LTLf Synthesis under Partial Observability: From Theory to Practice

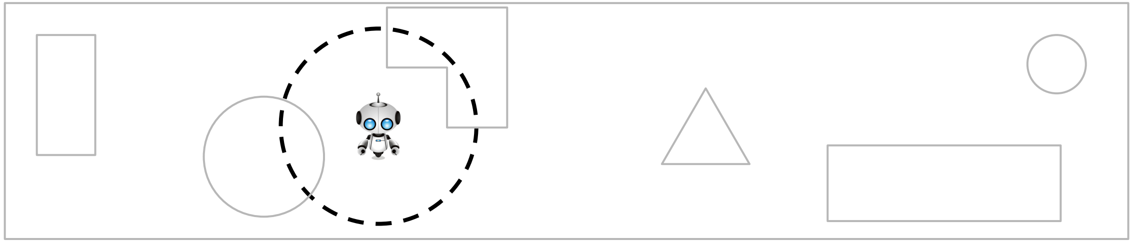
Lucas M. Tabajara

Joint work with Moshe Y. Vardi

Rice University (now at Runtime Verification, Inc.)

LTLf Synthesis under Partial Observability [DV, 2016]

Generalization of both LTLf synthesis and planning under partial observability.



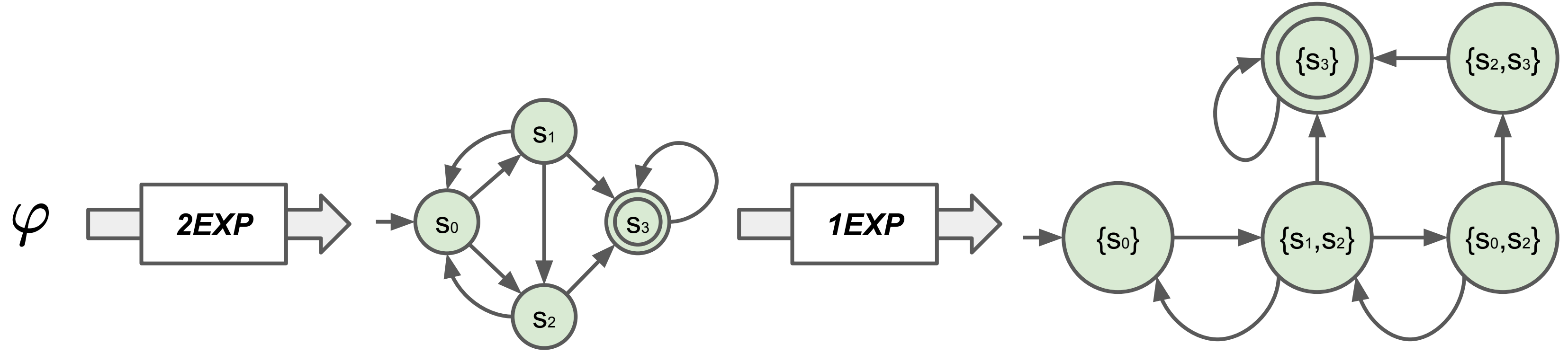
Example: Robot can only sense its local vicinity.

Key difference to regular LTLf synthesis:

- Input variables partitioned into *observable inputs **I*** and *unobservable inputs **U***.
- Agent strategy has to satisfy the specification φ without seeing the unobservable inputs.
- Equivalent to synthesizing ($\forall u_1,\ldots,u_n : \varphi$).

Algorithms for LTLf Synthesis under Partial Observability [DV, 2016]

1. Belief-states construction ***(3EXPTIME* algorithm)**:



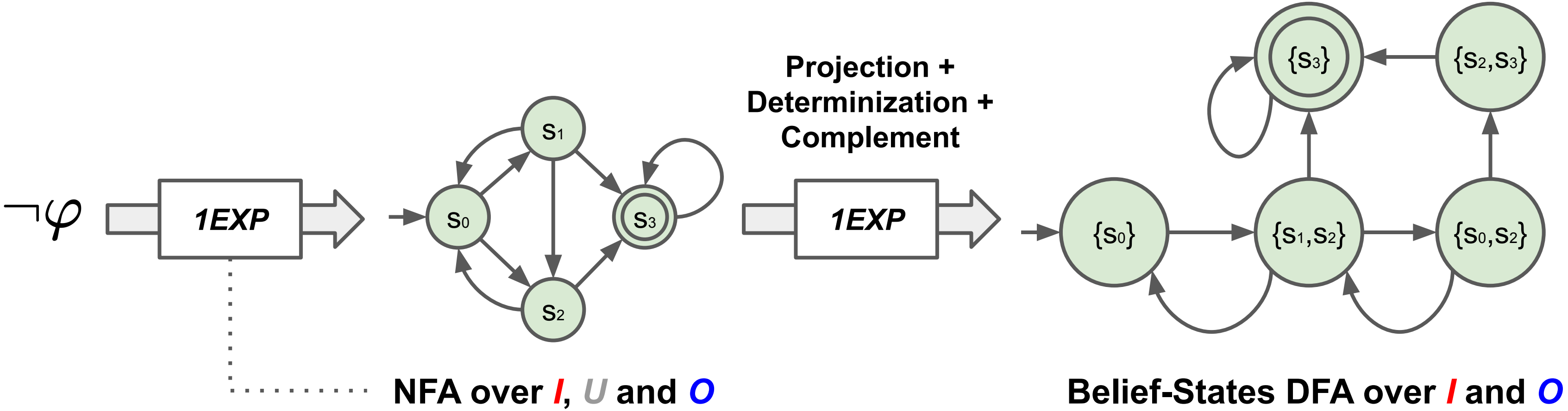
DFA over *I*, *U* and *O*

Belief-States DFA over *I* and *O*

- Belief state is a *set of possible states* the DFA can be in.
- Belief-States DFA is a DFA for ($\forall u_1,\ldots,u_n : \varphi$).

Algorithms for LTLf Synthesis under Partial Observability [DV, 2016]

2. Projection-based approach (***2EXPTIME* algorithm**):



- Construct Belief-States DFA as $\neg(\exists u_1,\dots,u_n : \neg\varphi) \equiv (\forall u_1,\dots,u_n : \varphi)$.
- Using NFA can save up to one exponential in the construction of the belief-states DFA.

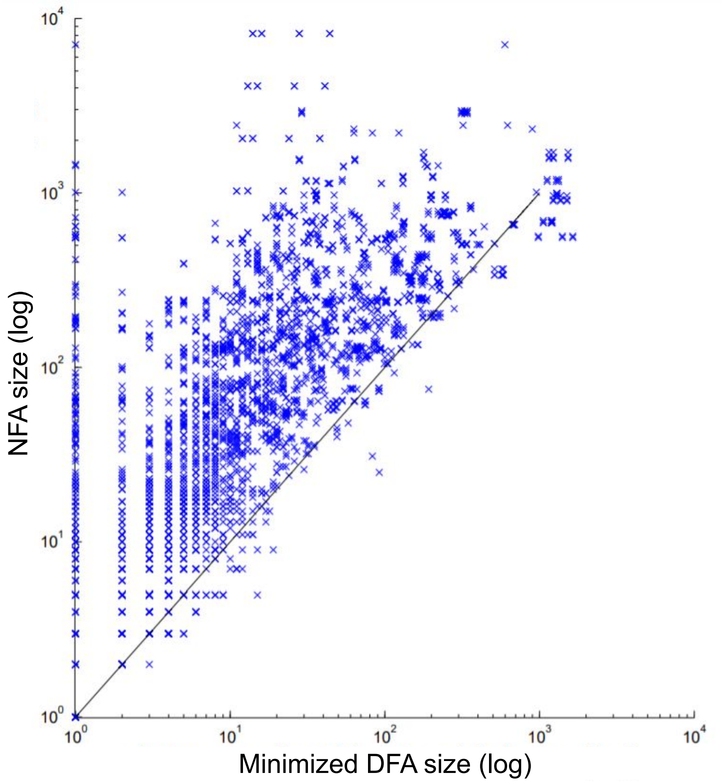
DFA vs. NFA - Theory vs. Practice

Complexity analysis depends on worst-case exponential gap between DFA and NFA.

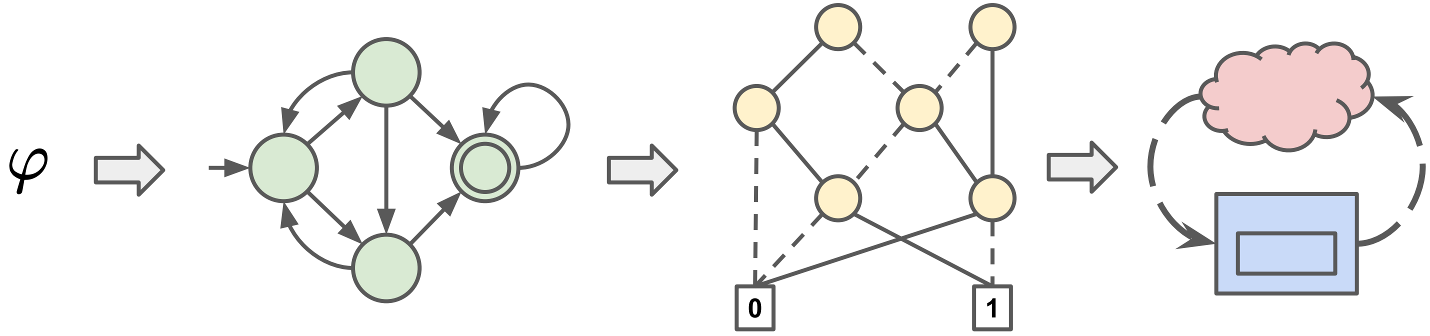
But DFAs have the advantage of being fully and efficiently minimizable.

Experimental analysis has shown that in practice determinizing and minimizing a finite automaton often makes it *smaller*. [TRV, 2011]

Question: Does the worst-case theoretical analysis of the two algorithms truly predict how they perform in practice?



Synthesis under Partial Observability in Practice [TV, 2020]



LTLf Specification

Explicit DFA/NFA over *I*, *U* and *O*

Symbolic Belief-States DFA over *I* and *O*

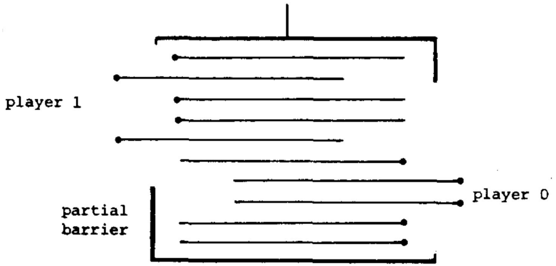
Synthesized System

- Construction of belief-states DFA is natural to implement symbolically.
- Symbolic construction can save one exponential (N BDD variables for 2^N DFA states).

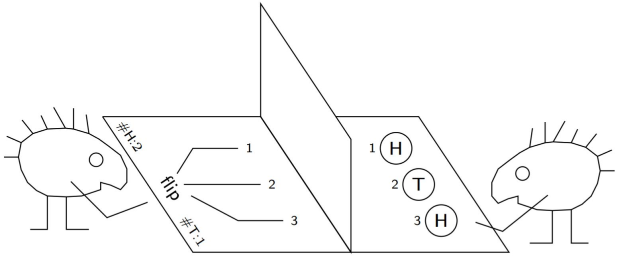
Empirical Evaluation

Benchmark families based on games with incomplete information:

Moving Target

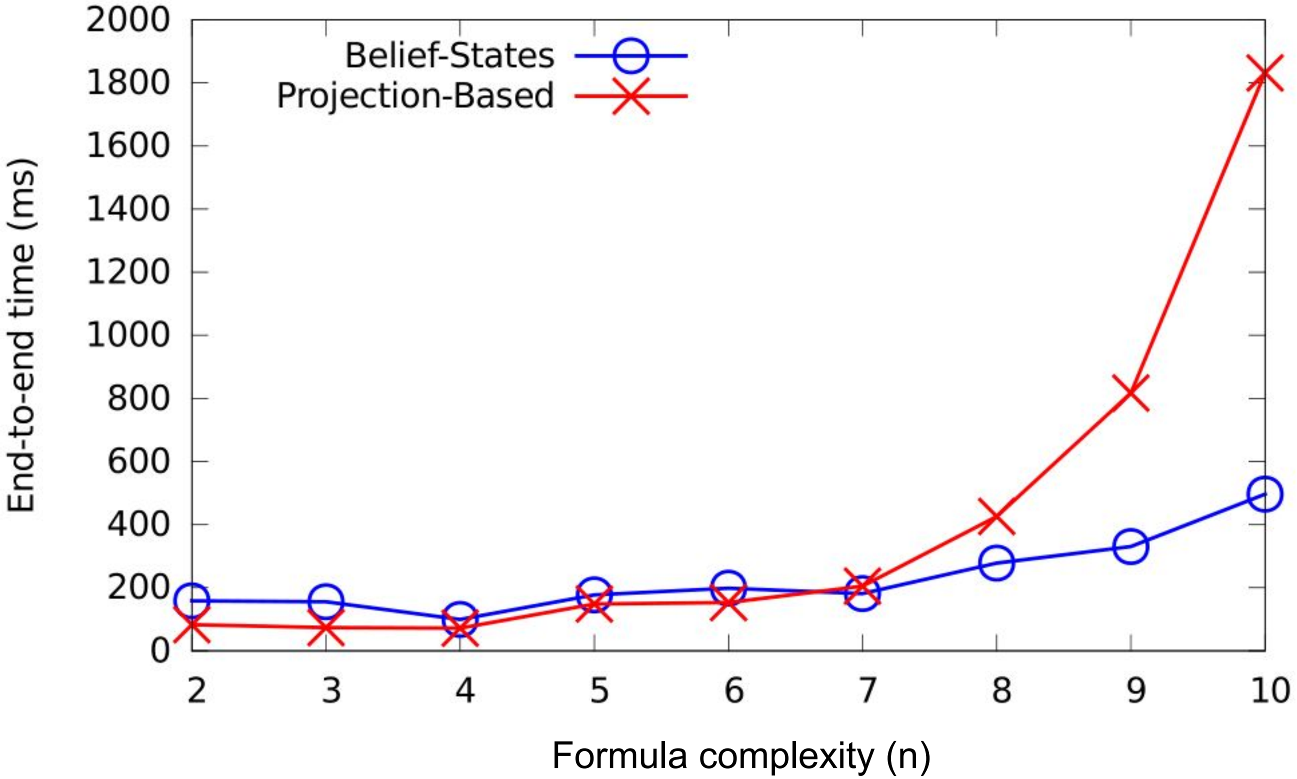


Private Peek [R, 1984]



Coin Game [DR, 2011]

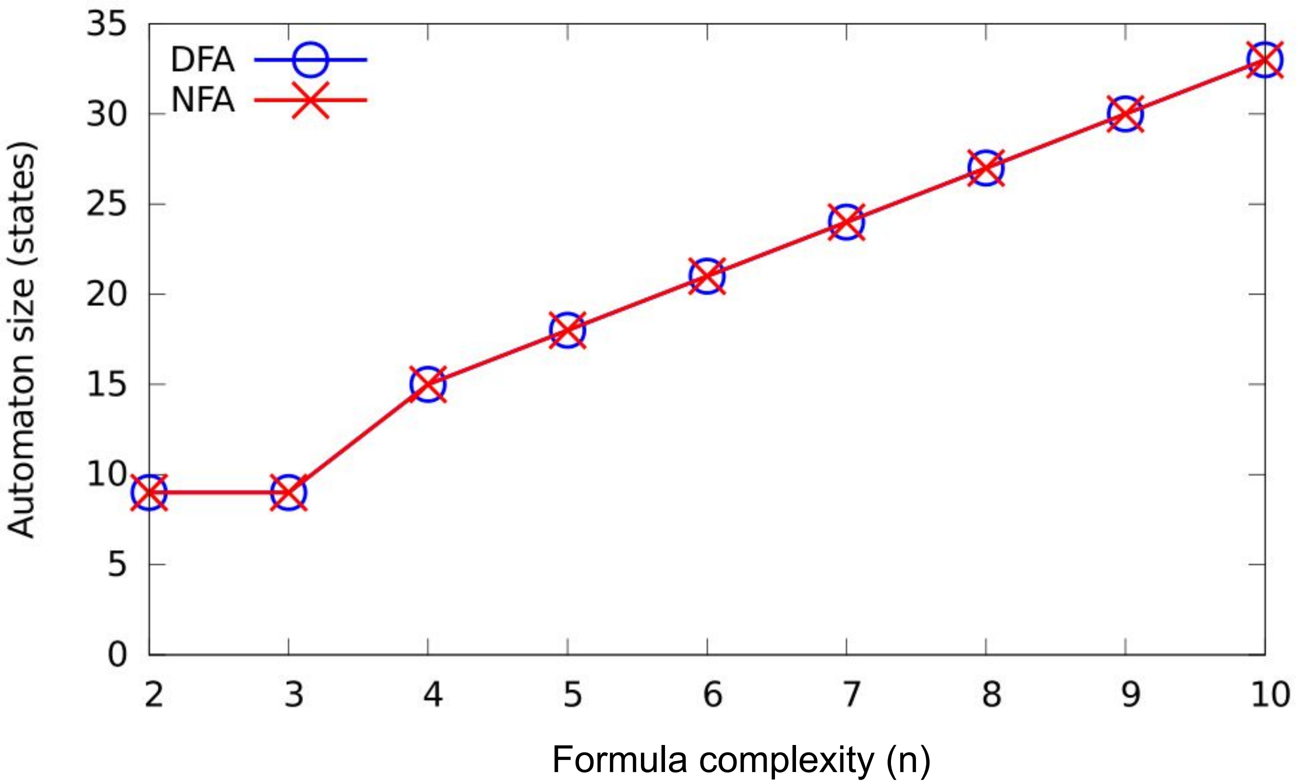
Synthesis Running Time



Using *Moving-Target* family as an example. Other families showed similar results.

Contrary to theoretical analysis, belief-states scales significantly better than projection-based.

DFA vs. NFA Size



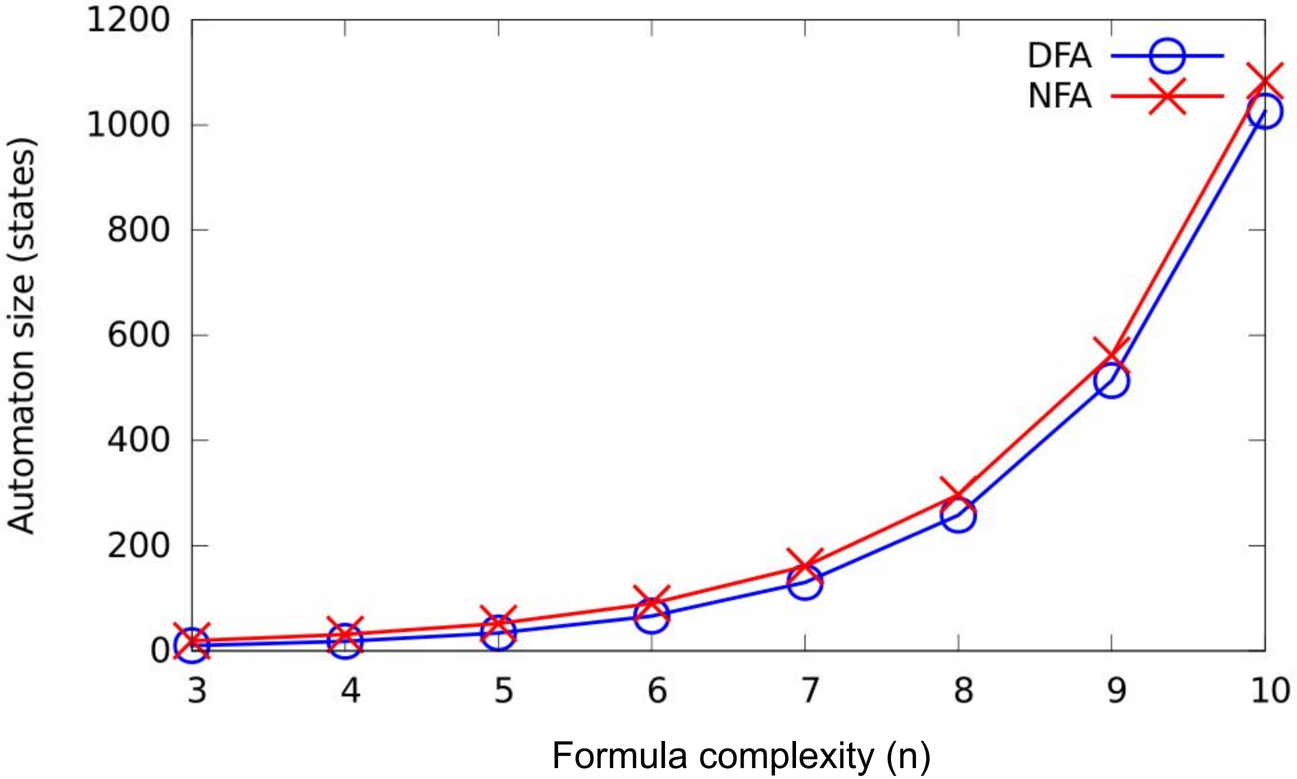
In theory:

NFA is exponentially smaller than DFA.

In practice:

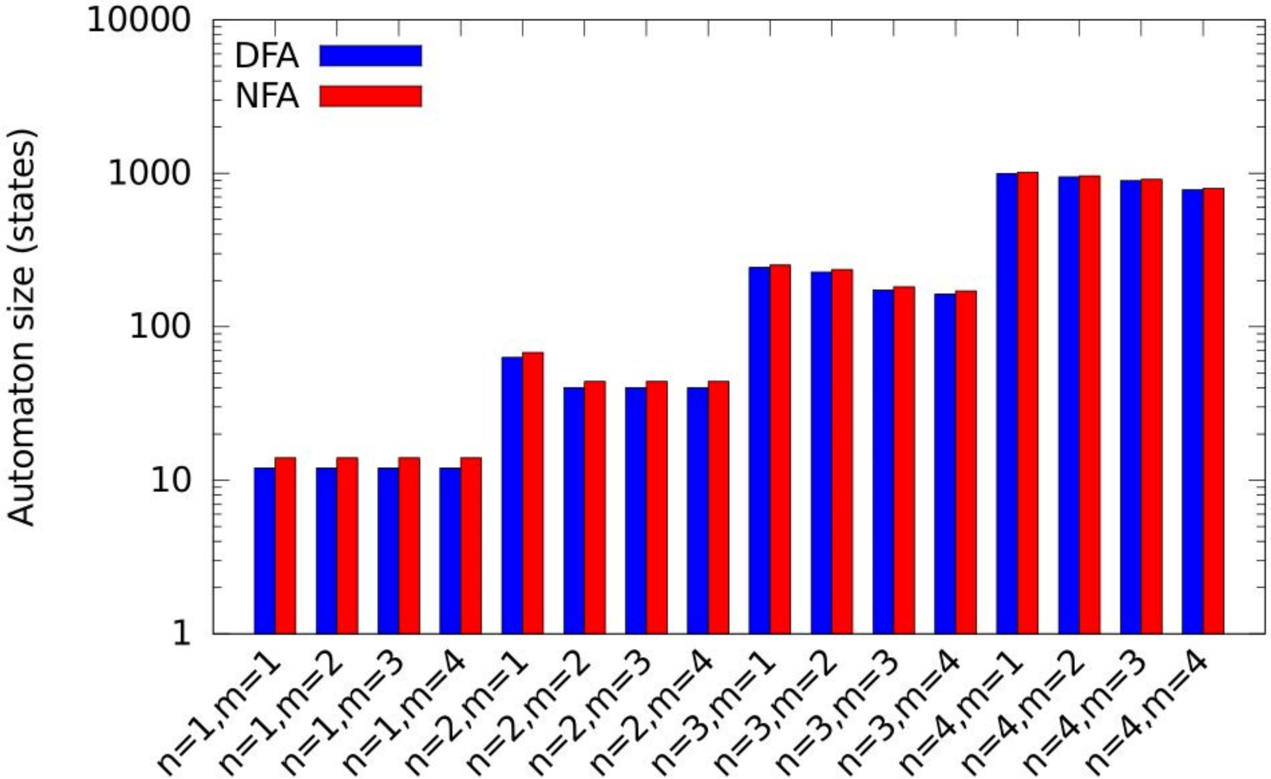
NFA and DFA are exactly the same size.

DFA vs. NFA Size



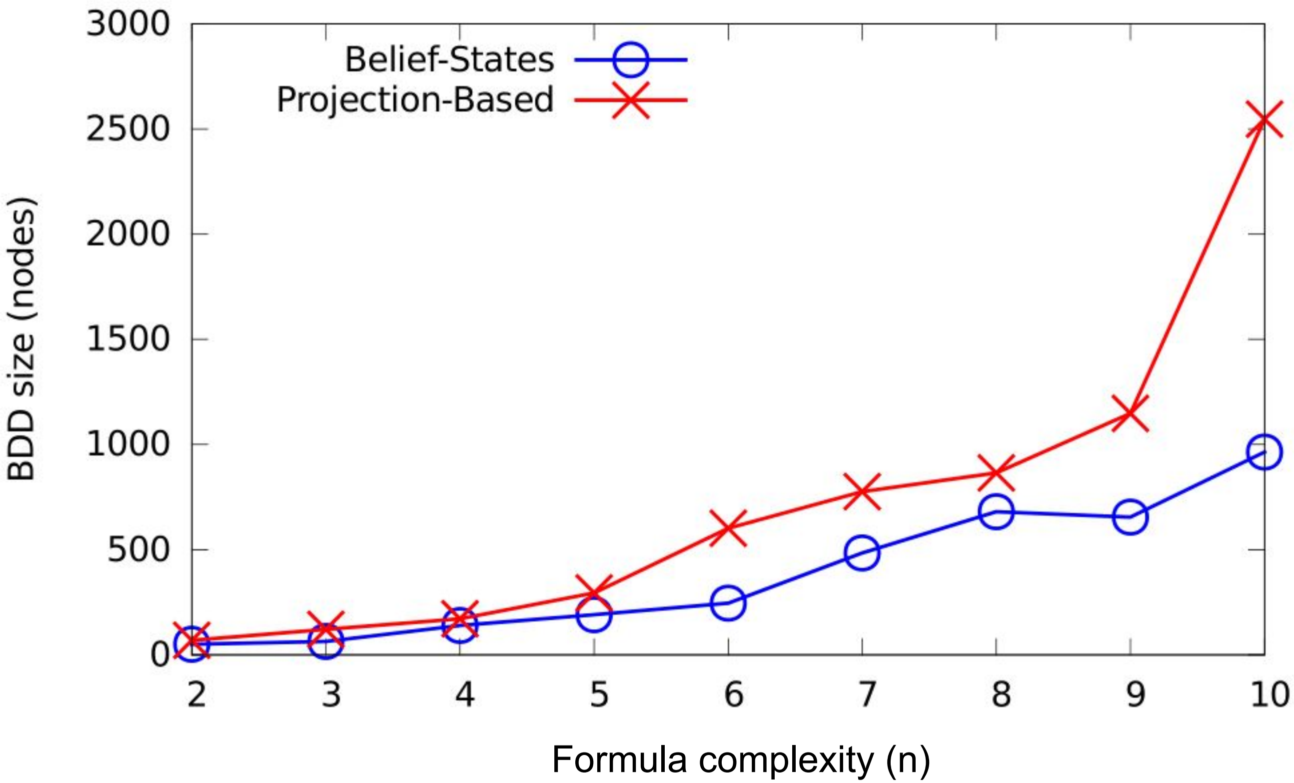
For other benchmark families, NFA is slightly larger than DFA.

DFA vs. NFA Size



For other benchmark families, NFA is slightly larger than DFA.

BDD Representation Size



Projection-based approach produces a larger and less efficient BDD representation.

LTLf Synthesis under Partial Observability - Takeaways

Theoretical results don't always tell the whole story.

- Practical considerations can have a greater impact than worst-case complexity.
- Important to complement theoretical analysis with empirical evaluation.

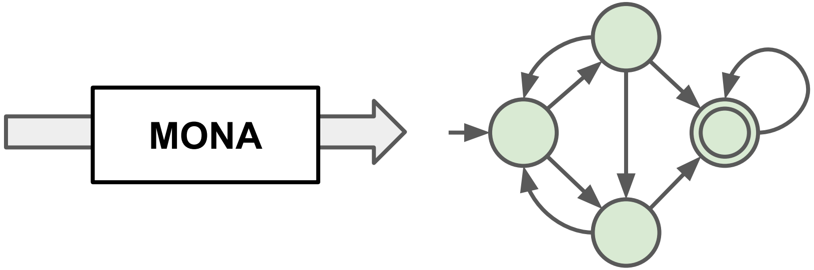
LTLf allows exploring more complex synthesis scenarios in practice.

- LTL synthesis with incomplete information had never left the realm of theory.
- LTLf enables extensions that are impractical in the infinite-horizon domain.

Extra Slides

A Third Algorithm: MSO Approach

$\forall u_1, \ldots, u_n . \varphi$

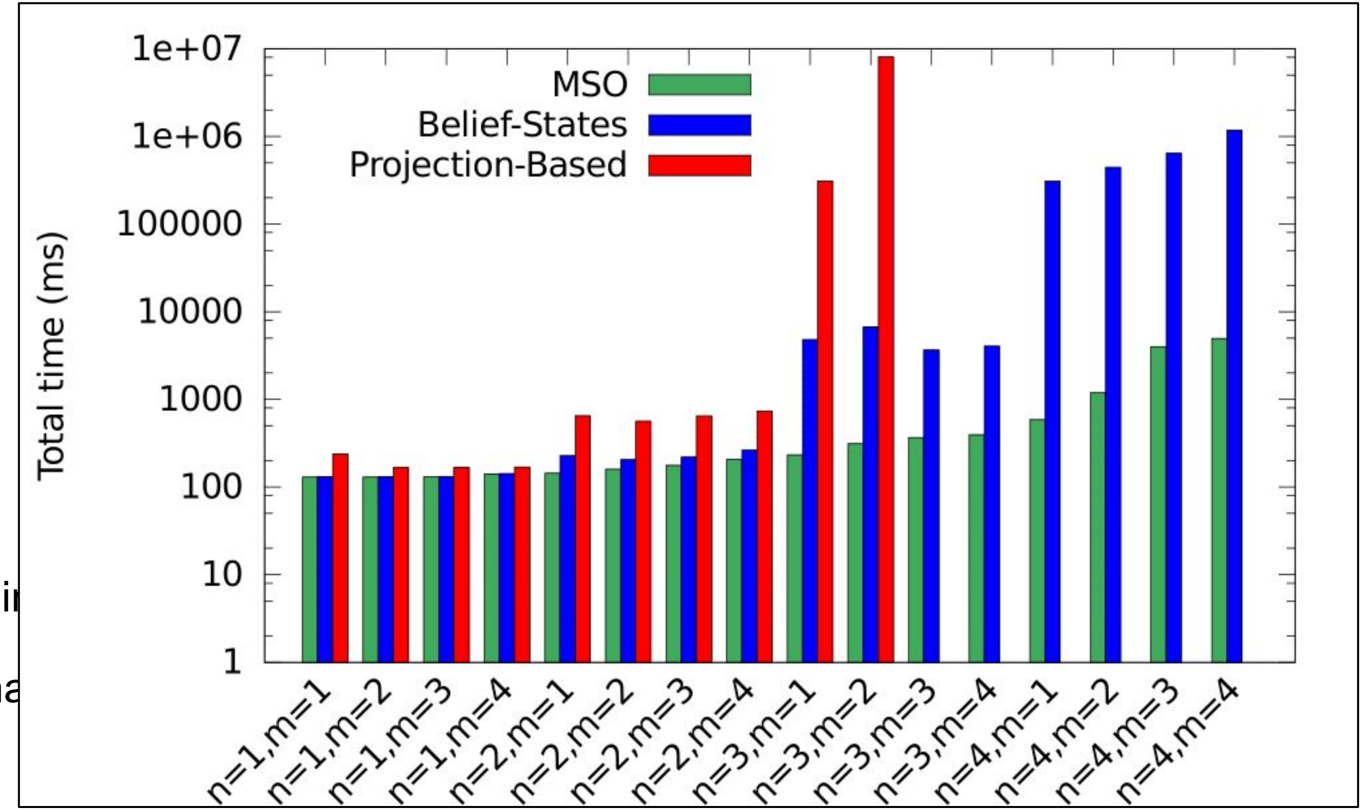


(Minimized) Explicit DFA over *I* and *O*

Pros: Minimized state space, so reachability game is easier to solve.

Cons: Final DFA constructed explicitly, so construction is more expensive.

A Third Algorithm: MSO Approach



Pros: Mini

Cons: Fina