

NAT2TEST – GENERATING TEST CASES FROM NATURAL-LANGUAGE REQUIREMENTS

DeepSpec Summer School 2018
Princeton University (US)

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil



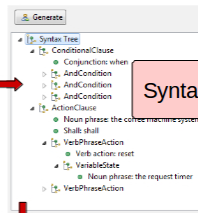
Centro de Informática (CIn) at UFPE | Brazil

- UFPE (1946)
 - 2,800 faculty members
 - 42,000 students
- Centro de Informática (1974)
 - 90 faculty members
 - 2,000 students
 - Strong research focus
(one of the **7 best in CS in Brazil**)
- Software Reliability Group (SRG)



NAT2TEST strategy (for data-flow reactive systems)

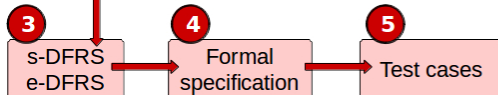
When the system mode is choice, and the coffee request button changes to pressed, and the request timer is lower than or equal to 30.0, the coffee machine system shall: reset the request timer, assign preparing weak coffee to the system mode.



2 Requirement frames

AGT	ACT	TOV	PAT
the coffee machine system	reset	-	the request timer
	AND		
the coffee machine system	assign	preparing weak coffee	the system mode

Actions



Console Variables Instances

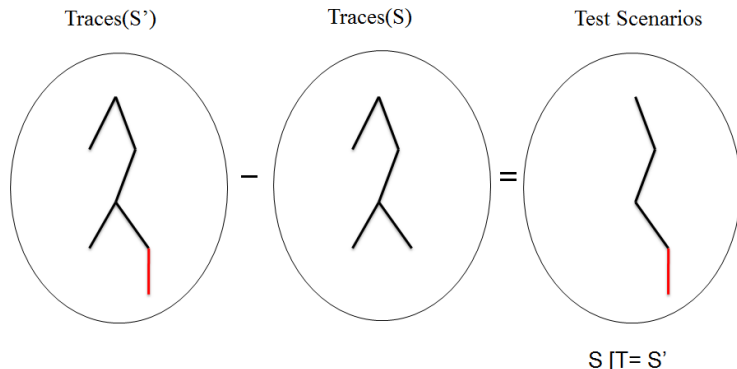
NAT2TEST - CONSOLE

TIME	the	coffee	request	button
0.0	false	false	1	0
1.0	false	true	0	0
2.0	true	true	3	0
12.0	true	true	1	1



NAT2TEST_{CSP}: sound test-case generation

- Test scenarios (traces) are obtained from the specification



NAT2TEST tool (short demo)

The screenshot displays the NAT2TEST tool interface. On the left, the 'Navigation View' shows a project structure for a 'Vending Machine' with sections for 'General Info', 'Requirements' (listing REQ001 to REQ005), 'Dictionary', 'Data-flow Reactive System' (with sub-items for Variables and Types, Functions, and Animation), 'SCR', and 'CSPm'. The 'Test Cases' section is currently selected.

The main workspace is divided into several panels:

- Requirements:** A list of requirements: REQ001, REQ002, REQ003, and REQ005.
- Selected Requirements:** A list containing REQ004.
- # Test cases per requirement:** A text input field containing the value '1'.
- Generate Test Cases - CSPm:** A button to initiate the test case generation process.
- Test Cases:** A list containing 'tc_REQ004_1'.
- Test Case Data:** A table showing the execution data for the selected test case.
- Console:** A log window showing the execution output.

Test Case Data Table:

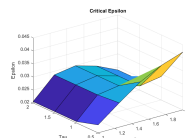
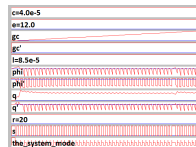
Time	I: the_coffee_request_button	I: the_coin_sensor	O: the_system_mode	O: the_coffee_machine_output
0.0	false	false	1	0
1.0	false	true	0	0
2.0	false	false	0	0
3.0	true	false	3	0
13.0	false	false	1	1

Console Output:

```
NAT2TEST - CONSOLE
2.0  false  false  0    0
3.0  true   false  3    0
13.0 false  false  1    1
-----
Time consumed by FDR: 5153
Time consumed by Z3: 496
```

Conclusion

- **NAT2TEST**: generating test cases from natural-language requirements
 - Evaluation considering examples from the **aerospace** (Embraer) and **automotive** (Mercedes) domains
- Current and future developments
 - h-NAT2TEST: considering hybrid systems
 - Compositional conformance relations
 - Compositional test case generation
 - Test case generation based on **Coq**



NAT2TEST – GENERATING TEST CASES FROM NATURAL-LANGUAGE REQUIREMENTS

DeepSpec Summer School 2018
Princeton University (US)

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil

