# AUTOMATIC REVIEW OF ABSTRACT STATE MACHINES BY META-PROPERTY VERIFICATION

Angelo Gargantini
University of Bergamo - Italy
Paolo Arcani, Elvinia Riccobene
Università degli Studi di Milano - Italy

Nasa Formal Methods. Washington DC, 13 April 2010

# 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**

# 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

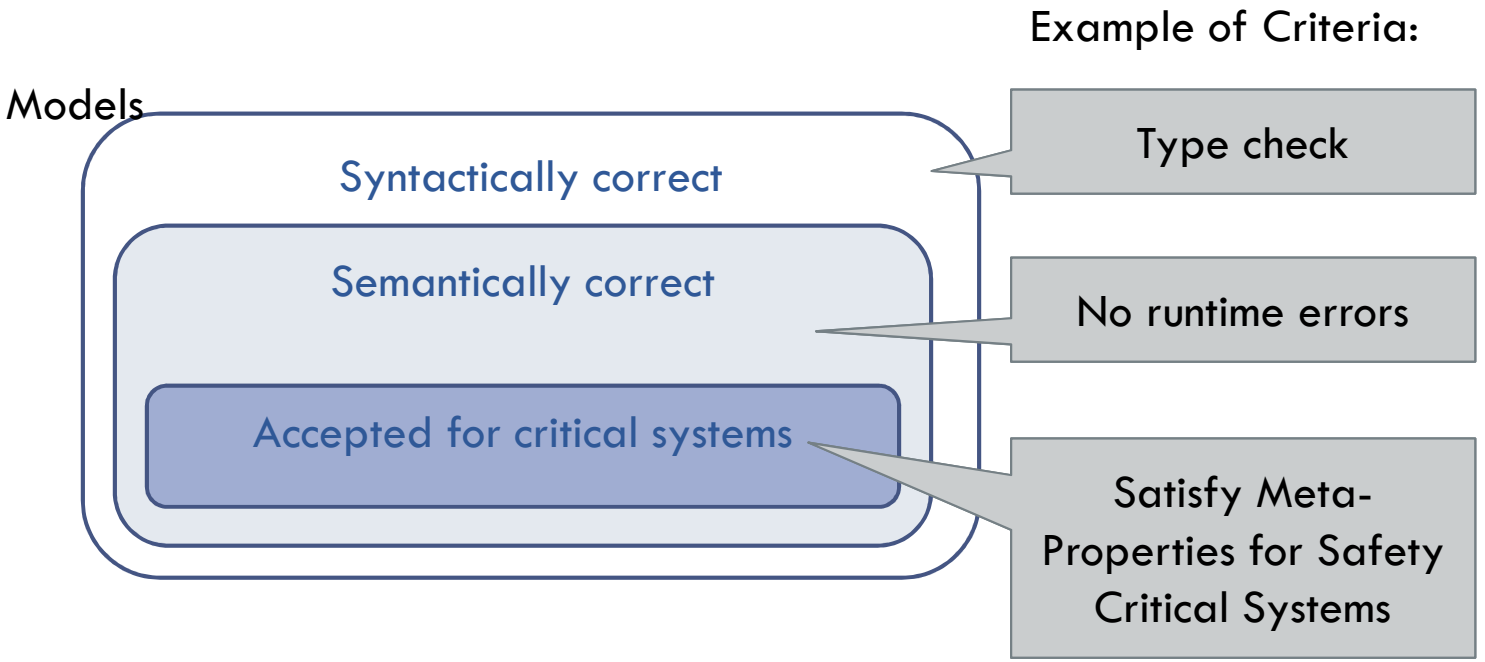


- Some properties should be true for any model
  - Parnas: "reviewers spent too much of their time and energy checking for simple, application-independent properties which distracted them from the more difficult, safety-relevant issues."
- We call these **meta-properties**
- Meta-property $\leftrightarrow$ 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

# Abastract State Machines (ASM)

# ASM history



- The concept of ASMs is due to Yuri Gurevich (1980)
  - ASM Thesis: every algorithm, no matter how abstract, is step-for-step emulated by an appropriate ASM
- AsmBook:

E. Börger, R. Stärk. *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer 2003

- ASMs used for
  - formal specification and analysis of (possibly critical) computer hardware and software.
  - specifications of programming languages (including Prolog, C, and Java) and design languages (UML and SDL) have been developed

# Abastract State Machines (brief)



- ASMs are an extension of FSMs
  - states are multi-sorted first-order structures, i.e. domains of objects with functions and predicates (Boolean functions) defined on them,
  - transition relation is specified by "**rules**" describing how functions change from one state to the next.
- Basic transition rule has the form of guarded update

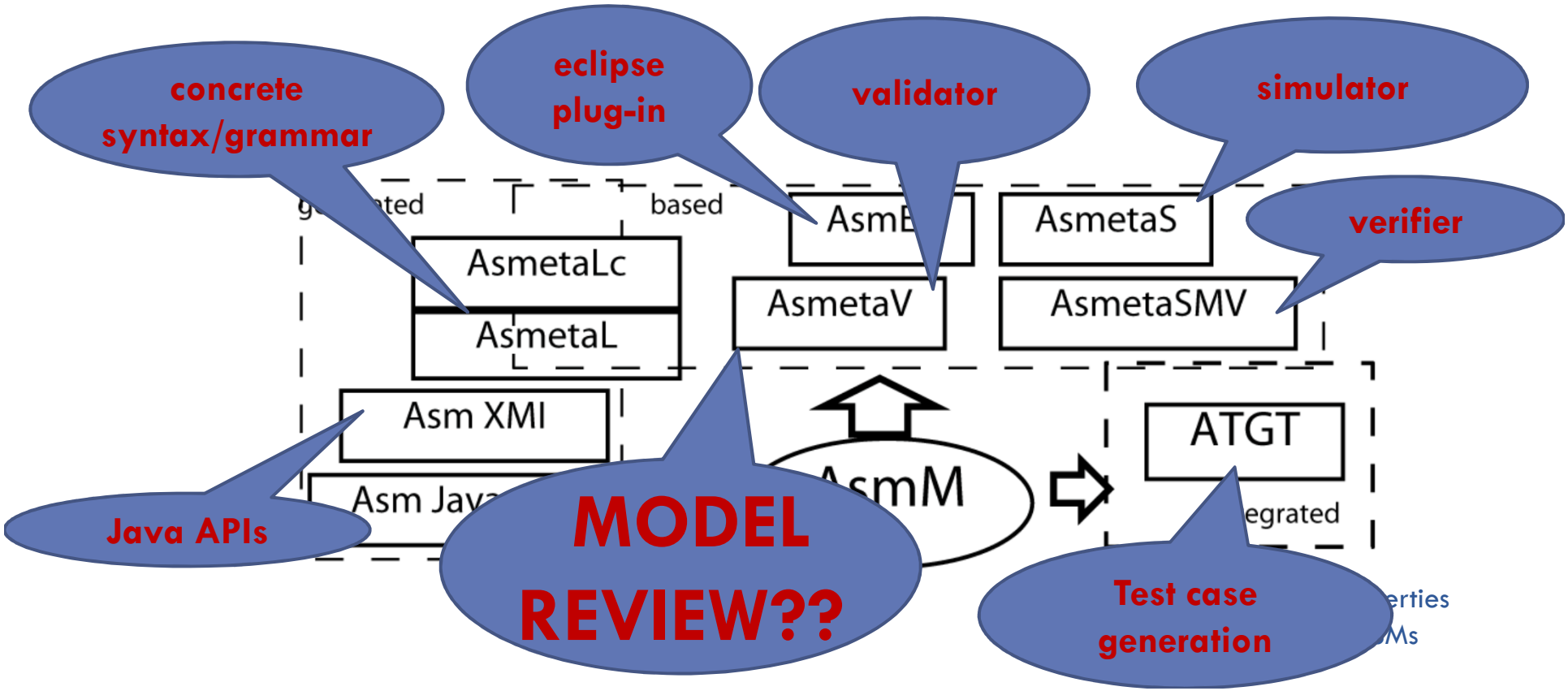
$$\boldsymbol{if}\ Condition\ \boldsymbol{then}\ f(t_1, \ldots, t_n) := t$$

- Parallel rule composition, non deterministic choice ...

# Asmeta Toolset asmeta.sourceforge.net



- Model Driven Engineering used to build a set of tools – mainly used by our students



# 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:

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

$Sometime(\phi)$ : $\phi$ 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**

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

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

$$Always\left(\begin{array}{c} RFC(R_1) \wedge RFC(R_2) \\ \wedge\ a_1 = a_2 \\ \rightarrow\ t_1 = t_2 \end{array}\right)$$

# 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:

**MP3**

$$Sometime(RFC(R))$$

# Other meta-properties



**MP2** Every conditional rule must be complete

**MP4** No assignment is always trivial

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

**MP7** Every controlled location is updated and every location is read

…

# Meta-Property Verification by Model Checking



- Meta-Property can be verified (or falsified) by model checking
  - We use the AsmetaSMV model checker which translates Asms to NuSMV

$$M \rightarrow M_{NuSMV}$$

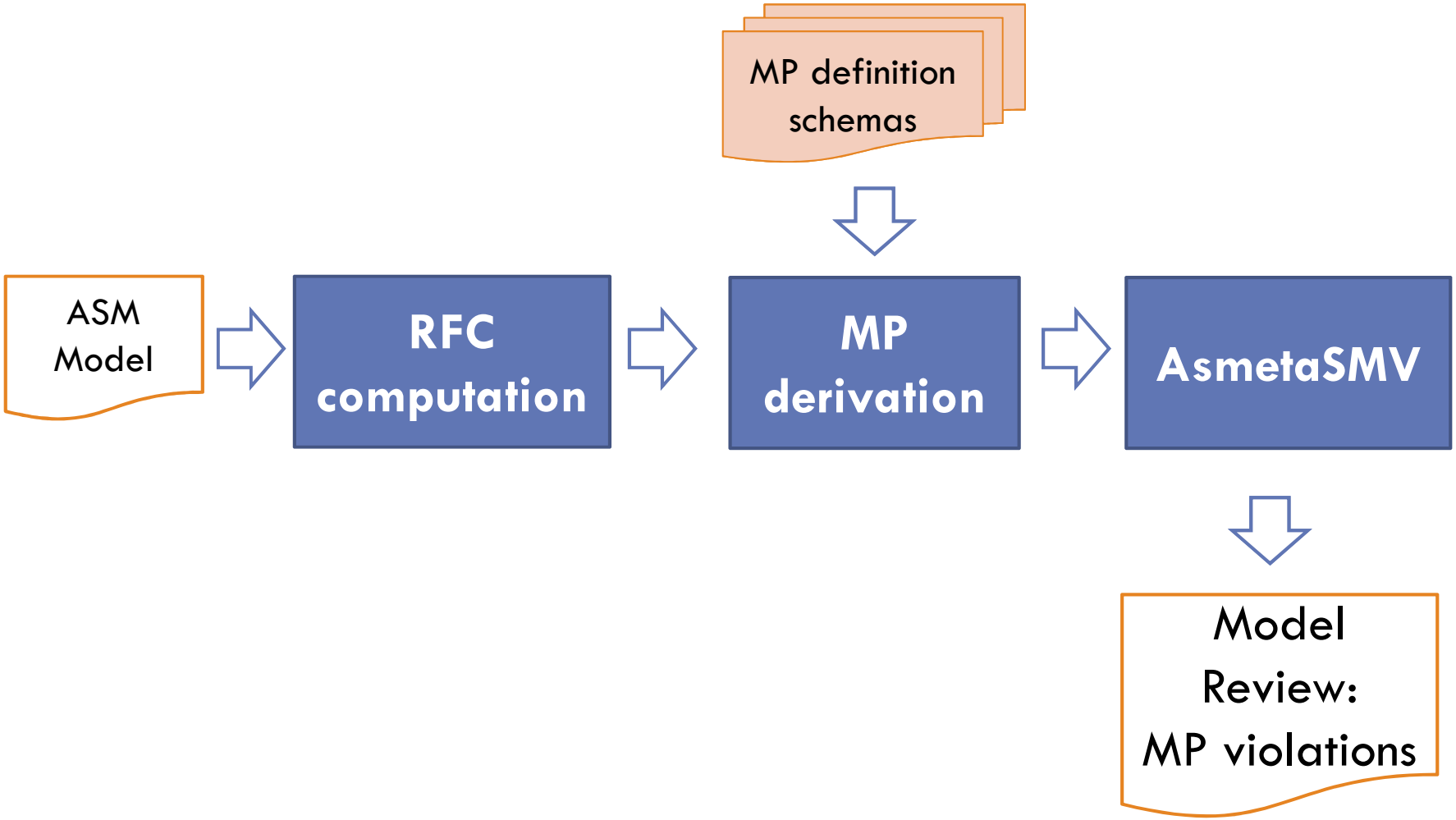
  - Meta-Properties to CTL:

$$M \vDash Always(\phi) \Leftrightarrow M_{NuSMV} \vDash \mathsf{AG}(\phi)$$

~~$$M \vDash Sometime(\phi) \Leftrightarrow M_{NuSMV} \vDash \mathsf{EF}(\phi)$$~~

$$M \vDash Sometime(\phi) \Leftrightarrow M_{NuSMV} \nvDash \mathsf{AG}(\neg\phi)$$

# MP verification



# Experiments

# Results



- 3 benchmark sets, with models
  1. specifically designed
  2. subset of examples of asmeta repository
  3. written by our students

| Spec Set | # spec | # rules | # violations | violated MPS |
|---|---|---|---|---|
| Bench | 21 | 384 | 61 | ALL |
| AsmRep | 18 | 506 | 29 | minimality |
| Stu | 6 | 172 | 38 | minimality but also some inconsistencies |

# Conclusions



- Model review of Formal Models
- Abstract State Machines
- Some quality meta-properties (of several classes) can be formalized as meta-properties
- Meta-properites may be model checked to find possible quality violations and prevent faults
- Applied to several examples and found several faults.