An integrated framework for analysing, simulating and testing UML models

Gustavo Carvalho[1], José Dihego[2], Augusto Sampaio[1]
(ghpc@cin.ufpe.br | jose.dihego@ifba.edu.br | acas@cin.ufpe.br)

[1] Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil

[2] Instituto Federal de Educação, Ciência e Tecnologia da Bahia, Brazil

Centro de Informática UFPE

Universidade Federal de Pernambuco

Motivation

Context: UML as a notation for modelling object-oriented software systems

Goal: enable analyses supported by formal methods

Challenges:

- formal semantics (e.g., fUML, PSSM, PSCS)
- integration between structural and behavioural models (i.e., a unified semantics)

Centro de Informática UFPE

Universidade Federal de Pernambuco

Contributions

Focus: behavioural (B) and structural (S) models

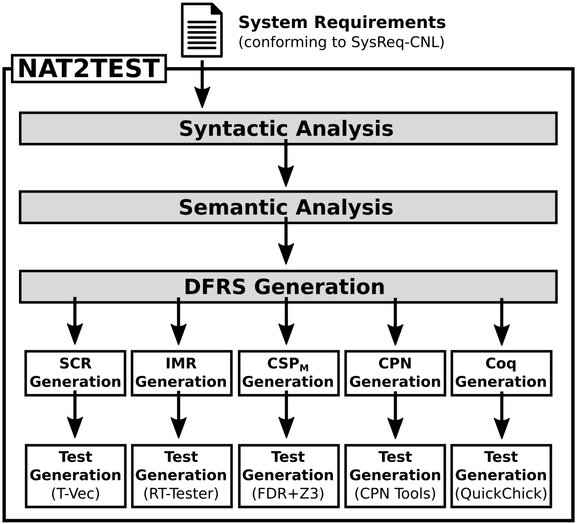
- B = state machine diagrams
- S = composite structure diagrams

Integration with the NAT2TEST strategy

Main contributions:

- Systematic and compositional process
- Validation of translation based on CSP
- Tool support along with the NAT2TEST tool
- Case studies:
  – dining philosophers
  – ring-buffer model

NAT2TEST strategy



SysReq-CNL: guarded actions

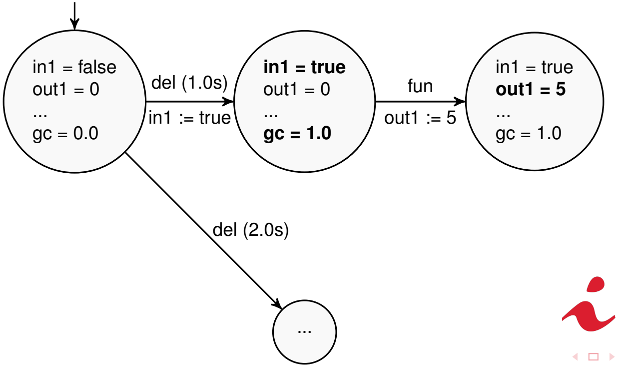
DFRSs: Data-flow Reactive Systems

s-DFRS: symbolic DFRS = $(I, O, T, gcvar, s_0, F)$

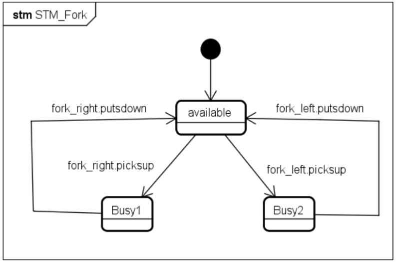
- $F = \{f_1, f_2, ...\}$, s.t. $f_1 = \{(static\_guard_1, timed\_guard\_1) \mapsto \{action_1, action_2, ...\}, ...\}$

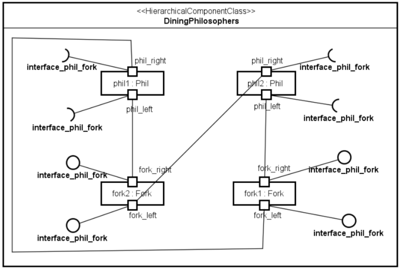
e-DFRS: expanded DFRS = $(I, O, T, gcvar, s_0, S, TR)$

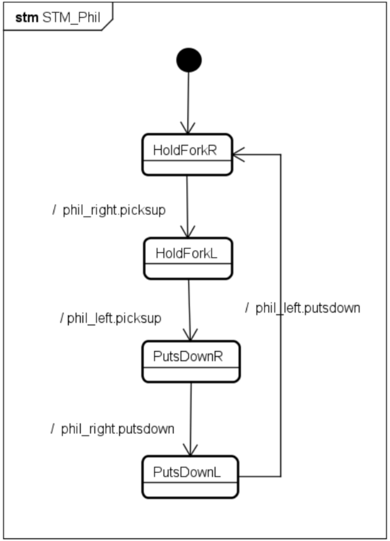
- del = delay transitions | fun = function transitions



Running example: dining philosophers







Centro de Informática UFPE

Universidade Federal de Pernambuco

From UML diagrams to CNL requirements: overview

DFRSs: communication via shared memory (events represented using variables)

Translation follows three phases[1]

1. SM: from transitions to sentences
   (conditions $\rightsquigarrow$ guards, effect $\rightsquigarrow$ actions)
2. CS: sentences replicated based on the number of instances
   (`STM_Fork_state` $\rightsquigarrow$ `STM_Fork_statefork1`, ...)
3. CS: sentences updated according to connections
   (sync.: one waits for the change, other performs the change)

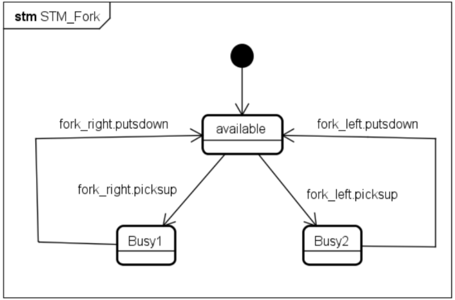
Mapping rules: presented in the paper

[1] SM = state machines, CS = composite structure

From UML diagrams to CNL requirements: phase 1

Phase 1 – SM: from transitions to sentences

```
When the STM_Fork_state is available,
 and the STM_Fork timer is greater than 0,
the STM_Fork_component shall:
 assign true to the fork_right_picksup,
 assign Busy1 to the STM_Fork_state,
 reset the STM_Fork timer.
```



From UML diagrams to CNL requirements: phase 2

Phase 2 – CS: sentences replicated based on the number of instances

```
When the STM_Fork_statefork1 is available, ...
the STM_Fork_componentfork1 shall:
 assign true to the fork_right_picksupfork1,
 assign Busy1 to the STM_Fork_statefork1, ...  .

When the STM_Fork_statefork2 is available, ...
the STM_Fork_componentfork2 shall:
 assign true to the fork_right_picksupfork2,
 assign Busy1 to the STM_Fork_statefork2, ...  .
```

Centro de Informática UFPE
Universidade Federal de Pernambuco

From UML diagrams to CNL requirements: phase 3

Phase 3 – CS: sentences updated according to connections

```
When the STM_Phil_statephil1 is HoldForkR, ...,
 and the STM_Fork_statefork1 is available,
the STM_Phil_componentphil1 shall:  assign true to the
 phil_right_picksupphil1__fork_left_picksupfork1, ...  .

When the STM_Fork_statefork1 is available, ...,
 and the phil_right_picksupphil1__fork_left_picksupfork1
  becomes true,
the STM_Fork_componentfork1 shall:  assign false to the
 phil_right_picksupphil1__fork_left_picksupfork1, ...  .
```

Centro de Informática UFPE

Universidade Federal de Pernambuco

From UML diagrams to CNL requirements: validation

Validation of translation based on a CSP semantics

- `S_NAT2TEST`: our semantics (derived by the NAT2TEST strategy)
- `S_UML`: obtained independently from the same diagrams [FLS+24]
- `S_Classical`: classical semantics written by Roscoe

Centro de Informática UFPE

Universidade Federal de Pernambuco

From UML diagrams to CNL requirements: validation

S_NAT2TEST

```
...
SPECIFICATION(memory) = FUN(...)
SPECIFICATION2(memory) = ... delayTransition ...
SPECIFICATION3(memory) = INPUTS(memory)
SPECIFICATION4(memory) = SPECIFICATION(memory)

SYSTEM = SPECIFICATION(seq(initialBinding))
SYSTEM' = SYSTEM [[ ... <- ... ]]
S' = SYSTEM' \ {| ... |}

transparent sbisim, diamond
sbdia(P) = sbisim(diamond(P))
S'' = (sbdia(S'))

S_NAT2TEST = S''
S_NAT2TEST' = S_NAT2TEST \ { ... }
```

S_Classical

```
...
PHILS = ||| i:PHILNAMES @ PHIL(i)
FORKS = ||| i:FORKNAMES @ FORK(i)
PHILS_SYSTEM = PHILS [| {|picks, putsdown|} |] FORKS

S_Classical = PHILS_SYSTEM
S_Classical' = S_Classical [[ ... <- ...]] \ { ... }
```

S_UML

```
...
processcomp =
  (inter_inter_inter_fork2_phil2_phil1_fork1)
  [| ... |] BFIO_INIT(fork_left.2,phil_right.1)

processcomp_com = processcomp
  [| ... |] BFIO_INIT(fork_right.2,phil_left.2)

processcomp_com_com =  processcomp_com
  [| ... |] BFIO_INIT(fork_right.1,phil_left.1)

processcomp_com_com_com = processcomp_com_com
  [| ... |] BFIO_INIT(fork_left.1,phil_right.2)

S_UML = processcomp_com_com_com
S_UML' = S_UML [[ ... <- ... ]] \ { ... }
```

Centro de Informática UFPE

Universidade Federal de Pernambuco

From UML diagrams to CNL requirements: validation

We prove (via FDR) that:

```
assert S_NAT2TEST' [T= S_UML'
assert S_UML' [T= S_NAT2TEST'

assert S_NAT2TEST' [T= S_Classical'
assert S_Classical' [T= S_NAT2TEST'
```

All models have precisely the same trace semantics

Centro de Informática UFPE

Universidade Federal de Pernambuco

Tool support

JavaScript code: obtain textual representation of AstahUML models

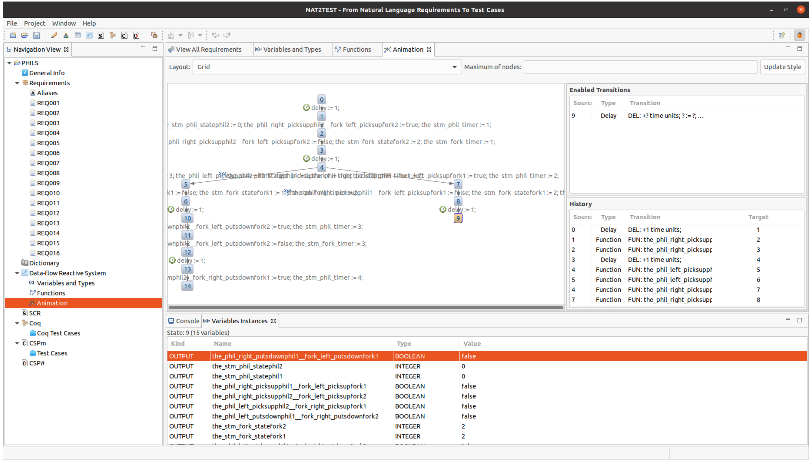
```
machine STM_Fork
state Available ...
transition Available fork_right.picksup Busy1 ...
```

Python scripts: implement the mapping rules in a very direct way

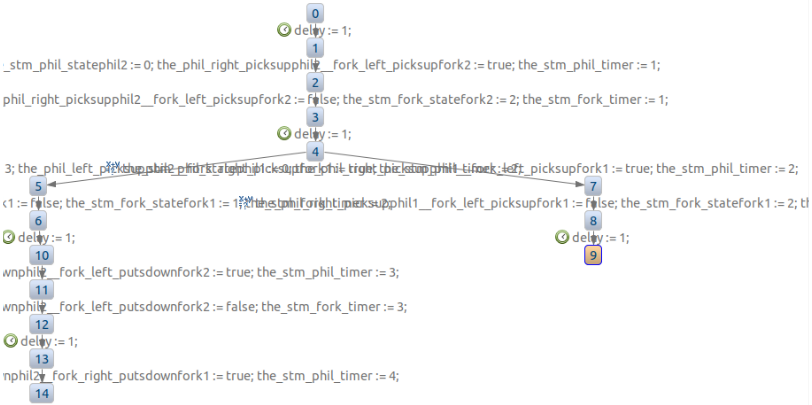
- Uses text templates

```
When the {1}_state is {2},
  and the {3} timer is greater than 0,
the {4}_component shall:
  assign true to the {5},
  assign {6} to the {7}_state,
  reset the {8} timer.
```

Tool support provided by the NAT2TEST strategy



Tool support provided by the NAT2TEST strategy



Related work

	B. diagr.	S. diagr.	Simul.	Testing	Verif.	Code gen.
CA21	SM	BD, IB	●	○	●	○
LMC$^+$17	Ac, SM, Sq	BD, IB	○	○	●	○
HMG$^+$23	Ac, SM, Sq	BD, IB	●	●	●	●
EMM24	Ac, SM, Sq	–	●	◑	○	○
FLS$^+$24	SM	CI, CS	●	○	●	○
Ours	SM	CS	●	●	●	○

CA21: *Direct Model-checking of SysML Models.*

LMC$^+$17: *An integrated semantics for reasoning about SysML design models using refinement.*

HMG$^+$23: *Pragmatic verification and validation of industrial executable SysML models.*

EMM24: *To Do or Not to Do: Semantics and Patterns for Do Activities in UML PSSM State Machines.*

FLS$^+$24: *A formal component model for UML based on CSP aiming at compositional verification.*

Conclusion

Analysing, simulating, and testing UML models

- Combination of behavioural + structural diagrams
- Enabled by the NAT2TEST strategy

Future work:

- Complex constructs
- More UML diagrams
- Additional case studies
- Further explore the NAT2TEST strategy

Centro de Informática UFPE

Universidade Federal de Pernambuco

An integrated framework for analysing, simulating and testing UML models

Gustavo Carvalho[1], José Dihego[2], Augusto Sampaio[1]
(ghpc@cin.ufpe.br | jose.dihego@ifba.edu.br | acas@cin.ufpe.br)

[1]Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil

[2]Instituto Federal de Educação, Ciência e Tecnologia da Bahia, Brazil

Centro de Informática UFPE

Universidade Federal de Pernambuco