Sciduction: Combining Induction, Deduction and Structure for Verification and Synthesis

(abridged version of DAC slides)

Sanjit A. Seshia

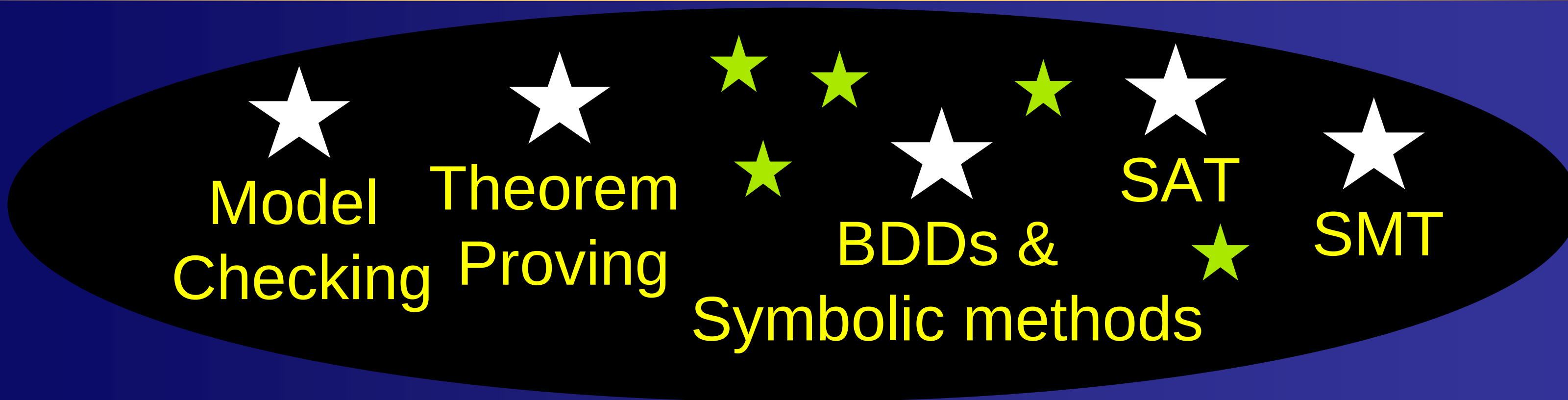
Associate Professor

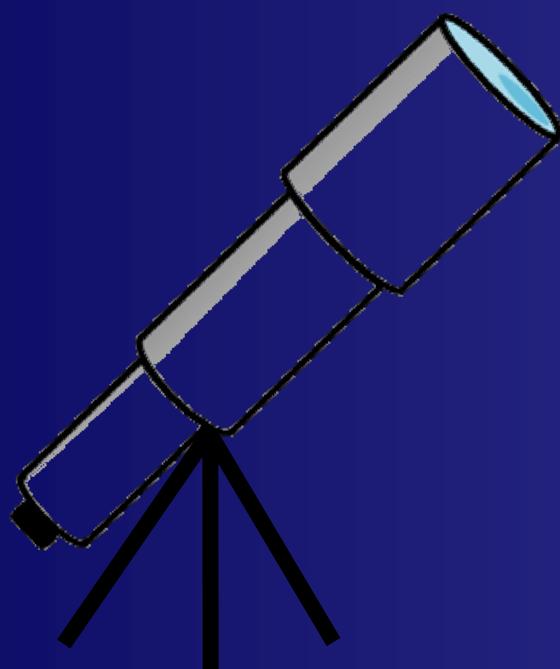
EECS Department

UC Berkeley

Design Automation Conference
June 5, 2012

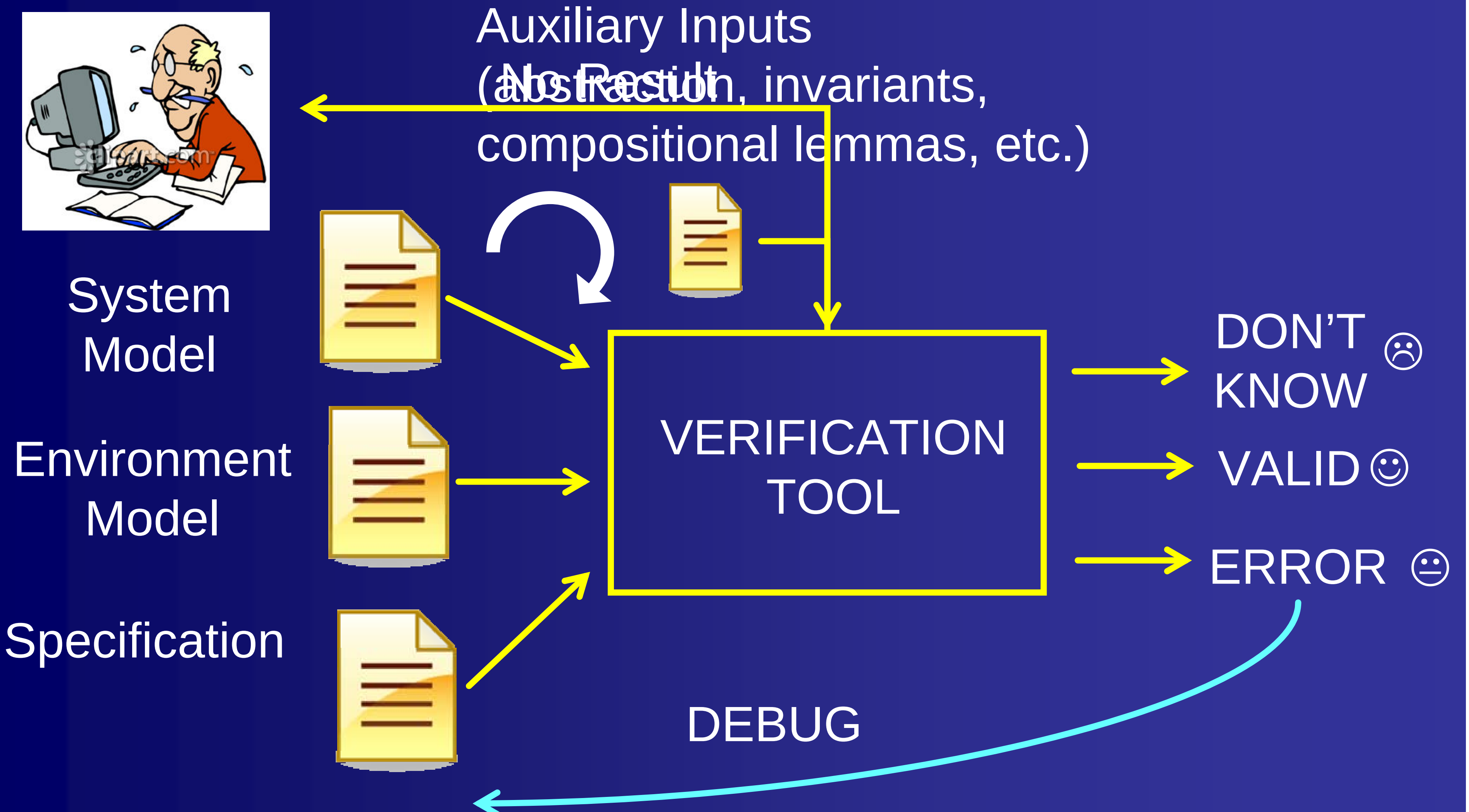
A Perspective on Formal Methods





What can we learn?
WHAT'S NEXT?

The Human Aspect



The End Goal

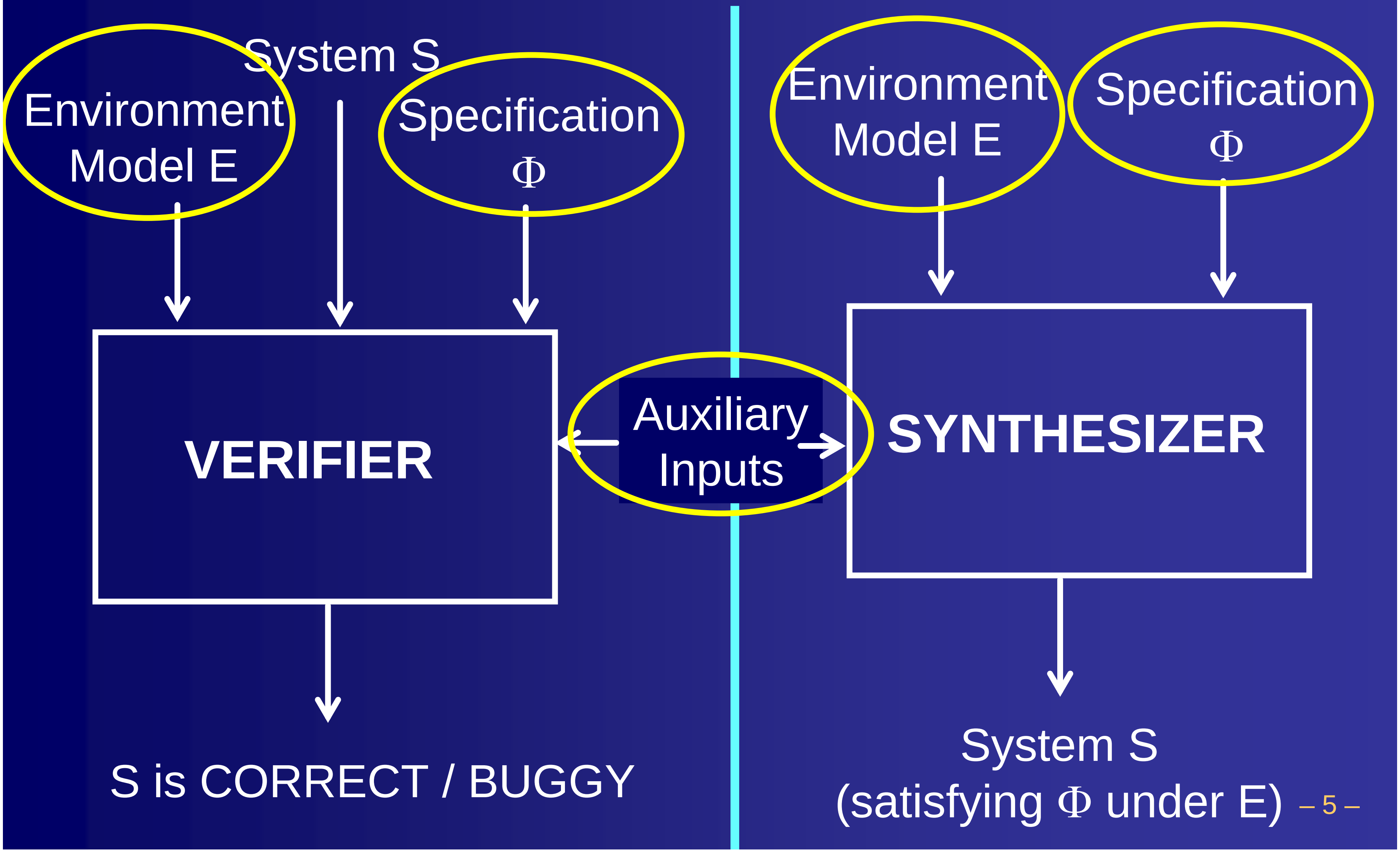
- **Improve designer / programmer creativity and productivity**
  - **Automate tedious tasks**
  - **Enable user to express creative insights**
  - **Correct-by-construction synthesis (from high-level spec.)**

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 original model checking paper)

Verification and Synthesis: Where Do We Spend Time?



Artifacts Synthesized in Verification

- Inductive / auxiliary invariants
- Auxiliary specifications (e.g., pre/post-conditions, function summaries)
- Environment assumptions / E
  interface specification
- Abstraction fu odels
- Interpo
- g
- ate lemmas for compositional
  oning
- Theory lemma instances in SMT solving
- …

EVERYTHING IS A SYNTHESIS PROBLEM!

Perspectives, so far…

- **Verification "=" Synthesis**
  - **The hard parts of verification involve "synthesis sub-tasks"**

- **3 Challenges; Human input is crucial**
  - **Writing specifications**
  - **Modeling environment**
  - **Guiding verification/synthesis engine**

- **How to help users provide creative input while automating tedious tasks?**

The Lens: Examining Human-Computer Interaction in Verification

- User identifies synthesis sub-task
  - "Generate abstract model"

  and expresses creative insight
  - "Use localization abstraction"

STRUCTURE HYPOTHESIS

- Tool automates search
  - Counterexample-guided abstraction refinement (CEGAR) using DPLL-based SAT solving

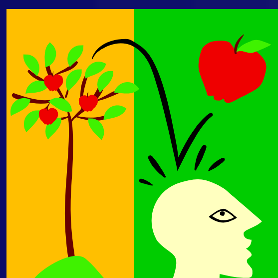
DEDUCTION: General to specific

+

INDUCTION: Specific to general

Sciduction

Structure-Constrained Induction and Deduction



Inductive Reasoning
(**Active Learning**: Generalizing from Examples)

+

Deductive Reasoning
(“Lightweight” **Logical** inference &
Constraint solving)

+

Structure Hypotheses
(on artifacts to be synthesized)

Demonstrated Applications

Floating-point
to fixed-point

Switching logic
synthesis

Structure Hypothesis
+
Inductive Inference
+
Deductive Reasoning

Timing analysis
of software

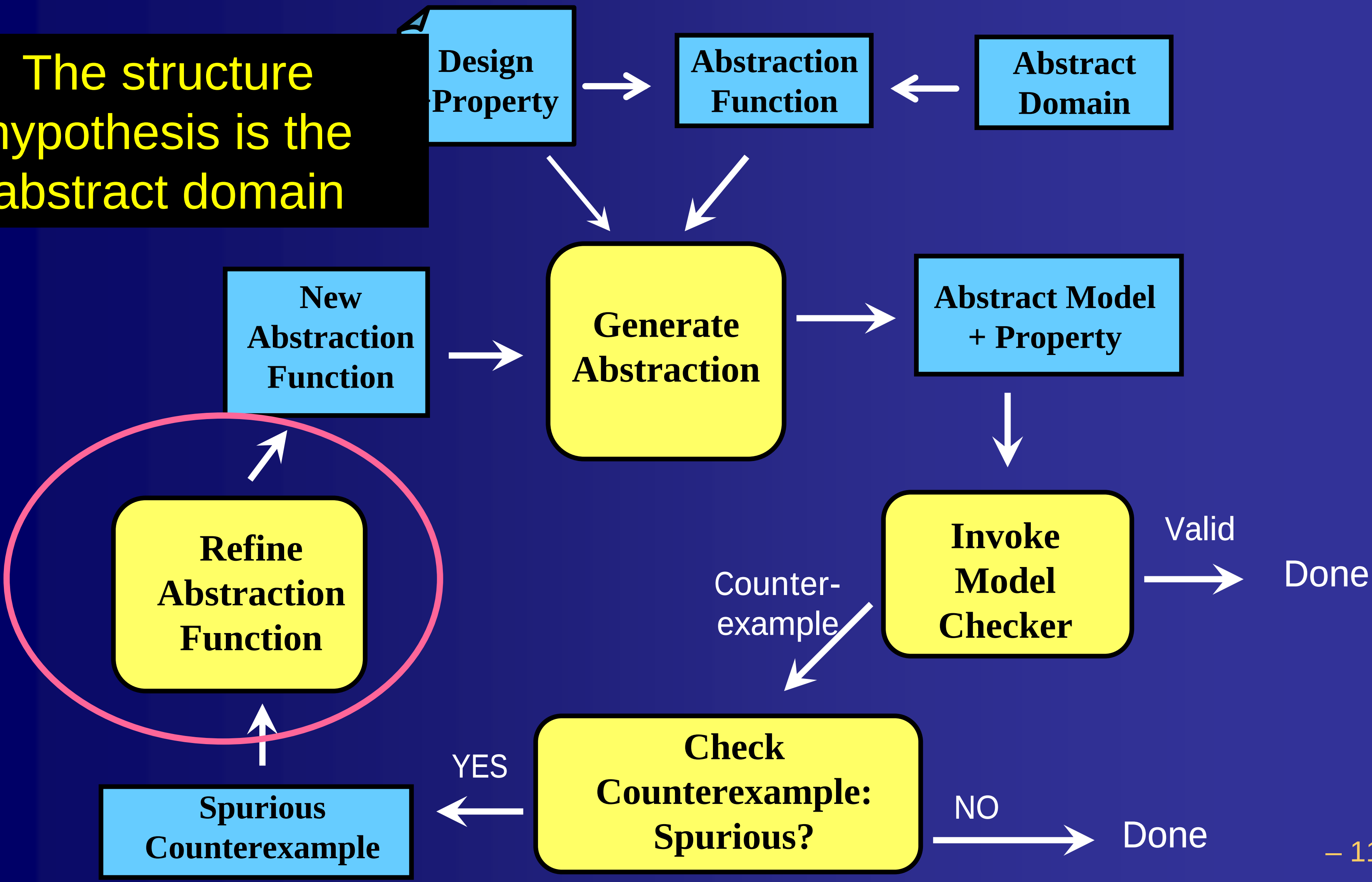
RTL
verification

Program
synthesis

Synthesis from
temporal logic

Counterexample-guided Abstraction Refinement involves Synthesis

The structure hypothesis is the abstract domain



Approach

Identify the synthesis sub-task(s)

↓

Make structure hypothesis

↓

Devise top-level synthesis strategy (inductive or deductive)

↓

Devise subroutines (inductive or deductive)

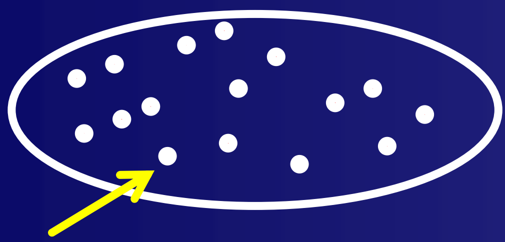
Synth. Task

Structure Hypothesis

Top-Level Strategy

Subroutines

Synthesis Sub-Task



C_S

- **Find artifact satisfying specification Ψ**

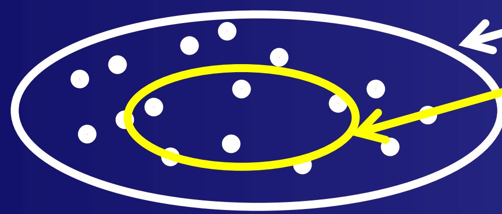
CEGAR

- **C_S = All (finite-state) abstract models**
- **Ψ = Abstract model must be**
  - **sound (over-approximate)**
  - **complete (no spurious counterexamples)**

Structure Hypothesis H

Synth. Task → **Structure Hypothesis** → Top-Level Strategy → Subroutines

- **Shrink set of artifacts from C_S to C_H**



CEGAR

- **H = The abstract domain (localization abstraction)**
- **C_H = Abstract models generated using H**

Top-Level Strategy

Synth. Task
↓
Structure Hypothesis
↓
Top-Level Strategy
↓
Subroutines

- **Top-level search strategy for:**

  $\exists\, c \in C_H$ s.t. c satisfies Ψ?

CEGAR

- **Learn from spurious counterexamples**
  – **most over-approximate model satisfying Ψ**
- **Soundness (trivial): by construction (over-approximation)**
- **Completeness: the original concrete system is in C_H**

Induction

Synth. Task

↓

Structure Hypothesis

↓

Top-Level Strategy

↓

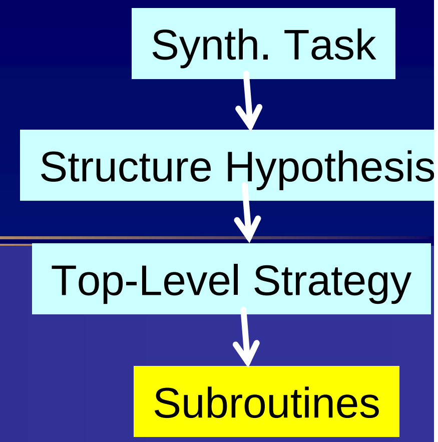
Subroutines

- **Learning algorithm**
  - **Active learning: choose examples to learn from**

CEGAR

- **Example: Spurious Counterexample**
- **Partially concretize abstract model to rule out spurious counterexample**
  - **CEGAR as Inductive Learning** [Anubhav Gupta, PhD thesis 2006]

Deduction

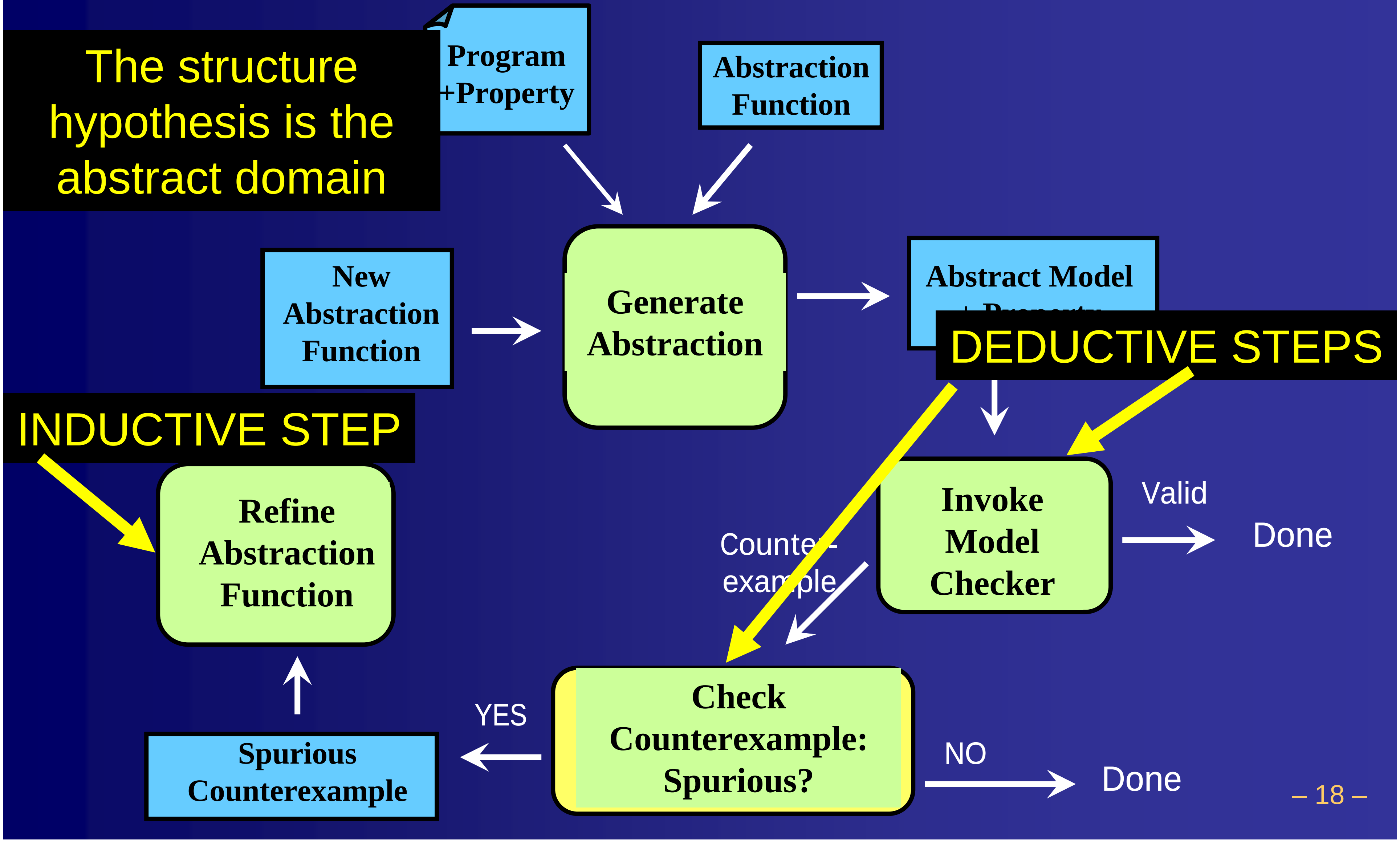


- **Lightweight decision procedure**
  - Solves decision problem that is “easier” than original
  - Generates examples, labels for examples, verifies artifact, etc.

CEGAR

- **Model checker**
  - Generates counterexample, if one exists
- **SAT solver**
  - Checks if counterexample is spurious

CEGAR and Sciduction



Related Work: A Sample

[see paper for details]

- **Instances of Sciduction (also inspiration!)**
  - **CEGAR [Clarke et al., ’00]**
  - **Compositional Reasoning, Invariant Generation based on Automata Learning (L*) [Cobleigh et al, ’03]**
  - **Counterexample-guided inductive synthesis (CEGIS) [Solar-Lezama et al., ’06]**

- **Purely Deductive Generalization**
  - **DPLL-based SAT solvers**
  - **Lazy SMT solvers -- DPLL(T)**
  - **Automata-theoretic synthesis from LTL**