Integrating Induction and Deduction in Model Checking

Sanjit A. Seshia

EECS Department

UC Berkeley

Clarke Symposium

September 19, 2014

Model Checking and Synthesis

E. M. Clarke and E. A. Emerson, 1981:

"We propose a method of constructing concurrent programs in which the ***synchronization skeleton of the program is automatically synthesized*** from a high-level (branching time) Temporal Logic specification."

(1st sentence of their seminal model checking paper)

Three Messages in this Talk

[Seshia DAC'12; Jha & Seshia, SYNT'14]

1. **Verification by Reduction to Synthesis**
   - **Many (verification) tasks involve synthesis**
2. **Induction + Deduction + Structure: An Effective Approach to Synthesis:**
   - ***Induction*: Learning from examples**
   - ***Deduction*: Logical inference and constraint solving**
   - ***Structure*: Hypothesis on syntactic form of artifact to be synthesized**
   - **"Syntax-Guided Synthesis"** [Alur et al., FMCAD'13]
     - Inspired by Counterexample-guided abstraction refinement (CEGAR) [Clarke et al., CAV'00]
3. **Machine Learning Theory + Formal Methods**
   - **Analysis of Counterexample-Guided Synthesis**
     - **Sample Complexity, Convergence, Search Strategies**

Artifacts Synthesized in Verification

- Inductive invariants
- Abstraction functions / abstract models
- Auxiliary specifications (e.g., pre/post-conditions, function summaries)
- Environment assumptions / Interface specifications
- Interpolants
- Ranking functions
- Intermediate lemmas for compositional proofs
- Theory lemma instances in SMT solving
- Patterns for Quantifier Instantiation
- …

Formal Verification as Synthesis

- Inductive Invariants

- Abstraction Functions

One Reduction from Verification to Synthesis

NOTATION

Transition system $M = (I, \delta)$

Safety property $\Psi = G(\psi)$

VERIFICATION PROBLEM

Does M satisfy Ψ?

↓

SYNTHESIS PROBLEM

Synthesize ϕ s.t.

$$I \Rightarrow \phi \wedge \psi$$

$$\phi \wedge \psi \wedge \delta \Rightarrow \phi' \wedge \psi'$$

Two Reductions from Verification to Synthesis

NOTATION

Transition system $M = (I, \delta)$, S = set of states

Safety property $\Psi = G(\psi)$

VERIFICATION PROBLEM

Does M satisfy Ψ?

SYNTHESIS PROBLEM #1

Synthesize ϕ s.t.

$$I \Rightarrow \phi \wedge \psi$$

$$\phi \wedge \psi \wedge \delta \Rightarrow \phi' \wedge \psi'$$

SYNTHESIS PROBLEM #2

Synthesize $\alpha : S \rightarrow \hat{S}$ where

$$\alpha(M) = (\hat{I}, \hat{\delta})$$

s.t.

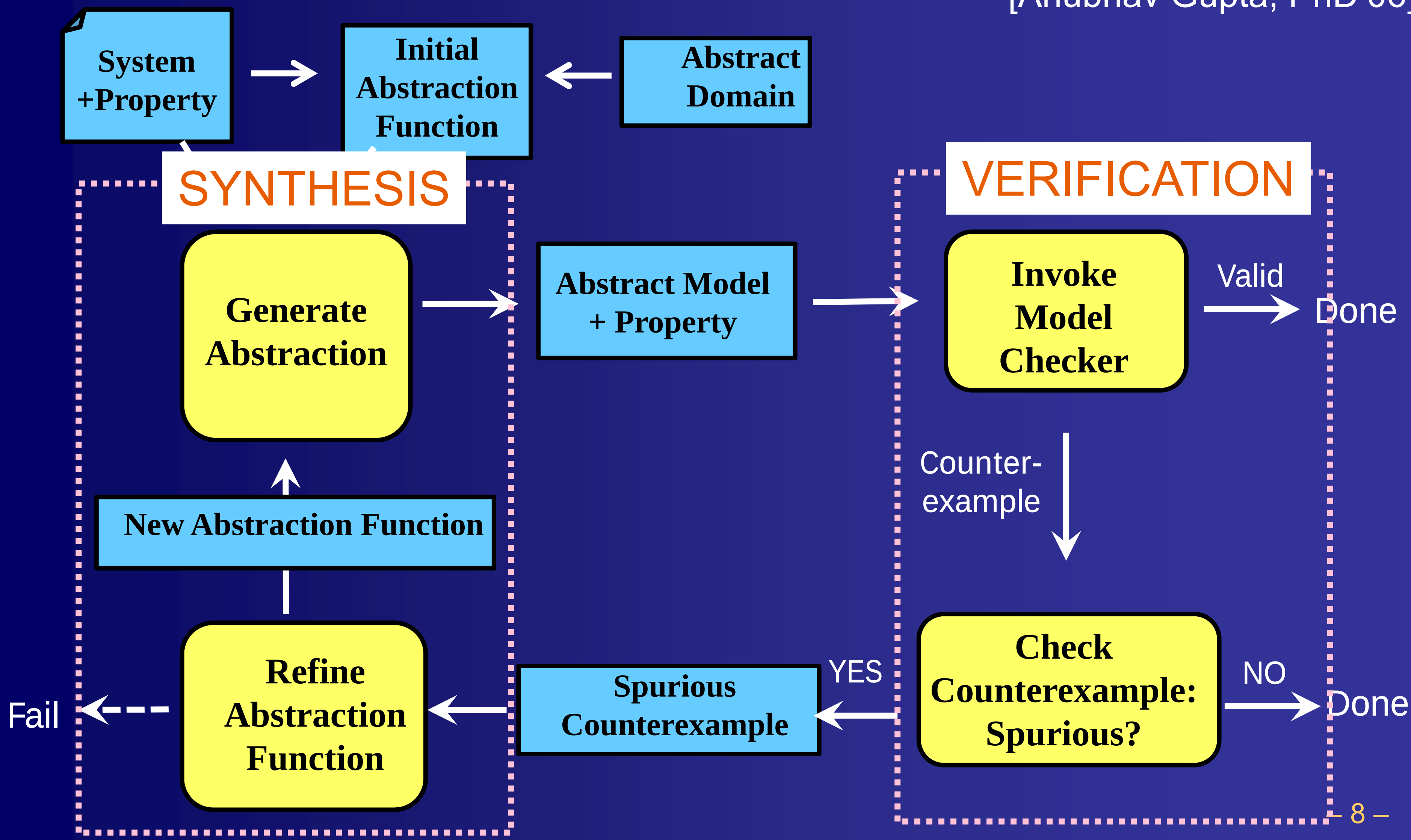
$\alpha(M)$ satisfies Ψ

iff

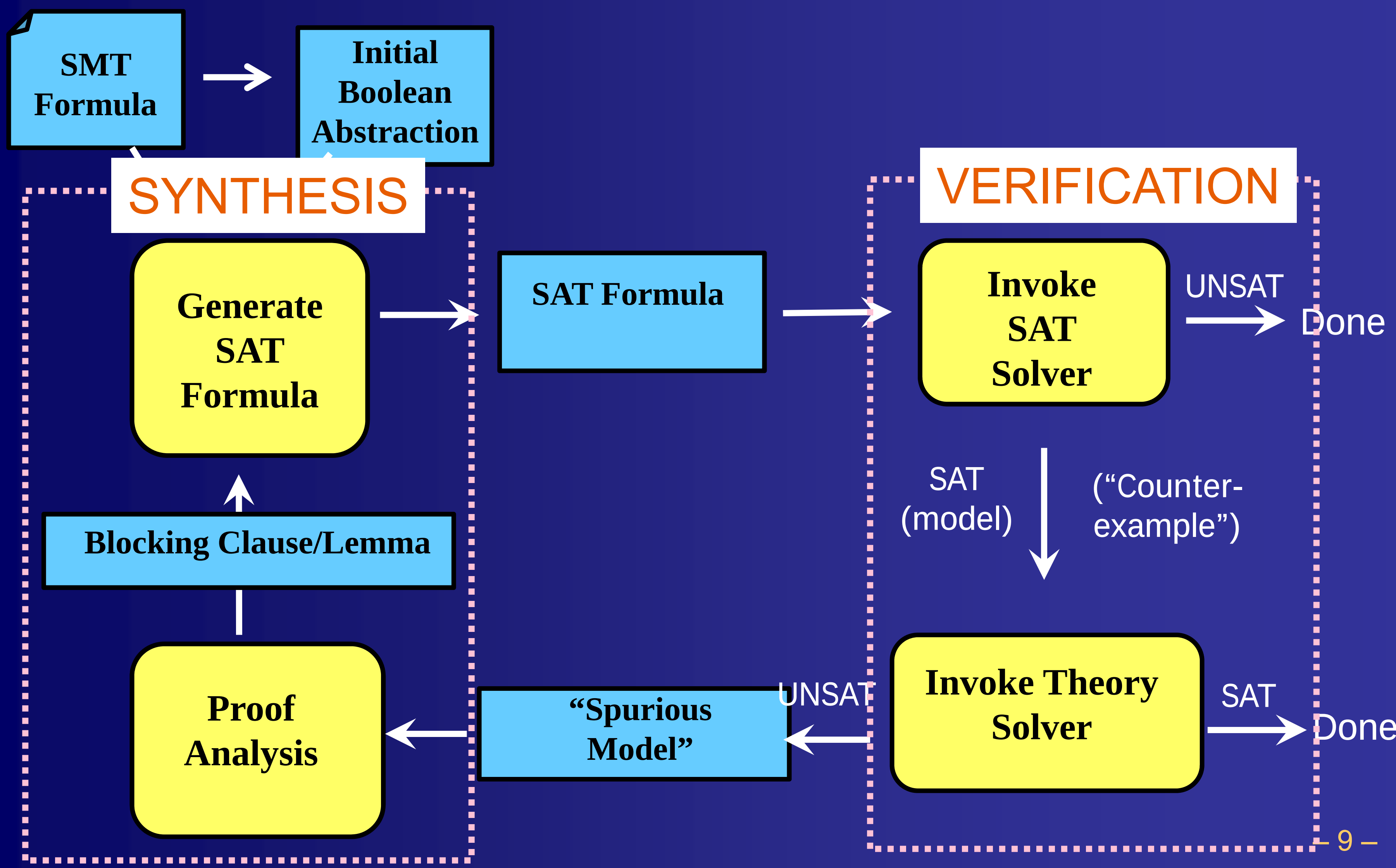
M satisfies Ψ

Counterexample-Guided Abstraction Refinement is "Inductive" Synthesis

[Anubhav Gupta, PhD'06]

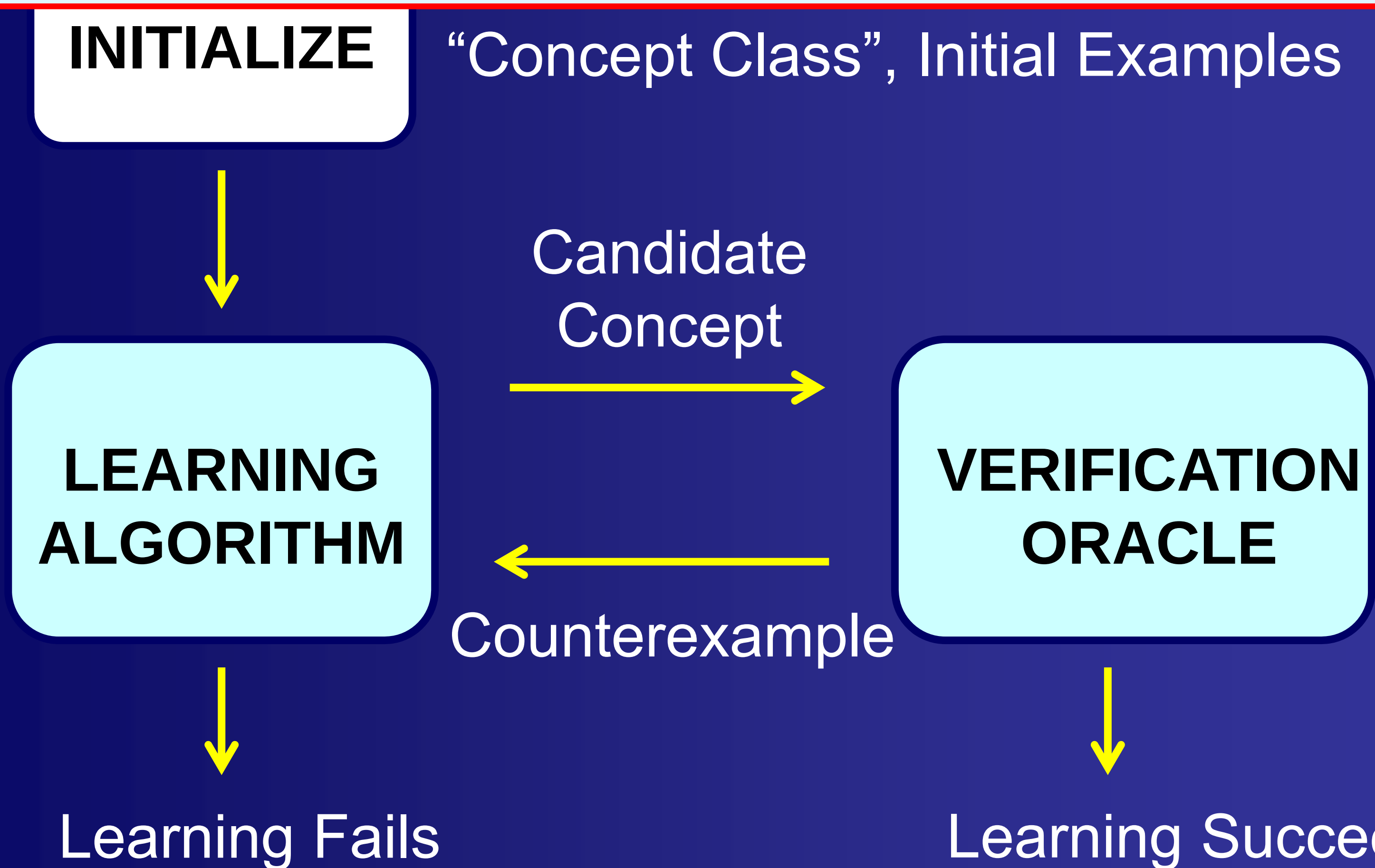


Lazy SMT Solving performs "Inductive" Synthesis (of Lemmas)



CEGAR & Lazy SMT perform Active Learning from (Counter)Examples

Difference from standard learning theory:
Learning Algorithm and Verification Oracle are typically "General" Solvers, independent of concept class



Machine Learning Theory ←→ Formal Methods: 2 Sample Results

- **Lower Bounds on Convergence of Counterexample-guided loop**
  - **Teaching Dimension (TD):** ***Minimum*** **number of (labeled) examples a teacher must reveal to** ***uniquely*** **identify any concept from a class**
  - **<u>Thm:</u> TD is a lower bound on # iterations for counterexample-guided synthesis**
- **Impact of "Quality" of Counterexamples**
  - **Does the type of counterexample affect convergence for** ***infinite-size*** **concept classes?**
  - **<u>Thm:</u> Minimum counterexamples are no better than arbitrary counterexamples**

[Jha & Seshia, SYNT'14]

Conclusion

- **Model Checking and Synthesis: many connections**
- **Verification by Reduction to Synthesis**
- **Approach for Synthesis: Induction + Deduction + Structure**
  - "Syntax-Guided Synthesis" [Alur et al., FMCAD'13]
  - Inspired by Counterexample-guided abstraction refinement (CEGAR) [Clarke et al., CAV'00]
- **Machine Learning Theory & Formal Methods: theoretical connections**
  - **Sample Complexity, Convergence, Search Strategies**