Model Checking LTL over the finite horizon

Suguman Bansal
Georgia Institute of Technology

Yong Li
University of Liverpool

Lucas M. Tabajara
Runtime Verification

Moshe Y. Vardi
Rice University

Andrew Wells
AWS

Linear-temporal logic over finite horizon (LTLf)

[Baier and McIlraith; 2006][De Giacomo and Vardi; IJCAI 2013]

- Specification language
  - Temporal logic over discrete time

- Syntax
  - Boolean variables and operators
  - Temporal operators: Always, Eventually, Next, Until …

Example: Always (Request → (Grant ∨ Next Grant))

- "Every request is granted within the two steps"

Linear-temporal logic over finite horizon (LTLf)

[Baier and McIlraith; 2006][De Giacomo and Vardi; IJCAI 2013]

- Specification language
  - Temporal logic over discrete time
- Syntax
  - Boolean variables and operators
  - Temporal operators: Always, Eventually, Next, …

Example: Always (Request → (Grant ∨ Next Grant))

- "Every request is granted within the two steps"

Request	T	F	T	F	T	T
Grant	T	F	F	T	F	T

Linear-temporal logic over finite horizon (LTLf)

[Baier and McIlraith; 2006][De Giacomo and Vardi; IJCAI 2013]

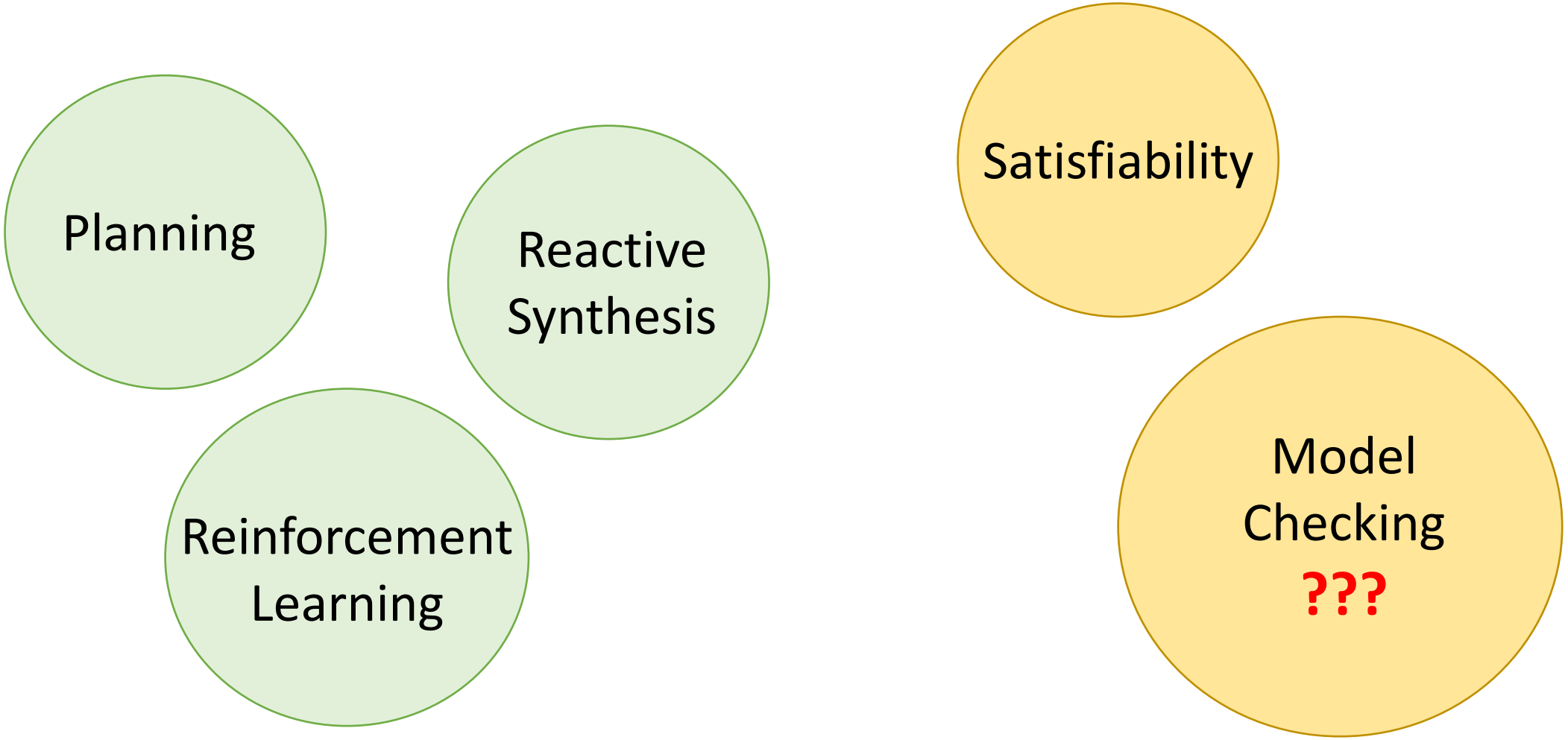
- Specification language
  - Temporal logic over discrete time
- Syntax
  - Boolean variables and operators
  - Temporal operators: Always, Eventually, Next, …

Example: Always (Request → (Grant ∨ Next Grant))

- "Every request is granted within the two steps"

Request	T	F	T	F	T	T
Grant	F	F	F	T	F	T

Prior works on LTLf



LTL vs. LTLf

LTLf is at most as hard as LTL, if not easier

	LTL	LTLf
Deterministic automata	(DPA) 2 Exponential	(DFA) 2 Exponential
Satisfiability	PSPACE-complete	PSPACE-complete
Synthesis	2EXPTIME-complete	2EXPTIME-complete

LTL vs. LTLf

LTLf is at most as hard as LTL, if not easier

	LTL	LTLf
Deterministic automata	(DPA) 2 Exponential	(DFA) 2 Exponential
Satisfiability	PSPACE-complete	PSPACE-complete
Synthesis	2EXPTIME-complete	2EXPTIME-complete
Model Checking	**PSPACE-complete**	**EXPSPACE-complete**

LTL model checking

[Vardi and Woolper; LICS 1986]

Model M satisfies LTL specification ψ if

Every execution of model satisfies ψ

$$M \subseteq \psi \quad \text{iff} \quad M \cap \underbrace{\neg\psi}_{\text{Counterexamples}} = \emptyset$$

LTL model checking is PSPACE-complete

The issue with LTLf model checking

Model M satisfies LTLf specification ψ if

Every execution of model satisfies ψ

Executions are of Infinite-length

Defined over finite-length executions

Non-terminating models are without terminating states

- Model checking of LTL is on non-terminating models
- Generated by most LTLf reactive synthesis algorithms/tool

LTLf model checking

Model M satisfies LTLf specification ψ if

Every execution of model **has a prefix** that satisfies ψ

$$M \subseteq \underbrace{\exists \text{ prefix, prefix satisfies } \psi}_{\text{Words of infinite-length}}$$

$$(M \cap \underbrace{\forall \text{ prefix, prefix satisfies } \neg\, \psi}_{\text{Counterexamples}}) = \emptyset$$

Complexity

Given LTLf formula ψ, generate counterexample automata

- Accepts infinite word if all prefixes satisfy $\neg\psi$

- Deterministic form is double exponential in ψ (upper bound)

- Non-deterministic form is also double exponential in ψ (lower bound)
  - Universal quantification on prefixes
  - LTLf can identify Last of a finite execution
  - Intuition: (For every prefix, Last a) $\equiv$ (Globally a) in LTL

Theorem.
For non-terminating models, LTLf model checking is EXPSPACE-hard

In a nutshell
Model Checking LTL over the Finite Horizon

- Among the first to study model checking of LTLf specifications

- For non-terminating models, LTLf model checking is EXPSPACE-hard
- For terminating models, LTLf model checking is PSPACE-hard

- Efficient and scalable algorithms for LTLf model checking
- Busted the myth that LTLf is easier than LTL: Examine implications