

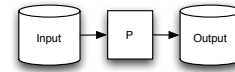
CMSC 631 – Program Analysis and Understanding Fall 2003

Model Checking
Slides from Adam Porter

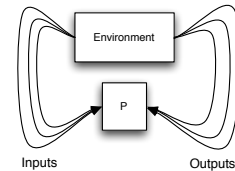
Reactive Systems

- Two kinds of systems:

- Transformational: Produce outputs from inputs



- Reactive: Infinite process, responding to environment
 - Model checking usually applied to reactive systems



CMSC 631, Fall 2003

2

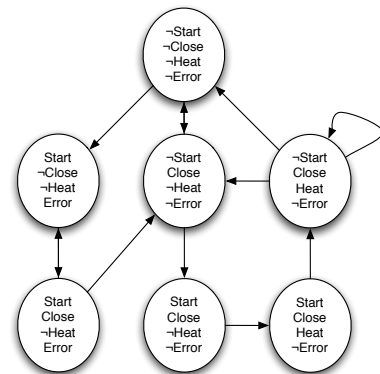
Program State

- The *state* of a program is the values of its variables at some point in time
- System state changes according to inputs
 - Takes *transitions* from one state to another

CMSC 631, Fall 2003

3

Example: Microwave Oven



CMSC 631, Fall 2003

4

Properties

- We would like to verify the following properties (among others):
 - The oven doesn't heat up until the door is closed
 - If the oven starts, it will eventually start cooking
 - It must be possible to correct errors

CMSC 631, Fall 2003

5

Temporal Operators

- Specifications in *temporal logic*
- Includes standard logical connectives
 - \wedge, \vee, \neg
- Plus path quantifiers, basic temporal operators
 - E (exists a path from here), A (for all paths from here)
 - $Fp - p$ holds sometime in the future
 - $Gp - p$ holds globally (always) in the future
 - $Xp - p$ holds next time
 - $pUq - p$ holds until q holds

CMSC 631, Fall 2003

6

Properties

- We would like to verify the following properties (among others):
 - The oven doesn't heat up until the door is closed
 - $(\neg \text{Heat}) \text{ U Closed}$
 - If the oven starts, it will eventually start cooking
 - $\text{AG}(\text{Start} \Rightarrow \text{AF Heat})$
 - It must be possible to correct errors
 - $\text{AG}(\text{Error} \Rightarrow \text{AF } \neg \text{Error})$

The Model Checking Problem

- Let M be a state transition graph
 - A.k.a. a Kripke structure
- Let f be the specification in temporal logic
- Find all states s of M such that $M, s \models f$

A Model Checking Algorithm

- Goal: For each state s , compute
 - $\text{lab}(s) = \{ \text{formulas true in } s \}$
- When algorithm terminates
 - $M, s \models f$ iff $f \in \text{lab}(s)$
- Algorithm: Iterate over subformulas of f inside-out, computing $\text{lab}(s)$

Checking Subformulas

- Lemma: Any CTL (see paper) formula can be expressed in terms of $\neg, \vee, \text{EX}, \text{EU}$, and EG
- Therefore, six cases:
 - Atomic proposition p
 - If p is true in s , then add p to $\text{lab}(s)$
 - $\neg f$ – If $f \notin \text{lab}(s)$, add $\neg f$ to $\text{lab}(s)$
 - $f_1 \vee f_2$ – If $f_1 \in \text{lab}(s)$ or $f_2 \in \text{lab}(s)$, add $f_1 \vee f_2$ to $\text{lab}(s)$
 - $\text{EX } f$ – If there exists a successor s' of s such that $f \in \text{lab}(s')$, add $\text{EX } f$ to $\text{lab}(s)$

Checking Subformulas (cont'd)

- $\text{E}[f_1 \text{ U } f_2]$
 - Find all states s for which $f_2 \in \text{lab}(s)$
 - Follow paths backward from s , finding all states that can reach s on a path in which every state is labeled with f_1
 - Label each of these states with $\text{E}[f_1 \text{ U } f_2]$

Checking Subformulas (cont'd)

- $\text{EG } f$
 - Idea: Look for an infinite path on which f holds
 - Divide M into nontrivial strongly-connected components
 - A strongly-connected component (SCC) C is
 - a maximal subgraph such that every node in C is reachable from everyone other node in C on a directed path contained entirely within C
 - C is nontrivial if either it has more than one node or it contains a node with a self loop
 - Compute M' from M by removing all states s in which $f \notin \text{lab}(s)$

Checking Subformulas (cont'd)

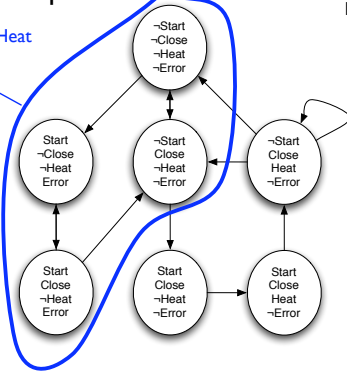
- Lemma: $M, s \models EG f$ iff
 - $s \in M'$
 - There exists a path in M' that leads from s to some node t in a nontrivial SCC
 - Idea of proof:
 - Need cycle to have infinite path (assuming finite state system)
 - So we need to find a path from s to a cycle on which f holds in every state
 - Then we've found an infinite path on which f holds

Example

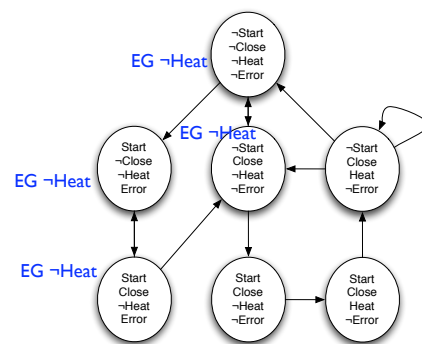
- Let's check $AG(Start \Rightarrow AF Heat)$
 - Rewrite to $\neg EF(Start \wedge EG \neg Heat)$
 - Rewrite to $\neg E[true U (Start \wedge EG \neg Heat)]$
- Compute labels for smallest subformulas
 - $Start, Heat, \neg Heat$
 - Just read these off the states

Example (cont'd)

- Next compute $EG \neg Heat$
- SCC where $\neg Heat$ holds
- No other state can reach this SCC

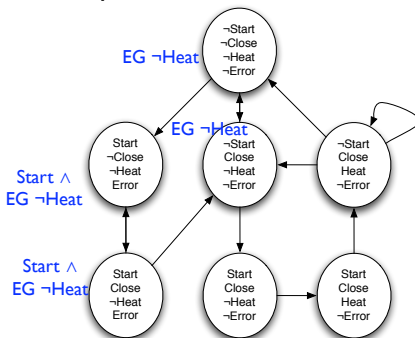


Example (cont'd)



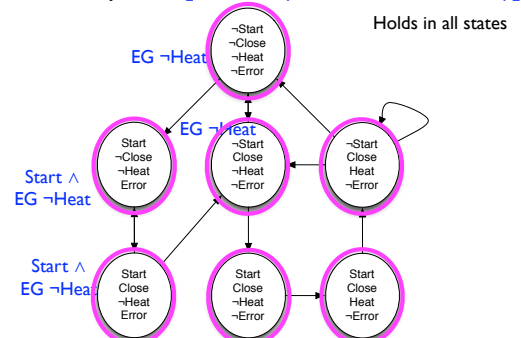
Example (cont'd)

- Next compute $Start \wedge EG \neg Heat$



Example (cont'd)

- Next compute $E[true U (Start \wedge EG \neg Heat)]$



Example (cont'd)

- Now compute $\neg E[\text{true } U (\text{Start} \wedge EG \neg \text{Heat})]$
 - All states satisfy $E[\text{true } U (\text{Start} \wedge EG \neg \text{Heat})]$
 - So no states satisfy its negation
 - So our safety property doesn't hold!

Features of Model Checking

- Advantages
 - Completely automatic
 - No proofs
 - Fast in practice
 - Generates counter-examples
 - Handles concurrency
- Disadvantages
 - State explosion problem
 - No dynamic allocation (heap), or recursion