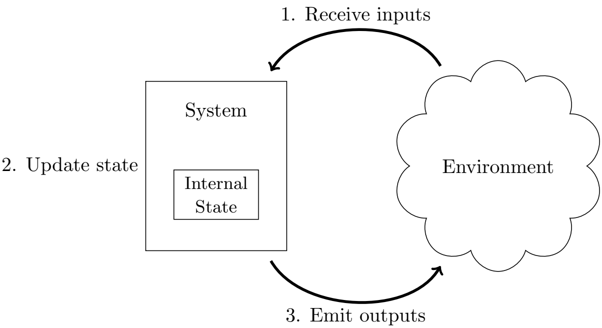
A Symbolic Approach to Safety LTL Synthesis

Shufang Zhu [1] **Lucas M. Tabajara** [2] Jianwen Li [2]
Geguang Pu [1] Moshe Y. Vardi [2]

[1]East China Normal University

[2]Rice University

Reactive Synthesis



Goal: Automatically design reactive systems that are guaranteed to follow a temporal specification.

LTL Synthesis

Linear Temporal Logic (LTL):

$$\varphi ::= \top \mid \bot \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid X\varphi \mid \varphi_1 R \varphi_2 \mid \varphi_1 U \varphi_2$$

$$G\varphi \equiv \bot R \varphi \qquad F\varphi \equiv \top U \varphi$$

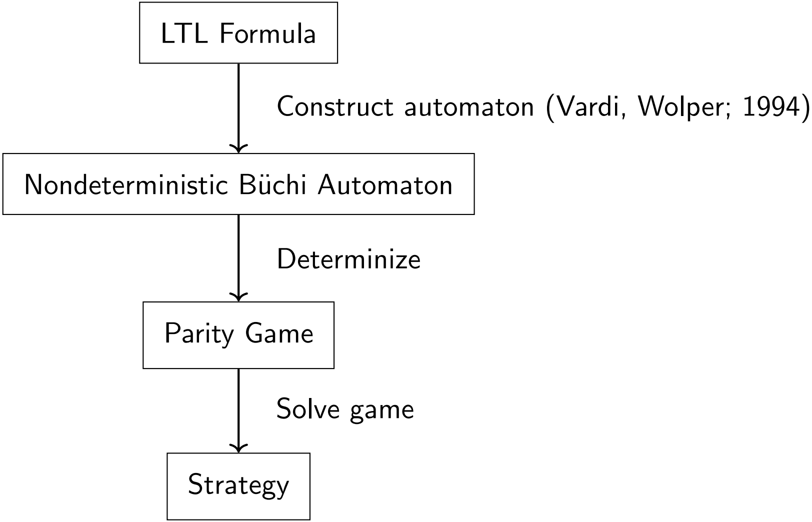
LTL Synthesis:

Given: LTL formula φ over a set of propositional variables $\mathcal{P} = \mathcal{X} \cup \mathcal{Y}$

- Input variables: $\mathcal{X}$
- Output variables: $\mathcal{Y}$

Obtain: Set of states S and strategy $g : 2^{\mathcal{X}} \times S \to 2^{\mathcal{Y}} \times S$ such that every trace satisfies φ.

Classical Approach to LTL Synthesis



Synthesis of LTL Fragments

LTL synthesis remains a challenging problem:

- 2EXPTIME theoretical complexity.
- Lack of scalable algorithms for determinization and solving games.

Solution: Focus on synthesis procedures for fragments of LTL.

Example: Generalized Reactivity(1) (GR(1)) fragment:

$$(\theta^e \wedge G\rho^e \wedge GF\varphi_1^e \wedge \ldots \wedge GF\varphi_m^e) \rightarrow (\theta^s \wedge G\rho^s \wedge GF\varphi_1^s \wedge \ldots \wedge GF\varphi_n^s)$$

- GR(1) games can be solved in time cubic in size of game graph.

Other easier fragments of LTL?

Synthesis of LTL Fragments

LTL synthesis remains a challenging problem:

- 2EXPTIME theoretical complexity.
- Lack of scalable algorithms for determinization and solving games.

Solution: Focus on synthesis procedures for fragments of LTL.

Example: Generalized Reactivity(1) (GR(1)) fragment:

$$(\theta^e \wedge G\rho^e \wedge GF\varphi_1^e \wedge \ldots \wedge GF\varphi_m^e) \rightarrow (\theta^s \wedge G\rho^s \wedge GF\varphi_1^s \wedge \ldots \wedge GF\varphi_n^s)$$

- GR(1) games can be solved in time cubic in size of game graph.

Other easier fragments of LTL? Safety LTL

Safety Properties

“Bad things don’t happen”

Safety property:

$$pRq$$

(q doesn’t become false until after p becomes true)

Non-safety property:

$$G(r \rightarrow Fg)$$

(every request is eventually granted)

Safety Properties

“Bad things don’t happen”

Safety property:

$$pRq$$

(q doesn't become false until after p becomes true)

Safety property:

$$G(r \rightarrow (g \vee Xg \vee XXg))$$

(every request is granted within two time steps)

Safety Properties

"Bad things don't happen"

Safety property:

$$pRq$$

(q doesn't become false until after p becomes true)

Safety property:

$$G(r \rightarrow (g \vee Xg \vee XXg))$$

(every request is granted within two time steps)

All eventualities are bounded.

Bad prefix

For a given temporal formula φ, a finite trace $\pi = \pi_1\pi_2 \ldots \pi_n$ is a *bad prefix* if π cannot be extended to a satisfying trace.

$$\varphi = pRq$$

$$\{q\}, \{q\}, \ldots, \{q\}, \{p, q\}, \{p\}, \ldots \models \varphi$$

$$\{q\}, \{q\}, \ldots, \{q\}, \{\}, \{p\}, \ldots \not\models \varphi$$

A temporal formula φ is *safe* if every trace that does not satisfy φ has a bad prefix.

Syntactical Safety

Purely *syntactical* sufficient condition for safety:

Theorem (Sistla; 1994)

If φ is an LTL formula in Negation Normal Form and φ is Until-free, then φ is safe.

Allows us to define an LTL fragment that guarantees safety.

Safety LTL

Linear Temporal Logic (LTL):

$$\varphi ::= \top \mid \bot \mid p \mid \boldsymbol{\neg\varphi} \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid X\varphi \mid \varphi_1 R \varphi_2 \mid \boldsymbol{\varphi_1 U \varphi_2}$$

Safety LTL:

$$\varphi ::= \top \mid \bot \mid p \mid \boldsymbol{\neg p} \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid X\varphi \mid \varphi_1 R \varphi_2$$

Safety LTL corresponds to the fragment of **Until-free** LTL formulas in **Negation Normal Form**.

Synthesis of the Safety LTL Fragment

Safety LTL Synthesis:

Given: Safety LTL formula φ over a set of propositional variables $\mathcal{P} = \mathcal{X} \cup \mathcal{Y}$

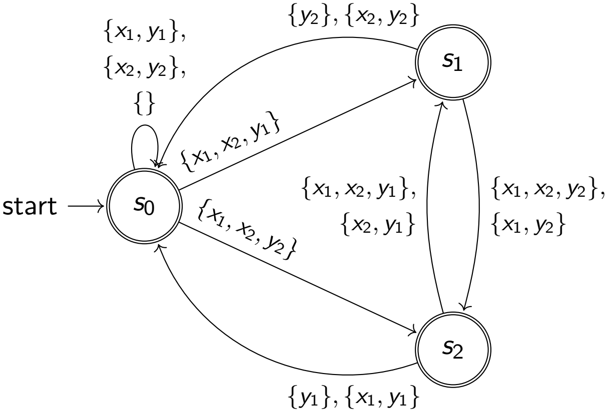
- Input variables: $\mathcal{X}$
- Output variables: $\mathcal{Y}$

Obtain: Set of states S and strategy $g : 2^{\mathcal{X}} \times S \rightarrow 2^{\mathcal{Y}} \times S$ such that every trace satisfies φ.

Our work: Safety LTL synthesis can be reduced to *safety games*.

Deterministic Safety Automata (DSA)

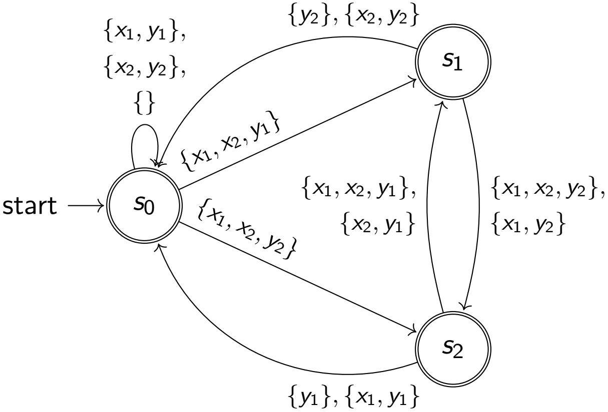
Every Safety LTL formula can be converted to a DSA:



- Büchi with partial transition function and all states accepting.

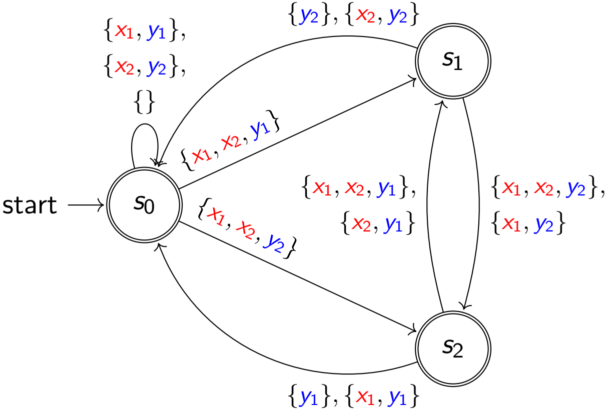
Deterministic Safety Automata (DSA)

Every Safety LTL formula can be converted to a DSA:



- Run is accepting iff never takes an undefined transition (bad prefix).

Safety Games



- *Environment* controls input variables $\mathcal{X}$, wins if automaton rejects.
- *System* controls output variables $\mathcal{Y}$, wins if automaton *never* rejects.

Safety Games for Safety LTL Synthesis

Winning strategy for the system encodes solution to Safety LTL synthesis:

$$
\begin{aligned}
\text{System wins} &\Rightarrow \text{Automaton never rejects}\\
&\Rightarrow \text{No undefined transition}\\
&\Rightarrow \text{No bad prefix}\\
&\Rightarrow \text{Formula is satisfied}
\end{aligned}
$$

Safety games can be solved efficiently: linear time in size of game graph.

Our goal: Efficient techniques for Safety LTL synthesis via safety games.

First Approach: Horn-SAT

Key idea: Reduce safety games to Horn-SAT.

Horn-SAT

Given a boolean formula $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_m$ where every φ_i is of the form $(p_1 \wedge \ldots \wedge p_n) \rightarrow q$, is φ satisfiable?

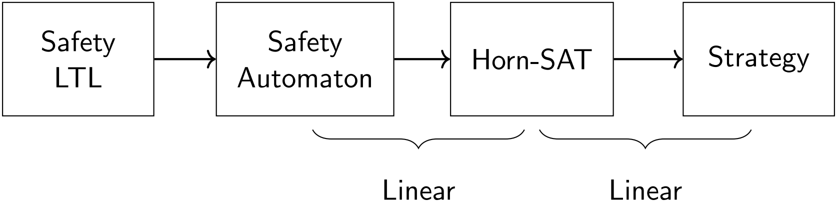
Horn-SAT can be solved in linear time by SAT solvers using constraint propagation.

First Approach: Horn-SAT

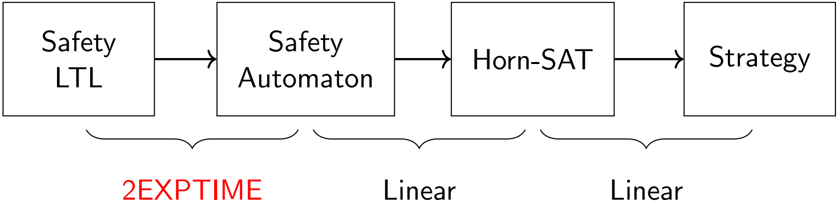
Key idea: Reduce safety games to Horn-SAT.

1. Use SPOT (Duret-Lutz, et al; 2016): LTL to Büchi automata.
   - Safety LTL is special case of LTL.
   - Safety automaton is special case of Büchi automaton.
2. Encode safety game as Horn formula.
   - Satisfying assignment encodes winning strategy.
3. Solve Horn-SAT using SAT solver.

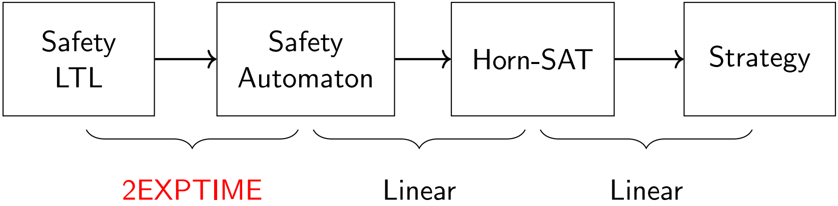
The State Explosion Problem



The State Explosion Problem



The State Explosion Problem



Solution: Represent the safety automaton symbolically using Binary Decision Diagrams (BDDs).

- State space of size n encoded using $\log_2(n)$ boolean variables $\mathcal{Z}$.
- Every state represented by an assignment $2^{\mathcal{Z}}$.
- Transition function as boolean function $2^{\mathcal{X}} \times 2^{\mathcal{Y}} \times 2^{\mathcal{Z}} \rightarrow 2^{\mathcal{Z}}$.

Second Approach: Symbolic Safety LTL Synthesis

Key idea: Leverage tools for symbolic construction of automata over *finite* words.

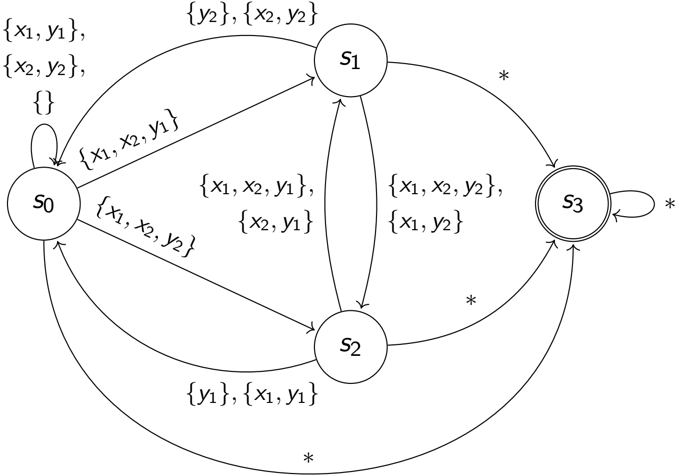
- MONA (Henrikson, et al; 1995): First-Order Logic over *finite* words to symbolic Deterministic Finite Automata (using BDDs).
- Safety LTL: like LTL, interpreted over *infinite* words.
- **However:** every falsifying trace of φ has *finite* bad prefix.

$$\{q\}, \{q\}, \ldots, \{q\}, \{\}, \{p\}, \ldots \not\models pRq$$

- **Therefore:** can translate $\neg\varphi$ to FOL over *finite* bad prefixes.

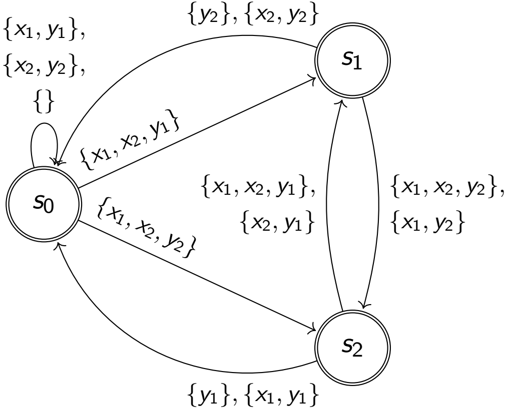
Finite Automaton to Safety Automaton

MONA constructs DFA for the bad prefixes of φ:



Finite Automaton to Safety Automaton

By deleting bad states, we can view DFA as DSA for φ:



Symbolic Safety LTL Synthesis

Given Safety LTL formula φ:

1. Use MONA to construct symbolic DFA for bad prefixes of φ.
2. Interpret symbolic DFA as symbolic DSA.
3. Compute winning states as a fixpoint:
   3.1 Start with set of all accepting states.
   3.2 At each step, remove states where Environment can move to bad state.
   3.3 Stop when fixpoint is reached.
4. Compute strategy as a boolean function using boolean-synthesis procedure (Fried, **Tabajara**, Vardi; CAV'2016).

Two Approaches for Safety LTL Synthesis

- Explicit synthesis framework:

Safety LTL → Safety Automaton → **Horn-SAT** → Strategy

- Symbolic synthesis framework:

Safety LTL → **Symbolic DFA** → Symbolic DSA → Strategy

Experimental Evaluation

Comparison between:

- Explicit approach using Horn-SAT.
- SSYFT tool implementing symbolic approach.
- LTL Synthesis tools UNBEAST (Ehlers; 2010) and ACACIA+ (Bohy, et al; 2012).

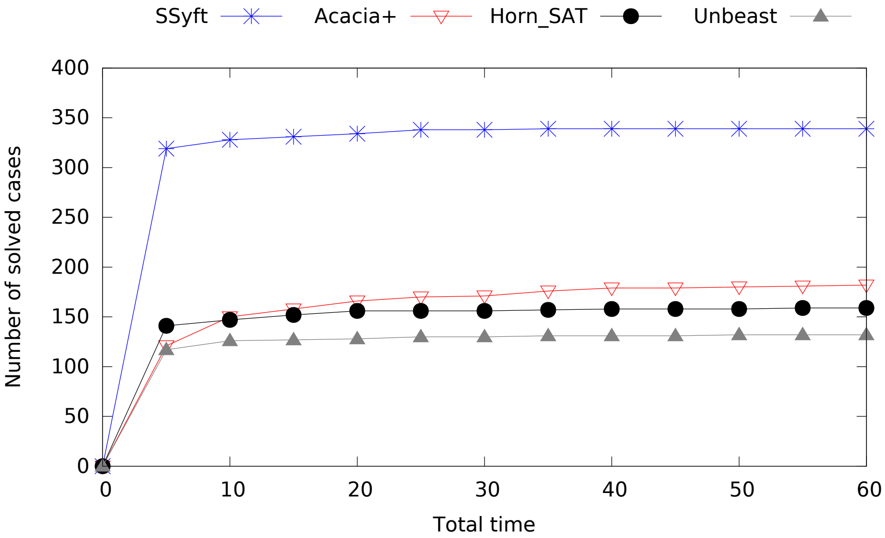
Benchmarks

LoadBalancer formulas from (Ehlers; 2010):

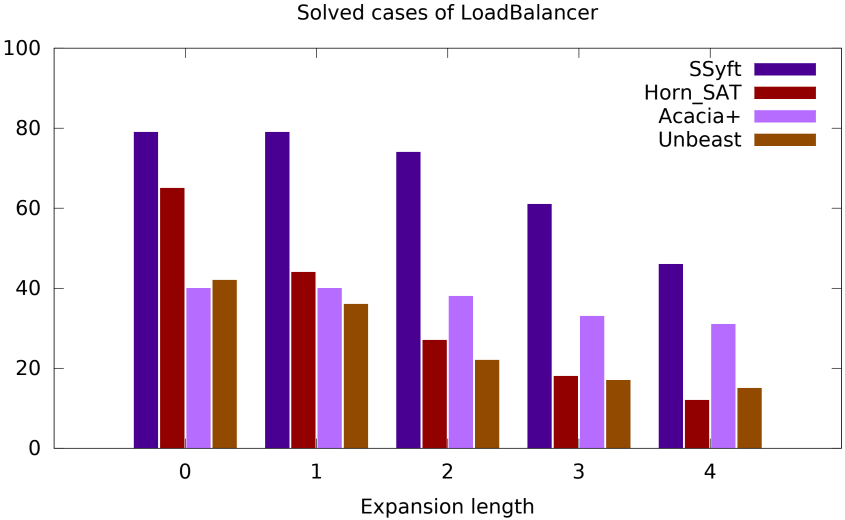
- Converted to Negation Normal Form.
- Since not all formulas are safe, expanded Until operator:

  Not safe: $\varphi_1 U \varphi_2$
  Expansion length 0: φ_2
  Expansion length 1: $\varphi_2 \vee (\varphi_1 \wedge X\varphi_2)$
  Expansion length 2: $\varphi_2 \vee (\varphi_1 \wedge X(\varphi_2 \vee (\varphi_1 \wedge X\varphi_2)))$
  ...
- Varied expansion length.

Symbolic Approach Dominates



Symbolic Approach Dominates



Summary

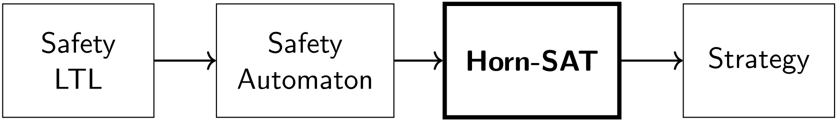
- **Contribution:** Two frameworks for Safety LTL synthesis - explicit and symbolic.
- **Results:** Symbolic framework outperforms tools for general LTL synthesis.
- **Conclusion:** Can benefit from focusing on specific LTL fragments for synthesis.

Future Work

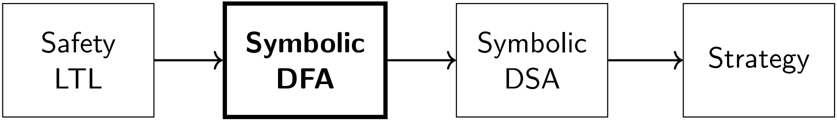
- *On-the-fly* synthesis to avoid bottleneck of automaton construction.
- Comparison with other LTL fragments, such as GR(1) (Bloem, Jobstmann, Piterman, Pnueli; 2012).
- Safety games as a subproblem of general LTL synthesis.

Questions?

- Explicit synthesis framework:



- Symbolic synthesis framework:



Extra Slides

Safety LTL vs. GR(1)

GR(1) formula:

$$(\theta^e \wedge G\rho^e \wedge GF\varphi_1^e \wedge \ldots \wedge GF\varphi_m^e) \rightarrow (\theta^s \wedge G\rho^s \wedge GF\varphi_1^s \wedge \ldots \wedge GF\varphi_n^s)$$

For $\alpha \in \{e, s\}$:

- θ^α: Safety
- $G\rho^\alpha$: Safety
- $GF\varphi^\alpha$: Non-safety

A GR(1) formula with $m = n = 0$ is a safety formula.

Safety Game to Horn-SAT

Given a Safety Automaton $\mathcal{A} = (2^{\mathcal{P}}, S, s_0, \delta)$, build a Horn formula where:

- Variables encode bad states:

$$\begin{aligned} b_s &: \quad s \text{ is a losing state for the System} \\ b_{(s,X,Y)} &: \quad Y \text{ is a losing move of the System on state } s \text{ for input } X \end{aligned}$$

- Constraints encode bad transitions:

$$b_{(s,X,Y)}, \quad \text{for } \delta(s, X \cup Y) \text{ undefined} \tag{1}$$

$$b_{s'} \rightarrow b_{(s,X,Y)}, \quad \text{for } \delta(s, X \cup Y) = s' \tag{2}$$

$$\left(\bigwedge_{Y \in \mathcal{Y}} b_{(s,X,Y)} \right) \rightarrow b_s, \quad \text{for every } s \in S,\ X \in 2^{\mathcal{X}} \tag{3}$$

$$b_{s_0} \rightarrow \bot \tag{4}$$