Run-Time Validation of Timing Constraints for VDM-RT Models

Augusto Ribeiro Kenneth Lausdahl Peter G. Larsen
{ari,kel,pgl}@iha.dk

Aarhus School of Engineering – Denmark

October 24, 2011

AARHUS SCHOOL OF ENGINEERING

Agenda

AARHUS SCHOOL OF ENGINEERING

VDM Real-Time

The Real-Time dialect is an extension to VDM++ where object oriented structure and concurrency was introduced.

Real-Time extensions adds

- Simulated real time; all instructions take time.
- Hardware architecture definition; CPUs and buses.
- Distribution; objects are deployed onto CPUs connected with buses.

Time in VDM-RT

- Statements take time to execute

Time in VDM-RT

- Statements take time to execute
- **duration**

Duration example

```
public AdjustVolumeUp : () ==> ()
AdjustVolumeUp () ==
 duration (10)
  (if volume < MAX then ( volume := volume + 1;
   RadNavSys`mmi.UpdateScreen(1)));
   );
```

AARHUS SCHOOL OF ENGINEERING

Time in VDM-RT

- Statements take time to execute
- **duration**

Duration example

```
public AdjustVolumeUp : () ==> ()
AdjustVolumeUp () ==
 duration (10)
  (if volume < MAX then ( volume := volume + 1;
   RadNavSys`mmi.UpdateScreen(1)));
   );
```

- Duration is typically an estimation

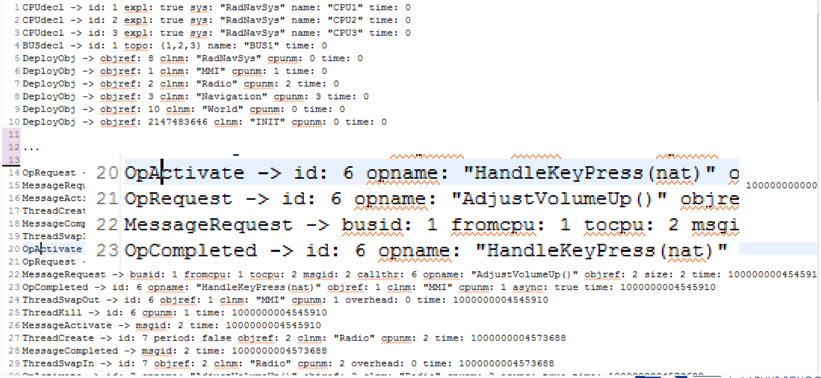
RT Log

- Log is produced when a VDM-RT model is interpreted.

```
CPUdecl -> id: 1 expl: true sys: "RadNavSys" name: "CPU1" time: 0
CPUdecl -> id: 2 expl: true sys: "RadNavSys" name: "CPU2" time: 0
CPUdecl -> id: 3 expl: true sys: "RadNavSys" name: "CPU3" time: 0
BUSdecl -> id: 1 topo: {1,2,3} name: "BUS1" time: 0
DeployObj -> objref: 8 clnm: "RadNavSys" cpunm: 0 time: 0
DeployObj -> objref: 1 clnm: "MMI" cpunm: 1 time: 0
DeployObj -> objref: 2 clnm: "Radio" cpunm: 2 time: 0
DeployObj -> objref: 3 clnm: "Navigation" cpunm: 3 time: 0
DeployObj -> objref: 10 clnm: "World" cpunm: 0 time: 0
DeployObj -> objref: 2147483646 clnm: "INIT" cpunm: 0 time: 0

...

OpRequest -> id: 5 opname: "HandleKeyPress(nat)" objref: 1 clnm: "MMI" cpunm: 0 async: true time: 1000000000000000
MessageRequest -> busid: 0 fromcpu: 0 tocpu: 1 msgid: 1 callthr: 5 opname: "HandleKeyPress(nat)" objref: 1 size: 3 time: 100000000000
MessageActivate -> msgid: 1 time: 1000000000000000
ThreadCreate -> id: 6 period: false objref: 1 clnm: "MMI" cpunm: 1 time: 1000000000000000
MessageCompleted -> msgid: 1 time: 1000000000000000
ThreadSwapIn -> id: 6 objref: 1 clnm: "MMI" cpunm: 1 overhead: 0 time: 1000000000000000
OpActivate -> id: 6 opname: "HandleKeyPress(nat)" objref: 1 clnm: "MMI" cpunm: 1 async: true time: 1000000000000000
OpRequest -> id: 6 opname: "AdjustVolumeUp()" objref: 2 clnm: "Radio" cpunm: 1 async: true time: 1000000004545910
MessageRequest -> busid: 1 fromcpu: 1 tocpu: 2 msgid: 2 callthr: 6 opname: "AdjustVolumeUp()" objref: 2 size: 2 time: 100000000454591
OpCompleted -> id: 6 opname: "HandleKeyPress(nat)" objref: 1 clnm: "MMI" cpunm: 1 async: true time: 1000000004545910
ThreadSwapOut -> id: 6 objref: 1 clnm: "MMI" cpunm: 1 overhead: 0 time: 1000000004545910
ThreadKill -> id: 6 cpunm: 1 time: 1000000004545910
MessageActivate -> msgid: 2 time: 1000000004545910
ThreadCreate -> id: 7 period: false objref: 2 clnm: "Radio" cpunm: 2 time: 1000000004573688
MessageCompleted -> msgid: 2 time: 1000000004573688
ThreadSwapIn -> id: 7 objref: 2 clnm: "Radio" cpunm: 2 overhead: 0 time: 1000000004573688
```

RT Log

- Log is produced when a VDM-RT model is interpreted.



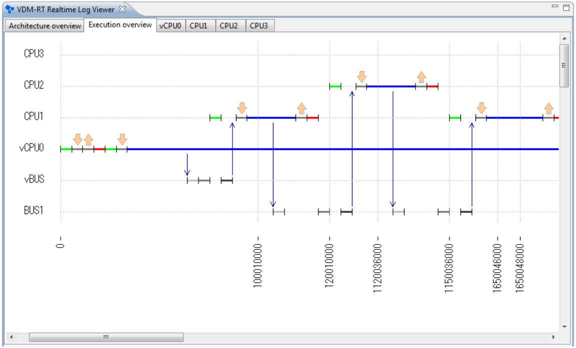
```
CPUdecl -> id: 1 expl: true sys: "RadNavSys" name: "CPU1" time: 0
CPUdecl -> id: 2 expl: true sys: "RadNavSys" name: "CPU2" time: 0
CPUdecl -> id: 3 expl: true sys: "RadNavSys" name: "CPU3" time: 0
BUSdecl -> id: 1 topo: {1,2,3} name: "BUS1" time: 0
DeployObj -> objref: 8 clnm: "RadNavSys" cpunm: 0 time: 0
DeployObj -> objref: 1 clnm: "MMI" cpunm: 1 time: 0
DeployObj -> objref: 2 clnm: "Radio" cpunm: 2 time: 0
DeployObj -> objref: 3 clnm: "Navigation" cpunm: 3 time: 0
DeployObj -> objref: 10 clnm: "World" cpunm: 0 time: 0
DeployObj -> objref: 2147483646 clnm: "INIT" cpunm: 0 time: 0

...

OpRequest -
MessageRequ
MessageActi
ThreadCreat
MessageComp
ThreadSwapI
OpActivate
OpRequest -
MessageRequest -> busid: 1 fromcpu: 1 tocpu: 2 msgid: 2 callthr: 6 opname: "AdjustVolumeUp()" objref: 2 size: 2 time: 100000000454591
OpCompleted -> id: 6 opname: "HandleKeyPress(nat)" objref: 1 clnm: "MMI" cpunm: 1 async: true time: 1000000004545910
ThreadSwapOut -> id: 6 objref: 1 clnm: "MMI" cpunm: 1 overhead: 0 time: 1000000004545910
ThreadKill -> id: 6 cpunm: 1 time: 1000000004545910
MessageActivate -> msgid: 2 time: 1000000004545910
ThreadCreate -> id: 7 period: false objref: 2 clnm: "Radio" cpunm: 2 time: 1000000004573688
MessageCompleted -> msgid: 2 time: 1000000004573688
ThreadSwapIn -> id: 7 objref: 2 clnm: "Radio" cpunm: 2 overhead: 0 time: 1000000004573688
```

```
20 OpActivate -> id: 6 opname: "HandleKeyPress(nat)" o
21 OpRequest -> id: 6 opname: "AdjustVolumeUp()" objre
22 MessageRequest -> busid: 1 fromcpu: 1 tocpu: 2 msgi
23 OpCompleted -> id: 6 opname: "HandleKeyPress(nat)"
```

100000000000

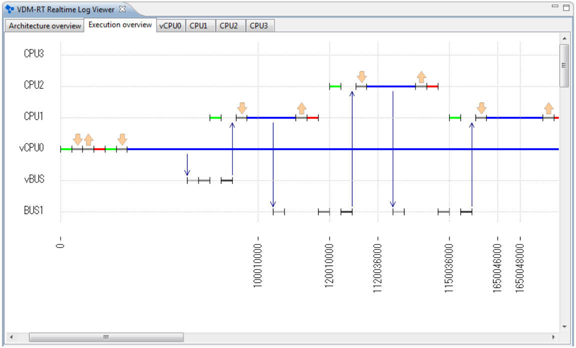
RT Log Viewer

- Overture tool to visualise RT logs - [Verhoef]



RT Log Viewer

- Overture tool to visualise RT logs - [Verhoef]



- Permits visually analysis of a model execution

AARHUS SCHOOL OF ENGINEERING

Motivation

- Real-time systems not only have functional requirements

Motivation

- Real-time systems not only have functional requirements
- Timing requirements cannot actually be recorded in VDM-RT explicitly

AARHUS SCHOOL OF ENGINEERING

Motivation

- Real-time systems not only have functional requirements
- Timing requirements cannot actually be recorded in VDM-RT explicitly
- Perform validation of these timing requirements

Agenda

First Idea for Timing Invariants

The idea of timing invariants for VDM-RT was originally described in:

[Fitzgerald et al., 2007] Fitzgerald, Larsen, Tjell, Verhoef
Validation Support for Real-Time Embedded Systems in VDM++
HASE 2007: 10th IEEE High Assurance Systems Engineering Symposium, 2007.

First Idea for Timing Invariants

The idea of timing invariants for VDM-RT was originally described in:

[Fitzgerald et al., 2007] Fitzgerald, Larsen, Tjell, Verhoef
Validation Support for Real-Time Embedded Systems in VDM++
HASE 2007: 10th IEEE High Assurance Systems Engineering Symposium, 2007.

- Post analysis of the RTLog

First Idea for Timing Invariants

The idea of timing invariants for VDM-RT was originally described in:

[Fitzgerald et al., 2007] Fitzgerald, Larsen, Tjell, Verhoef
Validation Support for Real-Time Embedded Systems in VDM++
HASE 2007: 10th IEEE High Assurance Systems Engineering Symposium, 2007.

- Post analysis of the RTLog
- It only exists at specification level

Timing Invariants - Basic Ingredients

Basic Ingredients:

- Trigger Event - T_t

Timing Invariants - Basic Ingredients

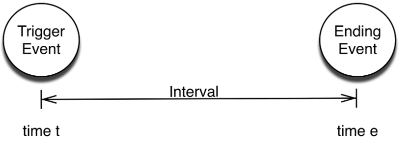
Basic Ingredients:

- Trigger Event - T_t
- Ending Event - T_e

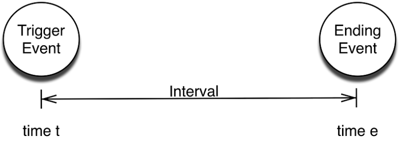
Timing Invariants - Basic Ingredients

Basic Ingredients:

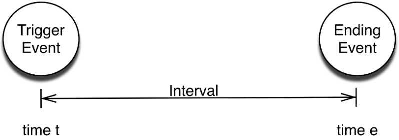
- Trigger Event - T_t
- Ending Event - T_e
- Interval - time



Timing Invariants - Basic Ingredients

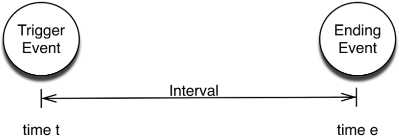


Timing Invariants - Basic Ingredients



- Invariants are relations between Trigger Time (T_t), Ending Time (T_e) and the Interval (i)

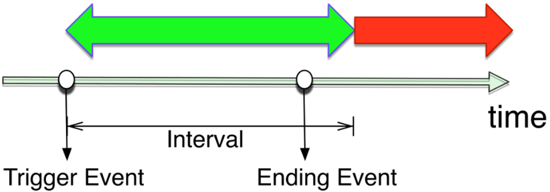
Timing Invariants - Basic Ingredients



- Invariants are relations between Trigger Time (T_t), Ending Time (T_e) and the Interval (i)
- Typically something like: $T_e - T_t \sqsubset i$

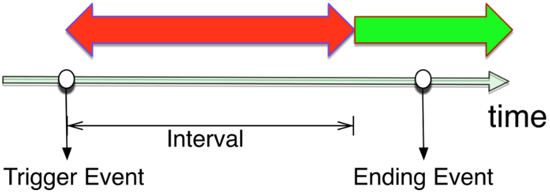
Timing Invariants - Deadline

- Deadline - $T_e - T_t \leq i$



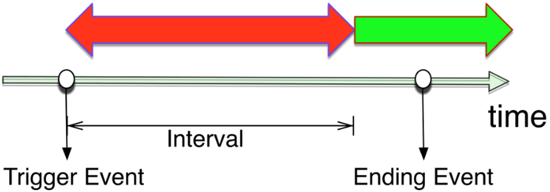
Timing Invariants - Separate

- Separate - $T_e - T_t > i$



Timing Invariants - Separate

- Separate - $T_e - T_t > i$



- Separate required - ending event is demanded

Invariant instances - lifecycle of the invariant

Lifecycle of an Invariant

AARHUS SCHOOL OF ENGINEERING

Invariant instances - lifecycle of the invariant

Lifecycle of an Invariant

- Trigger event occurs - an invariant instance is activated

AARHUS SCHOOL OF ENGINEERING

Invariant instances - lifecycle of the invariant

Lifecycle of an Invariant

- Trigger event occurs - an invariant instance is activated
- Ending event occurs - instance is deactivated and evaluated

Invariant instances - lifecycle of the invariant

Lifecycle of an Invariant

- Trigger event occurs - an invariant instance is activated
- Ending event occurs - instance is deactivated and evaluated

Problem

AARHUS SCHOOL OF ENGINEERING

Invariant instances - lifecycle of the invariant

Lifecycle of an Invariant

- Trigger event occurs - an invariant instance is activated
- Ending event occurs - instance is deactivated and evaluated

Problem

- Invariants can be triggered several times and instances of the same invariant coexist

Invariant instances - lifecycle of the invariant

Lifecycle of an Invariant

- Trigger event occurs - an invariant instance is activated
- Ending event occurs - instance is deactivated and evaluated

Problem

- Invariants can be triggered several times and instances of the same invariant coexist
- Definition of decommission policies of instances

Decommissioning Policies - Non-selective

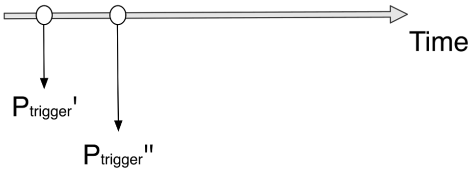
- Non-selective

Decommissioning Policies - Non-selective

- Non-selective

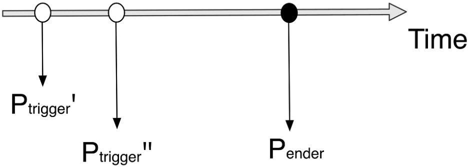
Decommissioning Policies - Non-selective

- Non-selective



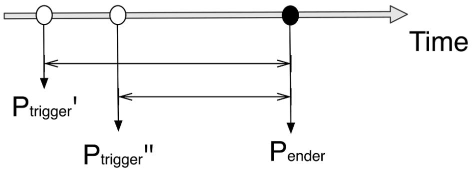
Decommissioning Policies - Non-selective

- Non-selective



Decommissioning Policies - Non-selective

- Non-selective

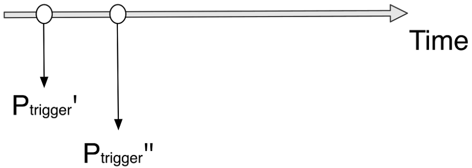


Decommissioning Policies - Matching

- Matching

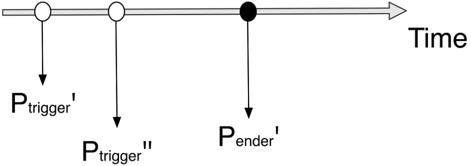
Decommissioning Policies - Matching

- Matching



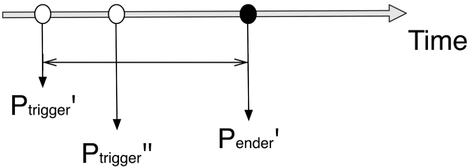
Decommissioning Policies - Matching

- Matching



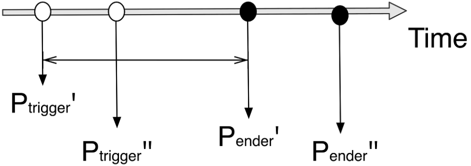
Decommissioning Policies - Matching

- Matching



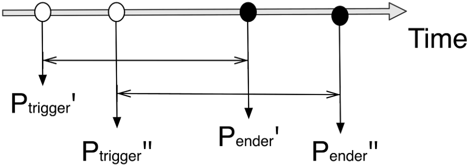
Decommissioning Policies - Matching

- Matching



Decommissioning Policies - Matching

- Matching



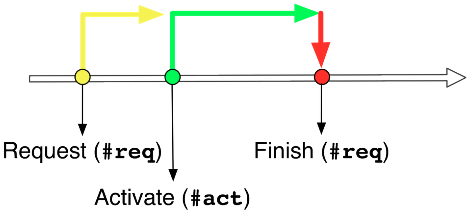
Decommissioning Policies - Others

Other decommissioning policies could be be:

- Matching thread
- Matching object

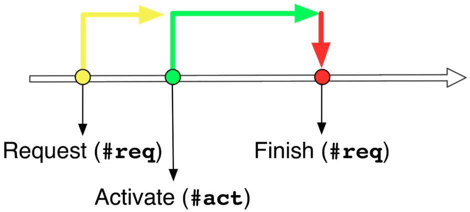
What can an event be? - Operation Events

- Operation Events



What can an event be? - Operation Events

- Operation Events



- Reference operation phases on classes or instances

What can an event be? - Predicate Events

- Predicates over instance variables

What can an event be? - Predicate Events

- Predicates over instance variables
- Predicates are evaluated when the variable changes

Timing Invariants - Syntax

General Syntax

property(trigger, ending, interval)

Examples

```
deadlineMet(
  ( #req(MMI`HandleKeyPressUp),
    RadNavSys`radio.volume < Radio`MAX
  ),
  #fin(MMI`AdjustVolumeUp),
  100 ms)
```

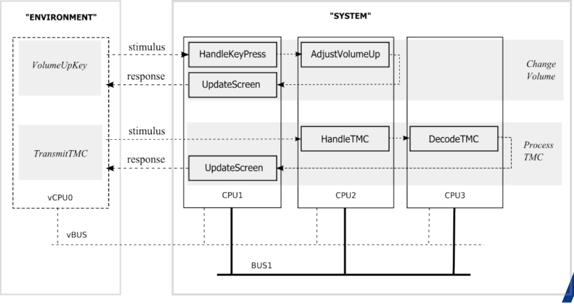
Agenda

1. Introduction
   - VDM-RT
   - Motivation
2. Timing Invariants
   - Invariant Types
   - Invariant Instances and Decommissioning Policies
   - Invariants Definition
3. Case Study and Results
4. Concluding remarks

AARHUS SCHOOL OF ENGINEERING

Case study - In car navigation radio

- Case appears in M. Verhoef PhD thesis
- Car radio with 3 CPUs
- Process Traffic Message Channel (TMC) and controls volume



Case study - Time Invariants

C1: *A volume change must be reflected in the display within 35 ms.*

```
deadlineMet(
  #fin(Radio`AdjustVolumeUp),
  #fin(MMI`UpdateScreen),
  35 ms)
```

AARHUS SCHOOL OF ENGINEERING

Case study - Time Invariants

C1: *A volume change must be reflected in the display within 35 ms.*

```
deadlineMet(
  #fin(Radio`AdjustVolumeUp),
  #fin(MMI`UpdateScreen),
  35 ms)
```

C2: *The screen should be updated no more than once every 500 ms.*

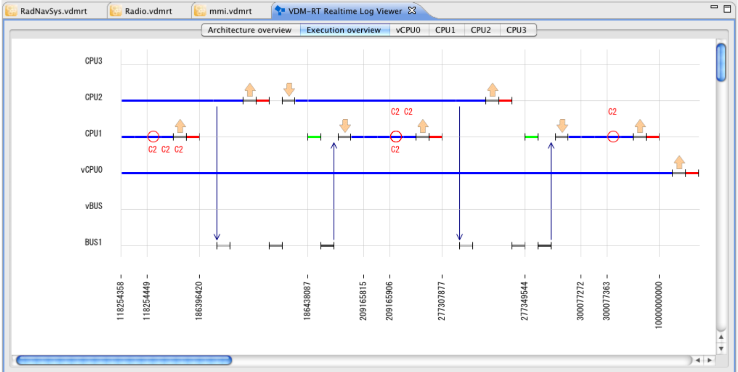
```
separate(
  #fin(MMI`UpdateScreen),
  #fin(MMI`UpdateScreen),
  500 ms)
```

Case study - Time Invariants

C3: *If the volume is to be adjusted upwards and it is not currently at the maximum, the audible change should occur within 100 ms.*

```
deadlineMet(
  ( #req(MMI`HandleKeyPressUp),
    RadNavSys`radio.volume < Radio`MAX
  ),
  #fin(MMI`AdjustVolumeUp),
  100 ms)
```

Achieved Results



status	name	expression	src time	src thread	dest time	dest thread
PASS	C1	deadlineMet(#fin(Radio`AdjustVolumeUp),#fin(MMI`UpdateScreen),35000000)				
FAIL	C2	separate(#fin(MMI`UpdateScreen),#fin(MMI`UpdateScreen),500000000)	118254449	12	209165906	13
FAIL	C2	separate(#fin(MMI`UpdateScreen),#fin(MMI`UpdateScreen),500000000)	209165906	13	300077363	14
PASS	C3	deadlineMet(#act(MMI`HandleKeyPressUp),ValueValidationExpression: RadNavSys`radio.volume LESS RadNavSys`ra				

AARHUS SCHOOL OF ENGINEERING

Timing Invariants

System Extension Suggestion

- Time invariants are system invariants
- They could possibly be added to the `system` class.

Timing Invariants

System Extension Suggestion

- Time invariants are system invariants
- They could possibly be added to the **system** class.

Examples

```
system Sys
...
timing invariants

deadlineMet(evTrigger1, evEnder1, 400 ms);
...
separate(evTrigger2, evEnder2, 1000 ms);
```

AARHUS SCHOOL OF ENGINEERING

Agenda

Concluding remarks and future work

- What we have done:
- Answered questions:
- Future work:

Concluding remarks and future work

- What we have done:
  - Prototype of time invariant checking during run-time
- Answered questions:
- Future work:

Concluding remarks and future work

- What we have done:
  - Prototype of time invariant checking during run-time
  - Defined behavior semantics of the invariants
- Answered questions:
- Future work:

Concluding remarks and future work

- What we have done:
  - Prototype of time invariant checking during run-time
  - Defined behavior semantics of the invariants
- Answered questions:
  - Should time invariants be included in the system?
- Future work:

AARHUS SCHOOL OF ENGINEERING

Concluding remarks and future work

- What we have done:
  - Prototype of time invariant checking during run-time
  - Defined behavior semantics of the invariants
- Answered questions:
  - Should time invariants be included in the system?
  - Are the defined properties enough?
- Future work:

Concluding remarks and future work

- What we have done:
  - Prototype of time invariant checking during run-time
  - Defined behavior semantics of the invariants
- Answered questions:
  - Should time invariants be included in the system?
  - Are the defined properties enough?
  - Should the policies of decommission be selected by the user?
- Future work:

AARHUS SCHOOL OF ENGINEERING

Concluding remarks and future work

- What we have done:
  - Prototype of time invariant checking during run-time
  - Defined behavior semantics of the invariants
- Answered questions:
  - Should time invariants be included in the system?
  - Are the defined properties enough?
  - Should the policies of decommission be selected by the user?
  - Should violations mean the interruption of the interpretation
- Future work:

ASE | AARHUS SCHOOL OF ENGINEERING

Concluding remarks and future work

- What we have done:
  - Prototype of time invariant checking during run-time
  - Defined behavior semantics of the invariants
- Answered questions:
  - Should time invariants be included in the system?
  - Are the defined properties enough?
  - Should the policies of decommission be selected by the user?
  - Should violations mean the interruption of the interpretation
- Future work:
  - Including run-time time invariant checks in the Overture interpreter.

ASE | AARHUS SCHOOL OF ENGINEERING

Ending

Questions?

AARHUS SCHOOL OF ENGINEERING