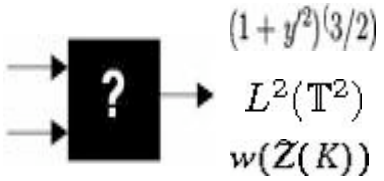


Using Invariant Detection Mechanism in Black Box Inference

Muzammil Shahbaz
France Telecom R&D
Grenoble

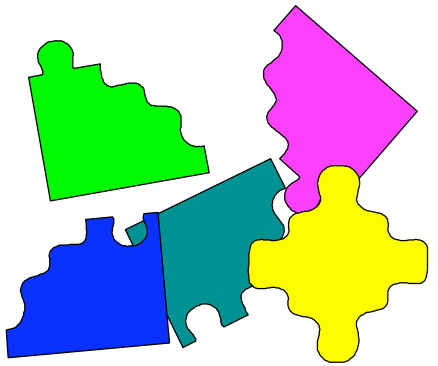
Roland Groz
LIG/ENSIMAG
Grenoble

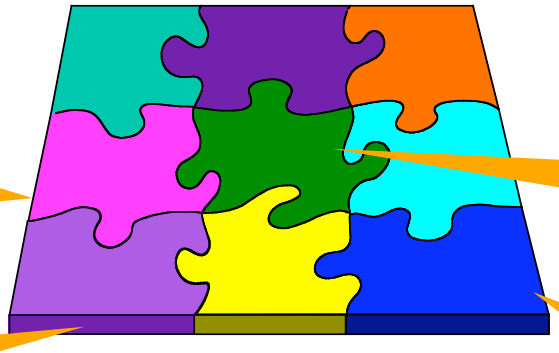
Component Based Software Engineering

Requirement Analysis

High Level System Design

Components selection





Integration Testing of System

System Analysis

Integrated System

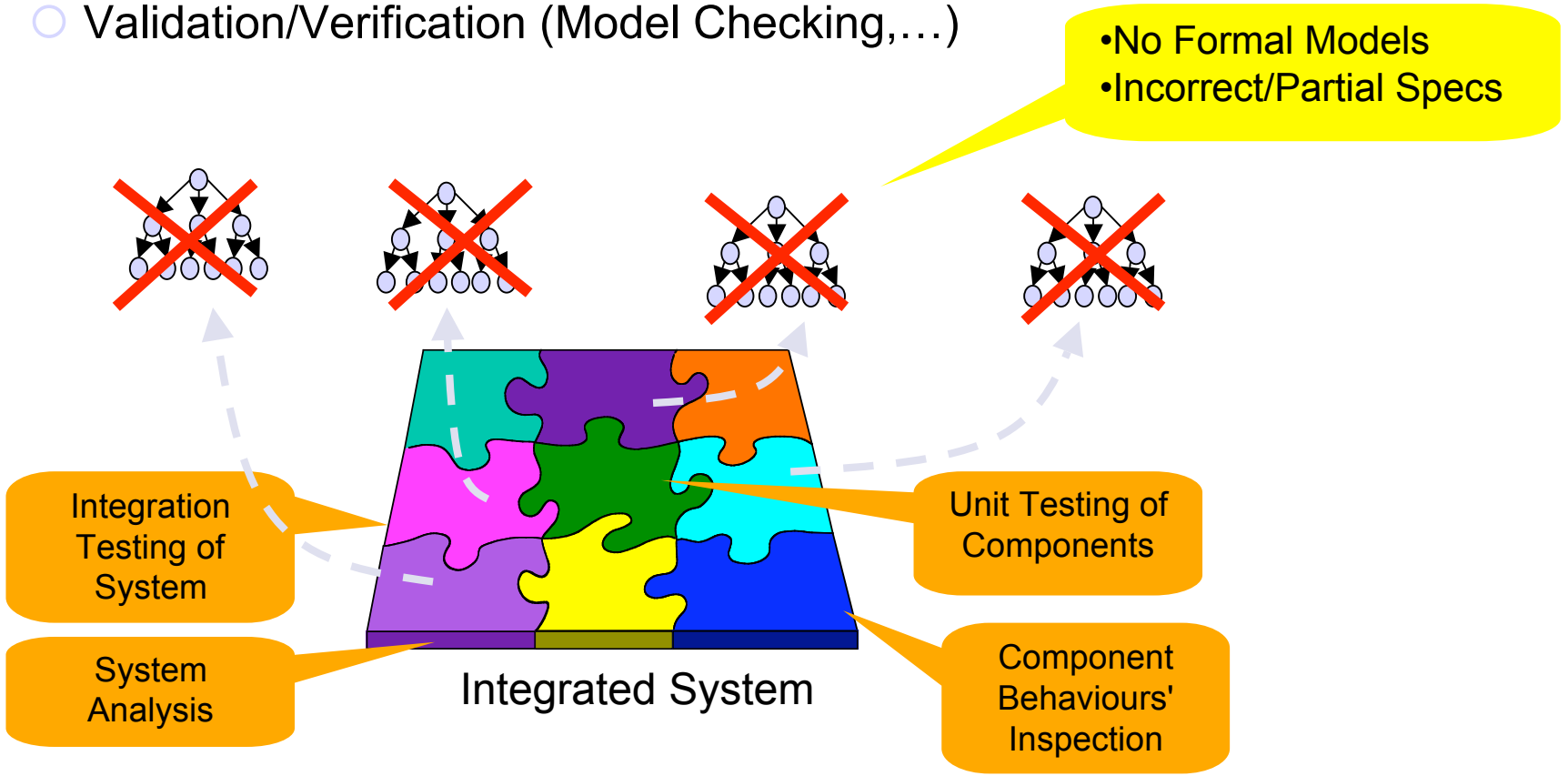
Unit Testing of Components

Component Behaviours' Inspection

Component Integration (System)

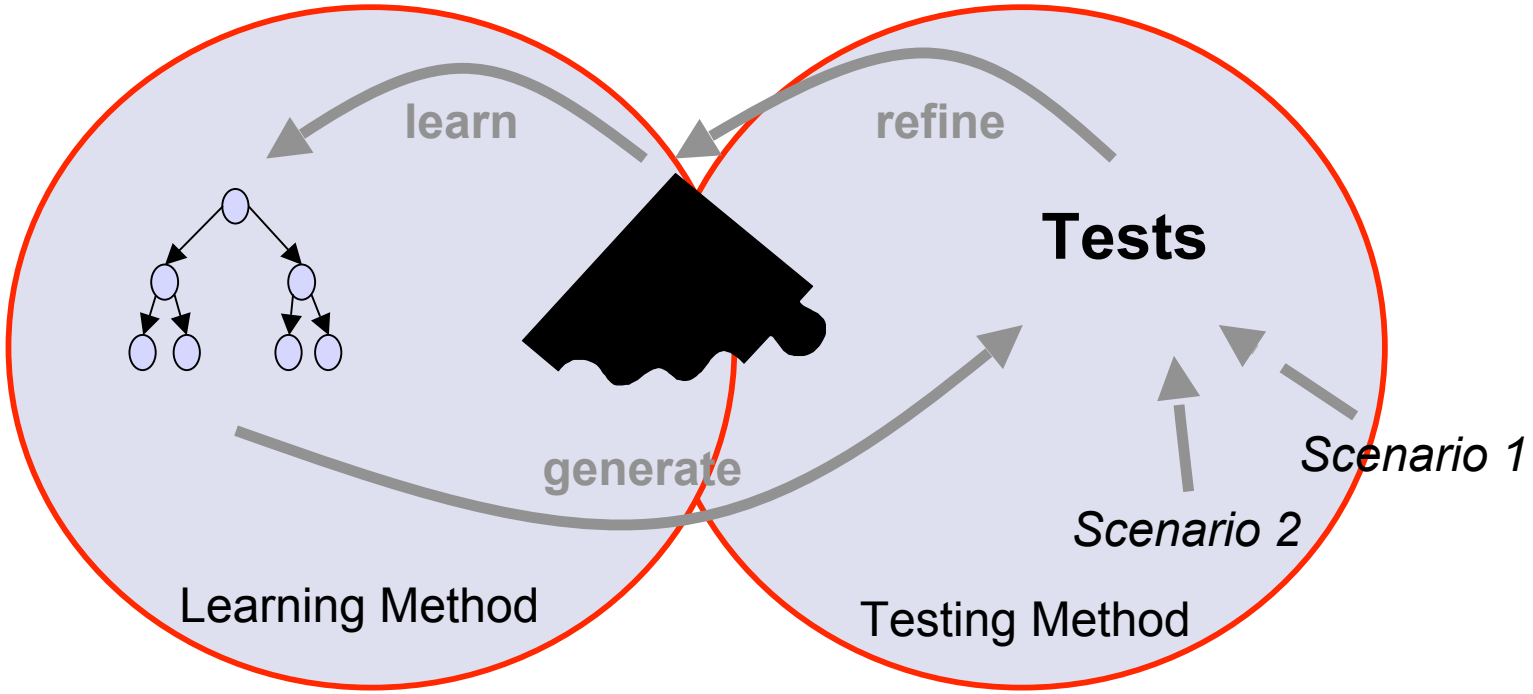
General Approach for CBSE

- **Formal Techniques**
  - Model Based Approach (MBT,…)
  - Validation/Verification (Model Checking,…)



Our Approach

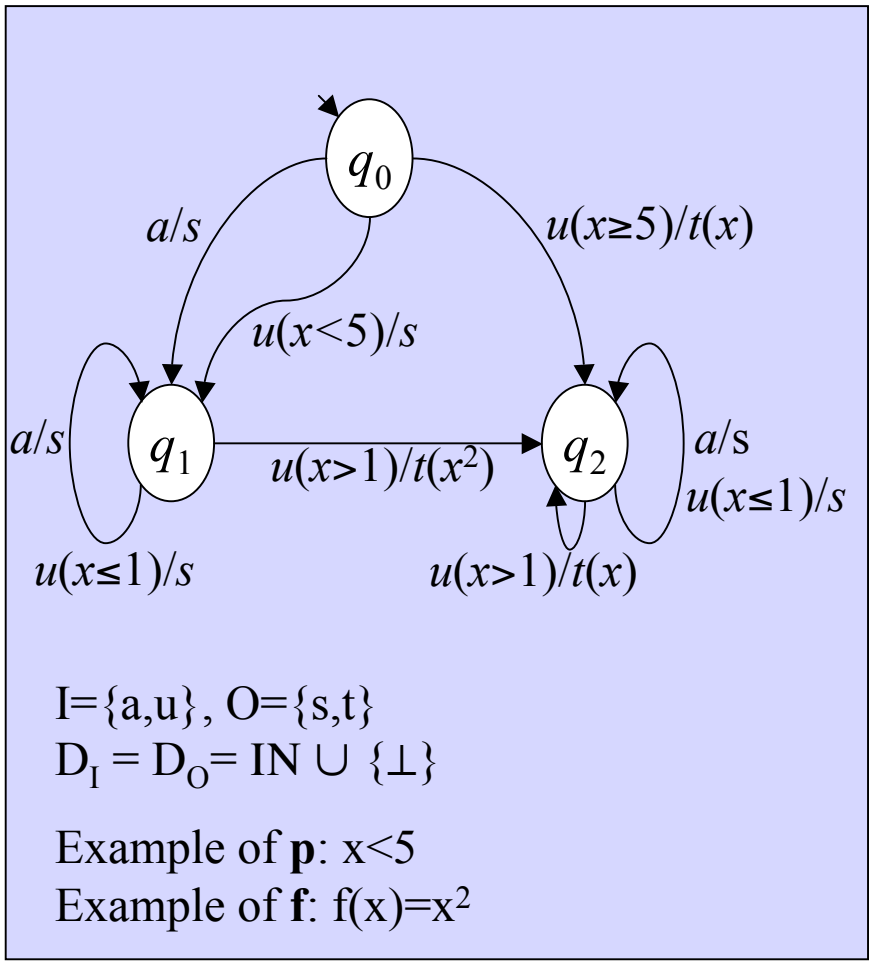
Incremental Inference of Black-Box Components to Support Integration Testing
[M. Shahbaz, TAIC PART 2006]



- **Benefits**
  - Components are Reverse Engineered
  - MBT/Verification Techniques can be applied on Inferred (partial) Models and can be refined incrementally

Parameterized Model (PFSM)

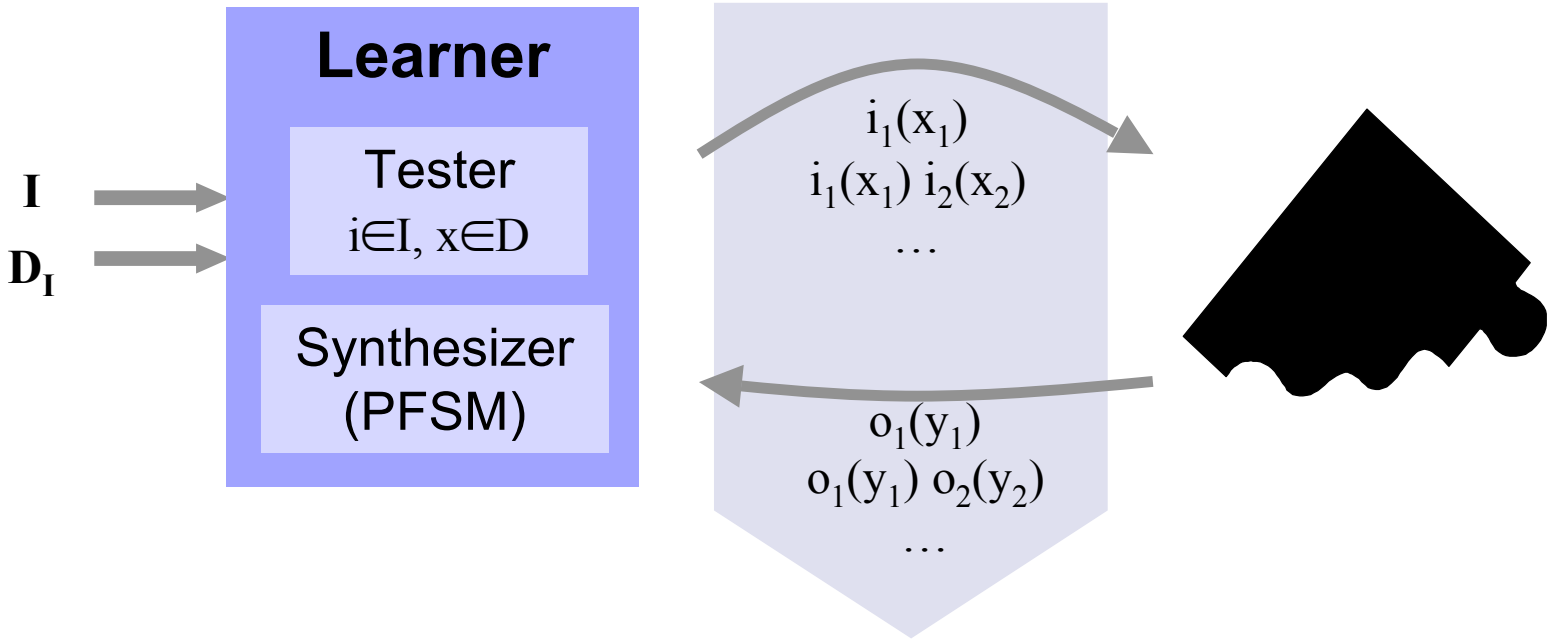
Learning and Integration of Parameterized Components through Testing
[M. Shahbaz, K. Li, R. Groz. TestCom 2007]

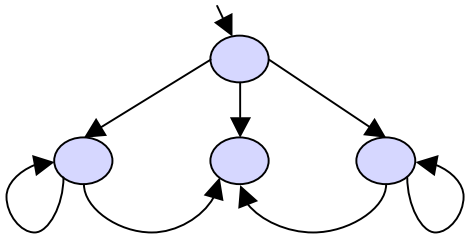


- Model: **M= (Q, I, O, D_I, D_O, T, q_0)**
  Q: States, q_0: Initial State
  I: Input Set **O**: Output Set
  D_I: Parameter Domain for I *(Finite/Infinite)*
  t ∈ T = (q, q', i, o, p, f),
  $p \subseteq D_I$ (predicate), $f: p \rightarrow D_O$
- **Input Enabled**: ∀ i(x) ∃t x∈p
- **Deterministic**: p∩p'=∅
- **Observable**: t≠t'⇒o≠o'

Learning Methodology

Learning and Integration of Parameterized Components through Testing
[M. Shahbaz, K. Li, R. Groz. TestCom 2007]

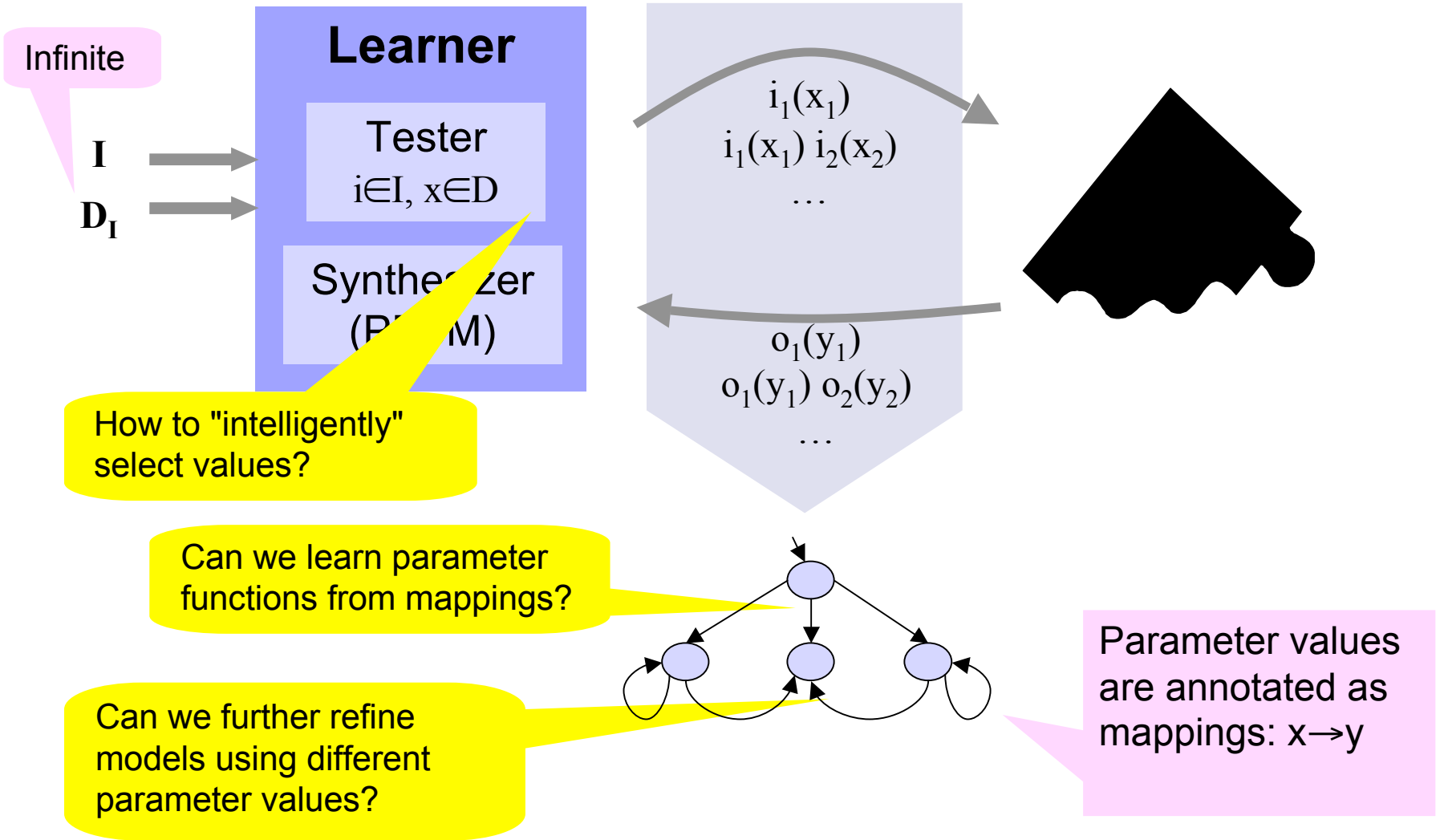




A partial PFSM model consistent with observations

Parameter values are annotated as mappings: x→y

Problem Statement



Invariant Detection

- Invariant is a property that holds at certain points in a program, e.g., x!=0, 5≤x≤10, y=2x+1, etc
- Invariant detection has been an active research for many years
- Detection of likely invariants can be done by observing variable values during program execution and reporting properties and relationships over values
- Used in Program Comprehension, Documentation and Maintenance tasks
- Applied in white box testing framework

Using Invariant Detection Mechanism

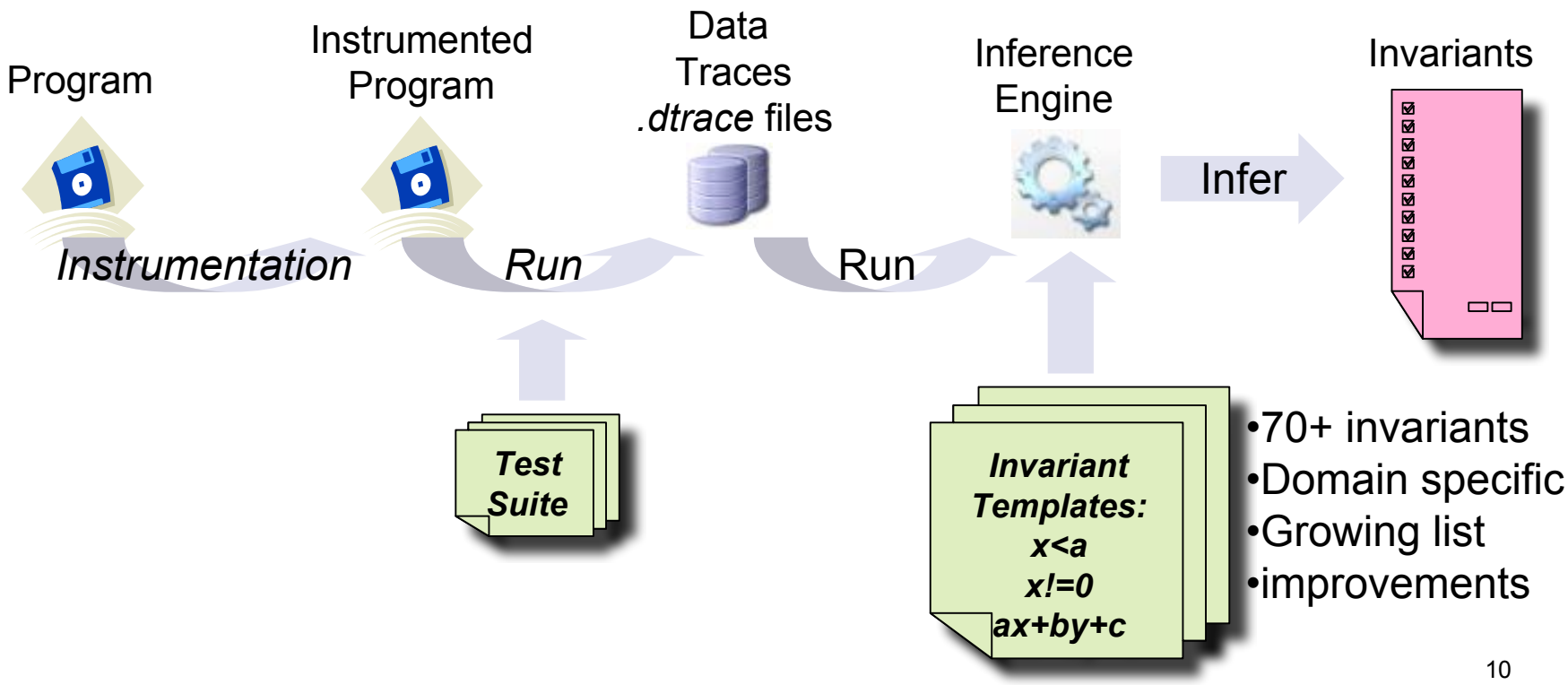
Using Invariant Detection Mechanism in Black Box Inference
[M. Shahbaz, R. Groz. ISOLA 2007] (to be appeared)

- Approach
  - Invariant Detection can be used in Black Box scenario, if enough data is available
  - Detected Invariants can guide new tests and hence leading towards new iterations of the learning algorithm
- Method:
  - Collect i/o traces from the initial run of the learning algorithm
  - Detect invariant over parameter values in traces using existing tools
  - Select parameter values guided by inferred invariants in the new learning iteration

The **Daikon** System

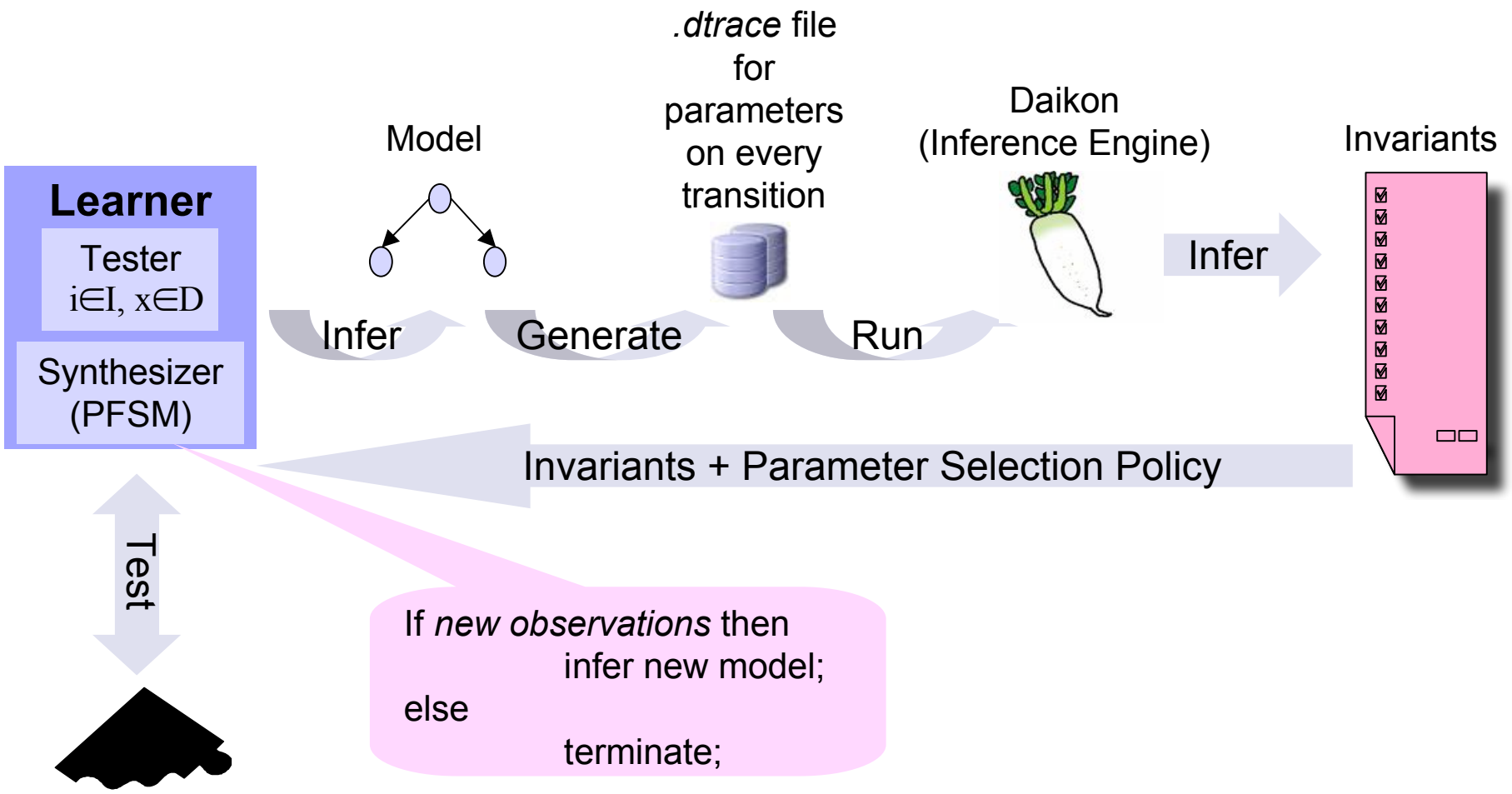
*"**dy**namic **con**jectures"*

- Developed by M. Ernst (Uni. of Washington, 2000)
  - The Daikon system for dynamic detection of likely invariants. [Science of Computer Programming, 2006]



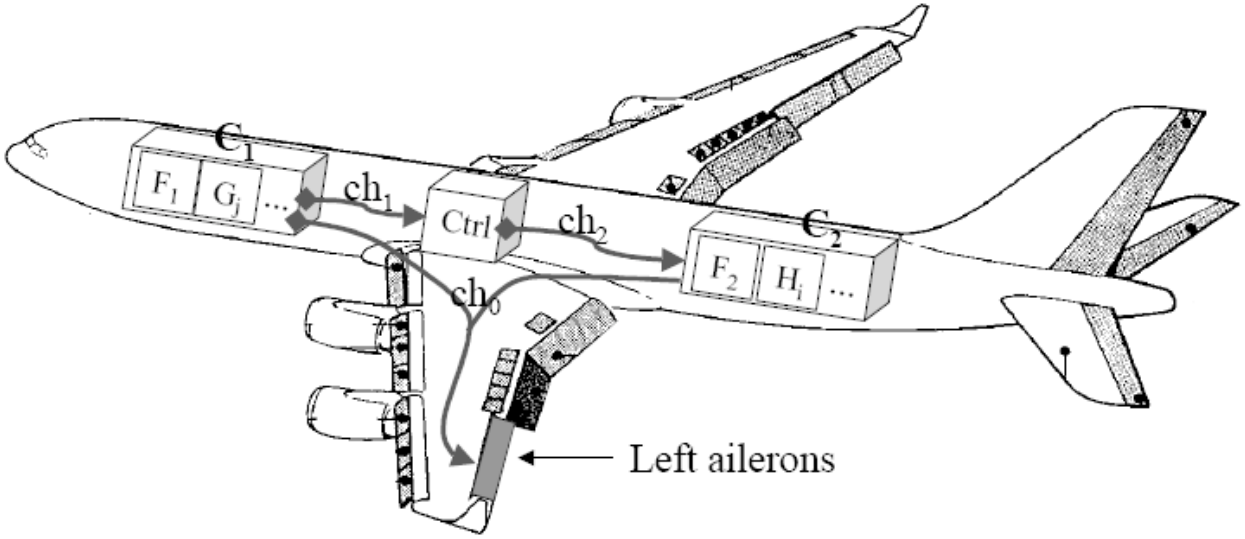
Using Daikon in Black Box Inference

Using Invariant Detection Mechanism in Black Box Inference
[M. Shahbaz, R. Groz. ISOLA 2007] (to be appeared)



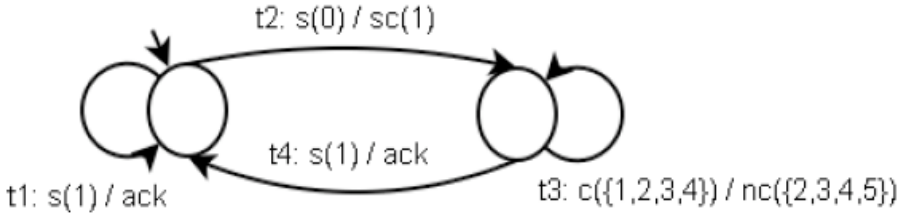
Ailerons Controller

TPAP: an algebra of pre-emptive processes for verifying real-time systems
[J. Ermont, F. Boniol. Electr. Notes Theor. Comput. Sci., 65(6), 2002]



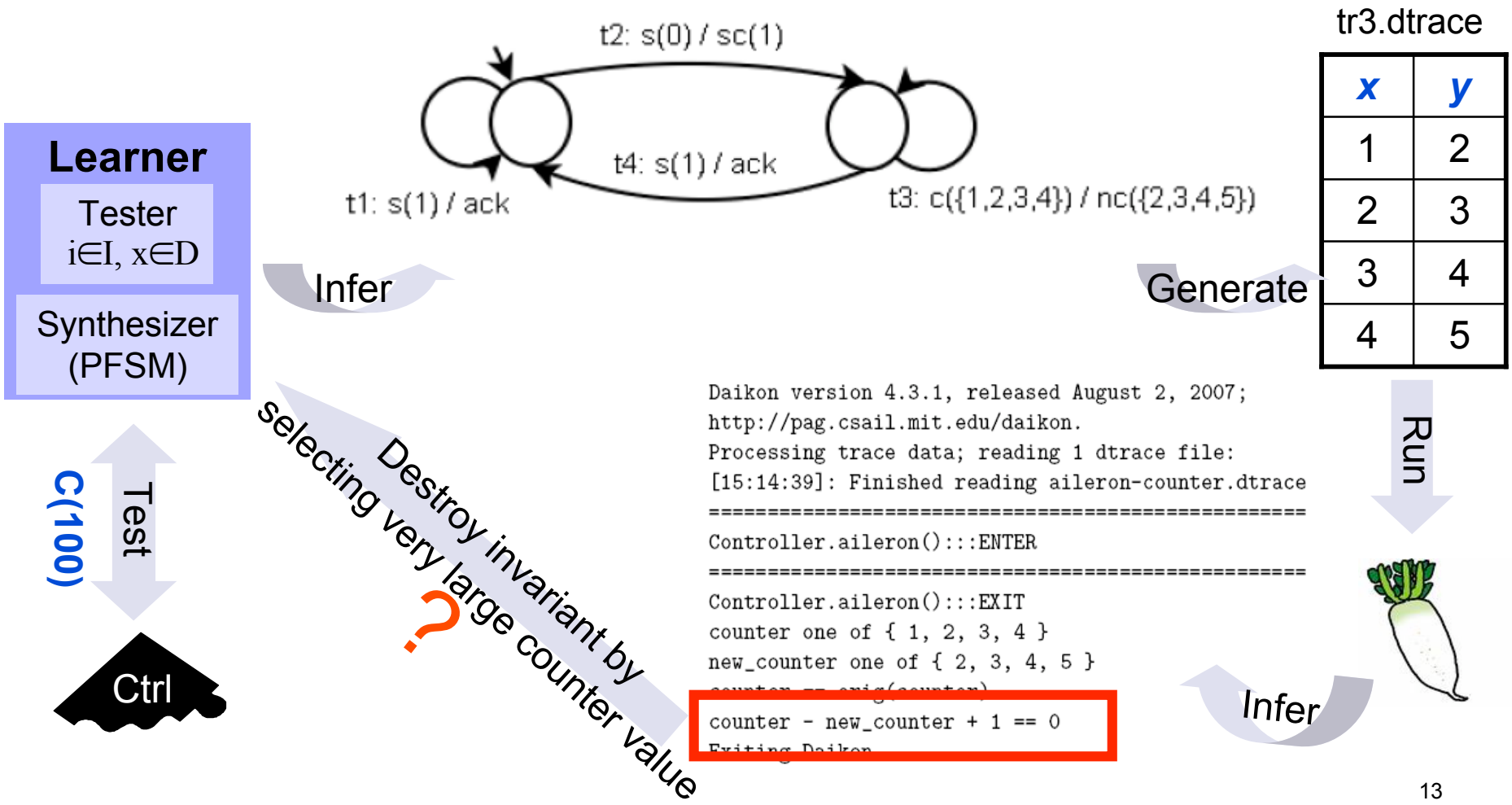
- C_1 sends periodically *s(1)* to *Ctrl*
- *Ctrl* sends *ack* to C_1
- When *Ctrl* receives *s(0)*, sets a counter *sc(1)*
- *Ctrl* checks the counter *c(x)* periodically and increments *x* every time
- When counter exceeds for certain threshold, *Ctrl* sends *Cmd* to C_2

- Inference of Ctrl

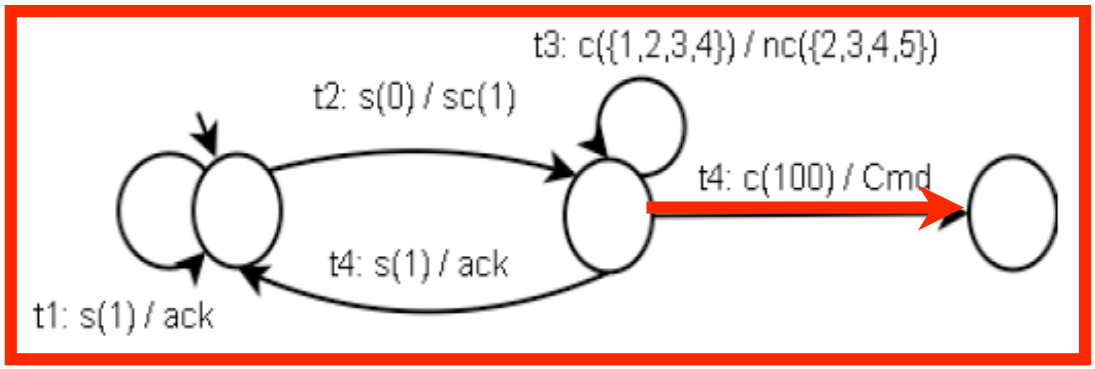


Generated from RALT

Experiment


t2: s(0) / sc(1)
t4: s(1) / ack
t1: s(1) / ack
t3: c({1,2,3,4}) / nc({2,3,4,5})
tr3.dtrace
x
y
1
2
2
3
3
4
4
5
Learner
Tester
i∈I, x∈D
Synthesizer
(PFSM)
Infer
Generate
Run
Test
C(100)
Ctrl
Destroy invariant by
selecting very large counter value
?
Daikon version 4.3.1, released August 2, 2007;
http://pag.csail.mit.edu/daikon.
Processing trace data; reading 1 dtrace file:
[15:14:39]: Finished reading aileron-counter.dtrace
==========
Controller.aileron():::ENTER
==========
Controller.aileron():::EXIT
counter one of { 1, 2, 3, 4 }
new_counter one of { 2, 3, 4, 5 }
counter - new_counter + 1 == 0
Infer

Experiment



tr3.dtrace

x	y
1	2
2	3
3	4
4	5

Learner

Tester
i∈I, x∈D

Synthesizer
(PFSM)

Infer

Generate

Run

Test

C(100)

Ctrl

Destroy invariant by selecting very large counter value

?

```
Daikon version 4.3.1, released August 2, 2007;
http://pag.csail.mit.edu/daikon.
Processing trace data; reading 1 dtrace file:
[15:14:39]: Finished reading aileron-counter.dtrace
================================================
Controller.aileron():::ENTER
================================================
Controller.aileron():::EXIT
counter one of { 1, 2, 3, 4 }
new_counter one of { 2, 3, 4, 5 }
counter - new_counter + 1 == 0
Exiting Daikon
```

Infer

Conclusion

- The selection of parameter values in Black Box Inference is crucial
- The conjecture can be different if different sets of values are used in learning method
- Invariants can be detected from the component using observations in the preliminary iterations of the learning method
- Detected Invariants can guide the selection of parameter values in new iterations of the Learning Method
- Testing with new parameter values in the learning method can refine the conjecture incrementally

Future Work

- Test Generation from Learned Invariants
- Study the limitations of the approach and improve the procedure
- Study other detectors, e.g.,
  - StInG (Standford Invariant Generator)
  - DIDUCE (Dynamic Invariant Detection Union Checking Engine)
  - Agitator? (Commercial Tool)
- Implementation and connecting with RALT
- More examples…. less stupid

Questions???