Hand in Glove – Complete Bounded Model Checking and Testing of Interlocking Systems

Jan Peleska
University of Bremen and Verified Systems International GmbH
jp@verified.de
2015-08-27

Motivation

- Testing interlocking systems is well known to require test case selection from an unmanageable multitude of possibilities

- How can we perform a well-justified test suite with acceptable effort . . .

- . . . while increasing the confidence into the strength of the resulting test suite ?

Overview

- Model-based testing
- Validated test models
- Complete test suites
- How many test cases do we need – naive approach
- A refined test strategy – compositional reasoning plus equivalence class testing plus randomisation plus boundary value selection
- Conclusion

Model-based Testing

Instead of writing test procedures,

- develop a **test model** specifying expected behaviour of SUT
- use **generator** to identify “relevant” test cases from the model and calculate concrete test data
- generate **test procedures** fully automatic
- perform **tracing** requirements ↔ test cases in a fully automatic way

Validated Test Models

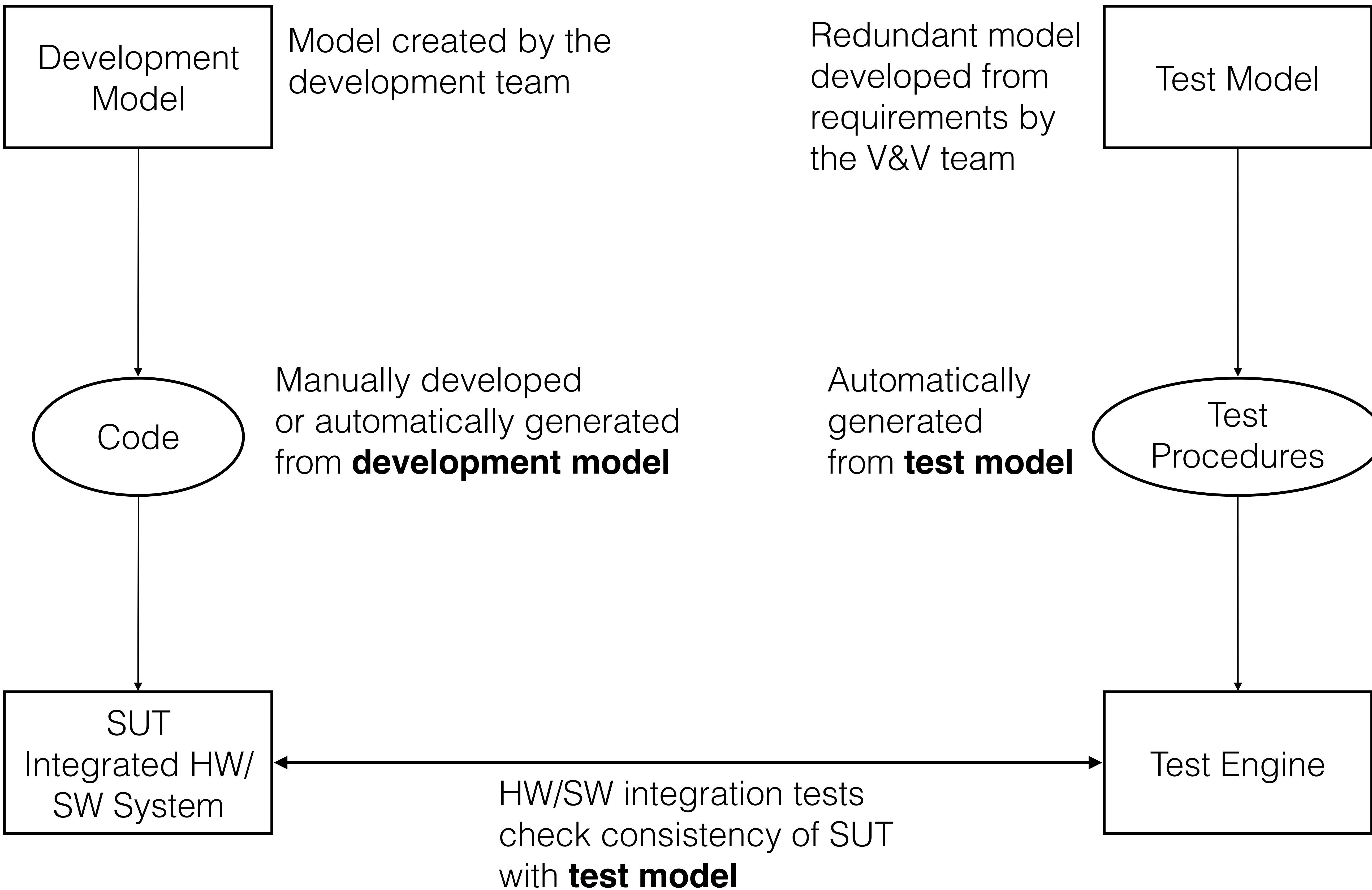
- The correctness and completeness of the test model is crucial for the success of a model-based testing strategy

- In a model-driven approach to development and V&V, there are two variants for arriving at trustworthy test models

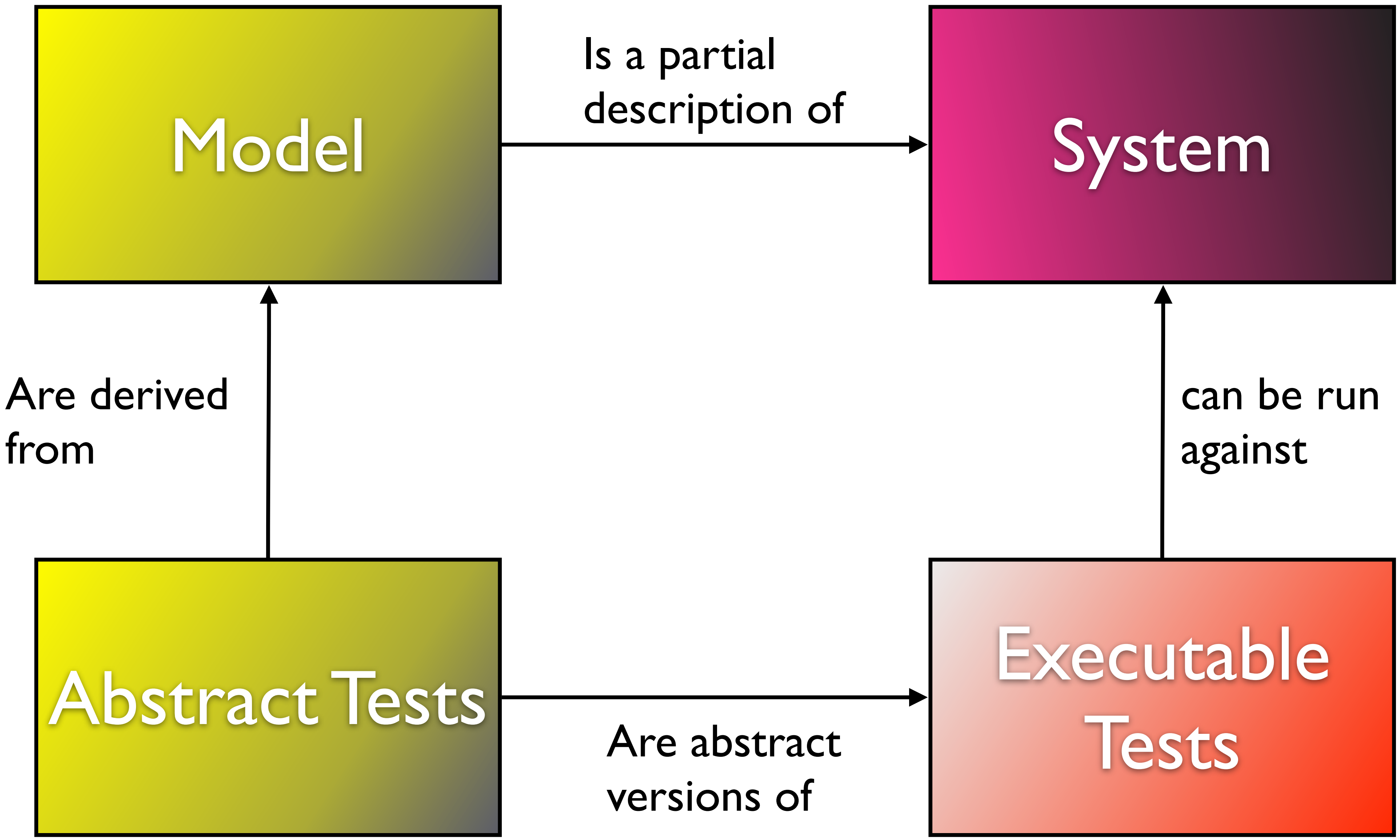
  1. Let the V&V team create a redundant model as test model

  2. Let the V&V team validate the existing design model and use that for test generation

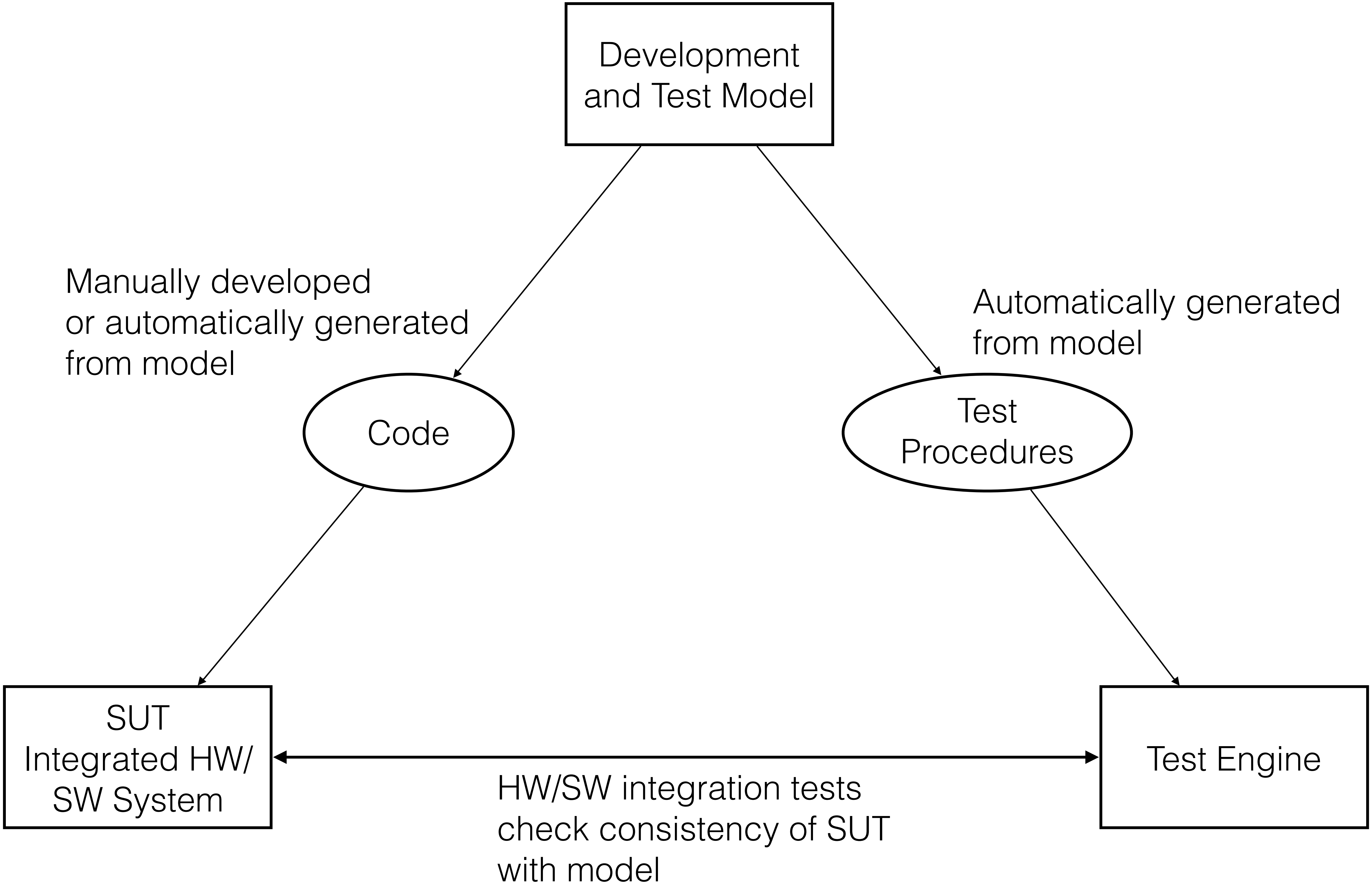
Validated Test Models – Variant 1



MBT-Paradigm



Validated Test Models – Variant 2



Validated Test Models

- We have seen in the RobustRailS presentations that complete verification of safety properties is possible for interlocking system designs of realistic size
  - This was achieved by bounded model checking in combination with inductive reasoning
  - ☞ Let’s take this model and use it for test case generation . . .
  - . . . so we advocate Variant 2 described above

Complete Test Suites

- For test suites created according to a certain strategy, we use the terms
  - **Sound** = correct implementations will not be rejected
  - **Exhaustive** = every faulty implementation will be detected
  - **Complete** = **Exhaustive** and **Sound**

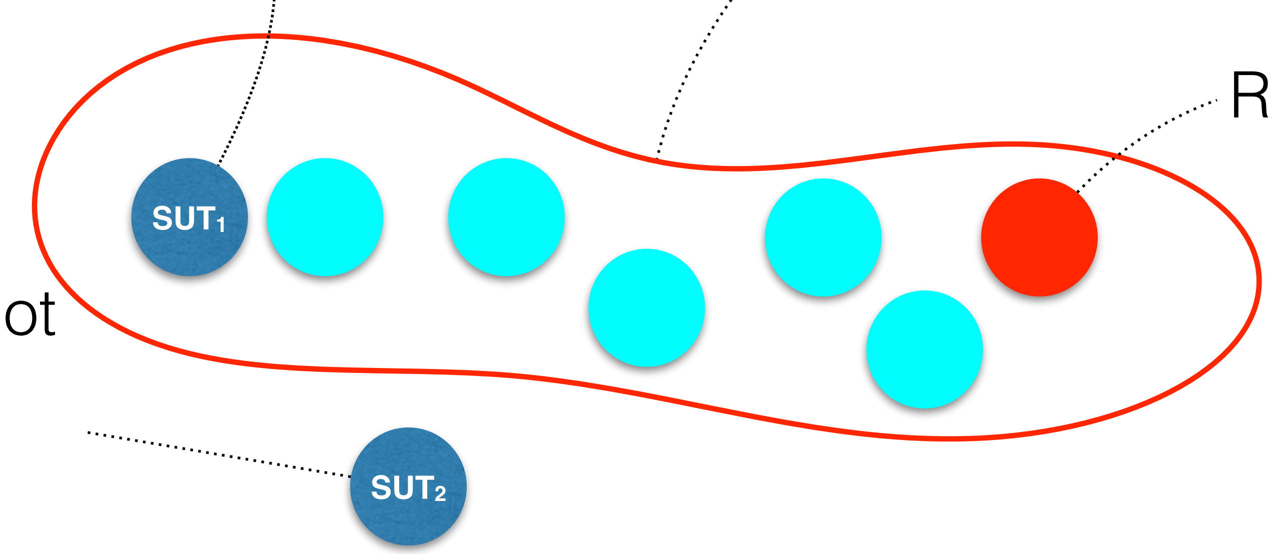
Complete Test Suites

- For black-box testing, completeness depends on a pre-specified **fault domain**
  - The true behaviour of the system under test must be captured in a (very large) class of models that may or may not be correct in relation to the given reference model

Fault domain *D*

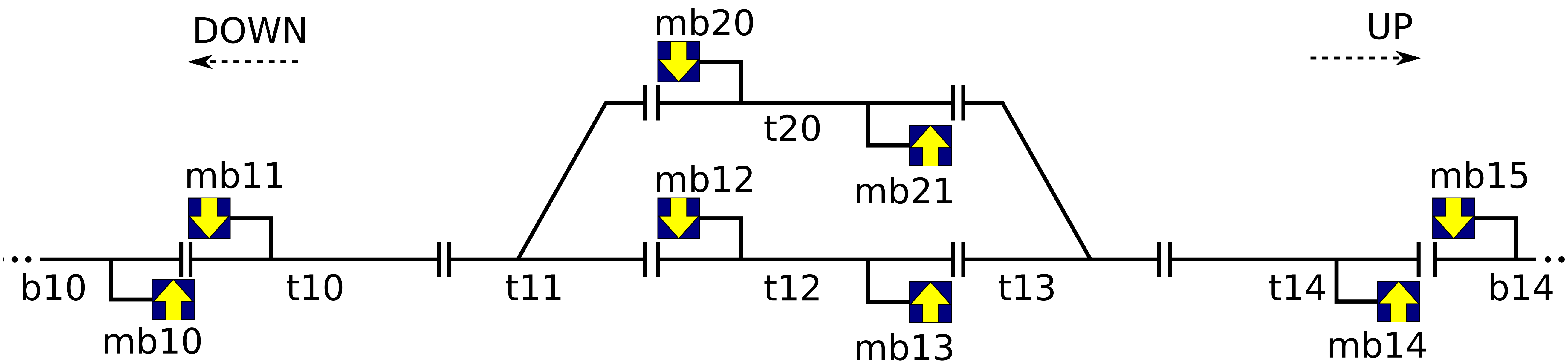
True behaviour of SUT_1 – complete test suite for *D* will uncover every deviation from reference model

Reference model

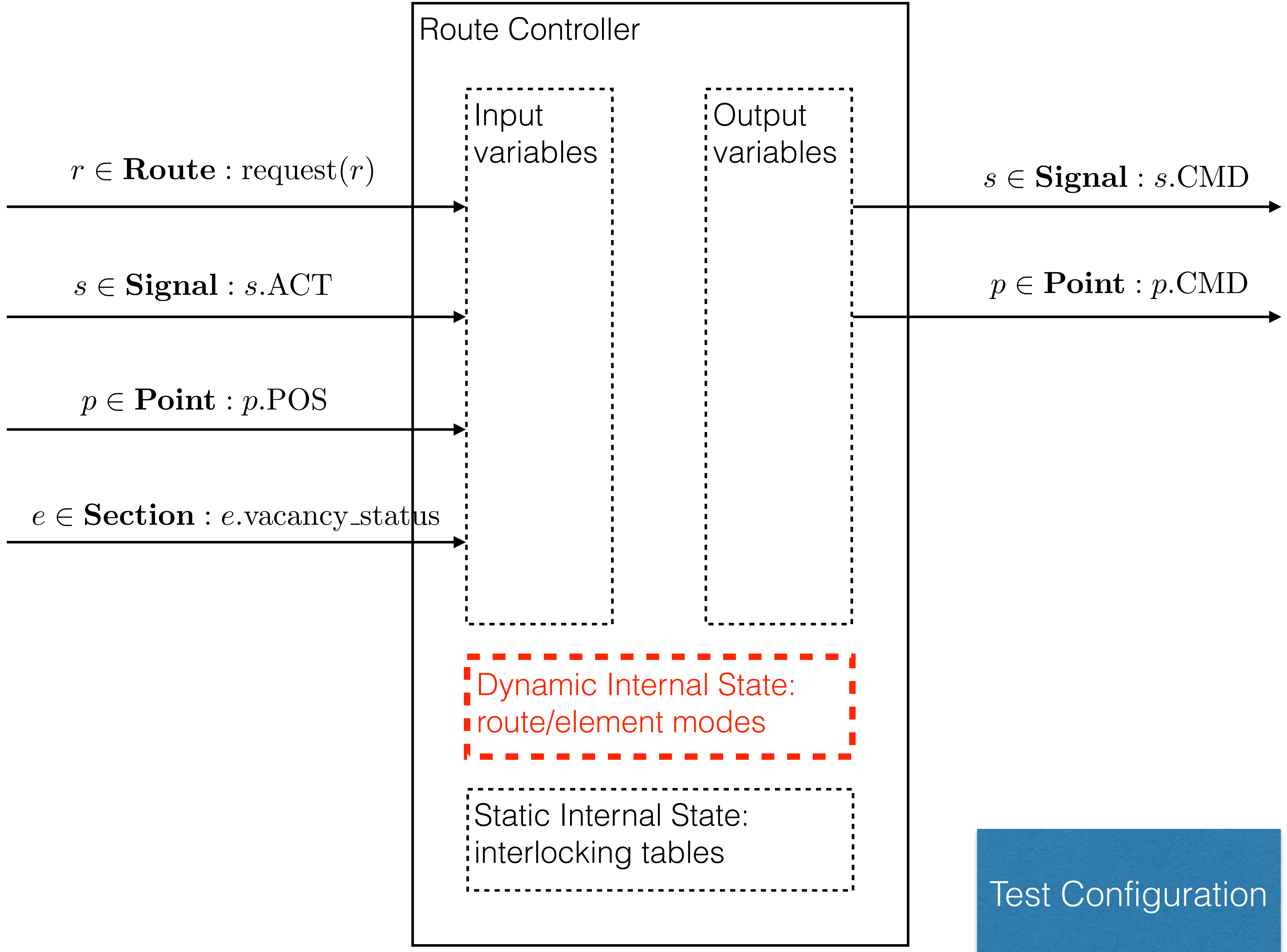


True behaviour of SUT_2 – complete test suite for *D* may not uncover every deviation from reference model

How many test cases do we need – naive approach



Example of railway network to be controlled



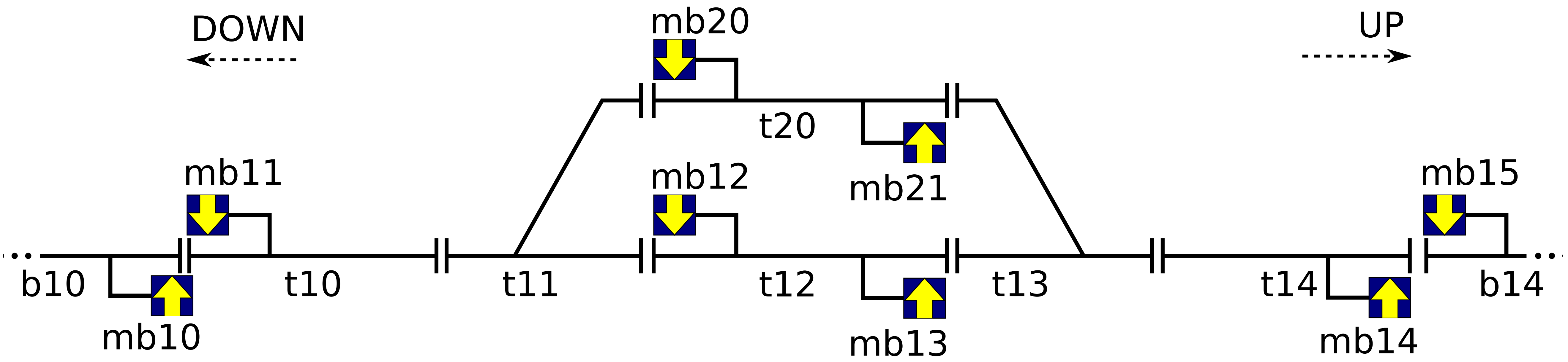
Route Controller
Input variables
Output variables
$r \in \textbf{Route} : \text{request}(r)$
$s \in \textbf{Signal} : s.\text{ACT}$
$p \in \textbf{Point} : p.\text{POS}$
$e \in \textbf{Section} : e.\text{vacancy\_status}$
$s \in \textbf{Signal} : s.\text{CMD}$
$p \in \textbf{Point} : p.\text{CMD}$
Dynamic Internal State: route/element modes
Static Internal State: interlocking tables
Test Configuration

How many test cases – naive approach

- A complete test suite requires test cases in the order of magnitude of at least

$$|\Sigma_I| \cdot n^2 \quad \text{test cases}$$

$\Sigma_I = \text{set of possible input vectors to the controller}$
$|\Sigma_I| = \text{Number of input vectors}$
$n = \text{number of internal states}$



Inputs.

$$\text{b10, t0, t11,}\ldots\text{,b14} \in \{\text{FREE, LOCKED, OCCUPIED}\}$$

plus

$$\text{t11.pos, t13.pos} \in \{\text{PLUS,MINUS}\}$$

plus

$$\text{mb11,}\ldots\text{,mb15} \in \{\text{HALT,GO}\}$$

This results in

$$|\Sigma_I| = (8 \cdot 3) \cdot (2 \cdot 2) \cdot (8 \cdot 2) = 1536$$

Internal State.

$$\text{Number of states in route controller}^{\text{Number of routes}}$$

This implies

$$n = (4 + \text{Number of segments in route})^8 \approx 6^8$$

Therefore order of magnitude for the number of test cases needed for complete test suite is

$$|\Sigma_I| \cdot n^2 = 1536 \cdot 6^{16} \approx 4.3 \cdot 10^{15}$$

Suppose a system test case execution needs 60s. Then you need

$$\frac{4.3 \cdot 10^{15}}{(60 \cdot 24 \cdot 365)} \approx 8 \cdot 10^9 \quad \text{years to execute the test suite}$$

Internal State.

$$\text{Number of states in route controller}^{\text{Number of routes}}$$

This implies

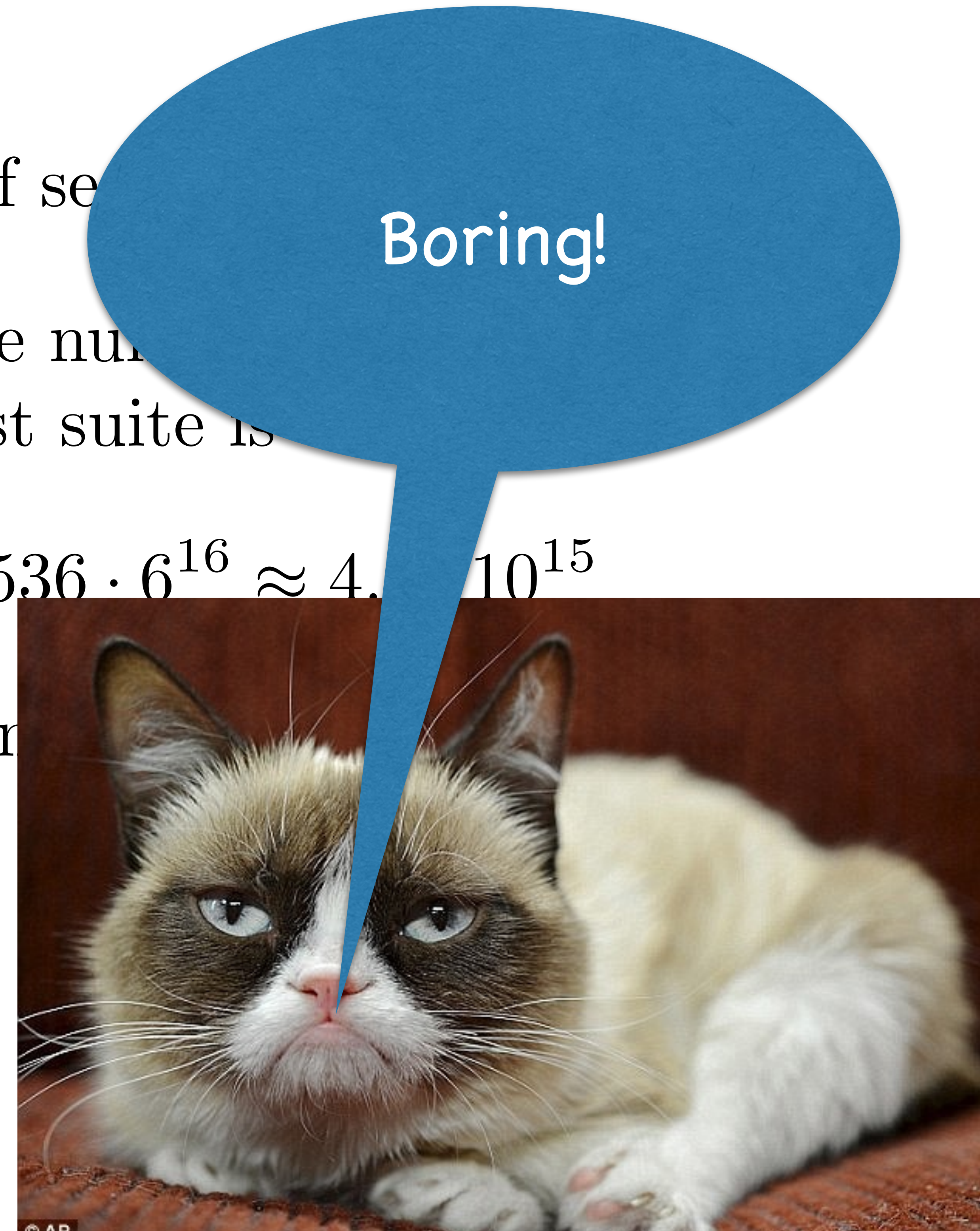
$$n = (4 + \text{Number of se}$$

Therefore order of magnitude for the nu
of test cases needed for complete test suite is

$$|\Sigma_I| \cdot n^2 = 1536 \cdot 6^{16} \approx 4.\;\;10^{15}$$

Suppose a system test case executio
Then you need

$$\frac{4.3 \cdot 10^{15}}{(60 \cdot 24 \cdot 365)} \approx 8 \cdot 10^9$$



A refined test strategy . . .

. . . in three steps

- Compositional Reasoning
- Equivalence Class Testing
- Randomisation in combination with boundary value selection

Compositional Reasoning

- From the knowledge about asserted behaviour of components . . .
- . . . conclude about the behaviour of the integrated system

More formally:

$$C_i \quad \mathbf{sat} \quad \text{Specification}_i, \quad i = 1, \ldots, n$$

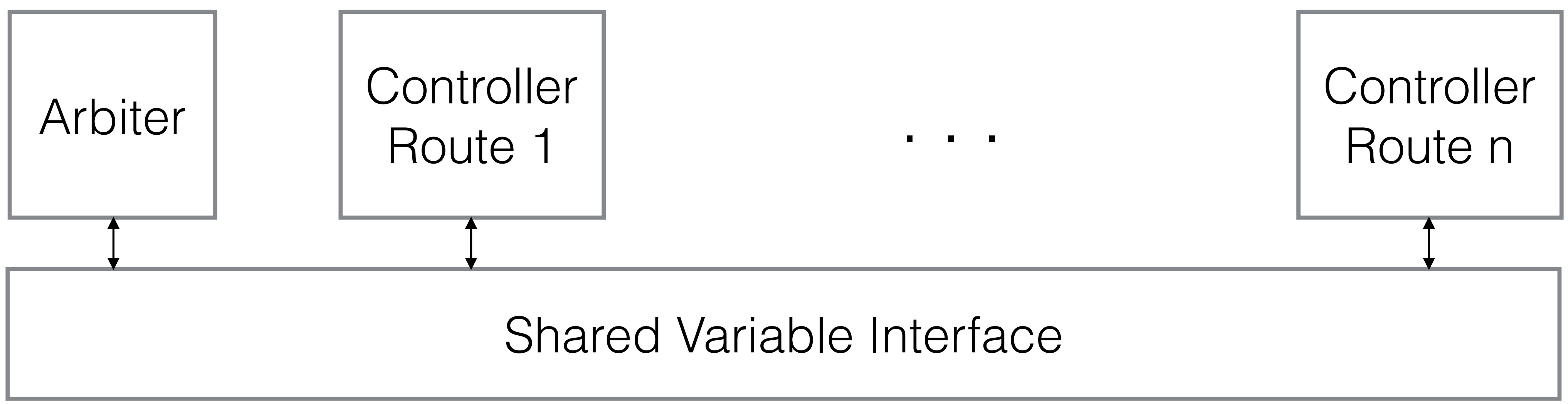
allows us to conclude that

$$(C_1 \parallel \cdots \parallel C_n) \quad \mathbf{sat} \quad \bigwedge_{i=1}^{n} \text{Specification}_i$$

provided that the integrated system is **compositional** – this is ensured, for example, if

- Components do not interfere with each other's internal state
- Data exchange over interfaces is synchronised

Application to Route Controller Tests



- synchronous execution
- synchronous data exchange over shared variables
- Arbiter acts a “semaphore” to ensure mutually exclusive route allocation

Refined Test Strategy

- **Refinement A**
  - Apply complete test suite on one route controller at a time
  - Conclude by compositional reasoning that whole system works correctly
  - This results in 8 x 1536 x 36 = 442368 test cases

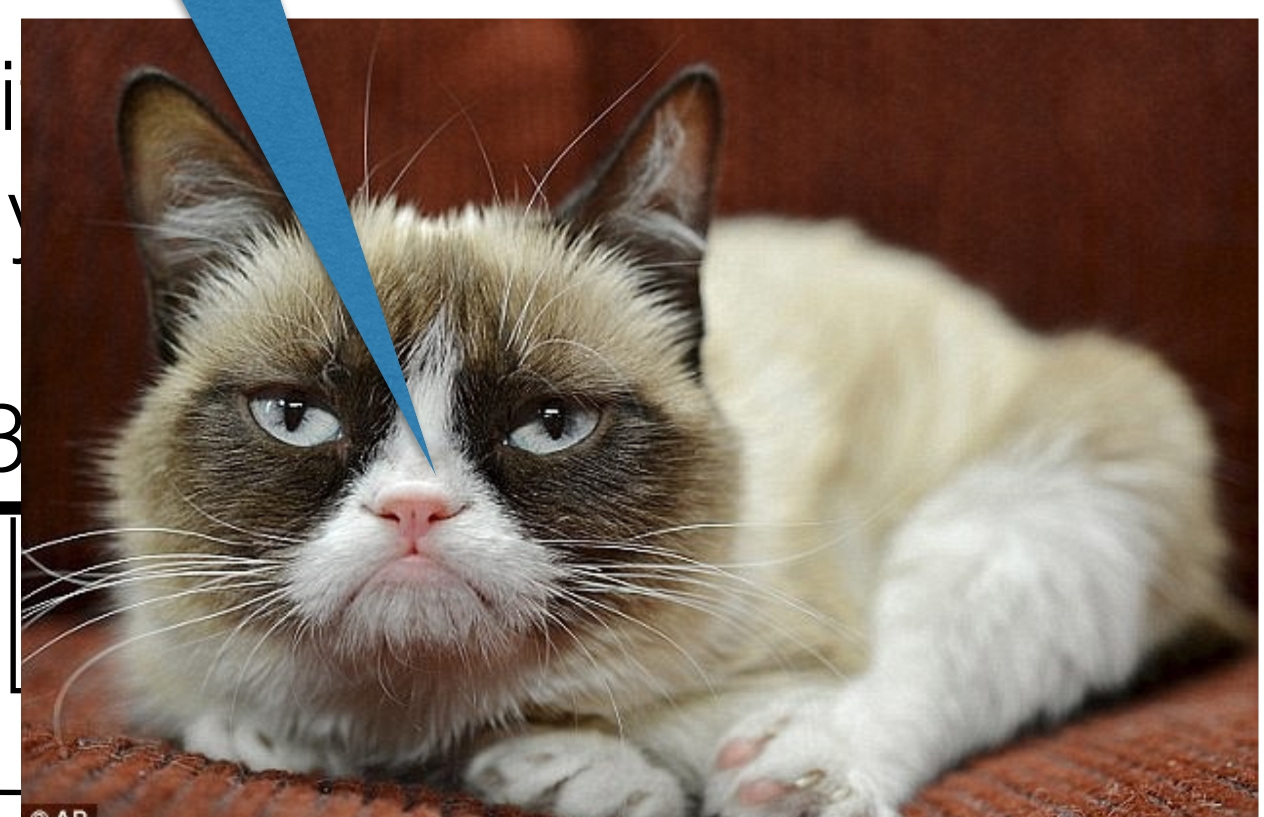
(Number of states)^2

Number of input vectors

Number of routes

Still not satisfied!
442368 test cases for such a small network configuration! I'm not impressed!

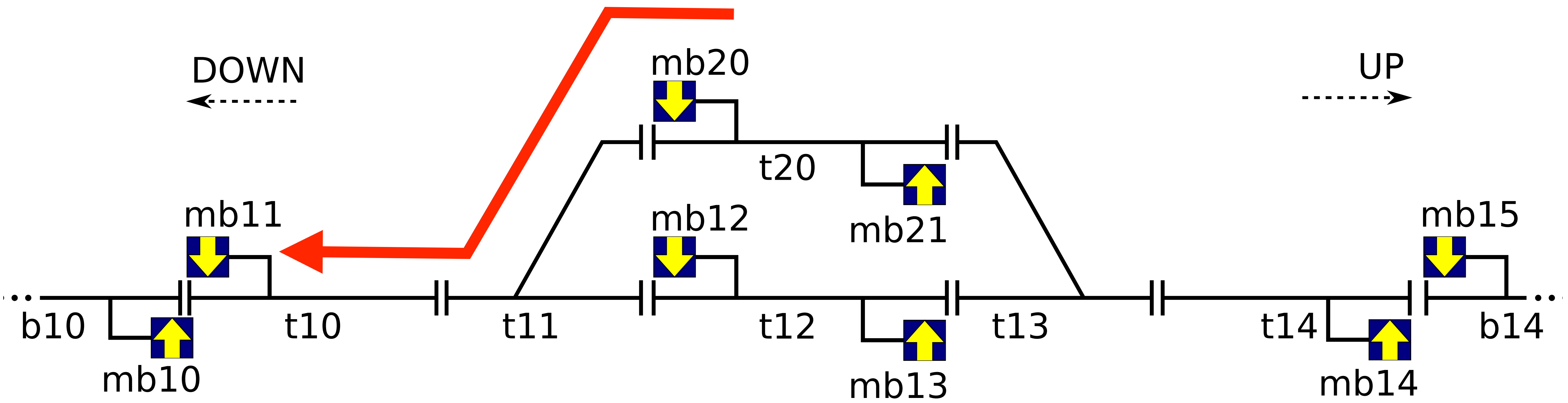
- **Refi**
  - Apply complete test su on one route controller at a time
  - Conclude by composi system works correctl
  - This results in 8 x 153 cases

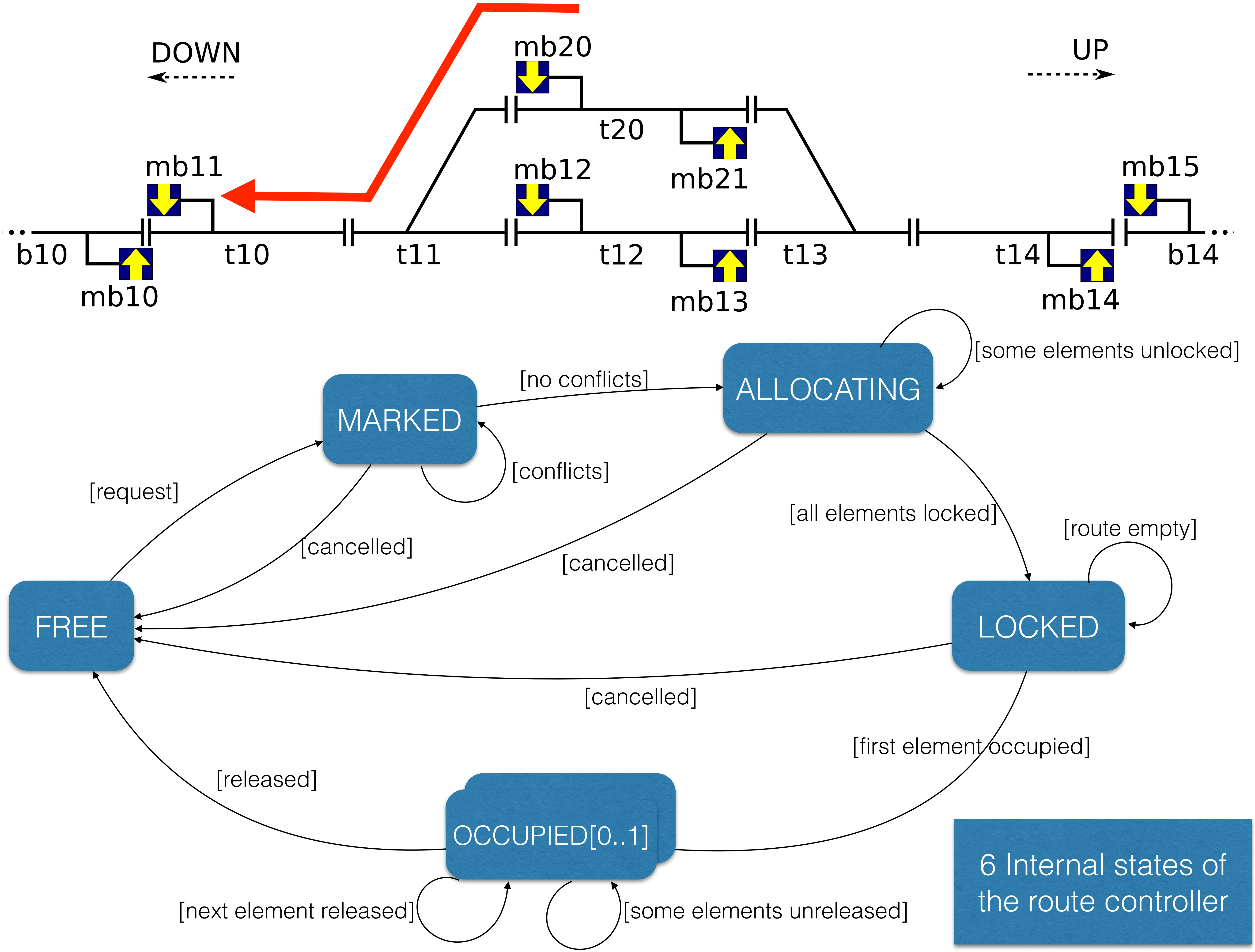


Refinement B – Input Equivalence Classes

- Recall
    - Input equivalence classes are constructed under the assumption that the SUT will process input of a class “in the same way”
- This intuitive concept can be formalised . . .
- . . . and also leads to a complete equivalence testing strategy

Example – Route(20,11)





DOWN
UP
mb20
t20
mb21
mb11
mb12
mb15
b10
t10
t11
t12
t13
t14
b14
mb10
mb13
mb14
MARKED
ALLOCATING
FREE
LOCKED
OCCUPIED[0..1]
[no conflicts]
[some elements unlocked]
[conflicts]
[request]
[all elements locked]
[cancelled]
[cancelled]
[route empty]
[cancelled]
[first element occupied]
[released]
[next element released]
[some elements unreleased]
6 Internal states of the route controller

Calculation of Input Equivalence Classes

$$\begin{aligned}
\text{conflicts} \quad \equiv \quad & \text{b10} \in \{L, O\} \vee \text{t10} \in \{L, O\} \vee \text{t11} \in \{L, O\} \vee \\
& \text{route}(10, 13) \in \{A\} \vee \\
& \text{route}(10, 21) \in \{A, L, O\} \vee \\
& \text{route}(12, 11) \in \{A, L, O\}
\end{aligned}$$

$$\begin{aligned}
\text{some elements unlocked} \quad \equiv \quad & \text{t10} = F \vee \text{t11} = F \vee \\
& \text{t11.pos} \neq \text{minus} \vee \\
& \text{mb12} \neq \text{HALT} \vee \\
& \text{mb10} \neq \text{HALT}
\end{aligned}$$

Calculation of Input Equivalence

. . . and all other
track elements
and internal state variables with
arbitrary values

$$
\begin{aligned}
\text{conflicts} \quad \equiv \quad & \text{b10} \in \{L, O\} \vee \text{t10} \in \{L, O\} \vee \text{t11} \in \{L, O\} \vee \\
& \text{route}(10, 13) \in \{A\} \vee \\
& \text{route}(10, 21) \in \{A, L, O\} \vee \\
& \text{route}(12, 11) \in \{A, L, O\}
\end{aligned}
$$

$$
\begin{aligned}
\text{some elements unlocked} \quad \equiv \quad & \text{t10} = F \vee \text{t11} = F \vee \\
& \text{t11.pos} \neq \text{minus} \vee \\
& \text{mb12} \neq \text{HALT} \vee \\
& \text{mb10} \neq \text{HALT} \qquad \ldots
\end{aligned}
$$

Calculation of Input Equivalence Classes

Every non-empty true/false combination of the six guard conditions defines one input equivalence class

- ☞ 6 guard conditions introduce approx. 64 input classes
- ☞ Number of test cases reduced to **$64x6^2$ = 2304 test cases per route**
- ☞ **18432 test cases for all 8 routes**
- ☞ These can be automatically executed in **307h –** or executed in parallel on 8 HW/SW integration test benches in 39h

Calculation of Input Equivalence Classes

- 6 guard conditions introduce approx. 64 input classes
- Number of test cases reduced to
  $64 \times 6^2 =$ 2304 test cases per route
- **18432 test cases for all 8 routes**
- These can be automatically executed in … or
  HW/S…

Should we refine the input classes and the assumptions about internal SUT states?

- Refining the input classes and assuming more internal states in the SUT would widen the fault domain – the probability that the SUT is inside the domain would be increased
- But this refinement would lead to an **exponential growth in the number of test cases**

Refinement C – Combination With Random and Boundary Value Testing

- Instead of always using the same representative of each input class representative, **select a random value of this class**, whenever it is used in the test case – combine this technique with **boundary value tests**

- Completeness is still guaranteed for SUTs inside the fault domain

- **For SUTs outside the fault domain, the test strength is significantly increased**

Side Remark: Boundary Values of Logical Formulas

- These are the so-called MC/DC conditions of a formula
- A and B has MC/DC valuations (0,1), (1,0), (1,1)
- A or B has MC/DC valuations (0,0), (1,0), (0,1)
- Basic idea: check predicate valuations where exactly one atom is responsible for the formula to evaluate to true or false

Refinement C – Combination With Random and Boundary Value Testing

- **Experimental results**

  - Mutation score (= number of uncovered SUT failures) up to 99%, where naive random testing only achieves a score of 68%

  - Published in *Felix Hübner, Wen-ling Huang, and Jan Peleska: Experimental Evaluation of a Novel Equivalence Class Partition Testing Strategy. In Blanchette and Kosmatov (eds.): Proceedings of the TAP 2015, Springer LNCS, Vol. 9154, pp. 155-173, 2015.*

Conclusion

- Testing route controllers for interlocking systems can be improved with respect to
  - Compositional strategy – from component tests to system integration tests
  - Application of a novel complete equivalence class testing strategy
  - Combination of this strategy with randomised value selection from input classes, including boundary values

Conclusion

- As a result,
  - Test cases are better justified (because they have been derived by complete strategy)
  - The resulting test suites have higher test strength then suites based on informal test selection criteria

Conclusion

Now why would anybody
wish to ask
a question about this stuff ?

- Te... ...er justified (because they have been deriv... ...y complete strategy)
- The resulting test suite... then suites based on info... criteria