Introduction to Models and Properties

Computer Science and Artificial Intelligence Laboratory
MIT
Armando Solar-Lezama

Nov 23, 2015

Recap

	Type Inference	Symbolic Execution	Abstract Interpretation	Model Checking
Properties	Properties of variables	Properties at program points	Properties at program points	Properties of execution traces
Flexible	No	Yes	No	Yes
Push-button	Yes	No	Yes	Yes

Model Checking Today

Hardware Model Checking

- part of the standard toolkit for hardware design
  - Intel has used it for production chips since Pentium 4
  - For the Intel Core i7, most pre-silicon validation was done through formal methods (i.e. Model Checking + Theorem Proving)
- many commercial products
  - IBM RuleBase, Synopsys Magellan, …

Software Model Checking

- Static driver verifier now a commercial Microsoft product
- Java PathFinder used to verify code for mars rover

This doesn't mean Model Checking is a solved problem

- Far from it

Model Checking Genesis

The paper that started it all

- Clarke and Emerson, *Design and Synthesis of Synchronization Skeletons using branching time temporal logic*

"Proof Construction is Unnecessary in the case of finite state concurrent systems and can be replaced by a model-theoretic approach which will mechanically determine if the system meets a specification expressed in propositional temporal logic"

Intellectual Roots

Two important developments preceded this paper

- Verification through exhaustive exploration of finite state models
  - G. V. Bochmann and J. Gecsei, *A unified method for the specification and verification of protocols*, Proc. IFIP Congress 1977

- Development of Linear Temporal Logic and its application to specifying system properties
  - A. Pnueli, *The temporal semantics of concurrent programs*. 1977

Model Checking

The model checking approach

(as characterized by Emerson)

- Start with a program that defines a finite state graph M
- Search M for patterns that tell you whether a specification f holds
- Pattern specification is flexible
- The method is efficient in the sizes of M and hopefully also f
- The method is algorithmic

So what exactly is a model?

Remember our friend $\vdash$?

- What does this mean? $\vdash x \wedge y \Rightarrow x$
  - The statement above can be established through logical deduction
  - Axiomatic semantics and type theory are deductive
    - The program, together with the desired properties make a theorem
    - We use deduction to prove the theorem
- What about this; is it true? $\vdash x + y == 5$
  - We can not really establish this through deduction
  - We can say whether it's true or false under a given *model*

$$[x=3, y=2] \vDash x + y == 5$$

You have seen this symbol too $\vDash$

- In operational semantics, the variable assignments were the model
- The program behavior was the theorem we were trying to prove under a given model

Basic Notions of Model Theory

Consider the following sentence:

- S := The class today was awesome

Is this sentence true or false?

- that depends
  - What class is “the class”? What day is “today”?

We can give this sentence an Interpretation

- I := The class is 6.820, Today is Tuesday Nov 22

When an interpretation I makes S true we say that

- I satisfies S
- I is a model of S
- $I \models S$

The model checking problem

We are interested in deciding whether $I \models S$ for the special case where

- I is a Kripke structure
- S is a temporal logic formula

Today you get to learn what each of these things are

But the high level idea is:

- Unlike axiomatic semantics, where the program was part of the theorem,
- The program will now be the *model*
  - Well, not the program directly, but rather a kripke structure representing the program

Kripke Structures as Models

Kripke structure is a FSM with labels

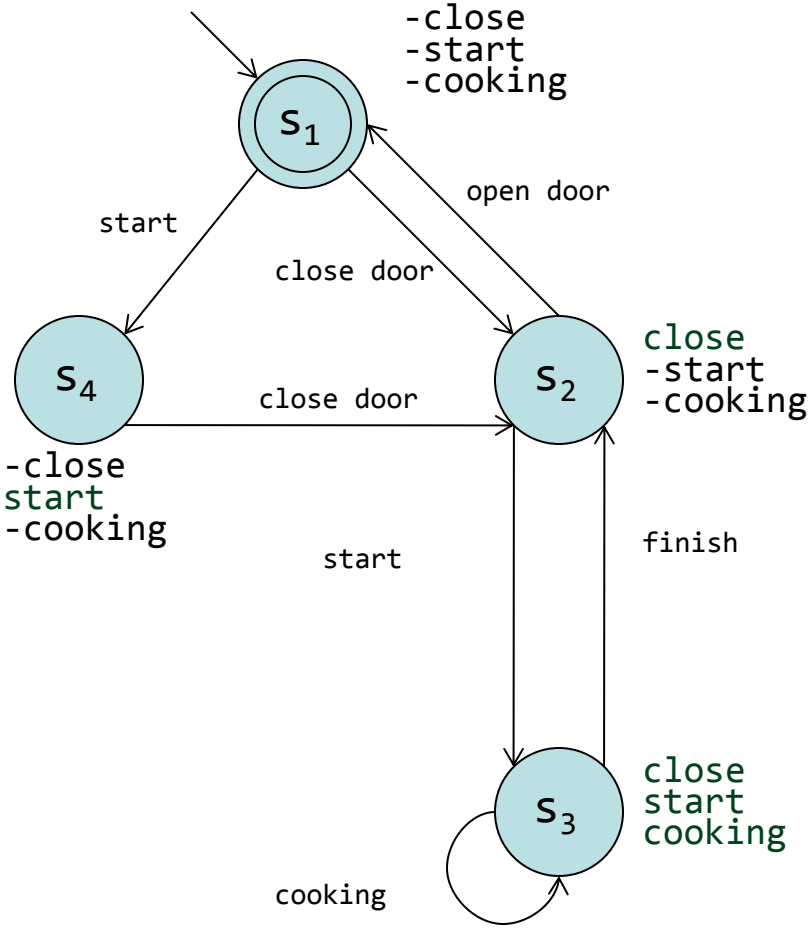
Kripke structure = (S, S0, R, L)

- S = finite set of states
- S0 $\subseteq$ S = set of initial states
- R $\subseteq$ S x S = transition relation
- L :S$\rightarrow$ 2^{AP} = labels each state with a set of atomic propositions

Microwave Example

- $S = \{s_1, s_2, s_3, s_4\}$
- $S0 = \{s_1\}$
- $R = \{ (s_1,s_2), (s_2,s_1), (s_1,s_4), (s_4,s_2), (s_2,s_3), (s_3,s_2), (s_3,s_3)\}$
- $L(s_1) = \{$-close, -start, -cooking$\}$
- $L(s_2) = \{$close, -start, -cooking$\}$
- $L(s_3) = \{$close, start, cooking$\}$
- $L(s_4) = \{$-close, start, -cooking$\}$

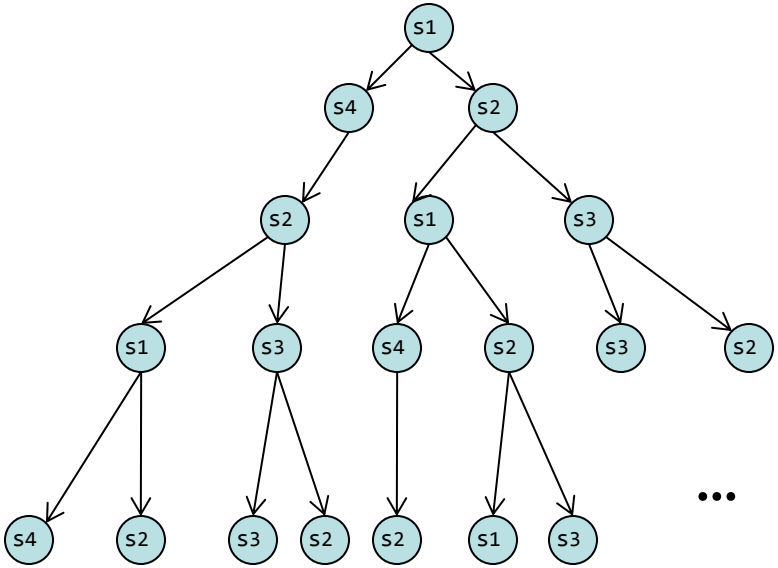
Can the microwave cook with the door open?

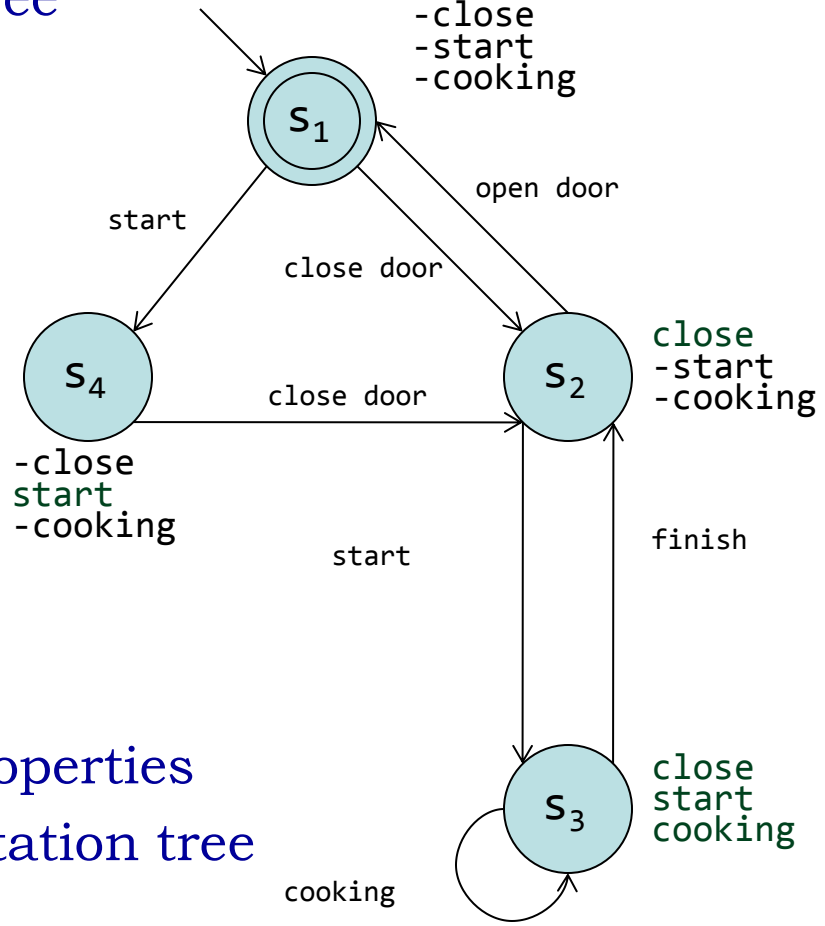


Kripke structures describe computations

A Kripke structure can describe an infinite process

- We can interpret it as an infinite tree





- We need a language to describe properties of paths down the computation tree

Linear Temporal Logic

Let π be a sequence of states in a path down the tree

- π := $s_0, s_1, s_2, \ldots$
- Let π_i be a subsequence starting at i

We are going to define a logic to describe properties over paths

Properties over states

State Formulas

- Can be established as true or false on a given state
- If p ϵ {AP} then p is a state formula
- if f and g are state formulas, so are (f and g), (not f), (f or g)
- Ex. (not closed and cooking)

For paths

Path formulas

- a state formula p is also a path formula
  - $p(\pi_i) := p(s_i)$
- boolean operations on path formulas are path formulas
  - f and $g(\pi_i)$:= $f(\pi_i)$ and $g(\pi_i)$
- path quantifiers
  - G f (π_i) := globally f (π_i) = forall k>= i f (π_k) (may abbreviate as $\Box$)
  - F f (π_i) := eventually f (π_i) = exists k>= i f (π_k) (may abbreviate as $\Diamond$)
  - X f (π_i) := next f (π_i) = f (π_{i+1}) (may abbreviate as $\circ$)
  - f U g (π_i) := f until g = exists k >= i s.t. g(π_k) and f(π_j) for i<=j<k

Given a forumula f and a path π,

- if f(π) is true, we say that $\pi \models f$

Examples

If you submit your homework (submit) you eventually get a grade back (grade)

- G (submit => F grade)

You should get your grade before you submit the next homework

- $G\ (submit \Rightarrow X\ (\neg submit\ U\ grade))$
  - What's wrong with $G\ (submit \Rightarrow\ (\neg submit\ U\ grade))$?

If assignment i was submitted before drop date, you should get your grade before drop date

- $\big(G(\ submit_i \Rightarrow F\ dropDate)\big) \Rightarrow \big(\big(G\ (\ grade_i \Rightarrow F\ dropDate)\big)\big)$
- and G (submit => F grade)

6.820 Fundamentals of Program Analysis

Fall 2015