Revisiting coverage criteria for *SCADE* models

Jean-Louis Colaço
7 December 2016
Context

- **Code coverage** is a measure that characterises how much a given *test suite* exercises a *code*.
- Lots of criteria exist, avioncs standard (DO-178) requires MC/DC for the most critical application.
- In DO-178C (2011), supplement DO-331 about *Model Based Design* now requires model coverage.
- **SCADE** proposes model coverage for about 10 years:
  - was based on ad’hoc criteria defined by the user per operator,
  - recent solution is inspired by work of Parissis et al.

A. Lakehal and I. Parissis,
Structural coverage criteria for LUSTRE/SCADE programs,
in *Software Testing, Verification and Reliability*, Wiley Interscience, 2009

J-L. Camus, C. Haudebourg and M. Schlickling
Data Flow Model Coverage Analysis: Principles and Practice
in *Embedded Real Time Software and Systems*, 2016
Why revisiting?

- current solution is based on *Paths* in the dataflow: quite complex objects;
- to study the relationship between model coverage and generated code coverage: paths are not well suited;
- to overcome some limitation of current implementation.
Why revisiting?

- current solution is based on Paths in the dataflow: quite complex objects;
- to study the relationship between model coverage and generated code coverage: paths are not well suited;
- to overcome some limitation of current implementation.

The idea we had for the rework was actually nicely presented in:

M. Whalen, G. Gay, Y. Dongjiang, M. P.E. Heimdahl and M. Staats
Observable modified condition/decision coverage
in *Proceedings of the 35th International Conference on Software Engineering*, 2013
Why revisiting?

- current solution is based on *Paths* in the dataflow: quite complex objects;
- to study the relationship between model coverage and generated code coverage: paths are not well suited;
- to overcome some limitation of current implementation.

The idea we had for the rework was actually nicely presented in:

M. Whalen, G. Gay, Y. Dongjiang, M. P.E. Heimdahl and M. Staats
Observable modified condition/decision coverage
in *Proceedings of the 35th International Conference on Software Engineering*, 2013

present work continues and extends it to full *SCADE 6* language.
Agenda

Intuition

Ideal definition of coverage

SCADE tagged semantics

Tag based definition of coverage

Static tag reduction

Conclusion
Intuition

Ideal definition of coverage

SCADE tagged semantics

Tag based definition of coverage

Static tag reduction

Conclusion
flow or stream: infinite sequence of values.
model: a SCADE program and a root node.
monitor: any construction that allows to observe a flow out of the model: (root node) outputs, probes, ... 
outcome (of a test) values taken by all the monitors of the model when running a test.
source designates any construction that introduces flow that that does not result from the combination of other flows. (root node) inputs, sensors, literal values, reference to constants.
The intuition

- Covering a stream occurrence $s$ requires exhibiting a test that shows its ability to influence a monitor (red bubles);
- Covering a model requires covering all its streams occurrences.
Criterion 1: Influence

A test $T$ shows the influence of stream $x$ of a model $M$ if:
- $T$ is such that $x$ is in situation to influence an output of $M$
- i.e. $T$ is such that modifying stream $x$ in the execution of the test changes the outcome.

A test suite $T_S$ covers a model $M$ if for all stream $x$ of $M$, $T_S$ contains a test $T$ that covers stream $x$. 
Criterion 2: OMC/DC

A pair of tests \((T_1, T_2)\) satisfies OMC/DC criterion for a Boolean stream \(b\) of a model \(M\) if \(T_1\) and \(T_2\) are such that:

- \(b\) takes different values in each test case and
- toggling \(b\) in both test cases changes the outcome.

A test suite \(T_S\) covers a model \(M\) in the sense of OMC/DC if for all Boolean stream \(b\) of \(M\), \(T_S\) contains two tests \(T_1\) and \(T_2\) such that satisfy the condition above.
Intuition

Ideal definition of coverage

SCADE tagged semantics

Tag based definition of coverage

Static tag reduction

Conclusion
Notations

- $\mathcal{D}^n$ represent the set of stream prefix of size smaller or equal to $n$.
- If $x$ is a stream prefix, $|x|$ represents its size.
- If $x$ is a stream prefix, $(x)_i$ where $i \leq |x|$ represents $i^{th}$ value.
- Let $\mathcal{M}$ be a SCADE model and $n_{in}$ its number of inputs.
- A test case $T$ of length $n$ cycle is a tuple of $n_{in}$ components of $\mathcal{D}^n$.
- $\mathcal{M}(T)$ represents the execution of test case $T$; the outcome of this execution is itself a tuple of values in $\mathcal{D}^n$ (one per monitor).
- If $\nu$ is a stream prefix of a Boolean stream, $\neg_i(\nu)$ represents the prefix with same length built from $\nu$ by negating its $i^{th}$ value.
- A stream occurrence is represented as $\lfloor e \rfloor_k$ where $k$ is an integer and $e$ is a stream expression.
Occurrences identification

Defined by function \textit{Streams}(.):

\[
\text{Streams}(x_1, \ldots, x_n = e;) \stackrel{\text{def}}{=} \text{Streams}(e)
\]

\[
\ldots
\]

\[
\text{Streams}(x) \quad \stackrel{\text{def}}{=} \{ \lfloor x \rfloor_k \}
\]

\[
\text{Streams}(1) \quad \stackrel{\text{def}}{=} \{ \lfloor 1 \rfloor_k \}
\]

\[
\text{Streams}(\text{'s};) \quad \stackrel{\text{def}}{=} \{ \lfloor \text{'s} \rfloor_k \}
\]

\[
\text{Streams} (\text{last 's;}) \quad \stackrel{\text{def}}{=} \{ \lfloor \text{last 's} \rfloor_k \}
\]

\[
\text{Streams}(\text{op}(e_1, \ldots, e_n)) \quad \stackrel{\text{def}}{=} \{ \lfloor \text{op}(e_1, \ldots, e_n) \rfloor_k \} \cup \text{Streams}(e_1) \cup \ldots
\]

\[
\ldots
\]
Occurrences identification example

Streams \((o = x*x + \text{pre}(2*x) + 1;\) =

\[
\begin{cases}
|x|_1, \ [x]_2, \ [x]_3, \ [2]_4, \ [1]_5, \\
[|x|_1 * |x|_2]_6, \ [2]_4 * [x]_3]_7, \ [\text{pre}([2]_4 * [x]_3)]_7]_8, \\
[|x|_1 * |x|_2]_6 + ([\text{pre}][2]_4 * [x]_3]_7]_8]_9, \\
[|x|_1 * |x|_2]_6 + ([\text{pre}][2]_4 * [x]_3]_7]_8]_9 + [1]_5]_10
\end{cases}
\]

Ideal definition of coverage
Stream occurrence mutation

Let $M$ be a model where:

- $\lfloor e \rfloor_k$ one of its stream occurrences: $\lfloor e \rfloor_k \in \text{Streams}(M)$,
- $v$ is a finite stream prefixe: $v \in \mathcal{D}^n$,
- $e$ and $v$ are of same type,
- $e'$ is a stream expression with same clock as $e$:

<table>
<thead>
<tr>
<th>$e$</th>
<th>$e_0$</th>
<th>$\cdots$</th>
<th>$e_n$</th>
<th>$e_{n+1}$</th>
<th>$e_{n+2}$</th>
<th>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v$</td>
<td>$v_0$</td>
<td>$\cdots$</td>
<td>$v_n$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$e'$</td>
<td>$v_0$</td>
<td>$\cdots$</td>
<td>$v_n$</td>
<td>$e_{n+1}$</td>
<td>$e_{n+2}$</td>
<td>$\cdots$</td>
</tr>
</tbody>
</table>

$M(v \triangleright \lfloor e \rfloor_k)$ represents the model obtained by substituting $\lfloor e \rfloor_k$ in $M$ by a $e'$; we called it a mutant of $M$ for the occurrence $\lfloor e \rfloor_k$. 

© ANSYS, Inc.
Influence ideal definition

Coverage of stream $x$ by $T$:

$$\text{Influence}(T, x, \mathcal{M}) \overset{\text{def}}{=} \exists n > 0. \exists v \in \mathcal{D}^n. \mathcal{M}(T) \neq \mathcal{M}^{(v\triangleright x)}(T)$$

Coverage of model $\mathcal{M}$ by a test suite $\mathcal{T}_S$:

$$\forall x \in \text{Streams}(\mathcal{M}). \exists T \in \mathcal{T}_S. \text{Influence}(T, x, \mathcal{M})$$
OMC/DC Ideal definition

Coverage of stream $x$ by $(T_1, T_2)$:

$$\text{Omcdc}(T_1, T_2, b, M) \overset{\text{def}}{=} \exists (i, j) \in \mathbb{N} \times \mathbb{N}. \left( (b_{T_1})_i \neq (b_{T_2})_j \land M(T_1) \neq M(\neg_i (b_{T_1}) \triangleright b)(T_1) \land M(T_2) \neq M(\neg_j (b_{T_2}) \triangleright b)(T_2) \right)$$

Coverage of model $M$ by a test suite $T_S$:

$$\forall b \in \text{Streams}(M). \exists (T_1, T_2) \in T_S \times T_S. ((b : \text{bool}) \Rightarrow \text{Omcdc}(T_1, T_2, b, M))$$
Limit of the ideal definition

Not really implementable:

- based on the existence of mutants without giving a way to build them (it is a guess);
- requires both executions on original model and on the mutant;
- needs one mutant per stream occurrence.
Intuition

Ideal definition of coverage

**Scade** tagged semantics

Tag based definition of coverage

Static tag reduction

Conclusion
Tagged semantics:
- based on tagged values;
- defines tag propagation rules.
- provides primitives for tag introduction;
Tagged values

The values used in a tagged $\text{SCADE}$ model $\mathcal{M}^\#$ are in $\mathcal{V}_{n,m}^\#$ defined by:

$$
\mathcal{V}_{0,m}^\# \overset{\text{def}}{=} \left( \text{bool} \cup \text{numeric} \cup \{\text{declared enum values}\} \right) \times \mathcal{P}(\text{Tags})
$$

$$
\mathcal{V}_{n+1,m}^\# \overset{\text{def}}{=} \mathcal{V}_{n,m}^\#
\quad \cup \quad \left\{ [v_1^\#, \ldots, v_p^\#] \mid 1 \leq i \leq p \leq m, \ v_i^\# \in \mathcal{V}_{n,m}^\# \right\} \times \mathcal{P}(\text{Tags})
\quad \cup \quad \left\{ l_1:v_1^\#, \ldots, l_p:v_p^\# \mid 1 \leq i \leq p \leq m, \ v_i^\# \in \mathcal{V}_{n,m}^\# \right\} \times \mathcal{P}(\text{Tags})
$$

where $\text{Tags}$ is a finite set of tags
Tag propagation of combinatorial operators

For most operators input tags propagate to the outputs:

$$\text{op}^#((v_1, \tau_1), \ldots, (v_n, \tau_n)) = (\text{op}(v_1, \ldots, v_n), \bigcup_{i \in [1..n]} \tau_i)$$
Tag propagation of temporal operators

Behave as usual but on tagged streams:

<table>
<thead>
<tr>
<th>$a, \tau^a$</th>
<th>$a_0, \tau^a_0$</th>
<th>$a_1, \tau^a_1$</th>
<th>$a_2, \tau^a_2$</th>
<th>$a_3, \tau^a_3$</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b, \tau^b$</td>
<td>$b_0, \tau^b_0$</td>
<td>$b_1, \tau^b_1$</td>
<td>$b_2, \tau^b_2$</td>
<td>$b_3, \tau^b_3$</td>
<td>\ldots</td>
</tr>
<tr>
<td><strong>pre</strong>$^#$ ($a, \tau^a$)</td>
<td>$\text{nil}, \emptyset$</td>
<td>$a_0, \tau^a_0$</td>
<td>$a_1, \tau^a_1$</td>
<td>$a_2, \tau^a_2$</td>
<td>\ldots</td>
</tr>
<tr>
<td>$a, \tau^a$ $\rightarrow$<strong>pre</strong>$^#$ ($b, \tau^b$)</td>
<td>$a_0, \tau^a_0$</td>
<td>$b_1, \tau^b_1$</td>
<td>$b_2, \tau^b_2$</td>
<td>$b_3, \tau^b_3$</td>
<td>\ldots</td>
</tr>
</tbody>
</table>
Specific propagation rules

and\# (also exists for or\#):

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b$</td>
<td>$a \text{ and} # b$</td>
</tr>
<tr>
<td>false, $\tau_a$</td>
<td>false, $\tau_b$</td>
<td>false, $\tau_a \cap \tau_b$</td>
</tr>
<tr>
<td>false, $\tau_a$</td>
<td>true, $\tau_b$</td>
<td>false, $\tau_a$</td>
</tr>
<tr>
<td>true, $\tau_a$</td>
<td>false, $\tau_b$</td>
<td>false, $\tau_b$</td>
</tr>
<tr>
<td>true, $\tau_a$</td>
<td>true, $\tau_b$</td>
<td>true, $\tau_a \cup \tau_b$</td>
</tr>
</tbody>
</table>

flow selection:

if \# (true, $\tau_c$) then \# ($v_1, \tau_1$) else \# ($v_2, \tau_2$) = ($v_1, \tau_c \cup \tau_1$)

if \# (false, $\tau_c$) then \# ($v_1, \tau_1$) else \# ($v_2, \tau_2$) = ($v_2, \tau_c \cup \tau_2$)
Sources are extended with an empty set of tags,
memories are initially extended with an empty set of tags,
new primitives $\text{tag}(e, t)$ and $\text{bool\_tag}(e, t_1, t_2)$ introduce tags:

$\text{tag}((v, \tau), t) = (v, \{t\} \cup \tau)$

$\text{bool\_tag}((\text{true}, \tau), t_1, t_2) = (\text{true}, \{t_1\} \cup \tau)$

$\text{bool\_tag}((\text{false}, \tau), t_1, t_2) = (\text{false}, \{t_2\} \cup \tau)$
Tagged semantics for coverage purpose

- introduce a tag for each stream occurrence and
- register tags when reaching a monitor.
A simple example of propagation

model
A simple example of propagation

tagged model
A simple example of propagation

first cycle
A simple example of propagation

second cycle
A simple example of propagation

other cycles
Intuition

Ideal definition of coverage

Scade tagged semantics

Tag based definition of coverage

Static tag reduction

Conclusion
Influence tagged definition

Coverage of stream $x$ by $T$:

\[
\text{Influence}^\#(T, x, \mathcal{M}) \overset{\text{def}}{=} t_x \in Otags(\mathcal{M}^\#(T))
\]

Coverage of model $\mathcal{M}$ by $\mathcal{T}_S$:

\[
\forall x \in \text{Streams}(\mathcal{M}). \exists T \in \mathcal{T}_S. \text{Influence}^\#(T, x, \mathcal{M})
\]
OMC/DC tagged definition

Coverage of stream \( x \) by \( (T_1, T_2) \):

\[
\text{Omcdc}^\#(T_1, T_2, b, \mathcal{M}) \overset{\text{def}}{=} t_o^b \in Otags(\mathcal{M}_\text{Bool}^\#(T_1)) \land t^b_\bullet \in Otags(\mathcal{M}_\text{Bool}^\#(T_2))
\]

Coverage of model \( \mathcal{M} \) by \( \mathcal{T}_S \):

\[
\forall b \in \text{Streams}(\mathcal{M}) . \exists (T_1, T_2) \in \mathcal{T}_S \times \mathcal{T}_S . ((b : \text{bool}) \Rightarrow \text{Omcdc}^\#(T_1, T_2, b, \mathcal{M}))
\]
There are situations where tags are propagated while no contribution can be observed:

- Absorption: $x \times 0$
- Unobservable selection: $\text{if } c \text{ then } x \text{ else } x$
There are situations where tags are propagated while no contribution can be observed:

- absorption: $x \times 0$
- unobservable selection: if $c$ then $x$ else $x$

Gaps exist but it still be a good compromise.
Intuition

Ideal definition of coverage

SCADE tagged semantics

Tag based definition of coverage

Static tag reduction

Conclusion
Reduction

- Criteria are based on tags on all the expressions and sub-expressions ⇒ big number of tags.
- Many tags are related: each time $t_1$ is observed $t_2$ is also observed.
- Reduction consists in removing tags whose observation can be deduced from other tags observation.
- Reduction is used in the model instrumentation phase.
Example

```plaintext
node N(a, b : bool; i : int16)
  returns (o : int16)
var m : int16;
let
  m = pre o;
  o = 0 -> (if a and b then 2 * i else i)
      + (if a or b then m / 4 else m);
tel
```
Example: initial tagging
Example: initial tagging

27 tags
Example: simple tag reduction
Example: simple tag reduction

15 tags
Example: + Boolean reduction
Example: + Boolean reduction

11 tags
Intuition

Ideal definition of coverage

**SCADE** tagged semantics

Tag based definition of coverage

Static tag reduction

**Conclusion**
Conclusion

- extends to all SCADE 6 language, including automata;
- implementation:
  - instrumentation of the model (addition of tag(...)) and code generation for the tagged semantics;
- static reduction is important, divides by 2 to 3 the number of tags;
- good scale up (tested on big industrial models).