Two formal semantics of a Subset of AADL

Z. Yang[1], K. Hu[1], J.-P. Bodeveix[2], L. Pi[2], D. Ma[1], J.-P. Talpin[3]

[1]BUAA, Beijing, China
[2]IRIT - Université Paul Sabatier, Toulouse, France
[3]INRIA-Rennes, Campus de Beaulieu, Rennes, France

April 2011

Plan

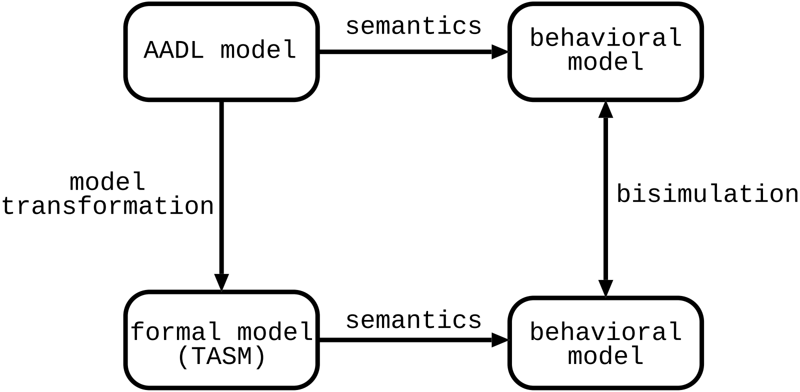
1. Introduction
2. Formalisms
   - TASM
   - Semantics domain: TTS
   - Coq
3. Verification
4. Conclusion

Context

Principles

- **Goal**: verification of AADL models
- **Method**: translation to an analysable formal language (TASM here)
- **Problem**: correctness of the transformation
- **Verification**: Semantics-preserving transformation

Semantics-preserving transformation



Formalisms

- AADL:synchronous subset (periodic threads, immediate/delayed communications)
- Formal analysable language: TASM (Timed abstract state machine)
- Semantics domain: TTS (Timed Transition systems)
- Meta-language: Coq (interactive proofs)

Plan

TASM (1)

TASM [Lundqvist, MIT]: Extension of abstract state machines

- system: set of concurrent machines
- machine: set of transitions updating local or shared variables
- time: duration associated to a transition
- resources: consumed during transition execution

TASM (2)

```
computed: bool
write: bool
read: bool
```

producer

```
R:
{
  t := 7;
  if write = false then
    write := TRUE;
}
```

compute

```
R0:
{
  t := 0;
  if write = true then
    write := false; read := true;
}

R1:
{
  t := 5; mem := 50;
  if read = true then
    read := false; computed := true;
}

R2:
{
  t := 0;
  else then skip
}
```

TASM (3)

Semantics

- One machine: transition system
  - select a rule with satisfiable guard
  - compute update set (next value of updated variables)
  - wait for transition duration, consume (additively) resources
  - update environment
- Composition of machines:
  - Asynchronous if zero-time transitions
  - Synchronous if non-zero time transitions

Timed Transition Systems

Definition

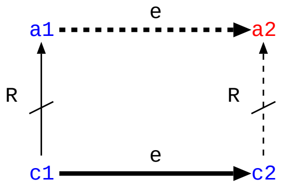
A TTS over a set of events Σ is a nuple

$$\langle Q, Q^0, P, \longrightarrow, \models \rangle$$

where

- Q is a set of states,
- $Q^0 \subseteq Q$ is the set of initial states
- P is a set of observables predicates
- $\_ \xrightarrow{-} \_ \subseteq Q \times (\mathbb{T} \cup \Sigma) \times Q$ is the timed transition relation
- $\models \subseteq Q \times P$ is a satisfaction relation.

Simulation relation



- $\forall c,\ c \in Q_c^0 \Rightarrow \exists a,\ R(c, a) \wedge a \in Q_a^0$
- $\forall c_1 c_2 a_1 e,$
  $$c_1 \xrightarrow{e} c_2 \wedge R(c_1, a_1)$$
  $$\Rightarrow$$
  $$\exists a_2,\ a_1 \xrightarrow{e} a_2 \wedge R(c_2, a_2)$$

Compositional proof

Principle

$$\frac{\forall i, \text{TTS\_TASM}_i \sim \text{TTS\_AADL}_i}{\Pi_{i=1..n}\text{TTS\_TASM}_i \sim \Pi_{i=1..n}\text{TTS\_AADL}_i}$$

Synchronous product of TTS

$$\langle Q_1, Q_1^0, P_1, \longrightarrow_1, \models_1 \rangle \otimes \langle Q_2, Q_2^0, P_2, \longrightarrow_2, \models_2 \rangle$$
$$=$$
$$\langle Q_1 \times Q_2, Q_1^0 \times Q_2^0, P_1 \cup P_2, \longrightarrow, \models_1 \cup \models_2 \rangle$$

where

$$\frac{q_1 \xrightarrow{e}_1 q_1' \quad q_2 \xrightarrow{e}_2 q_2'}{(q_1, q_2) \xrightarrow{e} (q_1', q_2')}$$

Coq and proof assistants

Several languages (maybe unified)

- typed functional language (definition of functions)
- rich type system (functions over types, dependent types, ...)
- assertional language (definition of properties)
- proof building language (assisted proofs)
- proof tactic definition language (proof automation)

Coq (2)

Use of Coq

- Definition of TTS, product, simulation
- Definition of TASM and AADL abstract syntax
- Definition of TASM and AADL semantics (as a product of TTS)
- Definition of AADL to TASM translation.
- Proof of bisimulaiton.

Feasability validated on a small subset of AADL.

Plan

1. Introduction
2. Formalisms
   - TASM
   - Semantics domain: TTS
   - Coq
3. Verification
4. Conclusion

Abstract syntax of an AADL subset (Coq)

```
Record Thread: Type := {
  WCET: Duration;
  period: Duration;
  dealine: Duration;
  Iports: Set;
  OPorts: Set
}.

Record Model: Type := {
  thId: Set;
  threads: thId → Thread;
  connections: ∀ th, Iports (threads th) →
    {th: thId & Oports (threads th)};
  connectionType: ∀ th, Iports (threads th) → bool
}.
```

Abstract syntax of a TASM subset

$$
\begin{array}{lll}
\texttt{P} & ::= & \texttt{Ident := exp} \\
& | & \texttt{skip} \\
& | & \texttt{channel!} \\
& | & \texttt{channel?} \\
& | & \texttt{if exp then P} \\
& | & \texttt{time min .. max} \rhd \texttt{P} \\
& | & \texttt{time next} \rhd \texttt{P} \\
& | & \texttt{resource r (min,max)} \rhd \texttt{P} \\
& | & \texttt{P} \oplus \texttt{P} \quad \textit{choice} \\
& | & \texttt{P} \otimes \texttt{P} \quad \textit{multi assignment}
\end{array}
$$

$$
\texttt{TASM ::=} \langle \texttt{Env, P} \parallel \texttt{...} \parallel \texttt{P} \rangle
$$

TASM semantics - TTS state space (Coq)

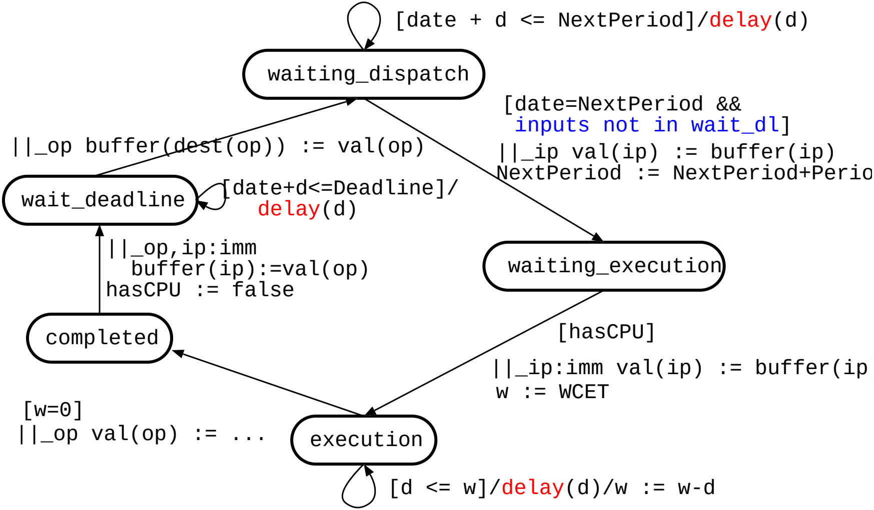
Variable to be updated at end of transition

```
Record UpdateSet: Type := mkUS {
  date: Time;
  updated: Vars sys -> Prop;
  update: forall v, updated v -> Value sys v
}.
```

TASM state

```
Record TASMState: Type := mkTASMState {
  currentTime: Time;
  updateSets: mmId sys -> option UpdateSet;
  currentEnv: Env (Vars sys) (Value sys)
}.
```

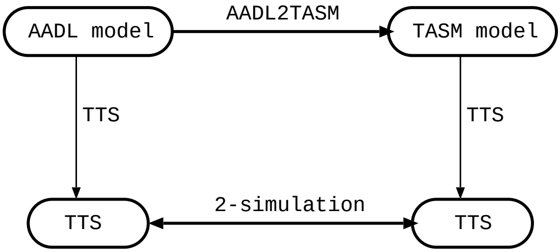
Operational Semantics of the AADL subset



Translation of AADL to TASM

```
Trans_Thread(th) =
// dispatch
time 0 ▷
if state(th) = waiting_dispatch and
  ∀ thi ∈ prec(th), state(thi)≠wait_deadline then
  state(th) := waiting_execution ⊗
  ⊗ip∈Iports(th) val(ip) := buffer(ip)
⊕
// waiting execution
time 0 ▷
if state(th)=waiting_execution and hasCPU(th) then
  state(th) := execution ⊗
  ⊗ip∈Iports(th) ∪ Imm val(ip) := buffer(ip)
⊕ ...
```

The main theorem (1)



The main theorem (2)

Coq statement (for one thread)

```
Theorem Thread2MM_simu1:
  ∀ th, simu _ _ A2T (ThreadPred sys th)
        (MM_TTS AADL2TASM th)
        (Thread_TTS sys th)
        (AP2TLP th)
        (P2LP th).

Theorem Thread2MM_simu2:
  ∀ th, simu _ _ T2A (ThreadPred sys th)
       (Thread_TTS sys th)
       (MM_TTS AADL2TASM th)
       (P2LP th)
       (AP2TLP th).
```

Plan

1. Introduction
2. Formalisms
   - TASM
   - Semantics domain: TTS
   - Coq
3. Verification
4. Conclusion

Conclusion and perspectives

Conclusion

- Definition of the semantics of a (small) fragment of AADL.
- Definition of the semantics of a fragment of TASM.
- AADL2TASM transformation and correctness proof.
- The proof is tedious and too dependent on AADL

Future work

- A higher level formalism to express the reference semantics.
- Larger coverage of AADL.
- Translation of this language to analysable languages.
- Verification of this translator.