Slicing AADL Specifications for Model Checking

Maximilian Odenbrett[1,2] Viet Yen Nguyen[2] Thomas Noll[2]

[1] Biomodeling and Bioinformatics Group
Eindhoven University of Technology, The Netherlands

[2] Software Modelling and Verification Group
RWTH Aachen University, Germany

NASA Formal Methods Symposium, April 15th, 2010

Partially funded by ESA/ESTEC under Contract No. 21171/07/NL/JD

AADL: Industry Standard for Modeling Embedded Systems

```
system RandomIntValue
 features
  value: out data port int
         default 2;
  update: in event port;
end RandomIntValue;

system implementation
  RandomIntValue.Impl
 modes
  loop: activation mode;
 transitions
  loop -[update then
         value := 0]-> loop;
  loop -[update then
         value := 1]-> loop;
  -- i.d. value := 2,...,29
  loop -[update then
         value := 30]-> loop;
end RandomIntValue.Impl;
```

AADL: Industry Standard for Modeling Embedded Systems

```
system RandomIntValue
 features
  value: out data port int
         default 2;
  update: in event port;
end RandomIntValue;

system implementation
  RandomIntValue.Impl
 modes
  loop: activation mode;
 transitions
  loop -[update then
         value := 0]-> loop;
  loop -[update then
         value := 1]-> loop;
  -- i.d. value := 2,...,29
  loop -[update then
         value := 30]-> loop;
end RandomIntValue.Impl;
```

Features

- Component-oriented
- HW/SW bindings
- Degraded operations
- Dynamic reconfiguration
- Error handling
- Fault injections
- Timing, probability, hybridity
- Formal semantics

Verification and Performability Evaluation of AADL

During the COMPASS Project 2008 - 2010:

- Defined formal semantics for a system-level modelling language

Verification and Performability Evaluation of AADL

During the COMPASS Project 2008 - 2010:

- Defined formal semantics for a system-level modelling language
- Developed a toolset

Verification and Performability Evaluation of AADL

During the COMPASS Project 2008 - 2010:

- Defined formal semantics for a system-level modelling language
- Developed a toolset
  - Requirements validation
  - Simulations
  - Model checking
  - (Dynamic) Fault Tree Analysis
  - Fault tolerance evaluation
  - Criticality evaluation
  - Failure modes and effects analysis
  - Fault detection, isolation and recovery
  - Diagnosability
  - Performability

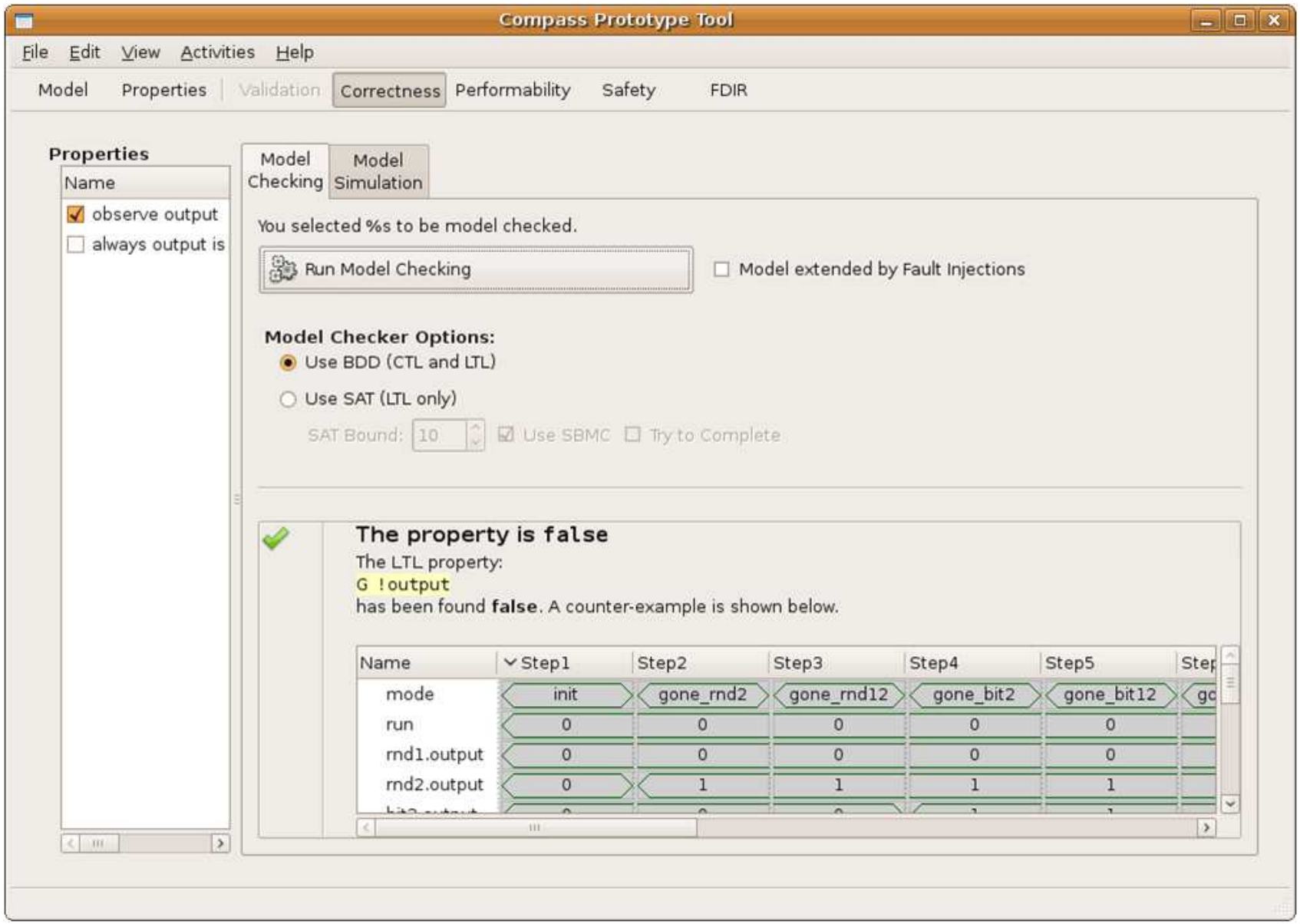
Verification and Performability Evaluation of AADL

During the COMPASS Project 2008 - 2010:

- Defined formal semantics for a system-level modelling language
- Developed a toolset
  - Requirements validation
  - Simulations
  - Model checking
  - (Dynamic) Fault Tree Analysis
  - Fault tolerance evaluation
  - Criticality evaluation
  - Failure modes and effects analysis
  - Fault detection, isolation and recovery
  - Diagnosability
  - Performability
- Applied case studies from Europe's space-industry

COMPASS Toolset 2.0

Based on NuSMV and Markov Reward Model Checker



History of Slicing

Given an input $\mathcal{M}$ and a slicing criterion Φ, compute a $\mathcal{M}^{\Phi}_{sliced}$ such that the following holds:

$$\mathcal{M} \approx \mathcal{M}^{\Phi}_{sliced}$$

History of Slicing

Given an input $\mathcal{M}$ and a slicing criterion Φ, compute a $\mathcal{M}^{\Phi}_{sliced}$ such that the following holds:

$$\mathcal{M} \approx \mathcal{M}^{\Phi}_{sliced}$$

Weiser 1981 Slicing sequential programs.

History of Slicing

Given an input $\mathcal{M}$ and a slicing criterion Φ, compute a $\mathcal{M}^{\Phi}_{sliced}$ such that the following holds:

$$\mathcal{M} \approx \mathcal{M}^{\Phi}_{sliced}$$

Weiser 1981 Slicing sequential programs.

Cheng 1993 Slicing concurrent programs.

History of Slicing

Given an input $\mathcal{M}$ and a property Φ, compute a $\mathcal{M}^{\Phi}_{sliced}$ such that the following holds:

$$\mathcal{M} \models \Phi \Leftrightarrow \mathcal{M}^{\Phi}_{sliced} \models \Phi$$

Weiser 1981 Slicing sequential programs.

Cheng 1993 Slicing concurrent programs.

Hatcliff et al. Property-driven slicing of (concurrent) programs.

History of Slicing

Given an input $\mathcal{M}$ and a property Φ, compute a $\mathcal{M}^{\Phi}_{sliced}$ such that the following holds:

$$\mathcal{M} \models \Phi \Leftrightarrow \mathcal{M}^{\Phi}_{sliced} \models \Phi$$

Weiser 1981 Slicing sequential programs.

Cheng 1993 Slicing concurrent programs.

Hatcliff et al. Property-driven slicing of (concurrent) programs.

Today Property-driven slicing of system-level models.

Property-Driven Slicing of AADL

Given an AADL model $\mathcal{M}$ and a CTL*-property Φ

1. Extract the *data elements* and *modes* of direct interest.
2. Compute the closure of data elements and modes of indirect influence (next slide).
3. Generate $\mathcal{M}^{\Phi}_{sliced}$

Static Analysis for Determining Interesting Parts

$D := \{d \in \textit{Dat} \mid d \text{ occurs in } \Phi\}$;
$E := \emptyset$;
$M := \{m \in \textit{Mod} \mid m \text{ occurs in } \Phi\}$;
} Initialization based on the property Φ.

repeat

until nothing changes;

Static Analysis for Determining Interesting Parts

$D := \{d \in \textit{Dat} \mid d \text{ occurs in } \Phi\}$;
$E := \emptyset$;
$M := \{m \in \textit{Mod} \mid m \text{ occurs in } \Phi\}$;

} Initialization based on the property Φ.

repeat

for all $m \xrightarrow{e,g,f} m' \in \textit{Trn}$ with $\exists d \in D : f$ updates d
or $\exists d \in D : d$ inactive in m but active in m'
or $e \in E$ **do**
$M := M \cup \{m\}$;

} Add source modes of transitions that affect interesting data elements.

until nothing changes;

Static Analysis for Determining Interesting Parts

$D := \{d \in \textit{Dat} \mid d \text{ occurs in } \Phi\}$;
$E := \emptyset$;
$M := \{m \in \textit{Mod} \mid m \text{ occurs in } \Phi\}$;
} Initialization based on the property Φ.

repeat

for all $m \xrightarrow{e,g,f} m' \in \textit{Trn}$ with $\exists d \in D : f$ updates d
or $\exists d \in D : d$ inactive in m but active in m'
or $e \in E$ **do**
$M := M \cup \{m\}$;
} Add source modes of transitions that affect interesting data elements.

for all $m \xrightarrow{e,g,f} m' \in \textit{Trn}$ with $m \in M$ or $m' \in M$ **do**
$D := D \cup \{d \in \textit{Dat} \mid g \text{ reads } d\}$
$\cup \{d \in \textit{Dat} \mid f \text{ updates some } d' \in D \text{ reading } d\}$;
$E := E \cup \{e\}$;
$M := M \cup \{m\}$;
} Add data elements, events and source modes of interesting transitions.

until nothing changes;

Static Analysis for Determining Interesting Parts

$D := \{d \in \textit{Dat} \mid d \text{ occurs in } \Phi\}$;
$E := \emptyset$;
$M := \{m \in \textit{Mod} \mid m \text{ occurs in } \Phi\}$;
} Initialization based on the property Φ.

repeat

for all $m \xrightarrow{e,g,f} m' \in \textit{Trn}$ with $\exists d \in D : f$ updates d
or $\exists d \in D : d$ inactive in m but active in m'
or $e \in E$ **do**
$M := M \cup \{m\}$;
} Add source modes of transitions that affect interesting data elements.

for all $m \xrightarrow{e,g,f} m' \in \textit{Trn}$ with $m \in M$ or $m' \in M$ **do**
$D := D \cup \{d \in \textit{Dat} \mid g \text{ reads } d\}$
$\cup \{d \in \textit{Dat} \mid f \text{ updates some } d' \in D \text{ reading } d\}$;
$E := E \cup \{e\}$;
$M := M \cup \{m\}$;
} Add data elements, events and source modes of interesting transitions.

for all $d := a \in \textit{Flw}$ with $d \in D$ **do**
$D := D \cup \{d' \in \textit{Dat} \mid a \text{ reads } d'\}$;
$M := M \cup \{m \in \textit{Mod} \mid d := a \text{ active in } m\}$;
} Add data elements and modes of interesting flows.

until nothing changes;

Static Analysis for Determining Interesting Parts

$D := \{d \in \textit{Dat} \mid d \text{ occurs in } \Phi\}$;
$E := \emptyset$;
$M := \{m \in \textit{Mod} \mid m \text{ occurs in } \Phi\}$;
} Initialization based on the property Φ.

repeat

for all $m \xrightarrow{e,g,f} m' \in \textit{Trn}$ with $\exists d \in D : f$ updates d
or $\exists d \in D : d$ inactive in m but active in m'
or $e \in E$ **do**
$M := M \cup \{m\}$;
} Add source modes of transitions that affect interesting data elements.

for all $m \xrightarrow{e,g,f} m' \in \textit{Trn}$ with $m \in M$ or $m' \in M$ **do**
$D := D \cup \{d \in \textit{Dat} \mid g \text{ reads } d\}$
$\cup \{d \in \textit{Dat} \mid f \text{ updates some } d' \in D \text{ reading } d\}$;
$E := E \cup \{e\}$;
$M := M \cup \{m\}$;
} Add data elements, events and source modes of interesting transitions.

for all $d := a \in \textit{Flw}$ with $d \in D$ **do**
$D := D \cup \{d' \in \textit{Dat} \mid a \text{ reads } d'\}$;
$M := M \cup \{m \in \textit{Mod} \mid d := a \text{ active in } m\}$;
} Add data elements and modes of interesting flows.

for all $e \rightsquigarrow e' \in \textit{Con}$ with $e \in E$ or $e' \in E$ **do**
$E := E \cup \{e, e'\}$;
$M := M \cup \{m \in \textit{Mod} \mid e \rightsquigarrow e' \text{ active in } m\}$;
} Add events and modes of interesting event ports.

until nothing changes;

Example: Adding Random Integers

```
system IntAdder
 features
  x: in data port int;
  y: in data port int;
  sum: out data port int;
end IntAdder;

system implementation
  IntAdder.Impl
 flows
  sum := x + y;
end IntAdder.Impl;
```

```
bus Bus
end Bus;

bus implementation
  Bus.Impl
end Bus.Impl;
```

Example: Adding Random Integers

```
system IntAdder
 features
  x: in data port int;
  y: in data port int;
  sum: out data port int;
end IntAdder;

system implementation
  IntAdder.Impl
 flows
  sum := x + y;
end IntAdder.Impl;
```

```
bus Bus
end Bus;

bus implementation
  Bus.Impl
end Bus.Impl;
```

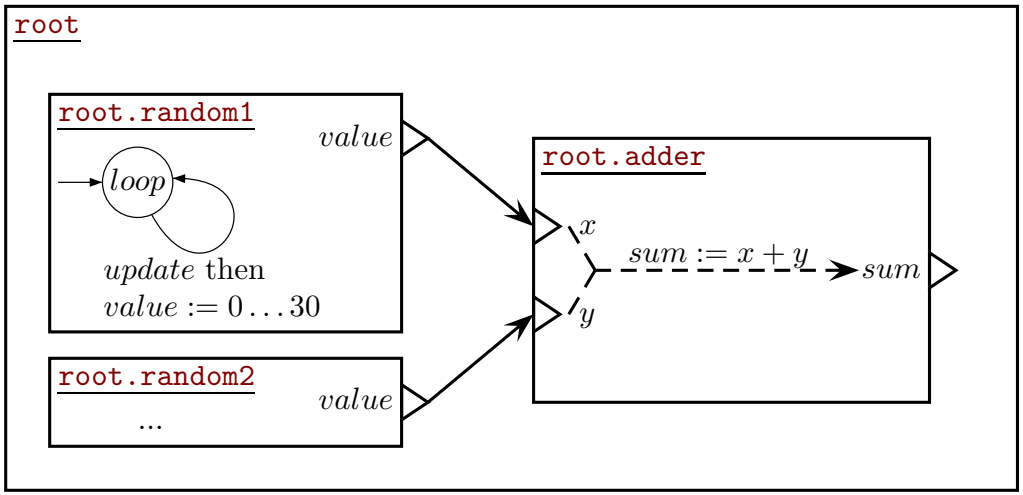
```
system IntAdderFrame
end IntAdderFrame;

system implementation IntAdderFrame.Impl
 subcomponents
  random1: system RandomIntValue.Impl
           accesses aBus;
  random2: system RandomIntValue.Impl
           accesses aBus;
  adder: system IntAdder.Impl
         accesses aBus;
  aBus: bus Bus.Impl;
 connections
  data port random1.value -> adder.x;
  data port random2.value -> adder.y;
 modes
  choose1: activation mode;
  choose2: mode;
 transitions
  choose1 -[random1.update]-> choose2;
  choose2 -[random2.update]-> choose1;
end IntAdderFrame.Impl;
```

Example: Adding Random Integers

$$\mathcal{M} \models \Box\,(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)$$

Example: Adding Random Integers

$$\mathcal{M} \models \Box\,(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)$$



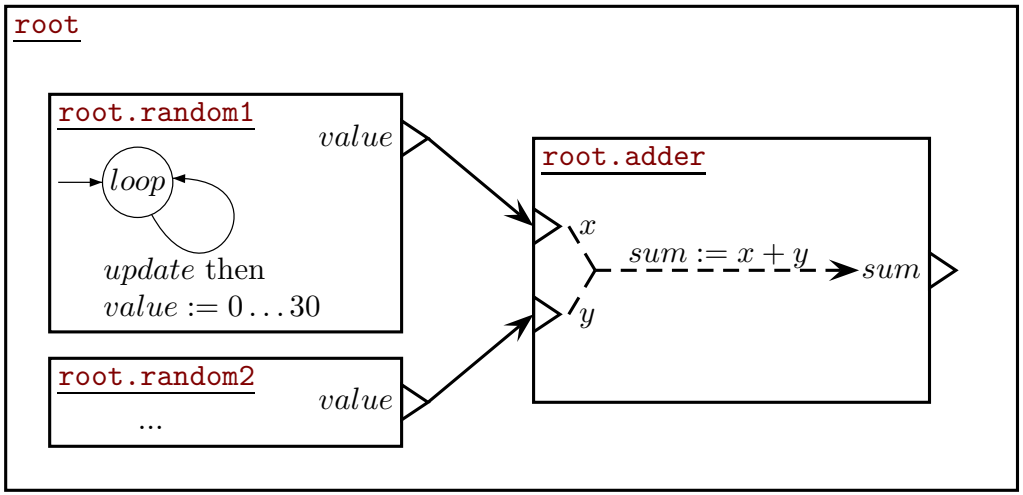
$D = \{$ `random1.value`, `random2.value` $\}$
$E = \emptyset$
$M = \emptyset$

Static Analysis Algorithm

1. Initialize D, E and M
2. Repeat until D, E and M do not change:
   - **2.1** Add source modes of transitions affecting interesting data.
   - **2.2** Add data, events and source modes of interesting transitions.
   - **2.3** Add data and modes of interesting flows.
   - **2.4** Add events and modes of interesting event ports.

Example: Adding Random Integers

$$\mathcal{M} \models \Box\,(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)$$



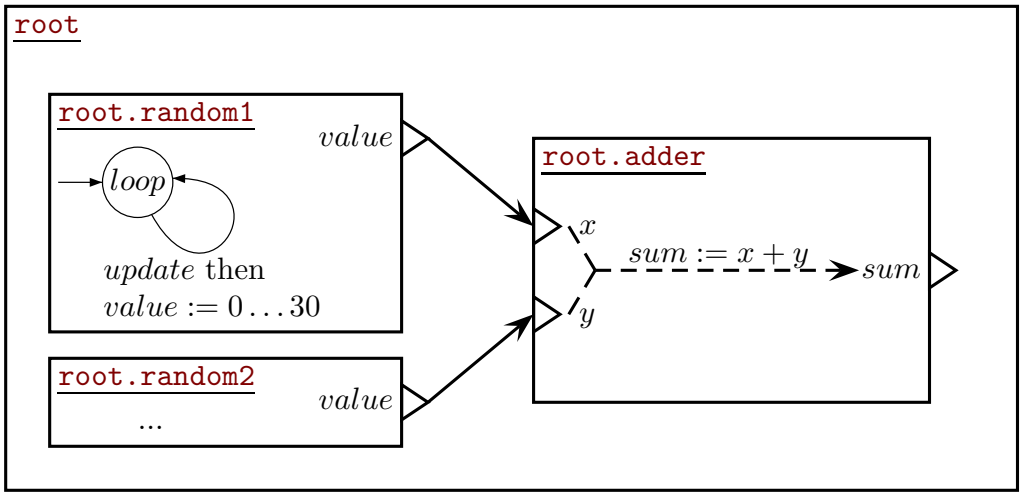
$D = \{$ `random1.value`, `random2.value` $\}$
$E = \emptyset$
$M = \{$ `random1.loop`, `random2.loop` $\}$

Static Analysis Algorithm

1. Initialize D, E and M
2. Repeat until D, E and M do not change:
   - **2.1** Add source modes of transitions affecting interesting data.
   - **2.2** Add data, events and source modes of interesting transitions.
   - **2.3** Add data and modes of interesting flows.
   - **2.4** Add events and modes of interesting event ports.

Example: Adding Random Integers

$$\mathcal{M} \models \Box\,(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)$$



$D = \{$ `random1.value`, `random2.value` $\}$

$E = \{$ `random1.update`, `random2.update` $\}$

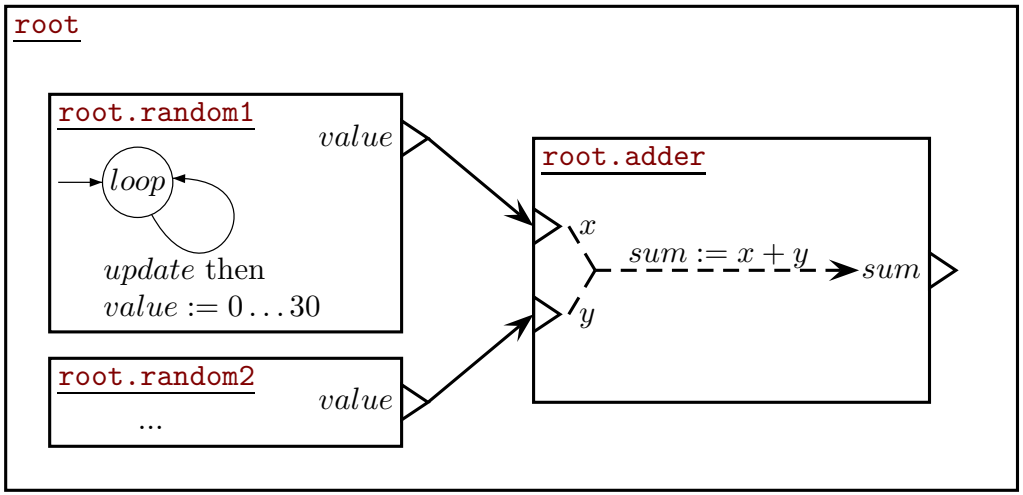
$M = \{$ `random1.loop`, `random2.loop` $\}$

Static Analysis Algorithm

1. Initialize D, E and M
2. Repeat until D, E and M do not change:
   - **2.1** Add source modes of transitions affecting interesting data.
   - **2.2** Add data, events and source modes of interesting transitions.
   - **2.3** Add data and modes of interesting flows.
   - **2.4** Add events and modes of interesting event ports.

Example: Adding Random Integers

$$\mathcal{M} \models \Box\,(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)$$



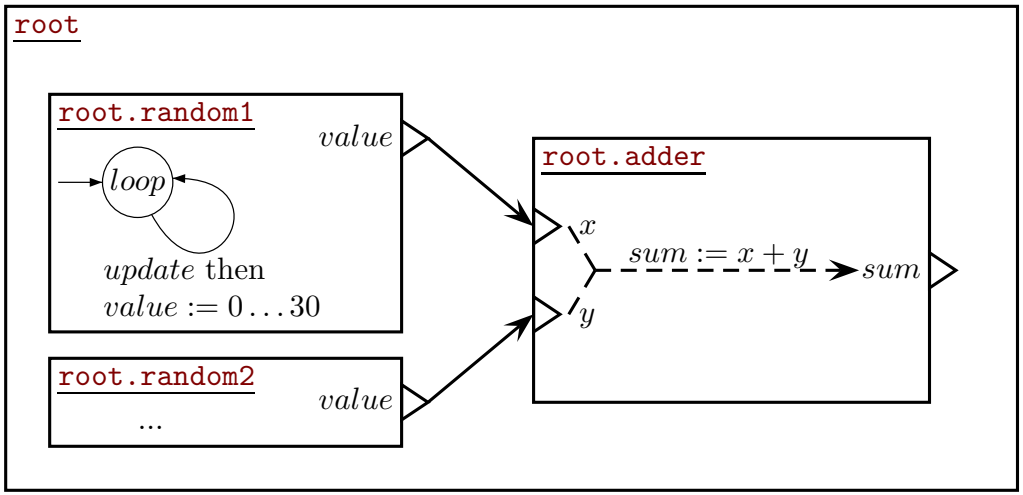
$D = \{$ `random1.value`, `random2.value` $\}$
$E = \{$ `random1.update`, `random2.update` $\}$
$M = \{$ `random1.loop`, `random2.loop` $\}$

Static Analysis Algorithm

1. Initialize D, E and M
2. Repeat until D, E and M do not change:
   - **2.1** Add source modes of transitions affecting interesting data.
   - **2.2** Add data, events and source modes of interesting transitions.
   - **2.3** Add data and modes of interesting flows.
   - **2.4** Add events and modes of interesting event ports.

Example: Adding Random Integers

$$\mathcal{M} \models \Box\,(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)$$



$D = \{$ `random1.value`, `random2.value` $\}$

$E = \{$ `random1.update`, `random2.update` $\}$

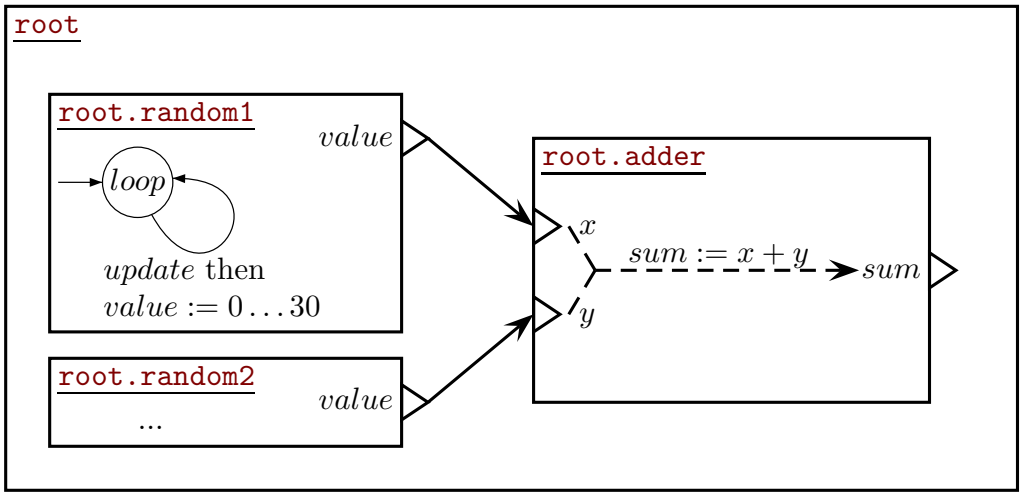
$M = \{$ `random1.loop`, `random2.loop`, `choose1`, `choose2` $\}$

Static Analysis Algorithm

1. Initialize D, E and M
2. Repeat until D, E and M do not change:
   - **2.1** Add source modes of transitions affecting interesting data.
   - **2.2** Add data, events and source modes of interesting transitions.
   - **2.3** Add data and modes of interesting flows.
   - **2.4** Add events and modes of interesting event ports.

Example: Adding Random Integers

$$\mathcal{M} \models \Box\,(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)$$



$D = \{$ `random1.value`, `random2.value` $\}$

$E = \{$ `random1.update`, `random2.update` $\}$

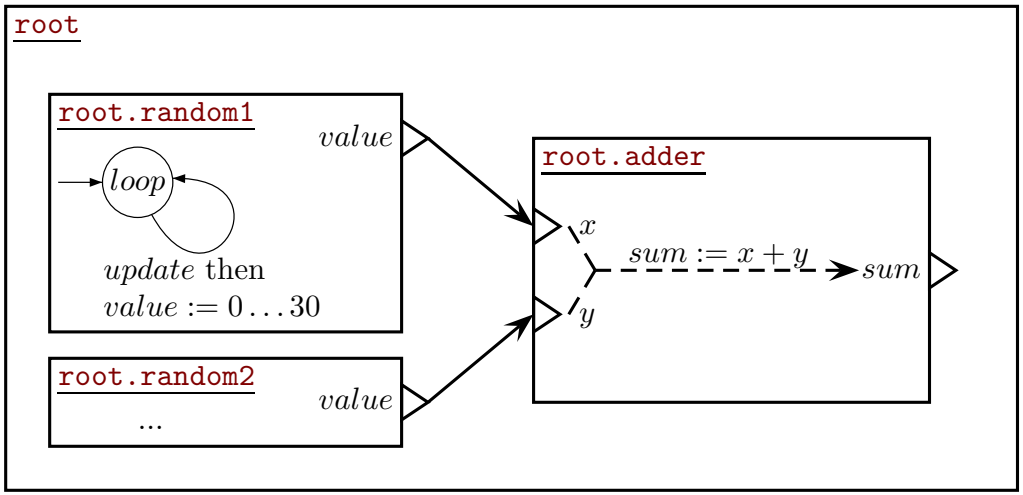
$M = \{$ `random1.loop`, `random2.loop`, `choose1`, `choose2` $\}$

Static Analysis Algorithm

1. Initialize D, E and M
2. Repeat until D, E and M do not change:
   - **2.1** Add source modes of transitions affecting interesting data.
   - **2.2** Add data, events and source modes of interesting transitions.
   - **2.3** Add data and modes of interesting flows.
   - **2.4** Add events and modes of interesting event ports.

Example: Adding Random Integers

$$\mathcal{M} \models \Box\,(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)$$



$D = \{$ `random1.value`, `random2.value` $\}$

$E = \{$ `random1.update`, `random2.update` $\}$

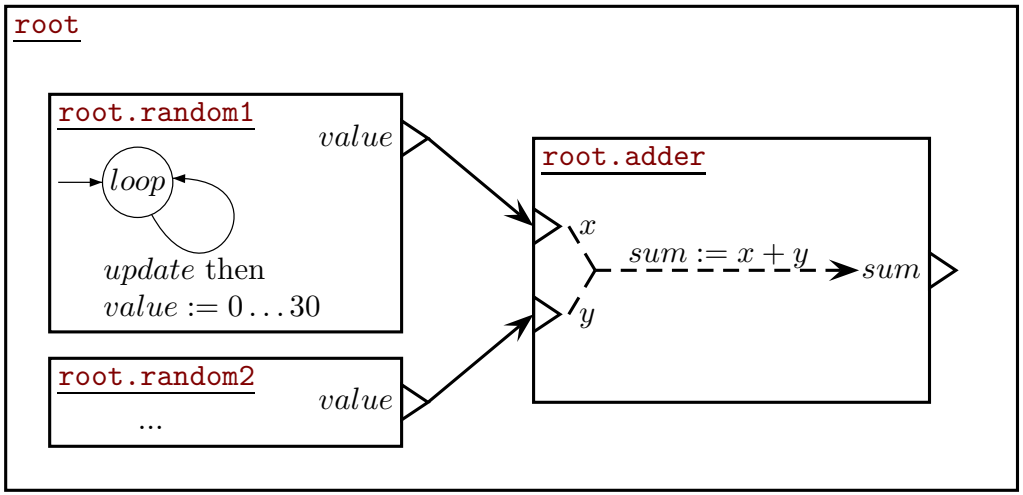
$M = \{$ `random1.loop`, `random2.loop`, `choose1`, `choose2` $\}$

Static Analysis Algorithm

1. Initialize D, E and M
2. Repeat until D, E and M do not change:
   - **2.1** Add source modes of transitions affecting interesting data.
   - **2.2** Add data, events and source modes of interesting transitions.
   - **2.3** Add data and modes of interesting flows.
   - **2.4** Add events and modes of interesting event ports.

Example: Adding Random Integers

$$\mathcal{M} \models \Box\,(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)$$



$D = \{$ `random1.value`, `random2.value` $\}$

$E = \{$ `random1.update`, `random2.update` $\}$

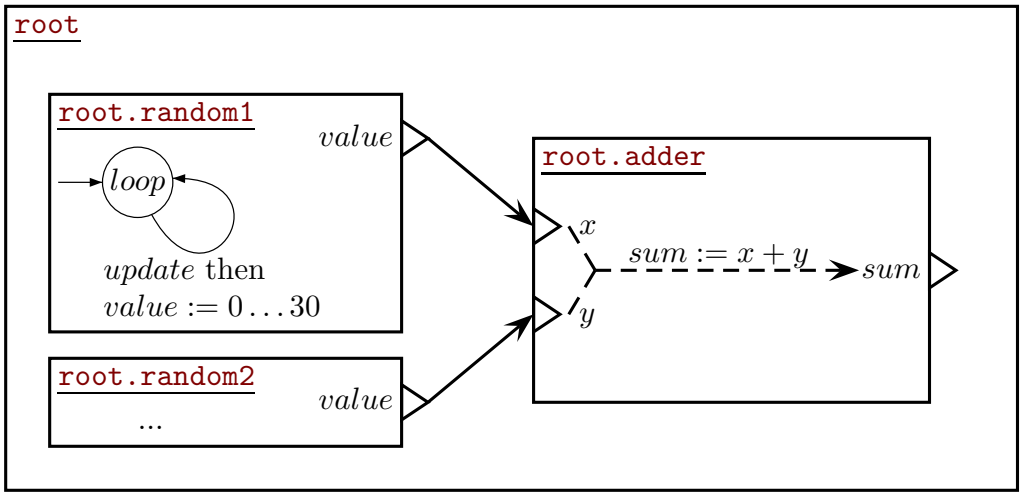
$M = \{$ `random1.loop`, `random2.loop`, `choose1`, `choose2` $\}$

Static Analysis Algorithm

1. Initialize D, E and M
2. Repeat until D, E and M do not change:
   - **2.1** Add source modes of transitions affecting interesting data.
   - **2.2** Add data, events and source modes of interesting transitions.
   - **2.3** Add data and modes of interesting flows.
   - **2.4** Add events and modes of interesting event ports.

Example: Adding Random Integers

$$\mathcal{M} \models \Box\,(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)$$



$D = \{$ `random1.value`, `random2.value` $\}$

$E = \{$ `random1.update`, `random2.update` $\}$

$M = \{$ `random1.loop`, `random2.loop`, `choose1`, `choose2` $\}$

Static Analysis Algorithm

1. Initialize D, E and M
2. Repeat until D, E and M do not change:
   - **2.1** Add source modes of transitions affecting interesting data.
   - **2.2** Add data, events and source modes of interesting transitions.
   - **2.3** Add data and modes of interesting flows.
   - **2.4** Add events and modes of interesting event ports.

Example: Adding Random Integers

$$\mathcal{M}_{sliced}^{\Box(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)}$$

Example: Adding Random Integers

$$\mathcal{M}_{sliced}^{\Box(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)}$$

```
system RandomIntValue
 features
  value: out data port int
         default 2;
  update: in event port;
end RandomIntValue;

system implementation
  RandomIntValue.Impl
 modes
  loop: activation mode;
 transitions
  loop -[update then
         value := 0]-> loop;
  loop -[update then
         value := 1]-> loop;
  -- i.d. value := 2,...,29
  loop -[update then
         value := 30]-> loop;
```

```
system IntAdderFrame
end IntAdderFrame;

system implementation
  IntAdderFrame.Impl
 subcomponents
  random1: system RandomIntValue.Impl;
  random1: system RandomIntValue.Impl;
 modes
  choose1: activation mode;
  choose2: mode;
 transitions
  choose1 -[random1.update]-> choose2;
  choose2 -[random2.update]-> choose1;
end IntAdderFrame.Impl;
```

Example: Adding Random Integers

Benchmarks Sliced versus Unsliced using AADL to Promela Translator[1]

Specification	Mem/State (bytes)	#States	Memory (MBs)	Time (seconds)
Unsliced	136	1,676,026	272	6.0 - 7.3
Sliced for Φ_1	116	1,437,691	211	5.4
Sliced for Φ_2	84	553,553	84	1.4
Sliced for Φ_3	76	9,379	33	0.1

$$\Phi_1 \equiv \Box\,(0 \leq adder.sum \leq 60)$$

$$\Phi_2 \equiv \Box\,(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)$$

$$\Phi_3 \equiv \Box\,(0 \leq random1.value \leq 30)$$

[1] AADL to Promela translator will be submitted to SPIN2010

Example: Adding Random Integers

Benchmarks Sliced versus Unsliced using AADL to Promela Translator[1]

Specification	Mem/State (bytes)	#States	Memory (MBs)	Time (seconds)
Unsliced	136	1,676,026	272	6.0 - 7.3
Sliced for Φ_1	116	1,437,691	211	5.4
Sliced for Φ_2	84	553,553	84	1.4
Sliced for Φ_3	76	9,379	33	0.1

$$\Phi_1 \equiv \Box\,(0 \leq adder.sum \leq 60)$$

$$\Phi_2 \equiv \Box\,(0 \leq random1.value \leq 30 \wedge 0 \leq random2.value \leq 30)$$

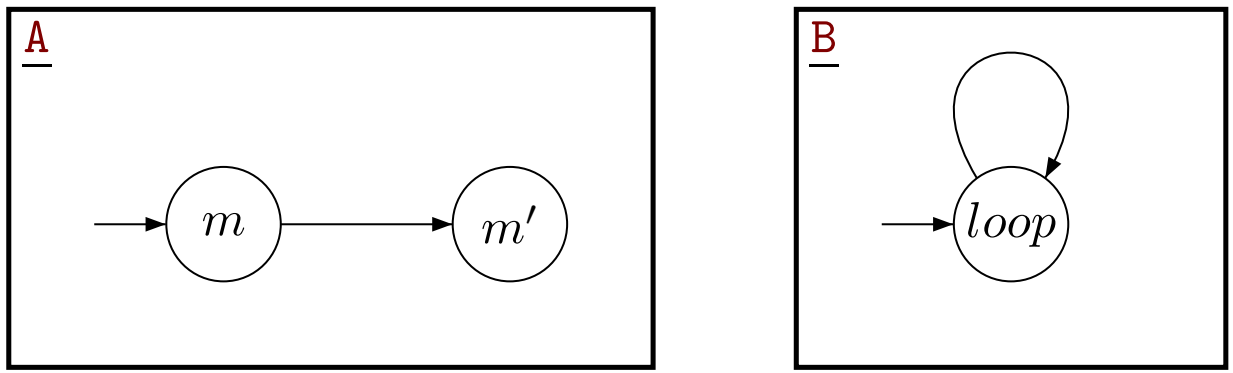
$$\Phi_3 \equiv \Box\,(0 \leq random1.value \leq 30)$$

More benchmarks with similar results: negator, redundant battery system, wind turbine, satellite's thermal regulator!

[1] AADL to Promela translator will be submitted to SPIN2010

Remarks

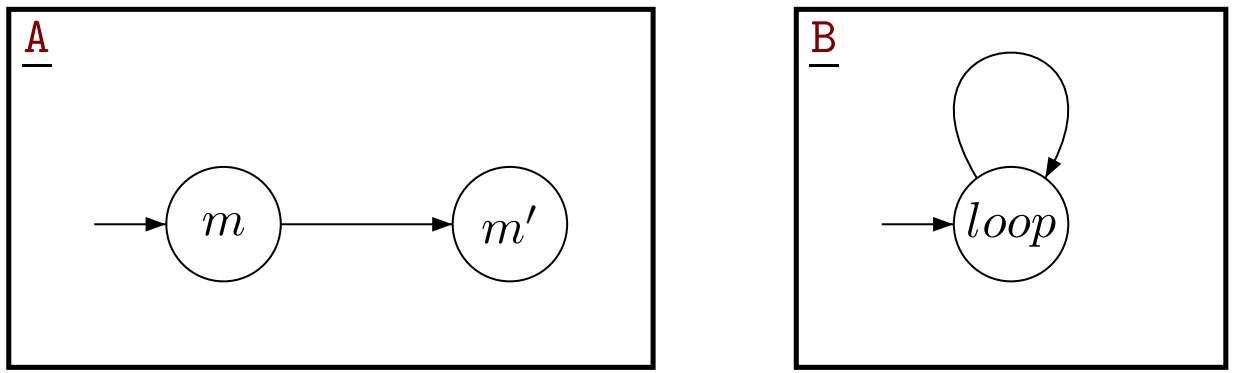
- Preserving divergence characteristics.
  For example, $M \not\models \Diamond m'$ for the model below but when component B is sliced away, $M \models \Diamond m'$.



Solution: static cycle detection and include all components with possible divergence behaviour.

Remarks

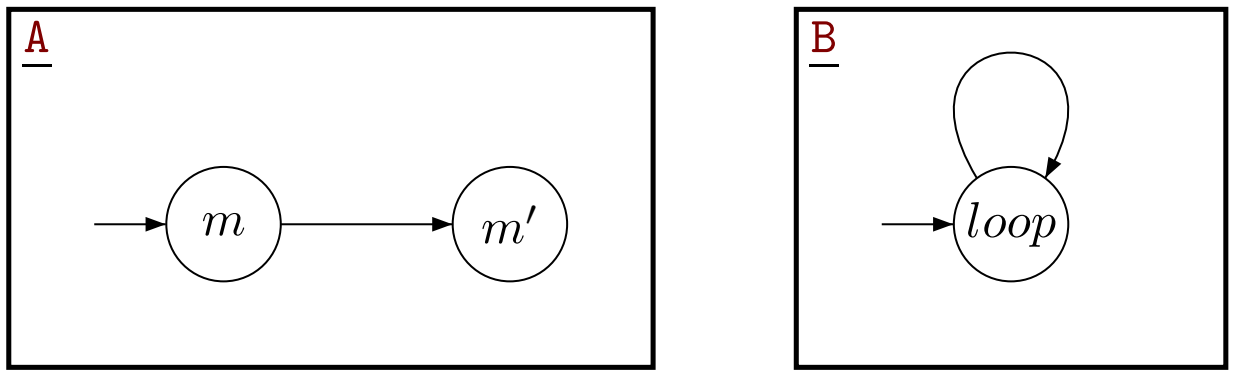
- Preserving divergence characteristics.
  For example, $M \not\models \Diamond m'$ for the model below but when component B is sliced away, $M \models \Diamond m'$.



  Solution: static cycle detection and include all components with possible divergence behaviour.
- Refinement by distinguishing component-local and global data elements.

Remarks

- Preserving divergence characteristics.
  For example, $M \not\models \Diamond m'$ for the model below but when component B is sliced away, $M \models \Diamond m'$.



  Solution: static cycle detection and include all components with possible divergence behaviour.
- Refinement by distinguishing component-local and global data elements.
- *Initial* work-progress on preserving hybrid behaviour.

Summary

We developed a property-driven slicing algorithm for AADL that preserves CTL*. Benchmarks show an expected significant reduction in model checking time and memory on the sliced models.

Summary

We developed a property-driven slicing algorithm for AADL that preserves CTL*. Benchmarks show an expected significant reduction in model checking time and memory on the sliced models.

References:

- M. Odenbrett, V.Y. Nguyen and T. Noll. *Slicing AADL Specifications for Model Checking*. In proceedings of NFM2010.
- M. Odenbrett. *Explicit-State Model Checking of an Architectural Design Language using SPIN*. Master's thesis, RWTH Aachen University 2010.
- M. Odenbrett, V.Y. Nguyen and T. Noll. *Model Checking AADL Specifications with SPIN*. To be submitted to SPIN2010.
- Bozzano et al. *Safety, Dependability, and Performance Analysis of Extended AADL Models*. In Oxford's Computer Journal 2010.
- COMPASS Website: `compass.informatik.rwth-aachen.de`.