OS Safety Verification

Safety and Correctness of an Embedded Operating System

Graduate Course CS5531- Advanced Operating Systems

Instructor – Rafida Zaman

Group – 08

- Aditya Nidugondi - 16322609
- Sai Sathvik Korrapati - 16345436
- Hrithik Reddy Janga - 16345458
- Mahesh Reddy Karepalli - 16345410

Introduction

- Traffic light control systems are electronic devices used to manage vehicular and pedestrian traffic at intersections by indicating when to stop and when to proceed. They follow a systematic timing mechanism to ensure orderly movement across roads. Modern systems are often intelligent, capable of adapting to real-time traffic conditions to optimize flow. The verification of these systems is crucial to prevent accidents and ensure safety.
- Verification of traffic light systems is essential to ensure they operate without faults, maintaining safety and preventing accidents at intersections. It helps in detecting and correcting errors before deployment, which can otherwise lead to traffic jams or collisions. Verification also ensures that the traffic lights adhere to specified timing and logic requirements. Ultimately, it contributes to the reliability and efficiency of traffic management in urban areas.
- NuSMV is a symbolic model checker used for the formal verification of finite state systems, employing advanced algorithms to check properties of a system's design. It allows for rigorous and automated checking of system specifications against desired behaviors.

NuSMV Overview

- NuSMV is a symbolic model checker used for the formal verification of finite state systems, employing advanced algorithms to check properties of a system's design. It allows for rigorous and automated checking of system specifications against desired behaviors.
- Key features of NuSMV :
- **Symbolic Model Checking**: NuSMV implements symbolic model checking algorithms, enabling the efficient verification of systems with large state spaces.
- **Temporal Logics Support**: It supports temporal logics like CTL (Computation Tree Logic) and LTL (Linear Temporal Logic), which are used to express properties of systems over time.
- **Modular Design**: The tool has a modular design that allows for the addition of new functionalities and the integration with other verification tools and frameworks.
- **Counterexamples**: In case of property violations, NuSMV provides counterexamples, which are essential for debugging and understanding why a system does not meet its specifications.

•Why NuSMV is suitable for traffic light system verification:
•NuSMV is suitable for traffic light system verification due to its capability to handle complex state spaces efficiently and its support for temporal logics, which are crucial for specifying the time-dependent behaviors of traffic lights.

```
C:\Program Files\NuSMV-2.6.0-win64\bin>tarffic-lights0.smv

C:\Program Files\NuSMV-2.6.0-win64\bin>NuSMV -int
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:37:51 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson
```

Formal Verification of the Operating System

- Formal verification of an operating system is a mathematical process used to prove its algorithms are correct and adhere to specifications. It abstracts the system into a model then uses theorem proving or model checking to validate properties like safety and security. This technique is crucial for ensuring reliability in complex and critical software environments, identifying bugs and vulnerabilities beyond the scope of conventional testing. It is particularly valuable in safety-critical applications where failure can have severe consequences.
- The 3 Ways for Formal Verification of the Operating System are:

1. Model Checking
2. Static Analysis
3. Dynamic Analysis

Advantages of Safety Verification

1. **Enhanced Reliability**: Safety verification ensures that systems operate without failure, increasing overall reliability.
2. **Prevention of Accidents**: By verifying safety aspects, potential hazards can be identified and mitigated before they cause accidents.
3. **Compliance with Standards**: It ensures systems comply with relevant safety standards and regulations, which is often legally required.
4. **Cost Savings**: Identifying and addressing issues early through verification can reduce the costs associated with failures and downtime.
5. **User Confidence**: Systems that are safety verified tend to instill greater confidence in users, as they are assured of their safe operation.

Traffic Light System Modeling

- How traffic light systems can be modeled in NuSMV.

Traffic light systems can be modeled in NuSMV by defining a finite state machine that captures all possible states and transitions of the lights. Each light and its possible colors (red, yellow, green) are represented as variables, with rules written in temporal logic to specify the allowed sequences and timing constraints. This model is then used for checking correctness against safety requirements, such as ensuring green lights never occur simultaneously in crossing directions.

- The basics of temporal logic used to specify the system.

Temporal logic in traffic light systems involves specifying the sequencing of light changes over time, ensuring conditions like "if one direction is green, then all crossing directions must be red" are always true. It uses modalities such as "eventually" and "until" to define the system's behavior across different time intervals.

- State diagram and explanation of states.
- **State Diagram Description**:

1. **State 1 - Red**: The default state where the traffic light is red, indicating a stop for traffic in that direction.
2. **State 2 - Green**: This state indicates that traffic in that direction is allowed to proceed.
3. **State 3 - Yellow**: This state is a transition indicating that the light is about to change to red, warning traffic to slow down and prepare to stop.

Verification Process:

1. Model the traffic light system in NuSMV, defining states for each light and transitions based on timing and sensor inputs.
2. Specify safety and liveness properties in temporal logic that the system must satisfy, such as mutual exclusivity of green lights in intersecting directions.
3. Run the NuSMV tool to check the model against these properties, with the tool exploring all possible states and transitions.
4. Analyze the output from NuSMV for any property violations and examine counterexamples to correct the model or system design as needed.

- In NuSMV, we input the model by writing a .smv file with the system's structure using its language constructs, and then you specify properties within the same file using CTL or LTL for the tool to verify.

```
NuSMV > read_model -i tarffic-lights0.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
```

```
NuSMV > pick_state -i

***************  AVAILABLE STATES  *************

  ================= State =================
  0) ------------------------
  turnInProgress = FALSE
  allowedDirection = NS
  NSDriving.current_traffic_light = Red
  EWDriving.current_traffic_light = Red
  NSPedestrian.current_walk_light = Dont_Walk
  EWPedestrian.current_walk_light = Dont_Walk
  light_timer = 0
  emergency = none


There's only one available state. Press Return to Proceed.

Chosen state is: 0
```

Code Comments

- MODULE main
- -- Traffic light system model for a simple intersection

- -- Define the states of the traffic light
- VAR
- light : {red, green, yellow};

- -- Define the initial state of the traffic light
- ASSIGN
- init(light) := red; -- The light starts at red

- -- Define the transition conditions for the traffic light states

```
TRANS
  case
    light = red : next(light) = green; -- Red changes to green
    light = green : next(light) = yellow; -- Green changes to yellow
    light = yellow : next(light) = red; -- Yellow changes to red
  esac;

-- Define safety properties to be verified
SPEC
  AG !(light = green & next(light) = green); -- The light cannot stay green forever

-- Define liveness properties to be verified
LTLSPEC
  G F (light = green); -- Eventually, the light will be green at some point in the future

-- Define fairness constraints if applicable
FAIRNESS
  running; -- Assuming a condition that the system is operational
```

Thank You