Automatic Review of Abstract State Machines by Meta-Property Verification

Corso tvsw

Angelo Gargantini
18/19

Outline

1. Foundations: concepts and principles
   - Model review and meta-properties
2. Abstract State Machines
3. Meta-Properties of ASMs
   - Definition and derivation
   - Verification by Model Checking
4. Experiments

1. Validation and Verification

- Validation:
  - the systems satisfies or fits the intended usage
- Validation should preceed formal property verification
  - Proving properties of wrong models?
- Validation activities include
  - Simulation
    - Interactive, random, scenario based …
  - **Model review – static analysis**
    - **Similar to static analysis of code like PMD**

2. Model review

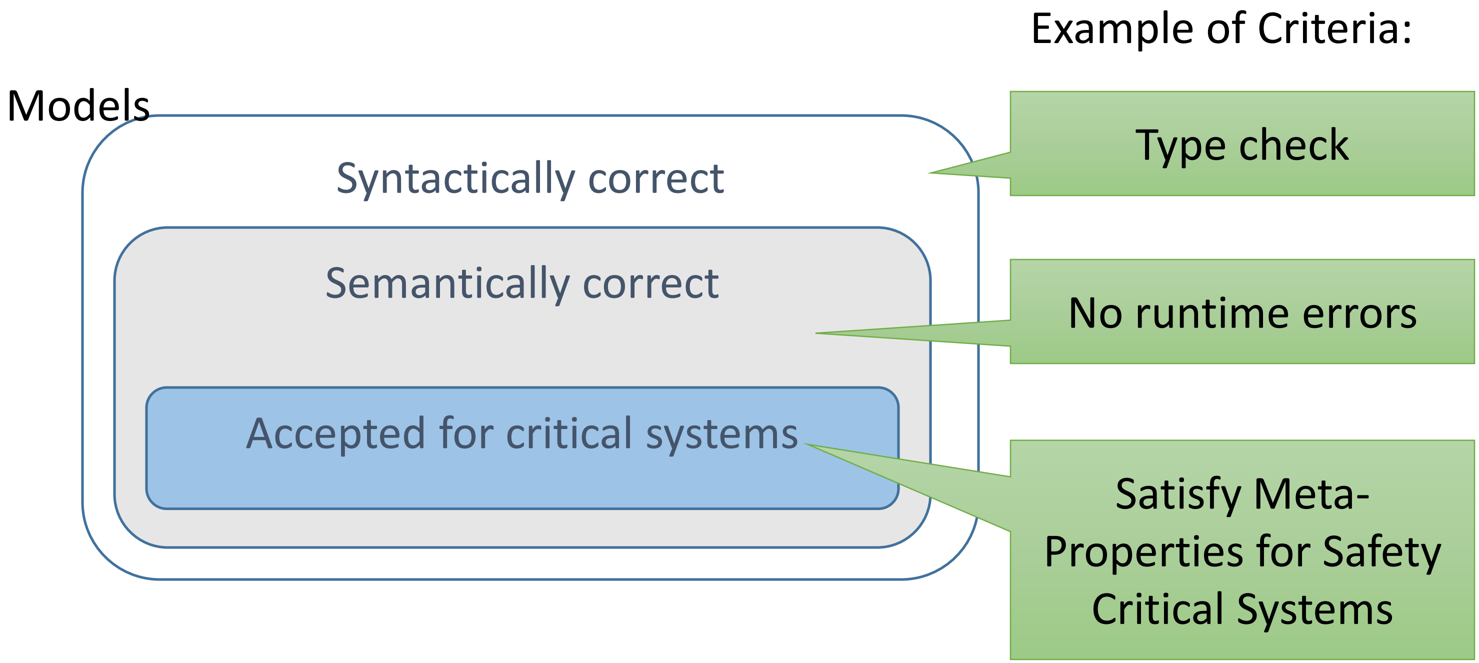
- "**model walk-through**" or "**model inspection**", is a validation technique
- Models are critically examined to determine if
  - fulfill the intended requirements
  - are of sufficient "quality" to be easy to develop, maintain, and enhance.
- Quality assurance process
  - allow defects to be detected early in the system development, reducing the cost of fixing them
- **What to check**?
  - Definition of "**properties**" of a good model

3. Meta-properties

- Some properties should be true for <u>any</u> model
  - Parnas: “reviewers spent too much of their time and energy checking for simple, application-independent properties which distracted them from the more difficult, safety-relevant issues.”
- We call these **meta-properties**
- Meta-property $\leftrightarrow$ quality attribute
- Tools that automatically perform such checks can save reviewers considerable time and effort, liberating them to do more creative work

4. Critical systems

- Safety critical systems may need more severe quality requirements
  - More severe meta-properties



5. Meta-properties and notation

- Meta-properties definition may be notation depedent
  - But most of them refer to general quality attributes
- In our case:
  - ABSTRACT STATE MACHINES (ASM)
- Largely inspired by the work done by Connie Heitmeyer at the NRL with SCR tabular notation

Rule Firing Condition

- For every rule is possible to **statically** compute the conditions under which it will fire:
- *Rule Firing Condition (RFC)*

$$RFC: Rules \rightarrow Conditions$$

  - RFC can be built by visting the model (details on the paper)

RFC – example

```
main rule R =
  if x > 0 then
     if y < 0 then
           x:= 5
     endif
  endif
```

Rule Firing Condition:

$x > 0 \text{ and } y < 0$

Meta-properties for ASMs

Meta-properties families

- **Consistency**
  locations are never simultaneously updated to different values (**inconsistent updates**).
- **Completeness**
  every behavior of the system is explicitly modeled.
  - E.g. listing of all the possible conditions in conditional rules
- **Minimality**
  the specification does not contain elements – e.g. transition rules, domain elements – defined or declared but never used (**over specification)**.

Meta-properties definition

- Two possible schemas for meta-properties:

$Always(\phi)$: ϕ must be true in **any** reachable state

$Sometime(\phi)$: ϕ must be true in **a** reachable state

MP1. No inconsistent update is ever performed

- An inconsistent update occurs when two updates clash, i.e. they refer to the same location but are distinct

Example

```
main rule R =
   par
      l:=1
      l:=2
   endpar
```

Inconsistent update

For every rule R1 and R2

MP1

$$\boldsymbol{R_1}: \quad \boldsymbol{f(a_1) := t_1}$$

$$\boldsymbol{R_2}: \quad \boldsymbol{f(a_2) := t_2}$$

$$Always\begin{pmatrix} RFC(R_1) \land RFC(R_2) \\ \land\ a_1 = a_2 \\ \rightarrow\ t_1 = t_2 \end{pmatrix}$$

MP2. Every conditional rule must be complete

- In a conditional rule R = if c then Rthen endif, without else, the condition c must be true if R is evaluated.
- Therefore, in a nested conditional rule, if one does not use the else branch, the last condition must be true.

MP3. Every rule can eventually fire

Example

```
main rule R =
  if x > 0 then
     if x < 0 then l:=1
     endif
  endif
```

Never fires

For every rule R in the model:

$$Sometime(RFC(R))$$

MP3

MP4. No assignment is always trivial

- An update l := t is trivial [7] if l is already equal to t, even before the update is applied. This property requires that each assignment which is eventually performed, will not be always trivial. Let R = l := tbe an update rule.
- Property

Sometime(RFC(R)) → Sometime(RFC(R)∧l!= t)

Other meta-properties

MP5 For every domain element e there exists a location which can take value e

MP6. Every controlled function can take any value in its co-domain

MP7 Every controlled location is updated and every location is read

…

MP verification