$ and λ such that $\sigma \models \lambda$

Symbolic State-Space Pruning

The set of reachable states r over state variables $\mathcal{Z}$

- $r_0(Z) = I(Z)$
- $r_{i+1}(Z) = r_i(Z) \vee \exists Z'.\exists X.\exists Y.r_i(Z') \wedge (\delta(Z', X \cup Y) = Z)$

Empirical Evaluation

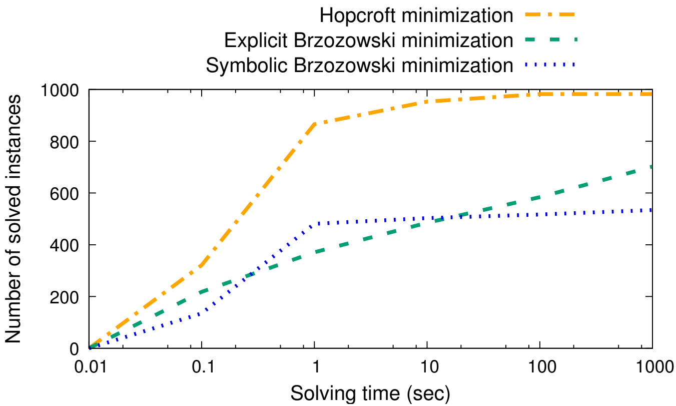
Candidates:

- Hopcroft minimization, represented by MONA
- Explicit Brzozowski minimization
- Symbolic Brzozowski minimization

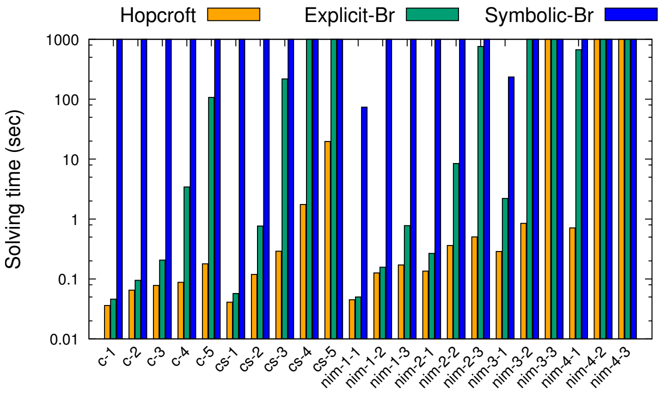
Synthesis Benchmarks:

- Random benchmarks
- Two-player-game benchmarks: Single-Counter, Double-Counter, Nim

Random Benchmarks



Two-player-game Benchmarks



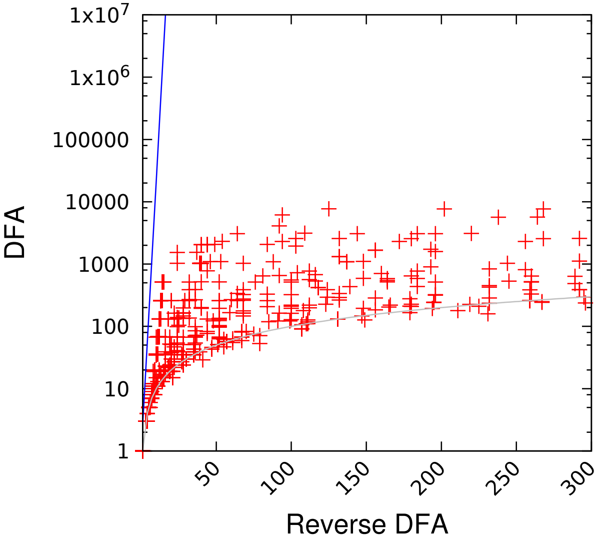
Observations

1. Hopcroft's approach dominates on all benchmarks
2. Explicit Brzozowski's approach even performs better than the symbolic version

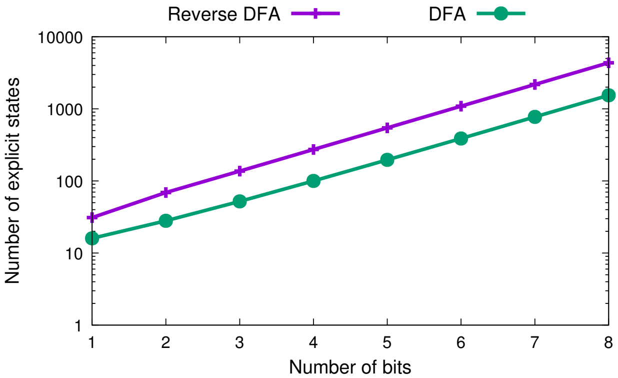
Recap on Brzozowski's Approach

1. **Reverse DFA construction**
   - Theoretical conclusion: Reverse DFA, exponentially smaller than DFA
2. Reversal into a co-DFA
3. Determinization and pruning
   - Second exponential blowup

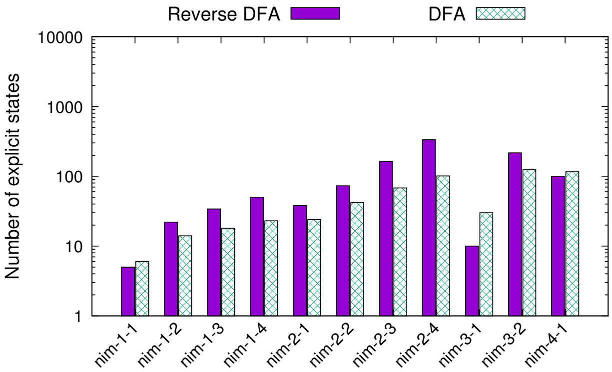
Reverse DFA vs. DFA on Random Benchmarks



Reverse DFA vs. DFA on Single-Counter Benchmarks



Reverse DFA vs. DFA on Nim Benchmarks



Reasons of Failure

1. Theoretical result: Reverse DFA can be exponentially smaller than DFA
2. Practical observation: No, this is not the case!
3. Subsequent effects: cannot handle further steps of determinization and pruning

Takeaways

- Using Hopcroft's algorithm is more promising for synthesis than employing Brzozowski's construction.
- Theoretical worst-case scenario might not be the common case in practice