EECS 219C: Computer-Aided Verification

Model Generation from Execution Traces

Sanjit A. Seshia

EECS, UC Berkeley

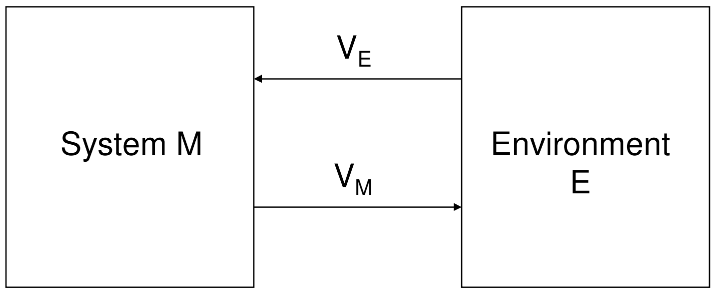
Acknowledgments: Avrim Blum

Today's Lecture

- Generating models of finite-state systems by observing execution traces
  - Based on a machine learning algorithm first proposed by D. Angluin in '87 and improved upon by Rivest & Schapire in '93
- Apr 4: Guest lecture by Anubhav Gupta on using learning in model checking

Setting

State variables $V = V_E \cup V_M$, $V_E \cap V_M = \phi$



Want to observe E and generate a good model of it
Usually easy to get a model of M

Why Learn Environment Models?

- As a middle ground between
  - Traditional, pessimistic (worst-case) verification
  - Optimistic verification ("does there exist an environment that makes my system work?")
- To generate *environment assumptions* for use in assume-guarantee reasoning
- To deal with *incorrect models* (of system modules or environment)
  - May miss behaviors and also include spurious behaviors

A Quote

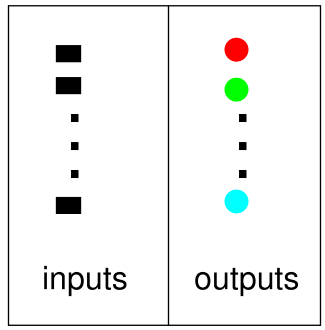
- “Assumptions are the things you don't know you're making”
  — Douglas Adams, Mark Cawardine,
  "Last Chance to See"

Learning Env. Model

- Model: (Deterministic) Finite Automaton
  – As a representation of the set of traces of env.
- What we can do:
  – Provide inputs to the environment
  – Observe (finite) prefixes of environment’s output trace
- Note:
  – Env. is a reactive system too, has infinitely long traces but we can only observe finite prefixes
  – So we are learning a finite automaton (not a Buchi automaton)

Another View

- Environment is a box, with input buttons and output lights
  – Outputs capture observable part of env state
- We can press some subset of input buttons at any time step
- Observe what lights turn on



Assumption for this lecture:
We can "reset" the environment at any time

Angluin's DFA Learning Algo.

(adapted to our setting)

- Input: A box as in the previous picture
  - inputs from an alphabet Σ
- Outputs: a DFA that accurately represents all (finite) output traces seen so far
- What it can do:
  – Generate environment traces by supplying inputs
  – Ask an oracle whether a candidate DFA is indeed correct (if not, get a counterexample)
  – Reset environment model to initial state

Angluin's DFA Learning Algo.

(adapted to our setting)

- Input: A box as in the previous picture
  - inputs from an alphabet Σ
- Outputs: a DFA that accurately represents all (finite) output traces seen so far
  - Given an oracle that precisely knows the environment, it learns the DFA representing exactly the output traces of the env.
- What it can do:
  - Generate environment traces by supplying inputs
  - Ask an oracle whether a candidate DFA is indeed correct (if not, get a counterexample)
  - Reset environment model to initial state

Formal Setup

- Want to learn (synthesize) a DFA (Q, Σ, δ, L)
  - Q : set of states
  - Σ : input alphabet
  - δ : transition function: Q x $\Sigma \rightarrow$ Q
  - L : labeling/output function
- What does it mean for two states of the DFA to be different?

  (In terms of the labels we observe)

Formal Setup

- Want to learn a DFA (Q, Σ, δ, L)
  - Q : set of states
  - Σ : input alphabet
  - δ : transition function: Q x Σ → Q
  - L : labeling/output function
- What does it mean for two states of the DFA to be different?
  - q and q' are different if there is a input sequence s.t. the states reachable on that sequence from q and q' respectively have different labels

S. A. Seshia 11

What defines a state

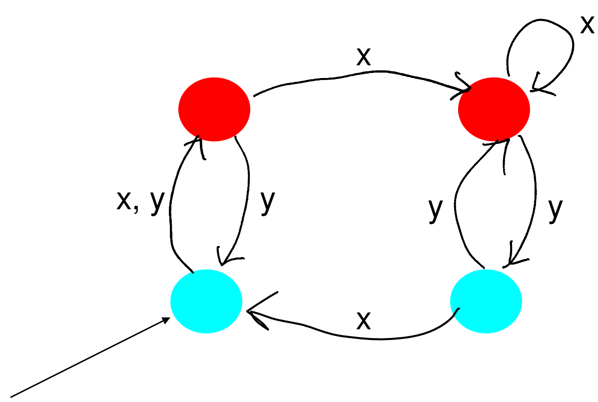
- Its label (observable part)
- What input sequence gets us to that state
  - Could be many, pick a representative
- What output sequence we see from that state
  - Perform "experiments" from that state to see this
- Angluin's algorithm "names" a state by the latter two things
  - A prefix and a suffix

S. A. Seshia 12

Algorithm Sketch

1. Start with only the DFA's initial state q_0
2. Generate a "new" state by supplying inputs
3. Check if its next states are observationally different from those of existing states
   - If yes, add it in
   - If not, ask the oracle if we have the correct DFA
     - If yes, we're done
     - If not, use the counterexample to figure out what new state(s) to add until that counterex goes away
   - Go back to step 2

An Example



This is the DFA we want to learn
(the correct environment model)

How the algorithm works on the previous example – worked out on board

Complexity

- Polynomial in size of environment model
- Good if environment model is small
  - This is why it is especially good for learning assumptions or concise env specifications

Some Refs. to Applications

- "Adaptive Model Checking" -- Groce, Peled, Yannakakis, TACAS'02
- "Learning Assumptions for Compositional Verification" -- Cobleigh et al., TACAS'03

Next: Part III of the course

- Next week: Decision procedures for fragments of first-order logic
  - Equivalent of Part I lectures on "SAT solving"
- After spring break:
  - Guest lecture
  - Your presentations