A TASM-based Requirements Validation Approach for Safety-critical Embedded Systems

Jiale (Joe) Zhou

zhou.jiale@mdh.se

School of Innovation, Design and Engineering
Mälardalen University, Sweden

Outline

- Introduction
- Overview Of the Timed Abstract State Machine (TASM) language
- TASM Extension
- Illustration Application
- The TASM-Based Approach
- Conclusion

zhou.jiale@mdh.se

Introduction

- Requirements Validation (RV) is vital
  - Continuum of requirements in the systems development life-cycle
    - High-level requirements describe system features
    - Low-level requirements state the system behaviors
  - Anomalies can be traced back to requirements specification
    - Contradictory or missing requirements
    - Infeasible requirements
- RV confirms the requirements correctness in terms of consistency and completeness
  - Consistency: No internal contradictions
  - Completeness: neither objects nor entities are left undefined, and low-level requirements can address high-level requirements.

zhou.jiale@mdh.se

Introduction cont’d

- TASM-based approach to requirements validation
  - TASM has shown its success with some distinctive features
    - Formal specification of behaviors and non-functional properties
    - A literate language without requiring mathematical training
    - TASM tool in progress
  - TASM is extended with newly defined constructs, TASM Event and TASM Observer
  - Three main steps
    - Low-level requirements modeling
    - High-level requirements modeling
    - Requirements validation

zhou.jiale@mdh.se

Outline

- Introduction
- Overview Of the Timed Abstract State Machine (TASM) language
- TASM Extension
- Illustration Application
- The TASM-Based Approach
- Conclusion

zhou.jiale@mdh.se

Overview Of TASM

- A TASM specification is a pair <E, ASM> where:
  - E is the environment, which defines
    - a set of variables,
    - type universe,
    - environment resources.
  - ASM is the abstract state machine, which defines
    - a set of machine rules by using the variables with property annotations.
      - The rule body is in the form of “if guard then action” or ”else then action”

zhou.jiale@mdh.se

Overview Of TASM cont’d

- TASM toy example
  - light switch

```
ENVIRONMENT:
  VARIABLES:
      light_status light := OFF;
      switch_status switch := DOWN;
  USER-DEFINED TYPES:
      light_status := {ON, OFF};
      switch_status := {UP, DOWN};
  RESOURCES:
      power:=[0,10]
```

```
MAIN MACHINE:
   MONITORED VARIABLES:
     switch;
   CONTROLLED VARIABLES:
     light;
   RULES:
         R1: Turn On{
            t:= 1;
            power:=[2,5];
            if light = OFF and switch = UP then
               light := ON; }
         R2: Turn Off {
            t:= [1,2];
            power:=[3,5];
            if light = ON and switch = DOWN then
               light := OFF; }
```

zhou.jiale@mdh.se

Outline

- Introduction
- Overview Of the Timed Abstract State Machine (TASM) language
- TASM Extension
- Illustration Application
- The TASM-Based Approach
- Conclusion

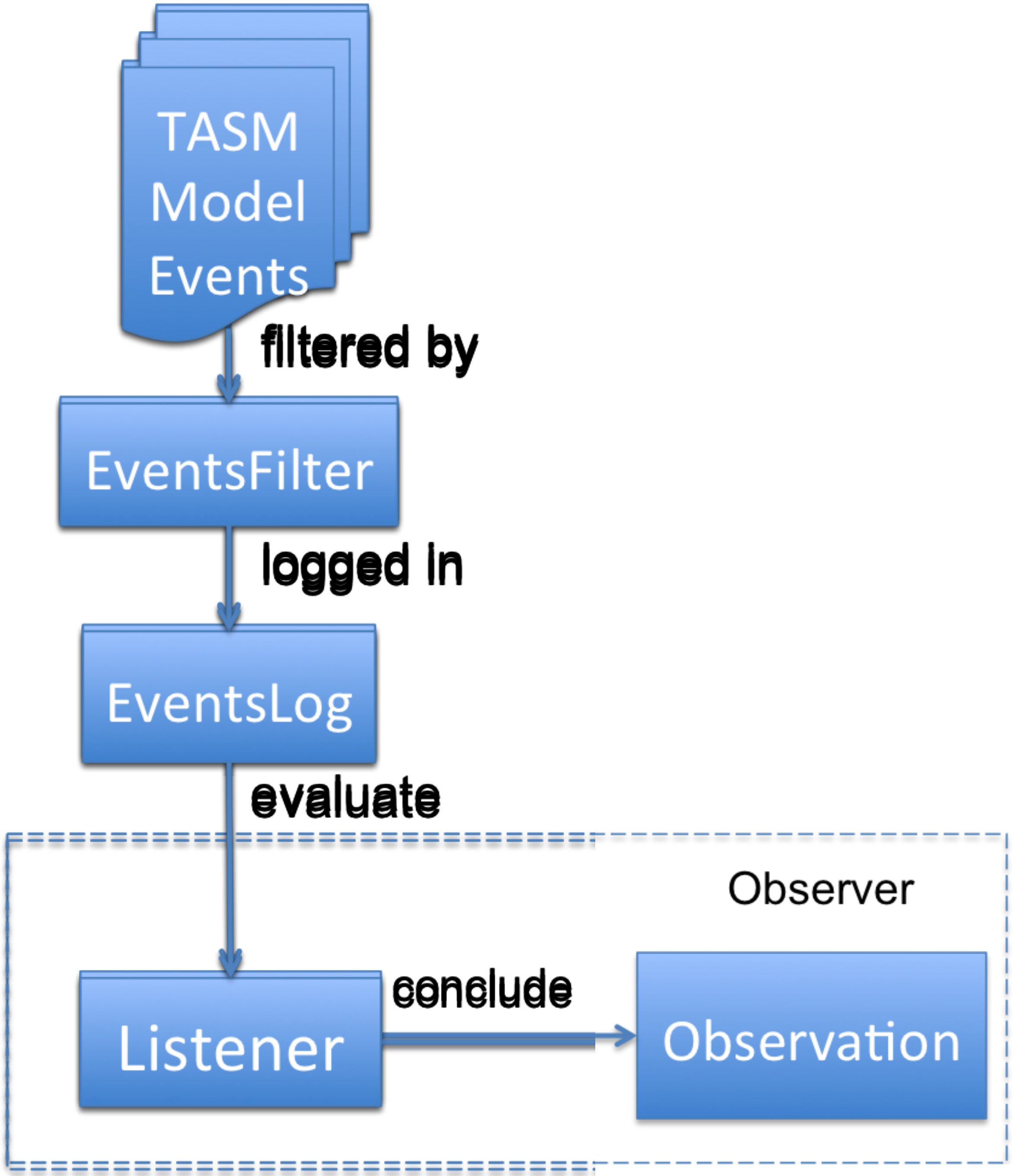
zhou.jiale@mdh.se

TASM Extension

- The extension includes two main parts
  - TASM Event has four types
    - ResourceUsedUpEvent, ChangeValueEvent, RuleEnableEvent, RuleDisableEvent
    - An event instance is generated by corresponding TASM constructs and will be time stamped.
  - TASM Observer is a tuple <OE, L, Ov>
    - OE denotes ObserverEnvironment which consists of ObserverVariable and EventsFilter,
    - L denotes Listener which specifies the observable scenario in the form of "**listening** *condition* **then** *action*",
      - *condition* is an expression describing the observable sequence of events
      - *action* updates the observer variables
    - Ov denotes Observation which is a predicate of the TASM model.

TASM Extension cont'd

- The execution semantics of TASM Observer
  - TASM model produces massive events
  - EventsFilter removes the irrelevant events and log the relevant events into the local database EventsLog
  - Listener will evaluate its condition based on the logged event sequence.
    - Regular expression is used as the sequential search pattern.
    - If the condition is satisfied, the observer variables will be updated.
  - The Observation will be concluded based on the updated variables
- A running TASM model can have several running observers.
- Offline monitoring



Outline

- Introduction
- Overview Of the Timed Abstract State Machine (TASM) language
- TASM Extension
- Illustration Application
- The TASM-Based Approach
- Conclusion

zhou.jiale@mdh.se

Illustration Application

- Brake-by-Wire (BbW) system
  - A demonstrator at a major automotive company
  - BbW aims at replacing the mechanical linkage between the brake pedal and the brake actuators
- High level requirements describe what the BbW system is required to do
  - **E.g.,** *the system shall provide a base brake functionality where the driver indicates that she/he wants to reduce speed so that the braking system starts decelerating the vehicle.*
- Low level requirements describe the behavior of each component of the BbW system
  - **E.g.,** *the brake torque calculator shall compute the driver requested torque and send the value to the vehicle brake controller, when a brake pedal displacement is detected.*

Outline

- Introduction
- Overview Of the Timed Abstract State Machine (TASM) language
- TASM Extension
- Illustration Application
- The TASM-Based Approach
- Conclusion

zhou.jiale@mdh.se

The TASM-based Approach

- Our approach consists of three main steps
  - Step 1: Low-level requirements modeling
    - Modeling system behaviors by using TASM
  - Step 2: High-level requirements modeling
    - Translating system features into TASM observers
  - Step 3: Requirements validation
    - Performing four kinds of validation checking
      - Logical Consistency Checking
      - Auxiliary Machine Checking,
      - Coverage Checking
      - Model Checking

The TASM-based Approach Cont’d

- Step 1: Low-level requirements modeling
  - Modeling the system behaviors
  - Five steps
    - Requirements preprocessing
      - Distinguishing functional requirements from non-functional requirements

The TASM-based Approach Cont’d

- Step 1: Low-level requirements modeling
  - Modeling the system behaviors
  - Five steps
    - Requirements preprocessing
    - Components Identification
      - Extracting possible system components
      - Two sub tasks
        - Identification of the external components
        - Identification of the internal components

The TASM-based Approach Cont’d

- Step 1: Low-level requirements modeling
  - Modeling the system behaviors
  - Five steps
    - Requirements preprocessing
    - Components Identification
    - Connection Identification
      - Identifying the connections between components
        - Port connection
        - Message passing

The TASM-based Approach Cont’d

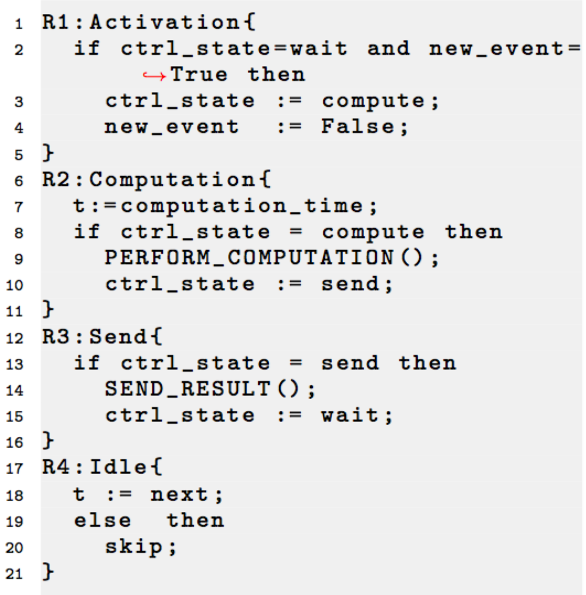
- Step 1: Low-level requirements modeling
  - Modeling the system behaviors
  - Five steps
    - Requirements preprocessing
    - Components Identification
    - Connection Identification
    - Behavior Specification
      - Specifying the behaviors of components
        - Identification of possible states
        - Identification of the transition conditions
        - Identification of the actions

The TASM-based Approach Cont’d

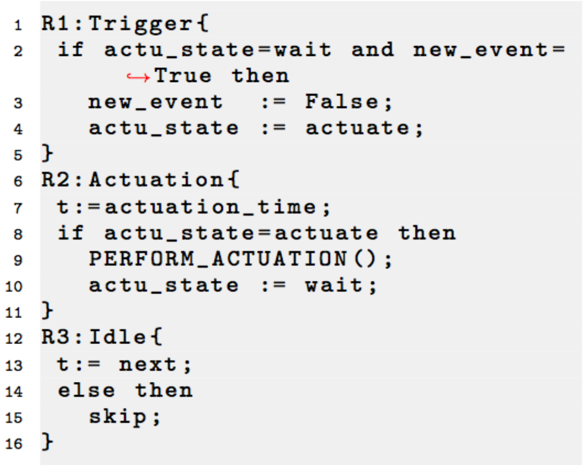
- Step 1: Low-level requirements modeling
  - Modeling the system behaviors
  - Five steps
    - Requirements preprocessing
    - Components Identification
    - Connection Identification
    - Behavior Specification
    - Property Annotation
      - Adding timing and resource consumption annotations

The TASM-based Approach Cont’d

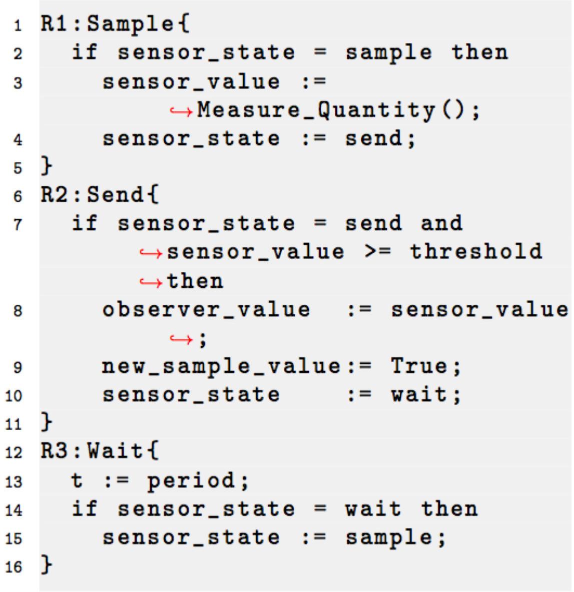
Main Machine	Quantity	Category	Description
DRIVER	1	External Entity	model the driver’s behavior
VEHICLE	1	External Entity	model the behavior of the vehicle
TORQUE_CALC	1	Micro-controller	calculate the driver’s requested torque
BRAKE_CTRL	1	Micro-controller	calculate the requested torque per wheel
ABS_CTRL	4	Micro-controller	calculate the brake force on each wheel
BRAKE_ACTU	4	Actuator	perform the brake force on each wheel
WHLSPD_SENSOR	4	Sensor	sense the rotating speed of each wheel
VCLSPD_SENSOR	1	Sensor	sense the moving speed of the vehicle
PEDAL_SENSOR	1	Sensor	sense the position of the brake pedal
COMMU_BUS	1	Bus	the communication bus



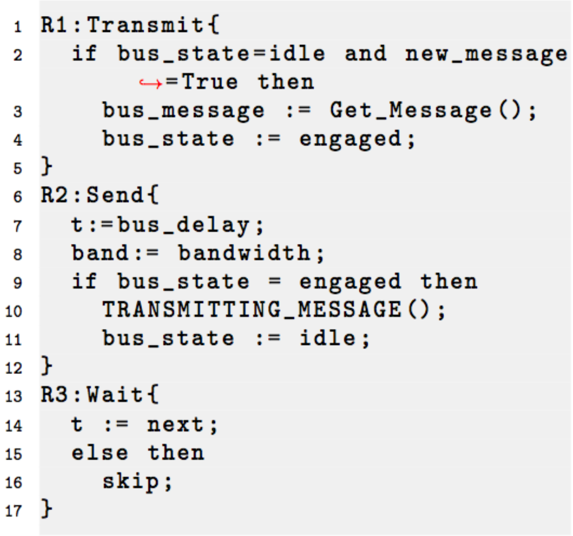
```
R1:Activation{
  if ctrl_state=wait and new_event=
      ↪True then
    ctrl_state := compute;
    new_event  := False;
}
R2:Computation{
  t:=computation_time;
  if ctrl_state = compute then
    PERFORM_COMPUTATION();
    ctrl_state := send;
}
R3:Send{
  if ctrl_state = send then
    SEND_RESULT();
    ctrl_state := wait;
}
R4:Idle{
  t := next;
  else  then
    skip;
}
```



```
R1:Trigger{
 if actu_state=wait and new_event=
     ↪True then
   new_event  := False;
   actu_state := actuate;
}
R2:Actuation{
 t:=actuation_time;
 if actu_state=actuate then
   PERFORM_ACTUATION();
   actu_state := wait;
}
R3:Idle{
 t:= next;
 else then
   skip;
}
```



```
R1:Sample{
  if sensor_state = sample then
    sensor_value :=
        ↪Measure_Quantity();
    sensor_state := send;
}
R2:Send{
  if sensor_state = send and
      ↪sensor_value >= threshold
      ↪then
    observer_value  := sensor_value
        ↪;
    new_sample_value:= True;
    sensor_state    := wait;
}
R3:Wait{
  t := period;
  if sensor_state = wait then
    sensor_state := sample;
}
```



```
R1:Transmit{
  if bus_state=idle and new_message
      ↪=True then
    bus_message := Get_Message();
    bus_state := engaged;
}
R2:Send{
  t:=bus_delay;
  band:= bandwidth;
  if bus_state = engaged then
    TRANSMITTING_MESSAGE();
    bus_state := idle;
}
R3:Wait{
  t := next;
  else then
    skip;
}
```

The TASM-based Approach Cont’d

- Step 2: High-level requirements modeling
  - Formalizing the system features
  - Four steps
    - Listener Specification
      - Specifying the possible events sequence representing the observable scenario,
      - Relevant observer variables will be updated if observed.

The TASM-based Approach Cont'd

- Step 2: High-level requirements modeling
  - Formalizing the system features
  - Four steps
    - Listener Specification
    - Observation Specification
      - Formalizing a predicate depending on the observer variables

The TASM-based Approach Cont’d

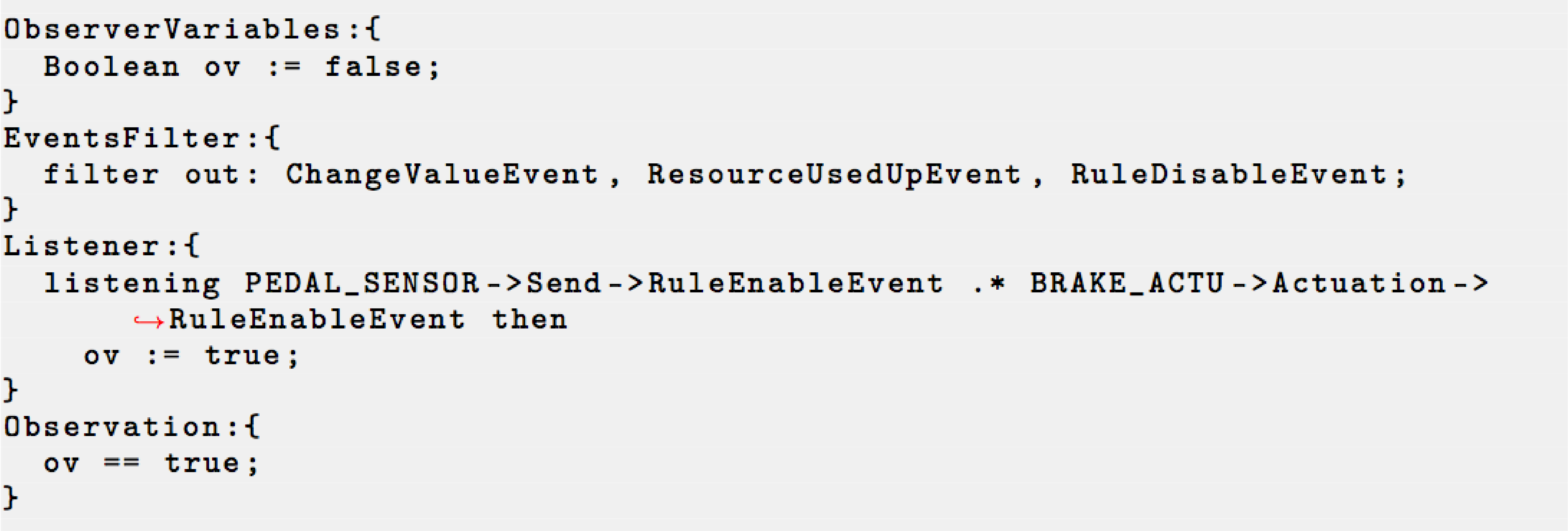
- Step 2: High-level requirements modeling
  - Formalizing the system features
  - Four steps
    - Listener Specification
    - Observation Specification
    - Events Filtering
      - Identifying the irrelevant events to the observable properties

The TASM-based Approach Cont’d

- Step 2: High-level requirements modeling
  - Formalizing the system features
  - Four steps
    - Listener Specification
    - Observation Specification
    - Events Filtering
    - Traceability Creation
      - Linking the specified Observer to the textual requirements

The TASM-based Approach Cont’d

- **E.g.,** *the system shall provide a base brake functionality where the driver indicates that she/he wants to reduce speed so that the braking system starts decelerating the vehicle.*



```
ObserverVariables:{
  Boolean ov := false;
}
EventsFilter:{
  filter out: ChangeValueEvent, ResourceUsedUpEvent, RuleDisableEvent;
}
Listener:{
  listening PEDAL_SENSOR->Send->RuleEnableEvent .* BRAKE_ACTU->Actuation->
      ↪RuleEnableEvent then
    ov := true;
}
Observation:{
  ov == true;
}
```

The TASM-based Approach Cont’d

- Step 3: Requirements validation
  - Logical Consistency Checking
    - Free of contradictions in the specification
      - Different machine rules are simultaneously enabled
      - Different values are assigned to the same variable simultaneously
  - Auxiliary Machine Checking
    - Free of undefined auxiliary machine
  - Coverage Checking
    - Checking whether all of the system features can be observed in the system behaviors model
  - Model Checking
    - Free of deadlock
    - Checking whether system features are satisfied by the system behaviors model

The TASM-based Approach Cont’d

- For the BbW system, the undefined auxiliary machine are found, which indicates the incompleteness of the BbW system requirements

Outline

- Introduction
- Overview Of the Timed Abstract State Machine (TASM) language
- TASM Extension
- Illustration Application
- The TASM-Based Approach
- Conclusion

zhou.jiale@mdh.se

Conclusion

- We have proposed a TASM-based approach to requirements validation
- We have extended the TASM language with TASM Event and TASM Observer constructs
- Our illustration application, namely the Brake-by-Wire system, shows that our approach can achieve the goal of requirements validation via
    - Logical Consistency Checking
    - Auxiliary Machine Checking
    - Coverage Checking
    - Model Checking
- Future work
    - Wider industrial validation of our approach
    - The improvement of our TASM toolset
    - Offline -> online
    - Discussion about the concept of “observable”

zhou.jiale@mdh.se

Thank You !
Tack !

zhou.jiale@mdh.se