Whitepaper
A timing annex for the AADL

AADL Workshop
Valencia, February 4-5th., 2013

Loïc Besnard, CNRS

Thierry Gautier, Paul Le Guernic, Jean-Pierre Talpin, INRIA
Whitepaper
A timing annex for the AADL

PLAN

Goal of the proposal

State of the art and background

A model of time and clock automata

Timing AADL concepts and annexes

Workplan
Goals

A formal definition of the AADL in a synchronous multi-clocked model of computation and communication

• A formal, thorough and unambiguous specification of the AADL

• A framework for verification
  • Multi-clocked synchronous automata
  • Compatible with PSL use with CA and Maude

• A framework for implementation
  • Synthesis of reactive and control systems
  • Globally asynchronous locally synchronous architectures

• A pivot model of automata: behavioral annex, SyncCharts, StateCharts, StateFlow, Esterel, Mealy machines, …
Recommendations

Discussions during the previous meeting (Peter, Jérome, …)

• Choose where a timing model best fits
  • core (property set)
  • annex (TA)
  • Review implications/links on/with annexes

• Isolate synchronous subset/profile of AADL v2

Consensus
• An abstract, relational, logical model of time
• Define clock domains and protocols
• Causal scheduling constraints between threads
• Propose a refinement methodology (from clocks to ports)
Approach

A software-centric model of time

• A syntactically light-weight proposal
  • Time as sensed from software
  • Synchronization, causality, clocks, relations

• Based on Polychrony and the MARTE/CCSL standard

• How does it extend the AADL synchronous core?
• Implications on the behavioral annex
  • Model of reactive automata
  • Model of constrained automata (controller synthesis)
• Implication on the constraint annex
  • Expression of timing relations
  • Expression of regular expressions over time
Implications and (modular) containment

- Refines (or abstracts) core timing property specifications
- Implications on the behavioral annex (timed/clocked automata)
- Implications on the constraint annex (timing invariants)
- Compatibility with the ARINC annex (previous work)
- A possibly extensible proposal (as CCSL)
  - Abstraction/refinement between time domains
  - Hardware real-time (discrete)
  - Physical real-time (continuous)
Whitepaper
A timing annex for the AADL

PLAN

Goal of the proposal

State of the art
A model of time and clock automata
Timing AADL concepts and annexes

Workplan
State of the Art – Verification

• “Expressing and enforcing user-defined constraints of AADL models”. Olivier GILLES, Jerome HUGUES. IEEE ICECCS, 2010.


Definition of specification formalisms based on PSL to formally express and verify (synchronous) timing properties of AADL objects
State of the Art – Semantics


Definition of an implicitly synchronous subset or profile of the AADL amenable to formal verification
Whitepaper
A timing annex for the AADL

PLAN

Goal of the proposal

Background
A model of time and clock automata
Timing AADL concepts and annexes
Workplan
Background on AADL


Synchronous multi-clocked interpretation, simulation, verification and scheduling analysis
Background on CCSL


Controller synthesis from timing constraints
Whitepaper
A timing annex for the AADL

PLAN

Goal of the proposal
State of the art and background

A model of time
Timing AADL concepts and annexes
Workplan
On the varieties of time domains

• Embedded system
  • Interacts with (distributed) physical systems: continuous times with uncertainty
  • Several (distributed) actual HW components (processors,...) each with its own physical clock
  • Several software components, interact wrt logical clocks

• Designing the architecture
  • Simulation time
  • Simulated time

• Designing the system (with or without HW architecture)
  • Modeller in the loop: simulated continuous time
  • (Synchronous) multi-level abstraction of architecture
Requirements for a model of time

- Must permit discrete behaviour
  - To model steps
  - To define delay

- Must allow synchronisation of events
- Must allow hierarchical computation
  Dense for « oversampling »

- Must permit interfacing with continuous time
- Must allow distributed implementation
  Partial order

- Must allow semantic fix point computation
Requirements for a model of time

- A (data) event associates a value to a logical instant (a tag)
- The set of tags is
  - Dense
  - Directed Complete Partial Order

- Typically
  - A poset in $\mathbb{R}^n$, where $n$ is the number of distributed physical clock
  - A (sequential) super-dense time
  - A (sequential) non standard time

- The set of (data) events in a run is a discrete Directed CPO
- Synchronisation is an equivalence relation over events
- A signal is a chain of (data) events
A model of time

How does software sense time?

- **Communicational synchronization**
  - Instantaneous (just wait)
  - Singular, sporadic
- **Computational causality**
  - Time-consuming
  - Repetitive, periodic

```
int flip = 1;
forever {
  if flip {
    read(&a);
    pause;
  } else {
    read(&b);
    pause;
  }
}
```
A model of time

- Event
  First occurrence $a_0$ of awaiting $a$

- Clock
  Awaiting $a_0\ a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ ...$

- Relation
  Alternation of reading $a$ and $b$

```c
int flip = 1;
forever {
  if flip {
    read(&a);
    pause;
  } else {
    read(&b);
    pause;
  } else {
    //...}
}
```

$\text{AnotB\_change := not (AnotB\_change \^= ^0)}$

```c
int flip = 1;
forever {
  if flip {
    read(&a);
    pause;
  } else {
    read(&b);
    pause;
  } else {
    //...}
}
```
A model of time

- Software-centric time
- Relation to computation time
- Relation to communication delay
- Abstraction/refinement of software time vs. real-time

```c
int flip = 1;
forever {
    if flip {
        read(&a);
        pause;
    } else {
        read(&b);
        pause;
    }
}
```
A model of time

An event algebra

[^*] infimum    ^+    supremum
[^0] minimum    ^1    maximum in the scope of V

A set of formulas $F_{V,S}$ on states $S$ and variables $V$

- $^0, ^1 \in F_{V,S}$ constant
- $S \subseteq F_{V,S}$ atom
- $^x, [x], [\neg x] \in F_{V,S}$ Boolean
- $\neg f \in F_{V,S}$ unary
- $f ^+ g, f ^* g, f ^- g \in F_{V,S}$ binary

States are exclusive atoms $(\forall s_1, s_2) s_1 ^* s_2 = ^0 \text{ or } s_1 = s_2$

Variables are free atoms $I_V = \sum ^v | v \in V$
Timing specifications are safety properties defined on clocks and relations between them

\[
\text{property} \quad ::= \text{constraint}\,^1 \, \text{relation}\,*
\]

\[
\text{relation} \quad ::= \text{forever} \, \text{clk} \quad \omega\text{-iteration on clk}
| \quad \text{never} \, \text{clk} \quad \text{complementary} \quad (\text{forever} \, ^\sim\text{clk})
| \quad \text{sync} \, \text{clk} \, \text{clk} \quad \text{synchronization} \quad (\text{forever} \, \text{clk}^\sim\text{clk}')
\]

\[^1\quad \text{or restrict of PSL or guarantee}\]
Clocks are Heiting-algebraic expressions on states, transitions, variables

\[
clk ::= \, ^0 \quad \text{never} \\
| \, ^x \quad \text{on the clock (of) } x \\
| \, \text{in } s \quad \text{in a state} \\
| \, \text{at } l \quad \text{firing a transition } l \\
| \, \text{when } x \quad \text{when } x \text{ is true} \\
| \, clk \, ^+ \, clk \quad \text{union} \\
| \, clk \, ^* \, clk \quad \text{intersection} \\
| \, clk \, ^- \, clk \quad \text{difference}
\]
A model of time

Clocks algebra naturally extends to a Kleene algebra of regular expressions on events (with counting)

<table>
<thead>
<tr>
<th>reg ::=</th>
<th>clk</th>
<th>event</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>reg ; reg</td>
<td>concatenation</td>
</tr>
<tr>
<td></td>
<td>reg + reg</td>
<td>sum</td>
</tr>
<tr>
<td></td>
<td>reg*</td>
<td>star</td>
</tr>
<tr>
<td></td>
<td>reg?</td>
<td>option (i.e. 1 + reg)</td>
</tr>
<tr>
<td></td>
<td>reg[n]</td>
<td>counting</td>
</tr>
<tr>
<td></td>
<td>reg : reg</td>
<td>fusion</td>
</tr>
<tr>
<td></td>
<td>reg</td>
<td>reg</td>
</tr>
</tbody>
</table>
A model of time for AADL

**Proposal (core)**
Boolean expressions are core AADL Boolean expressions
Extend AADL triggers with the `clk` grammar to build clocks
Allow AADL to access/store the previous/next value of a flow

**Proposal (constraints)**
Extend PSL SEREs (regular expressions over clocked Booleans with counting) to regular expressions on events
Represent events with Booleans and clocks
Extend AADL data types with events
Extend PSL data types to the AADL

**Objective**
a common language for safety properties
A Boolean layer extended with events
An event regular expression layer to define properties
Periodic clocks and affine relations

\[ \text{clk ::= } \ldots \mid \text{every } n \text{ clk } [\text{phase } m] \quad \text{affine period} \]

\[ \text{sync } c \text{ every } n \text{ clk } = \text{forever} \ (c^*\text{clk};\text{clk}[n-1]) \]

Counting automata

\begin{align*}
\text{S1 - clk [10]} & \rightarrow\rightarrow \text{ S2} & \text{skip 10 occurrences of clk to reach S2} \\
\text{S1 - clk[10..20]} & \rightarrow\rightarrow \text{ S2} & \text{skip 10 to 20 occurrences of clk} \\
& & \text{i.e. } \text{S1 - clk[10];clk?[10]} \rightarrow\rightarrow \text{ S2}
\end{align*}
Periodic and affine clocks

Methodology

Define abstraction/refinement relations between symbolic clock relations and real-time constraints

A period => 10ms implements sync B every 2A
B period => 20ms

forever (A^B; A^*B) + (A^*B; A^B)

forever (A^B; (A | B))

Checking time compatibility (e.g. between port and thread)
Checking timing refinement (e.g. from symbolic to real-time schedules)
Whitepaper
A timing annex for the AADL

PLAN

Goal of the proposal
State of the art and background
Clock automata
Timing AADL concepts and annexes
Workplan
钟自动机

未计时（未受限制）BA自动机是
• 反应式（时间由环境感应）
• 检查确定性
• 同步可组合的

计时（受限制）BA自动机是
• 内源性（时间可以被内部计算）
• 合成控制器从逻辑计时属性
• 同步可组合的

时序模型扩展了BA和CA
• 反应式自动化可以被表达
• PSL属性可以被表达
• 他们可以被组合
• 他们可以操纵部分相关的时钟域
A reactive single-clocked automata

Untimed (unconstrained) BA automata are
- Reactive (time is sensed from the environment)
- Checked deterministic and synchronously composable

Example: observe alternation of a and b

Example - a reactive automaton

\[
\begin{align*}
\text{awaiting\_AnotB} &:= (\neg \text{AnotB\_change}) \text{init true} \\
\text{AnotB\_change} &:= \text{a when awaiting\_AnotB} \\
& \quad \text{default not b when not awaiting\_AnotB}
\end{align*}
\]
Timed (constrained) BA automata are
- Endochronous (time can be computed internally)
- Synthesized controllers from logical timing properties
- Synchronously composable

Example: enforce alternation of a and b by a global constraint
A reactive multi-clocked automata

The timing model extends the BA and CA
  • They can be expressed and combined
  • They can manipulate partially related clock domains

Example: observe alternation on two clock domains
The timing model extends the BA and CA
- They can be expressed and combined
- They can manipulate partially related clock domains

Example: enforce alternation (by state-dependent constraints)
Another example: Esterel’s notorious ABRO

```plaintext
loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```

```
Esterel's ABRO example

Example – a multi-clocked permissive automaton

loop
   [ await A || await B ]; emit O
each R
```
An automaton $A$ is a tuple $A = (S_A, s_0, \rightarrow_A, V_A, T_A, \text{NULL}_A)$

- $S_A$ is the non empty set of states, $s_0$ is the initial state
- $\rightarrow_A \subseteq S_A \times S_A$ is the transition relation
- $V_A$ is the set of signal variables
- We denote by $F_{A,S}$ the set of formulas in $\Phi(V_A, S_A)$

- $T_A : \rightarrow_A \mapsto F_{A,S}$ is the function that assigns a formula to a transition. It satisfies $(\forall s, s1, s2 \in S_A) (s \text{ does not occur in } T_A((s1, s2)))$

- $\text{NULL}_A$ is a formula in $F_{A,S}$ that is (constrained to be) null
  - a formula $f$ in $F_{A,S}$ is null in $A$ iff $f \land \text{NULL}_A = f$
  - if $\text{NULL}_A$ is $^0$, the automaton is constraint free
  - if $\text{NULL}_A$ is $^1_{V_A}$ all formulas are null
An example of clock automata

Alternate = ( 
  S: {S_1, S_2},  
s_0: S_1,  
→: {(S_1, S_2), (S_2, S_1)},  
V: {a, b},  
T: (S_1, S_2) → a (S_2, S_1) → b,  
NULL: a^* b^+ [-a]^+ [-b] )

Transitions e.g. T = S_1 -- h ->> S_2,  
S1 is the source of T  
S2 is the target of T  
h is the trigger of T, denoted trigger(T) and a trigger in S_1  
T is enabled at k iff k^*h is not null and current state is S_1
Alternate = ( 
\begin{align*}
S: & \{S_1, S_2\}, \\
s_0: & S_1, \\
\rightarrow: & \{(S_1, S_2), (S_2, S_1)\}, \\
V: & \{a,b\}, \\
T: & (S_1, S_2) \rightarrow a \ (S_2, S_1) \rightarrow b, \\
NULL: & a^*b^+ [-a]^+ [-b] \\
\end{align*}

Steps

A step is an implicit not exiting reflexive transition
A step is enabled at k iff k^*h is not null, current state is S1 and there is no enabled transition.
All steps that are not explicitly forbidden are allowed

S1 -- b^-a ->> S1
S2 -- a^-b ->> S2
An example of clock automata

Alternate = (  
S: \{S_1, S_2\},  
s_0: S_1,  
\rightarrow:\{(S_1,S_2), (S_2,S_1)\},  
V:\{a,b\},  
T:\ (S_1,S_2)\rightarrow a (S_2,S_1)\rightarrow b,  
\text{NULL: } a^*b ^+ [-a] ^+ [-b] )

State S
Exiting on (one of the) enabled transitions is mandatory
The set of transitions of source S are “transitions from S”
trigger(s), the trigger of a state s is the upper bound of triggers in s
Properties of clock automata

For an automaton \( A \) and \( V_A \) its set of variables, a state \( s \) in \( A \)

\[ I_A(s) = \Delta \text{ NULL}_A \overset{\mathcal{V}_A}{\wedge} s \]

There exists at most one labeled transition from a given source \( s_1 \) to a given target \( s_2 \). Two labeled transitions \( h_1: s_1 \to_A s_2 \) and \( h_2: s_1 \to_A s_2 \)
can be replaced by \( (h_1 \overset{\mathcal{V}}{\wedge} h_2): s_1 \to_A s_2 \)

Automaton with empty set of transitions

\( \emptyset_A = (\{s\}, s, \emptyset, A, \emptyset, \emptyset, ^1 A) \) blocks all occurrences of all \( a \) in \( A \)

Automata with empty set of variables

\( \mathcal{I} = \mathcal{I}_\emptyset = (\{s\}, s, \emptyset, \emptyset, \emptyset, ^0) \)

\( \emptyset_\emptyset = (\{s\}, s, \emptyset, \emptyset, \emptyset, ^1 \emptyset) \equiv \mathcal{I} \)
Properties of clock automata

Labeled transitions “\(h:s_1 \xrightarrow{A} s_2\)” mean \((s_1, s_2) \in \xrightarrow{A}\) and \(T_A(s_1, s_2) = h\)

- The context clock of an automaton \(A\) is \(^1_A (= \sum_{x \in VA} (^x)))\)
- In \(h:s_1 \xrightarrow{A} s_2\), \(h\) is the trigger of \((s_1, s_2)\) and a trigger of \(s_1\)
- The trigger of a state \(s\), \(\text{trigger}(s)\) is the upper bound of triggers in \(s\)

- The null clock of a state \(s\) is \(\text{NULL}_A(s)\) defined as the simplified positive Shannon cofactor (for atom “[s]”) of \(\text{NULL}_A \hat{\ast} s\) occurrences of \(s\) are replaced by \(^1_A\) and if \(t\) is not \(s\), occurrences of \(t\) are replaced by \(^0\)

- When \(h = 1_A \hat{\ast} (\text{NULL}_A(s) \hat{\ast}\text{trigger}(s))\) is not null, there is a (virtual/stuttering) transition \(h:s \xrightarrow{A} s\) named step
A clock automata language

In summary, many nice properties

- Composable normal form
- Weak ands strong interpretation
- Priorities and hierarchies
- Composites states and modes (exported states)
- Morphism between clock automata and regular expressions

automaton alternate =
  state S1, S2;
  never {b in S1, a in S2};
  transition {
    S1 -- a --> S2;
    S2 -- b --> S1; }
end;

regexp alternate =
  (a^b) --> S2
  where {
    S1 = (a^-b) --> S2;
    S2 = (b^-a) --> S1;}

regexp short-alternate =
  ((a^-b) --> (b^-a) )*;
Whitepaper
A timing annex for the AADL

---

**PLAN**

Goal of the proposal

State of the art and background

A model of time and clock automata

**Timing the behavioral annex**

Workplan
A simple reactive automaton

thread alternate
annex behavior_specification {**
    states
    s1: initial state;
    s2: state;
    transitions
    t1: s1-[on dispatch a]->s2;
    t2: s2-[on dispatch b]->s1;
**};
end alternate;
Clock automata in the behavioral annex

Refined and constrained by a clock property

```plaintext
thread implementation alternate.alt1
annex behavior_specification {**
  constraints
    never a ^* b;
**}
end alternate.alt1;
```
Clock automata in the behavioral annex

A multi-clocked reactive automaton

Thread alternate.alt2
annex behavior_specification { **
  states
    s1: initial state;
    s2: state;
  transitions
    t1: s1-[a^b]->s2;
    t2: s1-[a^*b]->s1;
    t3: s1-[b^a]->s3;
    t4: s2-[b]->s1;
    t5: s3-[a]->s1;
  **};
end alternate.alt2;
Clock automata in the behavioral annex

Esterel’s ABRO (controlled multi-clocked)

Thread ABRO

annex behavior_specification {**

state S =

state S1, S2, S3, S4

constraint sync O T1^+T2^+T3

transition {
  S1 -[B^-A]-> S2;
  S1 -[A^-B]-> S3;
  T1::S1 -[A^*B]-> S4;
  T2::S2 -[A]-> S4;
  T3::S3 -[B]-> S4; }

end;

transition S -- R ->> S;

end;

loop
  [ await A
  || await B ];

emit O

each R
Whitepaper
A timing annex for the AADL

PLAN

Goal of the proposal
State of the art and background
A model of time and clock automata
Timing AADL concepts
Workplan
(Implicitly) synchronous profile/subset in core AADL [1,2]

- Periodic (dispatch) threads
- Immediate or delayed (next) data ports
- Mode changes
- Simple, deterministic, BA automata

Timing AADL concepts

A timing model allows to specify more

- Synchronous and delayed port
- Clock domains and relations among them
- Periodic, multi-rate, burst-mode ports and threads
- Explicit adapters/schedules and scheduling analysis
- Buffers/schedules synthesis from affine relations [1]
- Deterministic, reactive and controlled automata

=> verify and synthesize more properties

Examples

• *Synchronous* and *delayed* port
• Clock domains and relations among them
• *Periodic*, multi-rate, burst ports and threads

…/… examples
Whitepaper
A timing annex for the AADL

PLAN

Goal of the proposal
State of the art and background
A model of time and clock automata
Timing in related annexes
Workplan
Relation to annexes

Behavioral annex
   Extend automata with synthesizable control from timing constraints

Constraint annex
   Extend constraints with event algebra

Contracts in CA are suitable for compositional modeling and verification (possible collaboration)
   Timing compatibility
   Refinement checking
Whitepaper
A timing annex for the AADL

PLAN

Goal of the proposal
State of the art and background
A model of time and clock automata
Timing AADL concepts and annexes
Discussions
CCSL contains the proposed timing annex

\[
\begin{align*}
A & \text{ isSubClockOf } B & A & \triangleleft B \\
A & = B & A & \triangleleft= B \\
A & \neq B & A & \triangleleft B \\
A & = B \text{ clockunion } C & A & \triangleleft= B \text{ clockunion } C \\
A & = B \text{ clockinter } C & A & \triangleleft= B \text{ clockinter } C \\
A & = B \text{ clockdiff } C & A & \triangleleft= B \text{ clockdiff } C
\end{align*}
\]

The timing annex is its synthesizable subset.
In addition, the use of bounded \textit{delayfor, inf} and \textit{sup} could be discussed to express, e.g., causal relations and/or scheduling constraints?
A **process and its components** refer to a software time unit. In an unbound process these units have a discrete time domain. Their physical (discrete or continuous) semantic time domain is that of the system that contains it.

**A processor, a device, (a bus ?), a system** refer to physical time units. Distinct components have distinct time domains.
A process bound to a processor inherits the discrete/physical semantic time domain of this processor

Distinct time domains can be related by constraints

\[\text{sysIN.ms} \leq \text{sysOUT.ms} \text{ | } \text{sysIN.ms} \sim \leq \text{AMD65.ms} \%\text{less than 1 ms variation}\]
Data ports of components provided with a continuous time domain can hold continuous values.

Discrete/continuous data conversion could be a port property. A device can be a discrete behavior description, an interface to a physical component (an engine), an interface to the model of a physical component (the LMS model of an engine).
Whitepaper
A timing annex for the AADL

PLAN

Goal of the proposal
State of the art and background
A model of time and clock automata
Timing AADL concepts and annexes

Workplan
Workplan

What is available
- BA specification for reactive automata
- CA specification for timing constraints
- Core AADL timing properties

Proposed extensions
- Extend (core) triggers with the $\texttt{clk}$ grammar to build clocks and regular expressions over events with counting
- Reflect $\texttt{clk}$ in the constraints annex
- Allow AADL to access the previous value (pre/next) of a data-flow (w.r.t. some clock)

Contribution
- Combine BA and CA to synthesize controlled automata
- Synchronous semantics and time model for AADL, BA, CA
Conclusion

A light-weight syntactic proposal
- Focus on a time model (polychrony)
- Integrates with constraints and behavior
- Relates to MARTE/CCSL standard
- Expressive model of synchronous automata

A formal interpretation of constraints and behavior in a synchronous model of computation and communication

Extensible
- Controlled and hierarchical automata, regular expressions
- Relation to physical/hardware discrete/continuous time