Analysis of Software Artifacts

SMV continued (Lecture 5)
Discuss additional capabilities of SMV

Homeworks 2 and 3 will be based on this example

A small vending machine

Discuss another example

Agenda
Vending machine

takes two coins to buy a beverage

one vending machine

one user
- sprite, mountain-dew
- none, coke, diet-coke
  choice ●
- making-choice, waiting
- initial, one-coin, two-coins
  state ●

User (Variables)
Vending machine (Variables)
choice = none;

initial value for the variable choice

state = initial;

initial value for the variable state


User (Initial values)
initial \* one-coin \{ state = initial & (vending-machine-state = initial) \\ using non-determinism \\ in initial or transition to one-coin \\ vending machine is initial state \\ if state is initial and state of the \\ User (Transitions)
See the transitions for the variable choice

When state is making-choice the user makes choice

Indicates that the user has deposited the second coin

If state is one-coin move to two-coins

User (Transitions)
beverages has been dispensed •

go back to none when •

when state = making-choice •

make a non-deterministic choice between beverages •

see the state variable choice •

User (Transitions)
as parameters •

user-state and choice passed •

initial •

initial value of state is •

Vending machine (initial)
Go to state dispense after get-choice

If the user-state is one-choice

change state from initial to one-choice

can you locate them in the code?

only few transitions

Vending machine (Transitions)
Define

DisPense

in module vending-machine define a macro

Define

DisPense

in SML you can define macros using the keyword

Macros
module is like a type definition
create "real" state machines
instantiate the module user
instantiate the module vending-machine

Instantiating
machine.state

machine: user (machine.DISPENSED)

machine: user (machine.choice)

machine: vending-machine (user.state)

VAR

MODULE main

Instantiating
always eventually vending machine state is dispensed

 vending machine state is initial, then

 if the user state is one-cotn and

 Specification
(⇒ AP(machine.state = dispense)
   (machine.state = initial)
   & (seen-user.state = one-coin))

SPEC

Specification
Show me a trace when transaction T is finished.

For example:

- Behavior or trace from a spec
- Sometimes you want to demonstrate a certain

Enumerating behaviors
• assert that \( \text{machine} \) is never true

\( \text{machine} \). DISENSED

• negation of the property is
diseneses

e numerate a trace where the vending machine

Enumerating behaviors
The negation of the property

$\neg \text{dispensed} \implies \text{trace where the vending machine eventually so the counterexample to the previous spec is a}\$ 

$\neg \text{EF(machixe\_dispensed)}$