Attacking Complexity with Models

Features of a good Model

➢ A good model highlights what’s important
  • Elide secondary details

➢ A good model is unambiguous
  • It should be clear what parts of the behavior the model is attempting to describe
  • Model should provide concrete answers to concrete questions
Motivating Example

**Afghanistan, December 2001**

- US soldier uses PLGR* to mark Taliban position for air-strike
- notices battery-low warning, so replaces battery and calls in coordinates
- resulting strike kills user and two comrades and wounds 20 others

**What happened?**

- replacing battery reset to current position
- Was this a bug?
Failures in Complex Systems

accidents are complex

- rarely one cause, so be wary of simple explanations
- often human factors + technology

lessons from this case

- design for robustness against all likely failure modes
- defaults are dangerous: user should be warned
- describe and analyze all usage scenarios
Modeling With State Machines

- NOCOORD
- MYCOORD
- YOURCOORD

Transitions:
- locate: from NOCOORD to MYCOORD
- reset: from MYCOORD to YOURCOORD
- adjust: from YOURCOORD to MYCOORD
Behavior vs. Mechanism

The models focus on what the system should do
- design the behavior first
- then design the mechanism

The gap between behavior and mechanism is not trivial
- But many tools and methodologies exist to help you bridge it
Anatomy of a State Machine
Anatomy of a State Machine

\[ \langle S, S_0, E, T \rangle \]
Anatomy of a State Machine

\[ \langle S, S_0, E, T \rangle \]
Anatomy of a State Machine

\[ \langle S, S_0, E, T \rangle \]
Anatomy of a State Machine

Transition Relation

\[ \langle S, S_0, E, T \rangle \]

© 6.005 Staff 2007-09
State Machine as Set of Traces

\[
\langle \text{NC, MYC, YOURC, NC, ...} \rangle \quad \checkmark
\]

\[
\langle \text{NC, MYC, YOURC, MYC, YOURC, MYC, YOURC} \rangle \quad \times
\]
Car Tunnel Problem

One lane in the tunnel is closed
Operators on either end can not see each other
You must avoid collisions
Idea: **Use a Flag**
Car Tunnel Problem

Flag starts on one side
You can only let a car go if you have the flag
You give the flag to the car to take to the other side

Does this strategy work?

© 6.005 Course Staff 07-10
Modeling with Finite State Machines

**Events**

- ArriveLeft
- LetGoLeft
- ReachLeft
- ArriveRight
- LetGoRight
- ReachRight
Modeling with Finite State Machines

States

Right Queue

Tunnel

Left Queue

Flag
Modeling with Finite State Machines

States
- Right Queue
  - LCars
  - Lnot
- Left Queue
  - RCars
  - Rnot

Tunnel
- empty
- ltravel
- rtravel
- crash

Flag
- left
- middle
- right
Modeling with Finite State Machines

Transitions

ArriveLeft (AL)
LetGoLeft (GL)
ReachLeft (RL)

ArriveRight (AR)
LetGoRight (GR)
ReachRight (RR)
Does this strategy work?
state machine properties

what can we ask about a state machine?

➢ safety: does it do anything bad?
  • do cars crash in the middle?
➢ liveness: does it do anything good?
  • do cars ever get to go?

in practice, liveness rarely useful

➢ “eventually” isn’t good enough
➢ “happens before midnight” is a safety property (“no chime before op”)

how to formulate safety?

➢ abstractly, every trace satisfies a property
➢ concretely, every reachable state satisfies a property
Safety Properties

Can cars ever crash?

Can cars ever crash?
Liveness Properties

Could cars be left to wait forever?
Yes they could!
product machine

two machines can form a single product machine
- states are tuples
- one state from each machine

“state explosion”
- $k$ machines of $N$ states
- product machine has $N^k$ states

Ex:
Product Machine

- **ArriveLeft (AL)**
- **ArriveRight (AR)**
- **LetGoLeft (GL)**
- **LetGoRight (GR)**
- **ReachLeft (RL)**
- **ReachRight (RR)**

Diagram showing states and transitions such as:
- Lcars, left, Rcars, empty
- Lnot, left, Rcars, empty
- Lcars, left, Rnot, empty
- Lnot, left, Rnot, empty
- Lcars, mid, Rcars, empty
- Lnot, mid, Rcars, empty
- Lcars, mid, Rnot, empty
- Lnot, mid, Rnot, empty
- Lcars, right, Rcars, empty
- Lnot, right, Rcars, empty
- Lcars, right, Rnot, empty
- Lnot, right, Rnot, empty
- Lcars, left, Rcars, ltravel
- Lnot, left, Rcars, ltravel
- Lcars, left, Rnot, ltravel
- Lnot, left, Rnot, ltravel
- Lcars, mid, Rcars, ltravel
- Lnot, mid, Rcars, ltravel
- Lcars, mid, Rnot, ltravel
- Lnot, mid, Rnot, ltravel
- Lcars, right, Rcars, ltravel
- Lnot, right, Rcars, ltravel
- Lcars, right, Rnot, ltravel
- Lnot, right, Rnot, ltravel
- Lcars, left, Rcars, rtravel
- Lnot, left, Rcars, rtravel
- Lcars, left, Rnot, rtravel
- Lnot, left, Rnot, rtravel
- Lcars, mid, Rcars, rtravel
- Lnot, mid, Rcars, rtravel
- Lcars, mid, Rnot, rtravel
- Lnot, mid, Rnot, rtravel
- Lcars, right, Rcars, rtravel
- Lnot, right, Rcars, rtravel
- Lcars, right, Rnot, rtravel
- Lnot, right, Rnot, rtravel

States include:
- crash
- Err

© 6.005 Course Staff 07-10
MODELING OBJECTS WITH STATE MACHINES

© 6.005 Course Staff 07-10
List<String> l = ...;
for (String s : l) {
    System.out.println(s);
}

List<String> l = ...;
Iterator iter = l.iterator();
while (l.hasNext()) {
    String s = iter.next();
    System.out.println(s);
}
Infinite State Machine

- $i = 0$
- $i = 1$
- $i = n-1$
- $i = n$

Next

HasNext

© 6.005 Staff 2007-09
Finite State Machine