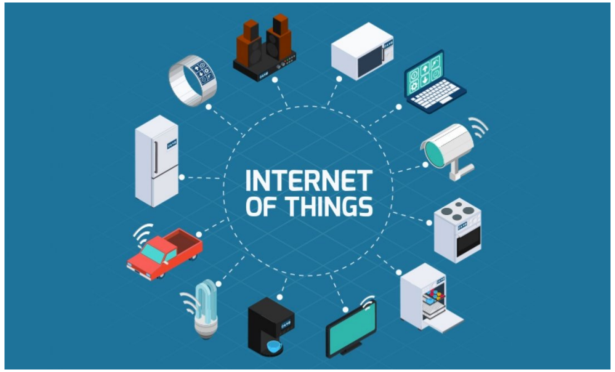
Verifying SysML/UML (Behavioural) Diagrams

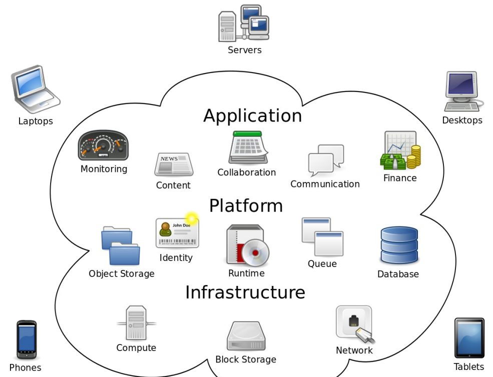
Lucas Lima

MSDL Summer workshop

01 September 2023

(Concurrent) System complexity concerns





Cloud computing

MSDL

Motivation

IN(SEMI)FORMAL MODELS

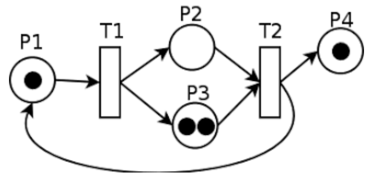
- Easy to learn and to create models
- Facilitates communication
- Property verification is limited and (usually) human-dependant

UNIFIED MODELING LANGUAGE UML™

OMG SYSTEMS MODELING LANGUAGE SysML™

FORMAL MODELS

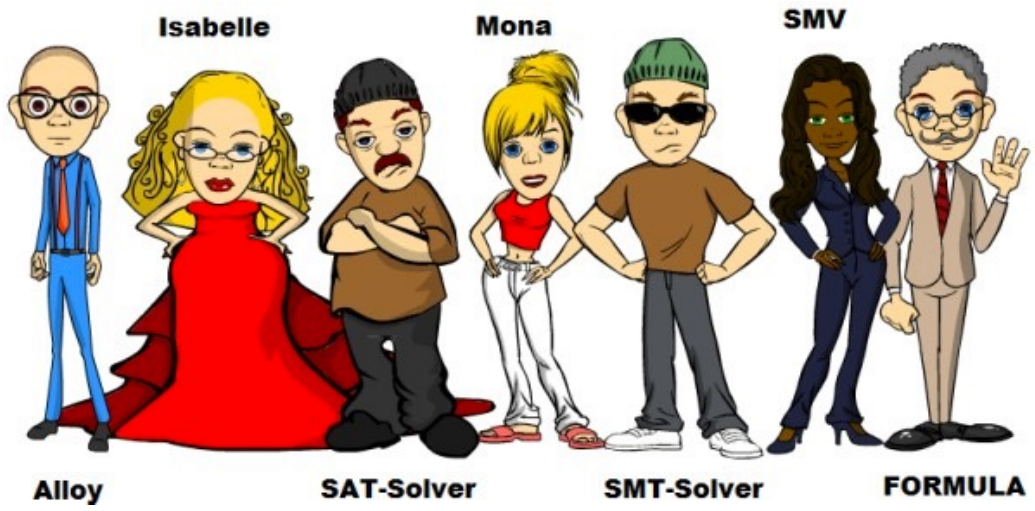
- Difficult to learn and manipulate
- Properties can be soundly verified
- Usually, supported by tools



CSP_M

MSDL

Formal Methods

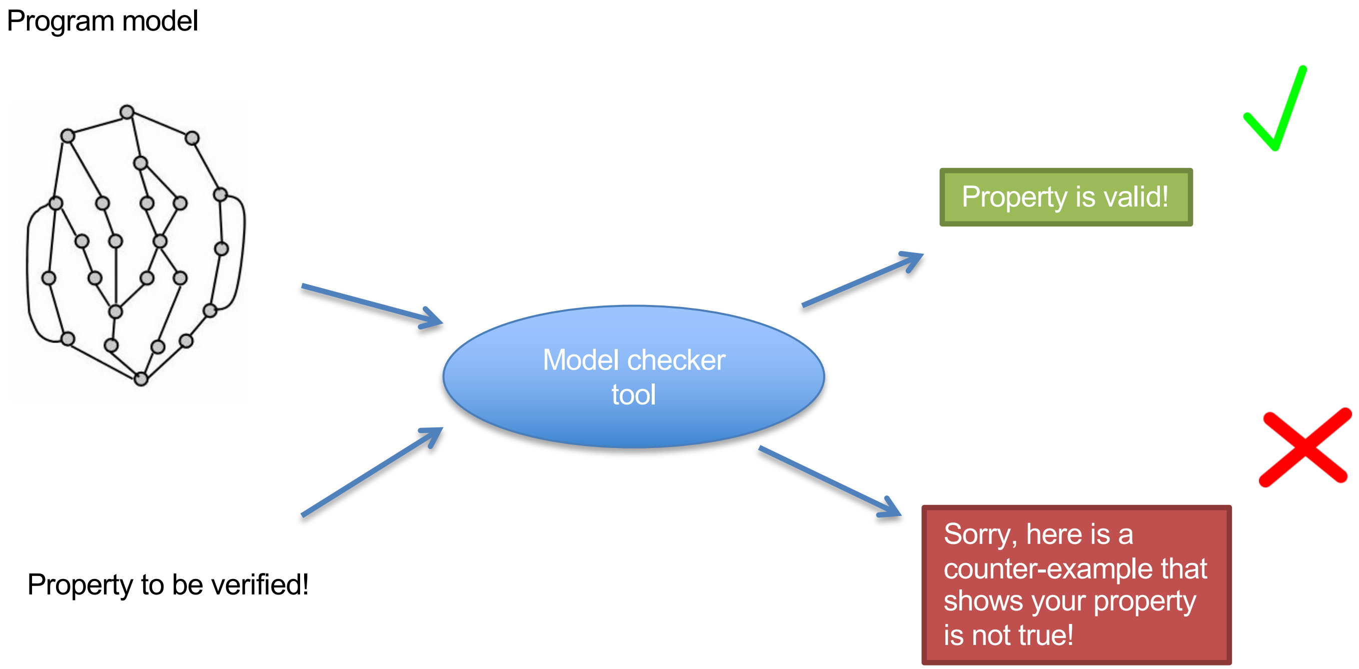


http://www.formal-methods.net/intro/

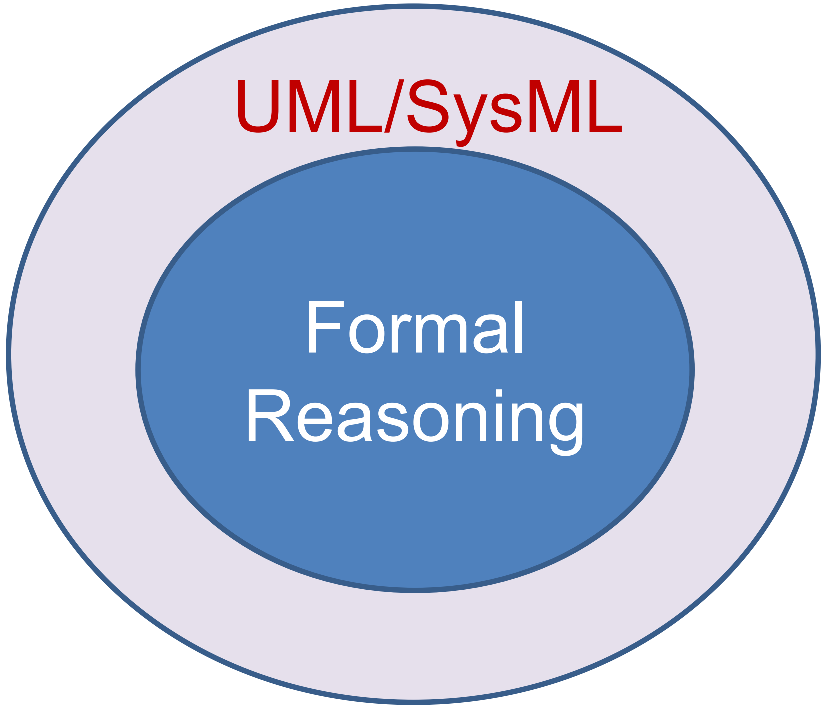
Mathematical approaches to software and system development which support the rigorous specification, design and verification of computer systems.

MSDL

Model checking !!!



Proposal



Formal Semantic Domain

- **CSP** – Communicating Sequential Process
  - Initially proposed by Tony Hoare in 1978
  - It has been applied in industry as a tool for <u>specifying and verifying</u> the concurrent aspects of systems
  - Influenced the design several languages, like occam, Limbo, RaftLib, Erlang, Go, Crystal, and Clojure's core.async
  - CSP_M is its machine-readable dialect
  - The *Failures/Divergence Refinement* (FDR) checker is the most well-known CSP tool

MSDL

Why CSP?

- Expressiveness of the language
- Compositional Operators
- Mature model checker (FDR)
- Established refinement theory

MSDL

CSP at a glance

```
NAT = {0..MAX}
MAX = 5
channel     put, get: NAT

Buffer(b) = ( length(b) < 5 & put?x -> Buffer(b^<x>) )
                  []
                ( length(b) > 0 & get!(head(b)) -> Buffer(tail(b)))

Producer = put!1 -> Producer
Consumer = get?x -> Consumer
System = (Buffer(<>) [|{|put,get|}|] (Producer ||| Consumer))
```

Types and Values

Channel declaration

PROCESSES

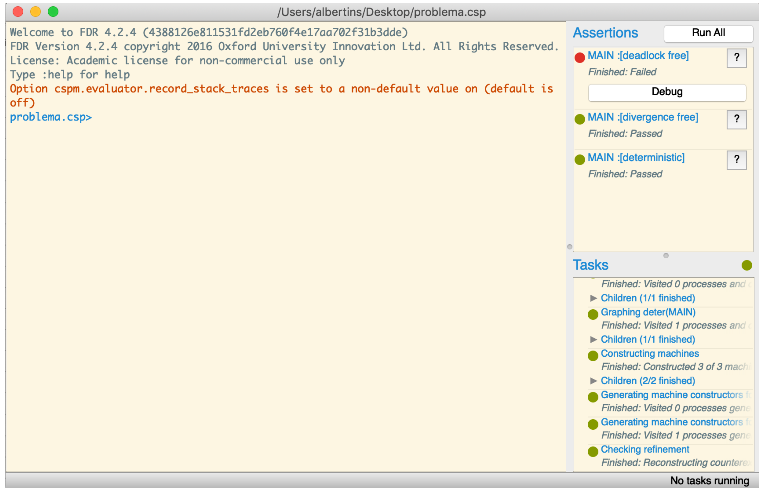
External Choice

Interleaving

Synchronized Parallelism

Verification - FDR

- FDR – Failures-Divergence Refinement
- User interface
  - animation
  - type checking
  - verification of properties like deadlock, divergence, determinism and refinement
- API
  - Java, Python and C++
  - Only works if executed from the FDR installation folder



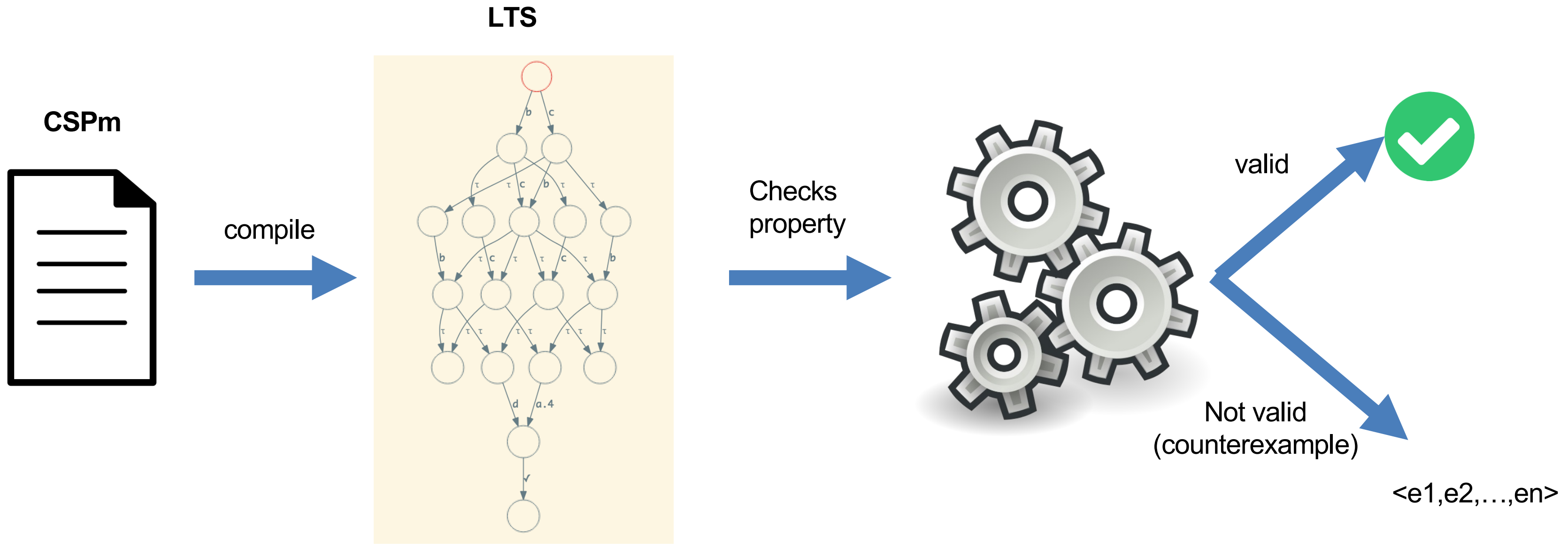
```
Welcome to FDR 4.2.4 (4388126e811531fd2eb760f4e17aa702f31b3dde)
FDR Version 4.2.4 copyright 2016 Oxford University Innovation Ltd. All Rights Reserved.
License: Academic license for non-commercial use only
Type :help for help
Option cspm.evaluator.record_stack_traces is set to a non-default value on (default is
off)
problema.csp>
```

```
Assertions
MAIN :[deadlock free]
Finished: Failed
MAIN :[divergence free]
Finished: Passed
MAIN :[deterministic]
Finished: Passed

Tasks
Finished: Visited 0 processes and
▶ Children (1/1 finished)
Graphing deter(MAIN)
Finished: Visited 1 processes and
▶ Children (1/1 finished)
Constructing machines
Finished: Constructed 3 of 3 mach
▶ Children (2/2 finished)
Generating machine constructors
Finished: Visited 0 processes gen
Generating machine constructors
Finished: Visited 1 processes gen
Checking refinement
Finished: Reconstructing countere
No tasks running
```

MSDL

Verification - FDR



MSDL

Verification - FDR

- Properties are checked using assertions
- Given that `MODEL` is the CSP process translated from an Activity
- Deadlock
  - `assert MODEL :[deadlock free]`
- Determinism
  - `assert MODEL :[deterministic]`
- In case a deadlock or nondeterminism is found, FDR returns a trace of events that leads to the issue

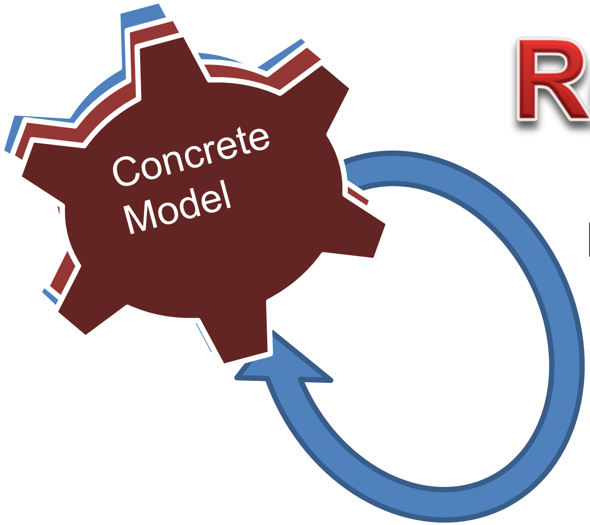
Application 1

Checking Sequence Diagram Refinement

https://link.springer.com/chapter/10.1007/978-3-319-49815-7_14

Concern

- Stepwise design



Refine *

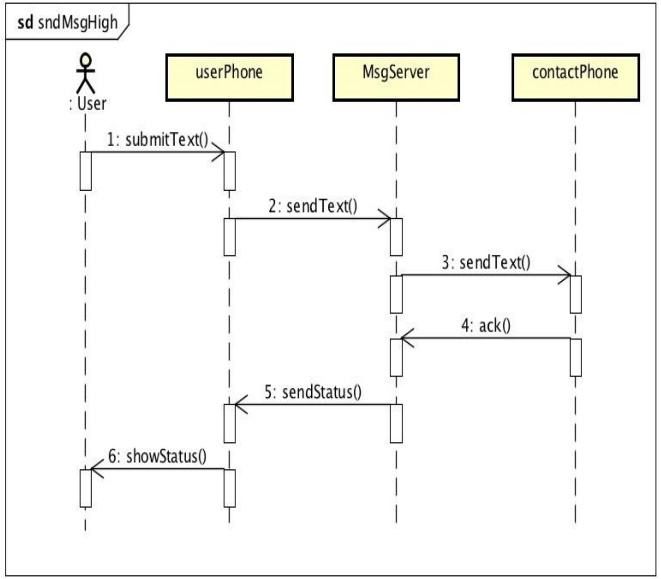
What is a Refinement?

MSDL

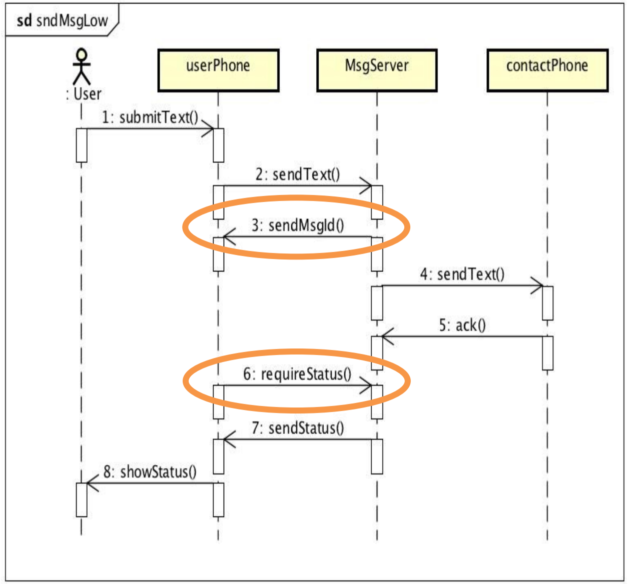
Refinement Notions

- Strict Increment Refinement - Example

Abstract Model



Refined Model

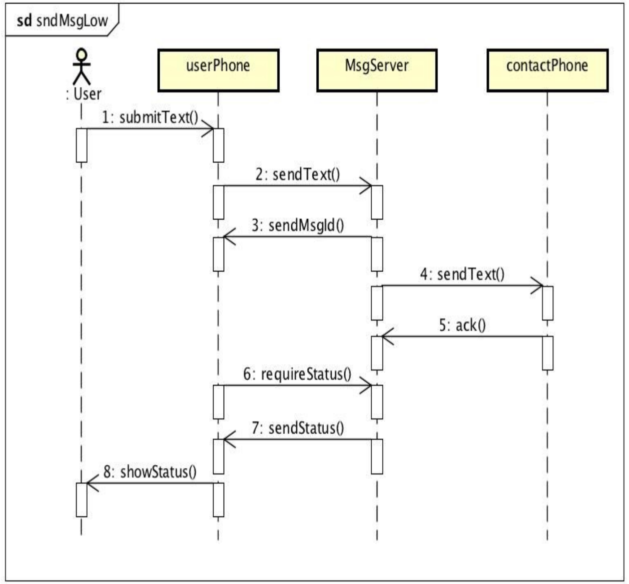


MSDL

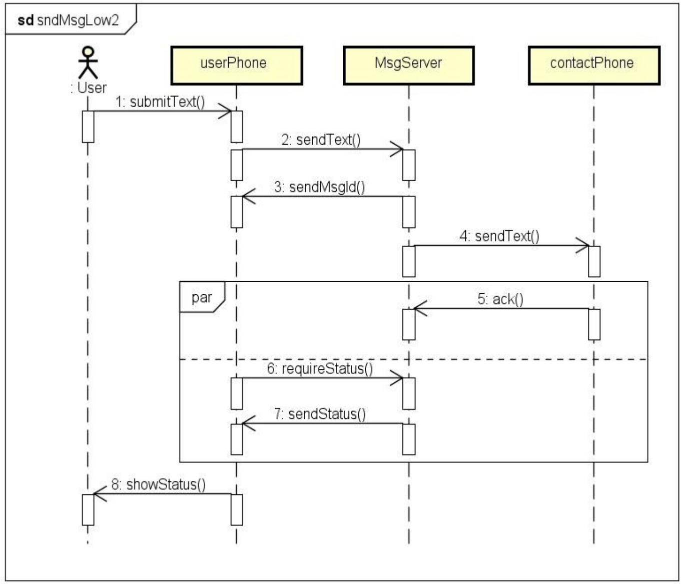
Refinement Notions

- Weak Increment Refinement - Example

Abstract Model

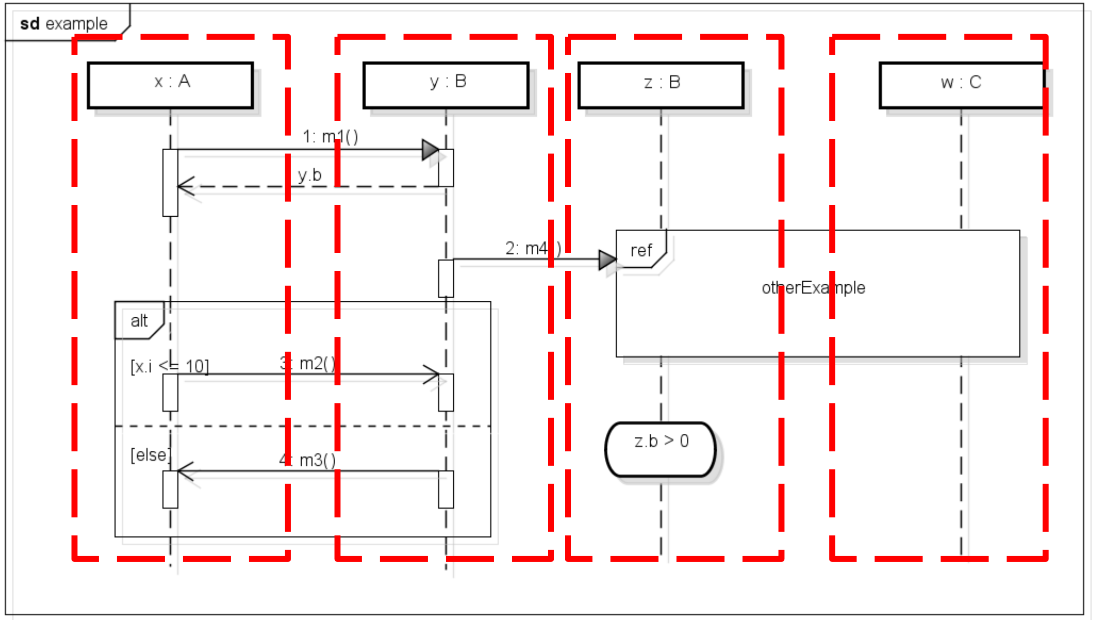


Refined Model



MSDL

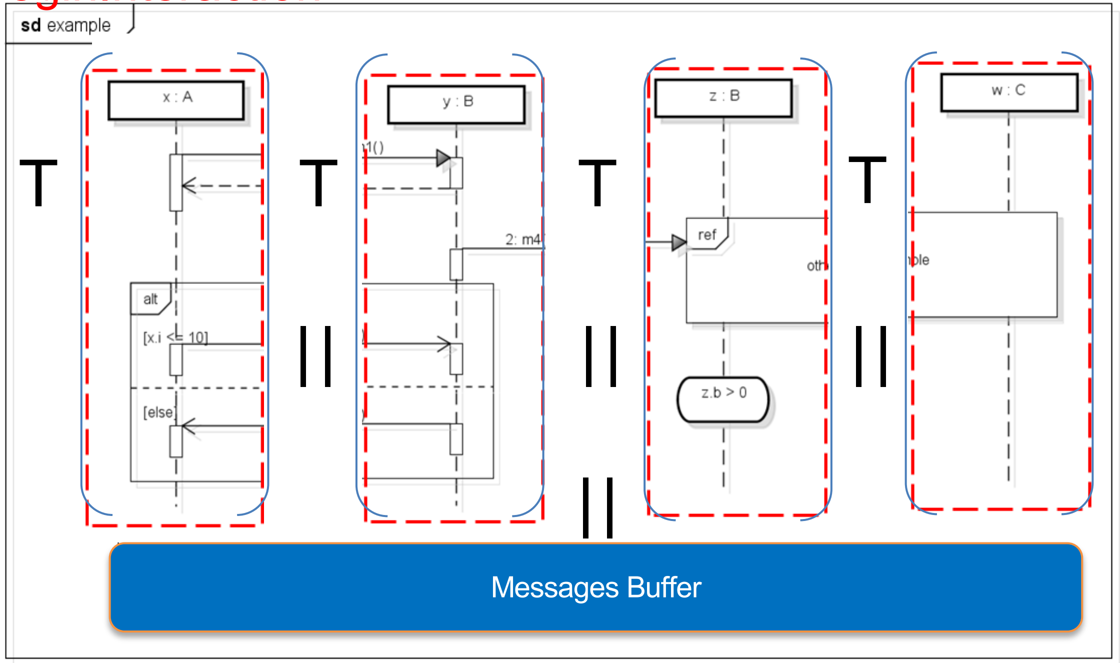
Overview on the CSP sequence diagram semantics



MSDL

Overview on the CSP sequence diagram semantics

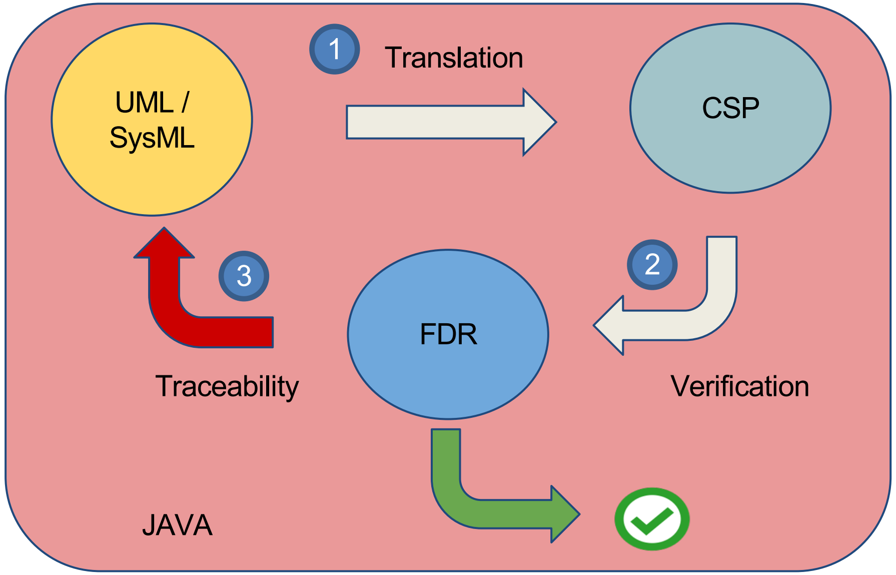
beginInteraction →



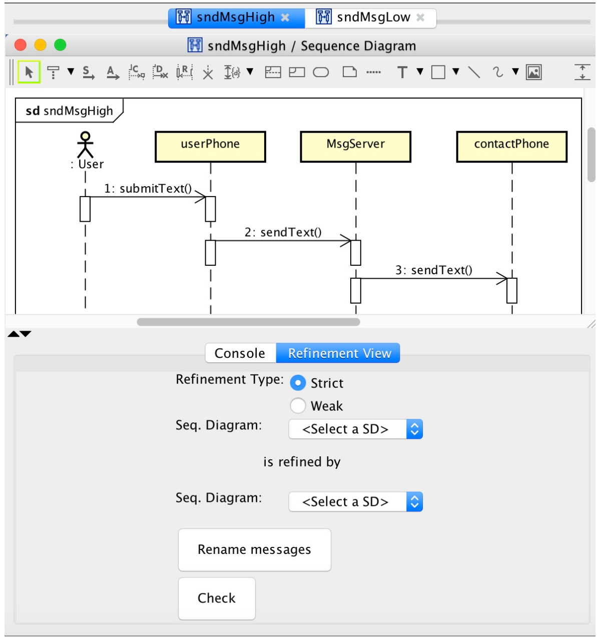
endInteraction → SKIP

MSDL

Tool Support



Tool Support



Refinement Type: [✓ Strict] Weak

Seq. Diagram: <Select a SD>

is refined by

Seq. Diagram: <Select a SD>

- Plug-in of the Astah Modeling Tool
- It requires the FDR3 tool

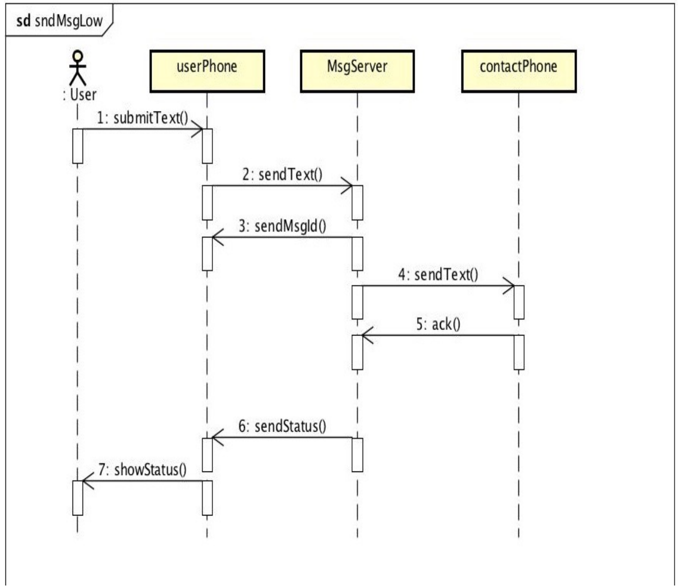
MSDL

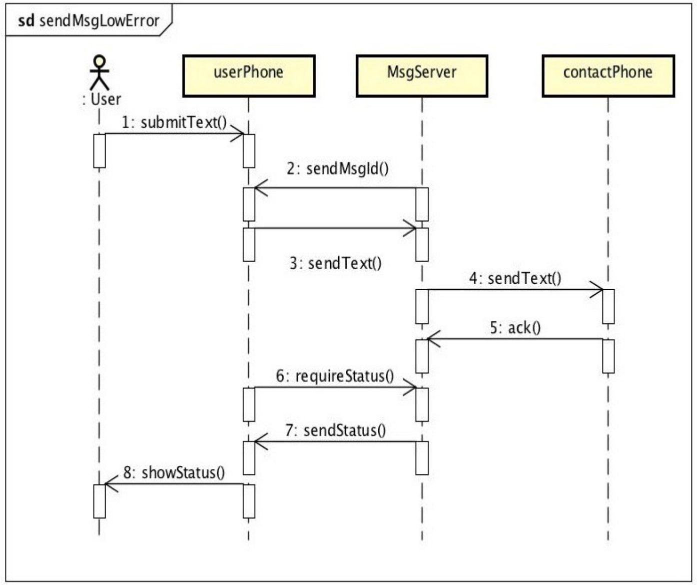
Example

- Strict Increment Refinement

Abstract Model

Refined Model



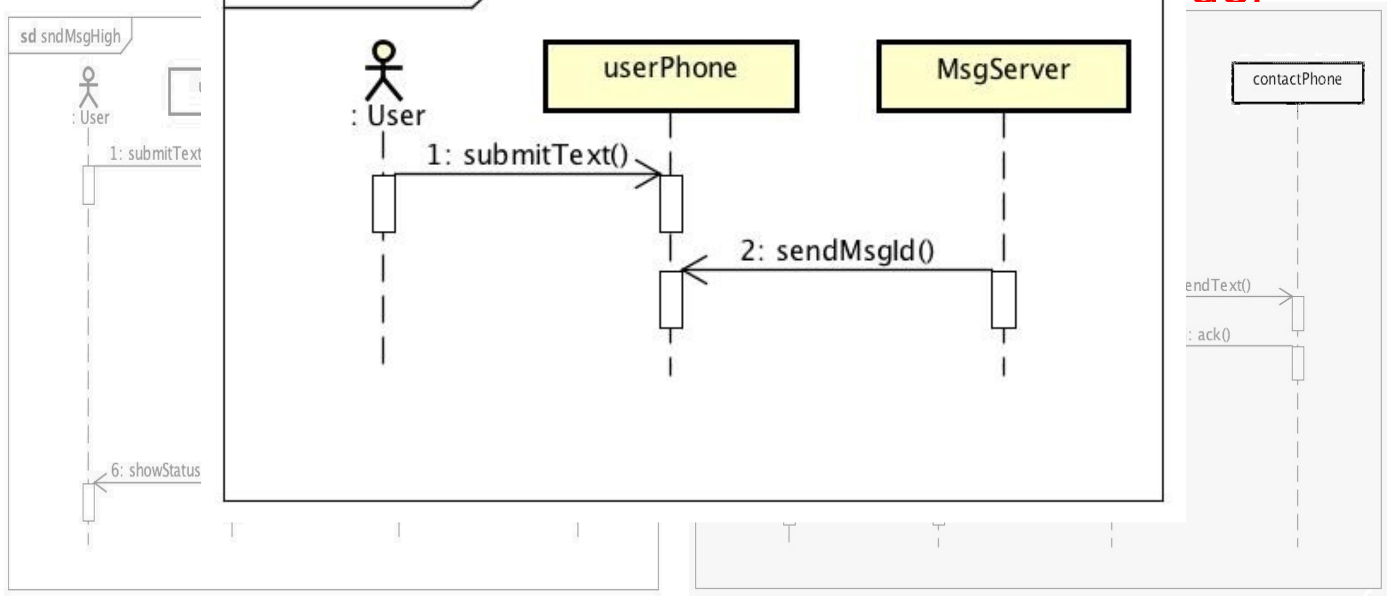


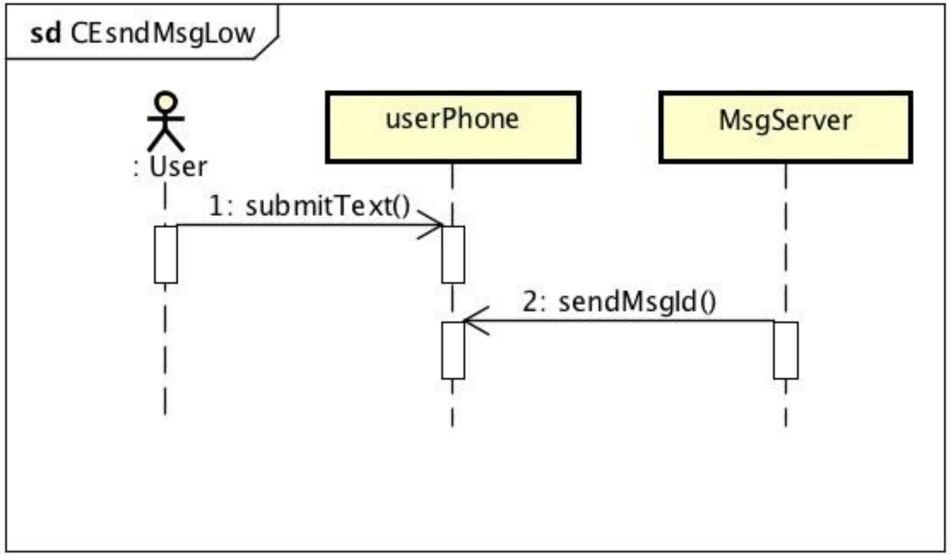
Example

- Strict Increment Refinement

A

del



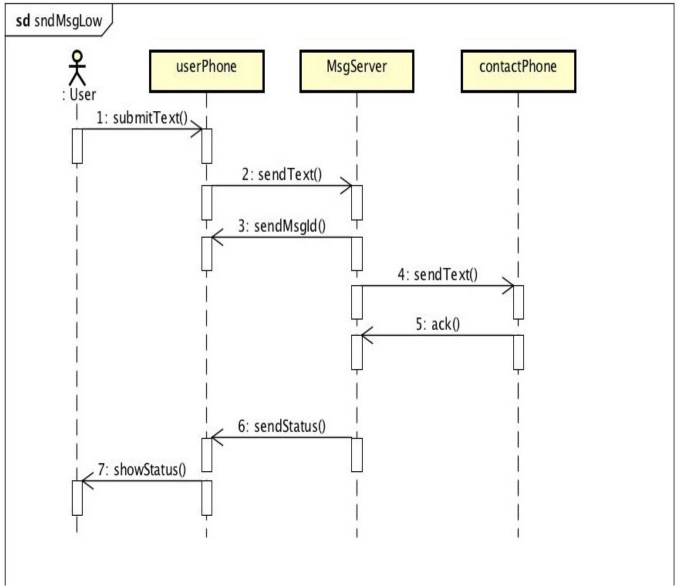


MSDL

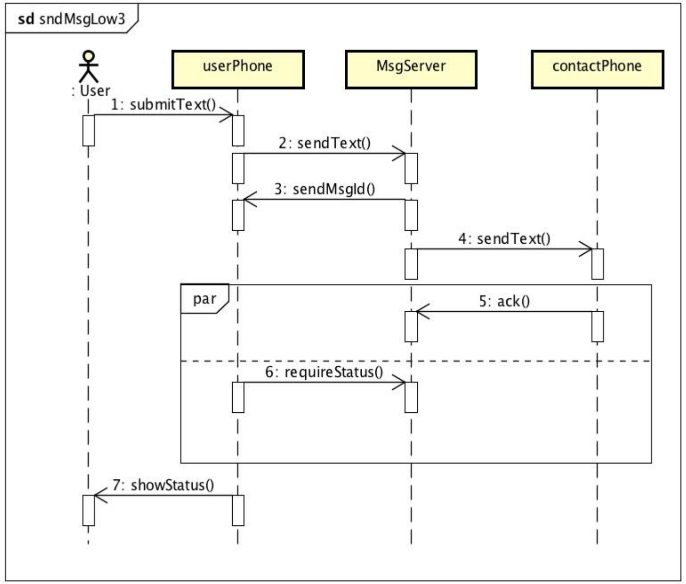
Example

- Weak Increment Refinement

Abstract Model



Refined Model



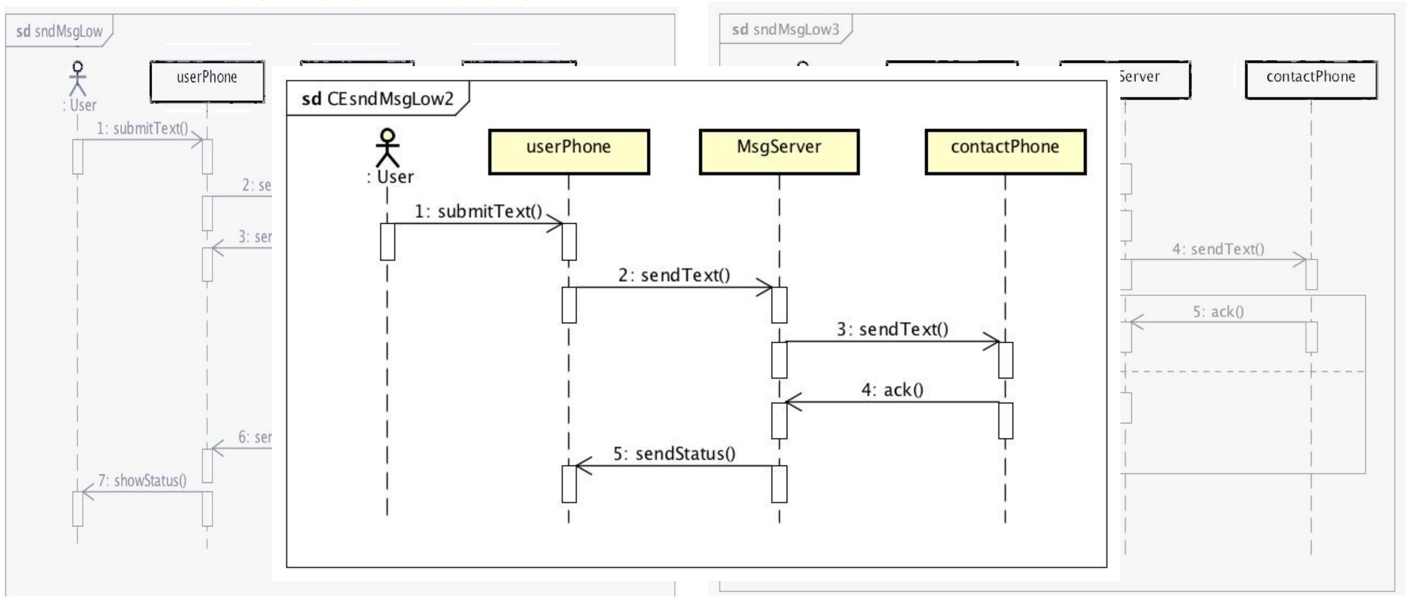
MSDL

Example

- Weak Increment Refinement

Abstract Model

Refined Model



MSDL

Application 2

Verifying Deadlock and Nondeterminism in Activity Diagrams

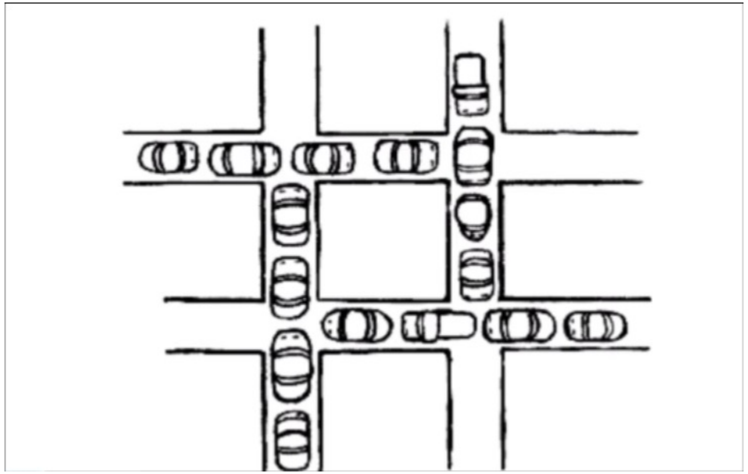
https://ieeexplore.ieee.org/document/8904590

https://www.sciencedirect.com/science/article/abs/pii/S0167642320301064

MSDL

Current concerns

Deadlock



the system can’t make any progress, because each process is waiting for communication with others.

It can happen for instance due to competition for resources

remains one of the most common and feared issues in concurrent systems.

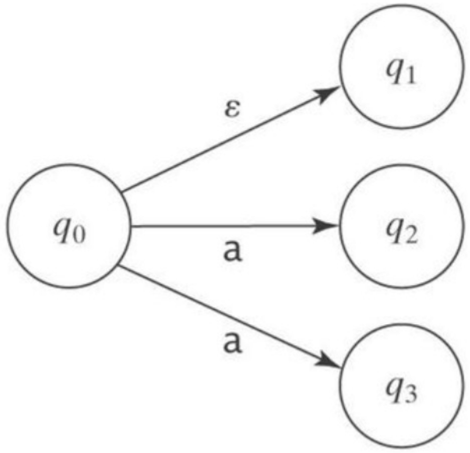
Current concerns

even for the same input, the system can exhibit different behaviors on different runs

Unpredictability

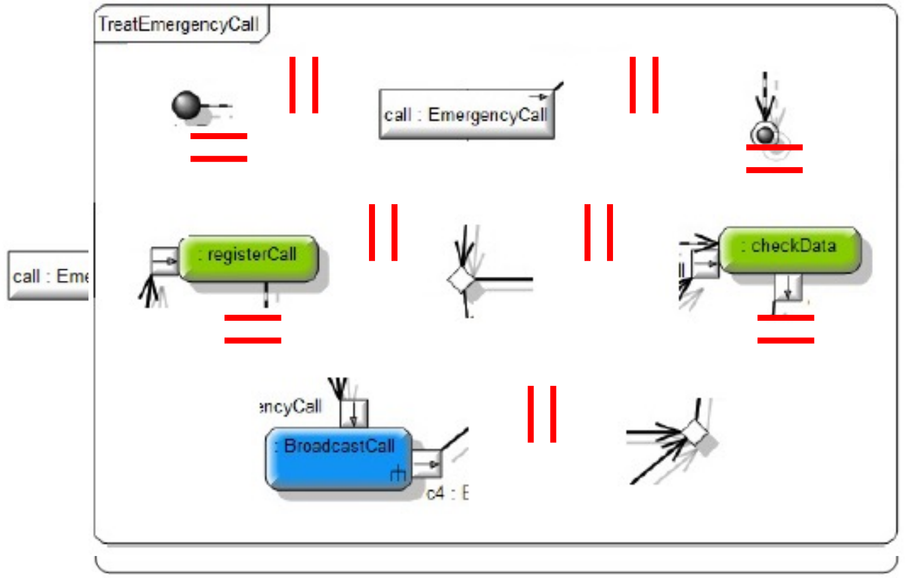
Cannot be tackled with standard verification approaches like testing

Nondeterminism



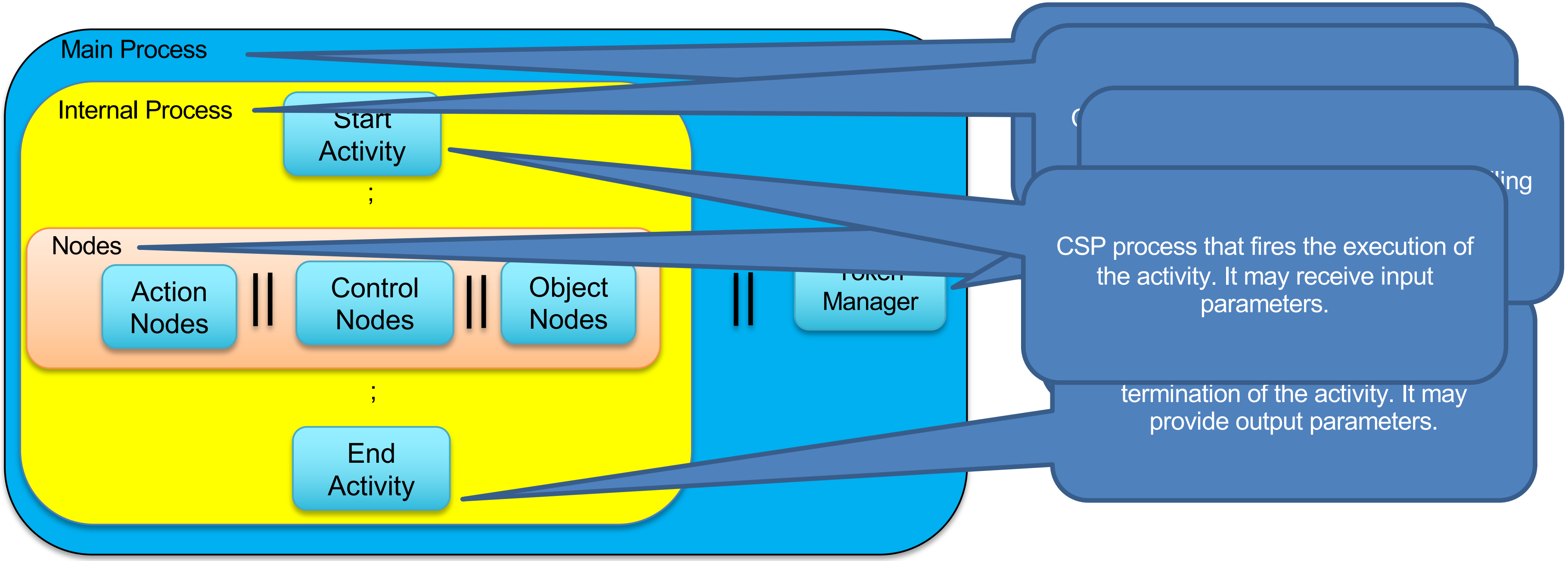
MSDL

Overview on the CSP activity diagram semantics



MSDL

Overview on the CSP activity diagram semantics


Main Process
Internal Process
Start Activity
;
Nodes
Action Nodes
||
Control Nodes
||
Object Nodes
;
End Activity
||
Token Manager
C
ling
CSP process that fires the execution of the activity. It may receive input parameters.
termination of the activity. It may provide output parameters.

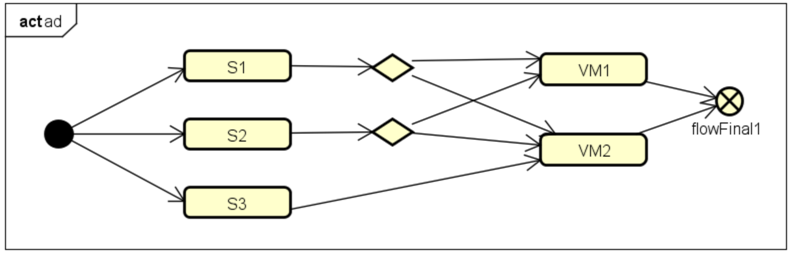
MSDL

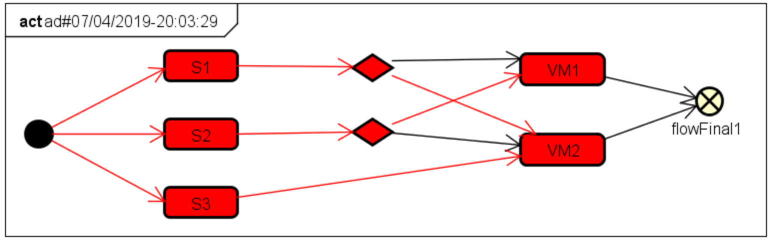
Traceability

- Mechanism to show the results in terms of UML/SysML
- Avoid any contact with formalism (CSP)
- Events need to allow traceability
  - Unique Identifiers
  - Table describing mappings
- When a counterexample is returned be FDR:
  - Create a copy of the activity
  - Highlight the path to the problem traversing the trace given by the counterexample

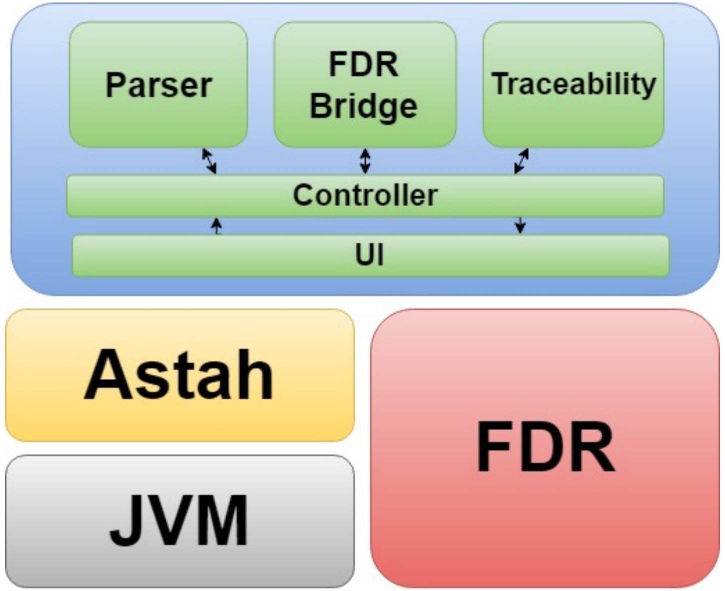
MSDL

Traceability





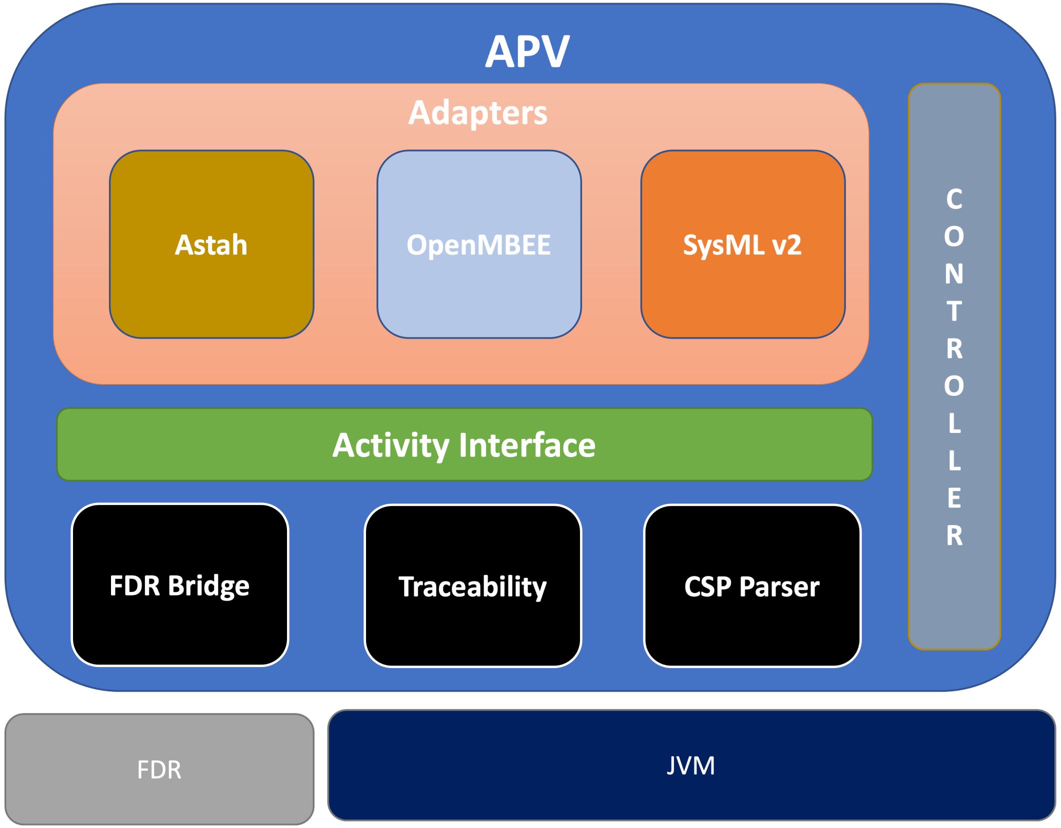
Activity Property Verifier (APV) Architecture



MSDL

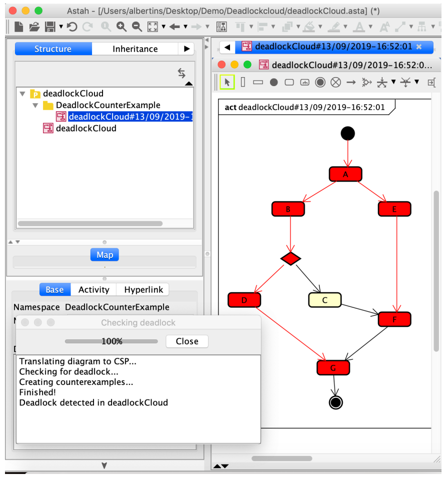
APV Architecture

- - Adapters to support different environments/tools
- - Common Activity Interface isolate the formal semantics (CSP Parser)
- - Traceability module maps counterexample trace to activity identifiers
- - FDR Bridge manages communication with FDR



MSDL

Tool demonstration



Astah - [/Users/albertins/Desktop/Demo/Deadlockcloud/deadlockCloud.asta] (*)
Structure
Inheritance
deadlockCloud
DeadlockCounterExample
deadlockCloud#13/09/2019-
deadlockCloud
Map
Base
Activity
Hyperlink
Namespace DeadlockCounterExample
deadlockCloud#13/09/2019-16:52:01
act deadlockCloud#13/09/2019-16:52:01
A
B
E
C
D
F
G

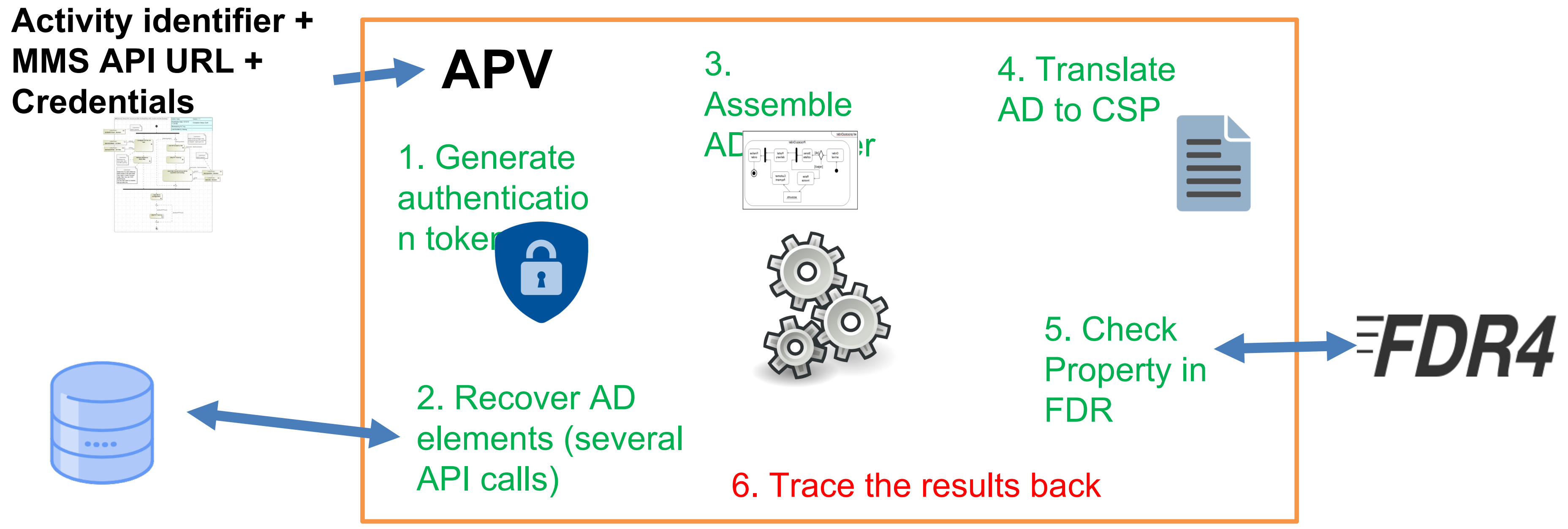
Checking deadlock

100% Close

```
Translating diagram to CSP...
Checking for deadlock...
Creating counterexamples...
Finished!
Deadlock detected in deadlockCloud
```

MSDL

OpenMBEE Module Overview


Activity identifier + MMS API URL + Credentials
APV
1. Generate authenticatio n toke
2. Recover AD elements (several API calls)
3. Assemble AD
er
4. Translate AD to CSP
5. Check Property in FDR
FDR4
6. Trace the results back

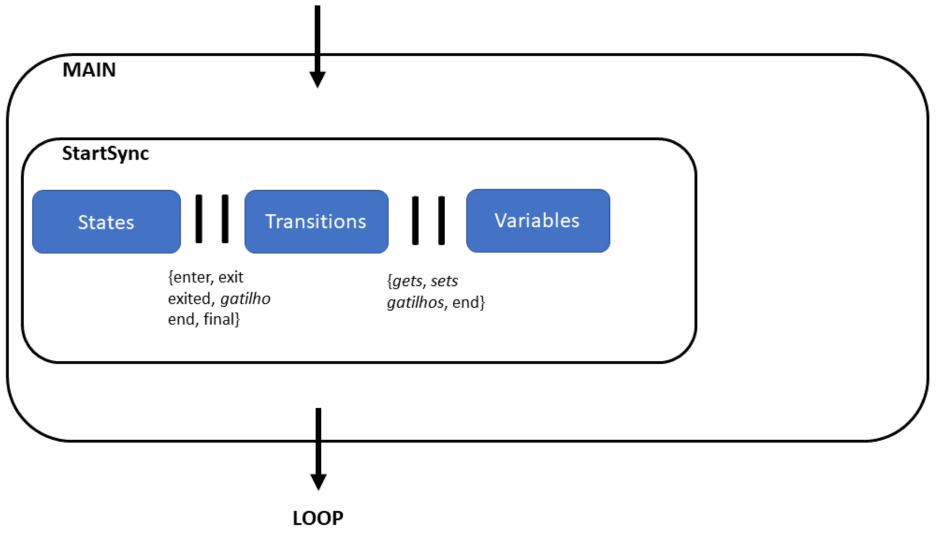
MSDL

Application 3

Verifying Deadlock and Nondeterminism in State Machines

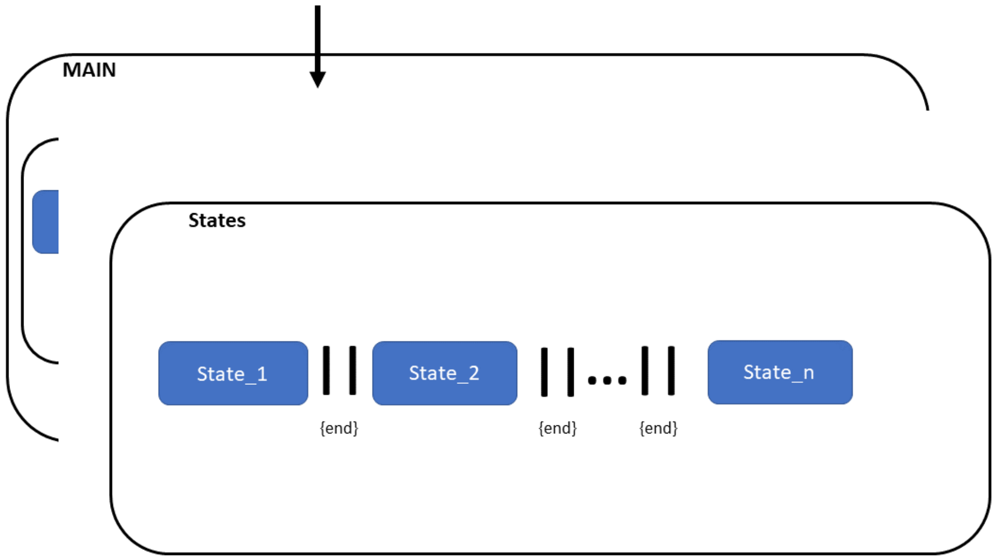
MSDL

Overview on the CSP state machine diagram semantics


MAIN
StartSync
States
Transitions
Variables
{enter, exit exited, *gatilho* end, final}
{*gets*, *sets* *gatilhos*, end}
LOOP

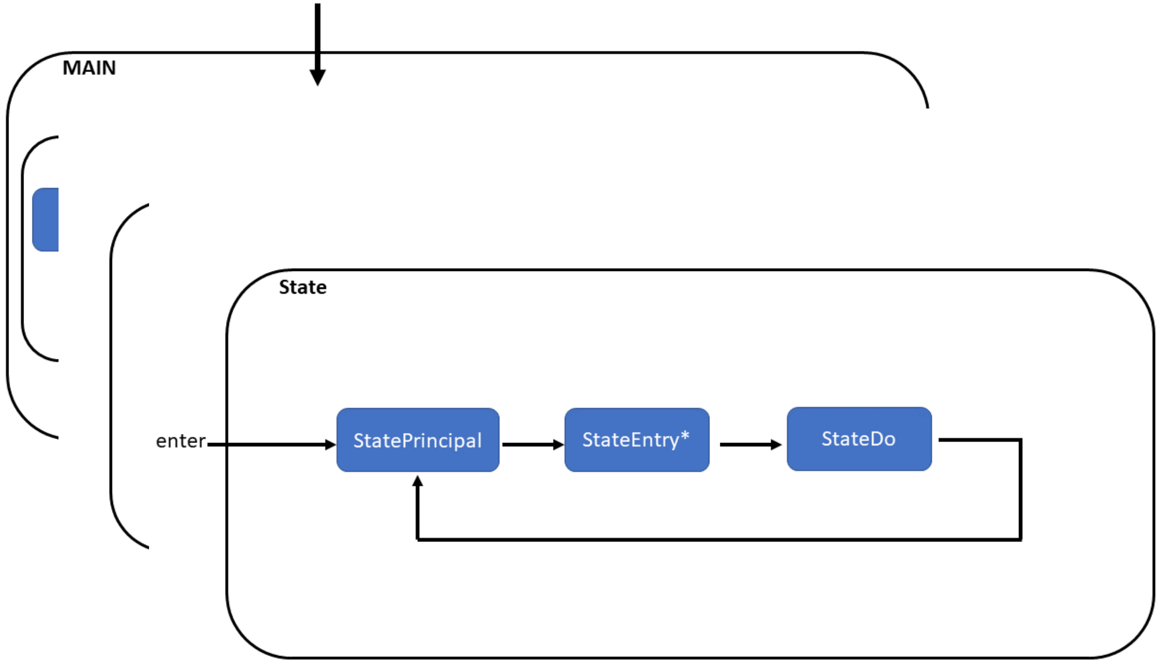
MSDL

Overview on the CSP state machine diagram semantics

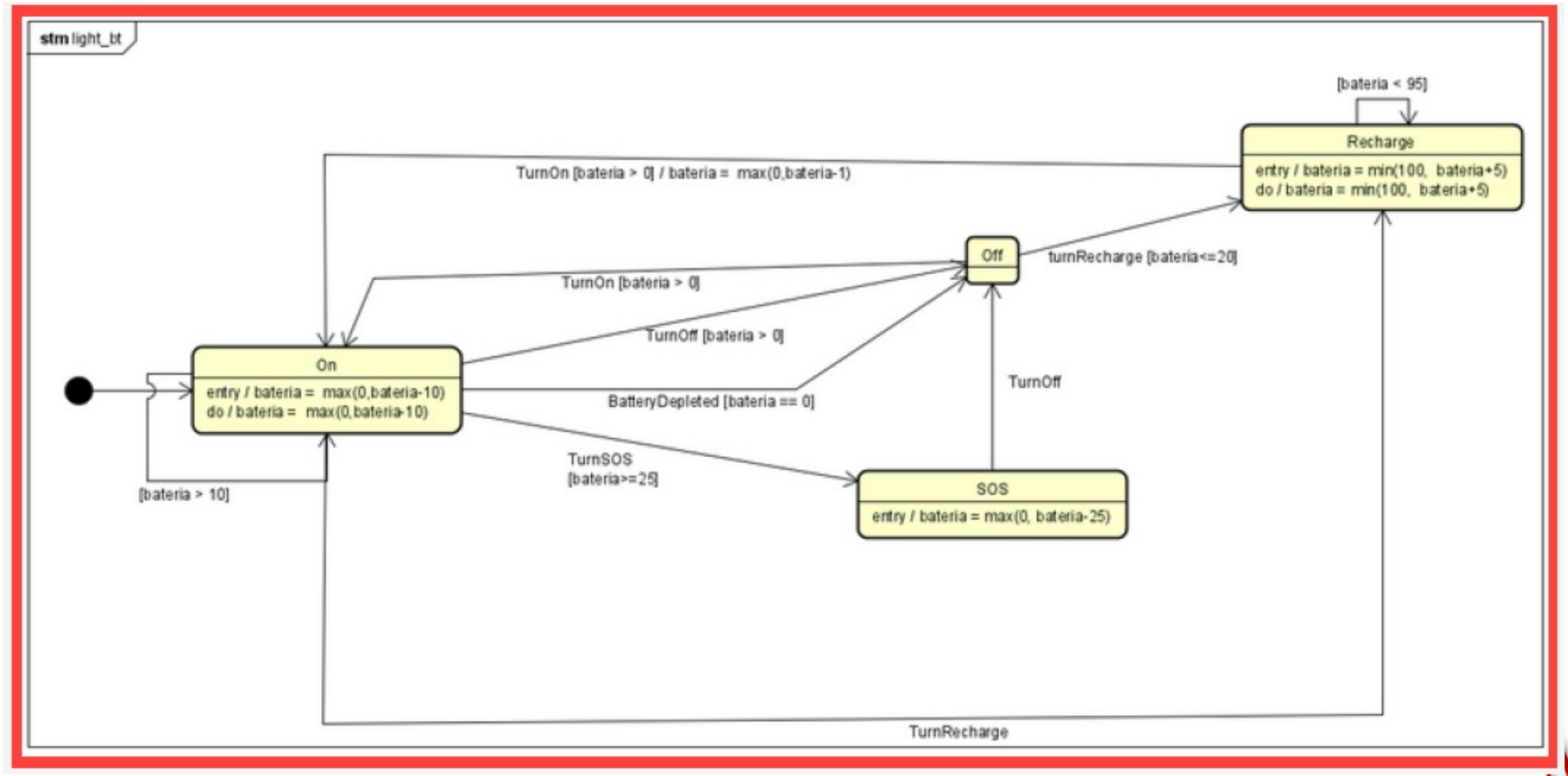


MSDL

Overview on the CSP state machine diagram semantics

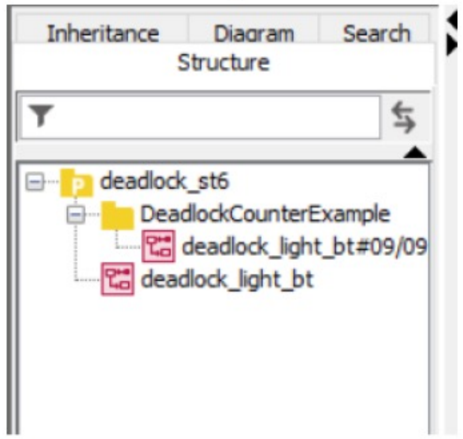


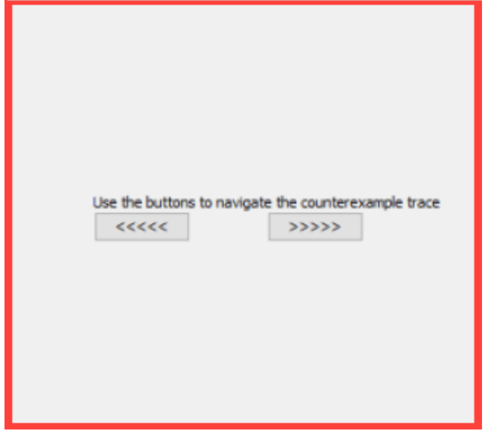
Example

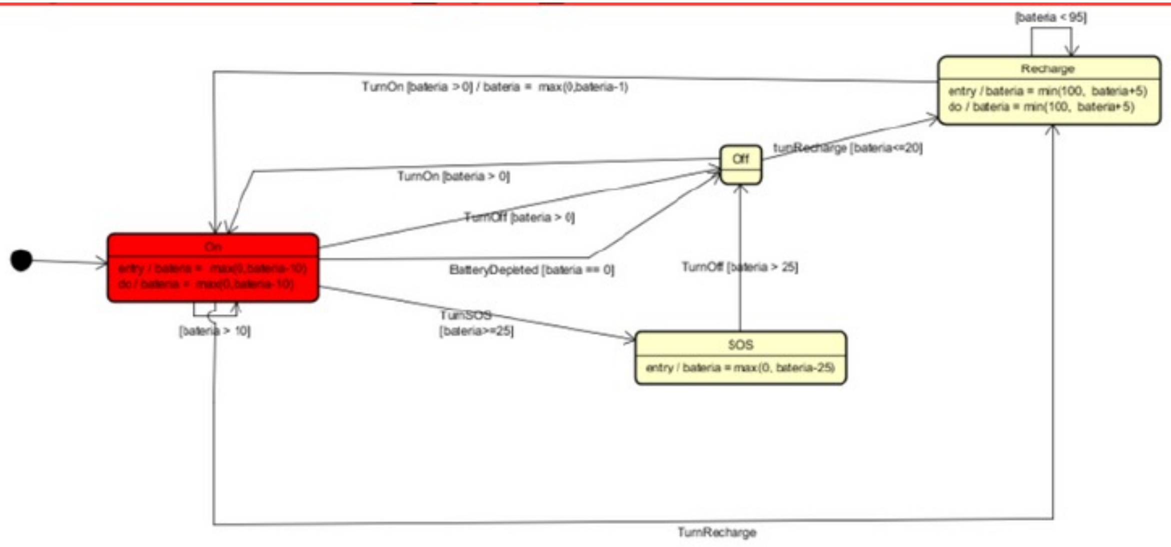

stm light_bt
[bateria < 95]
Recharge
entry / bateria = min(100, bateria+5)
do / bateria = min(100, bateria+5)
TurnOn [bateria > 0] / bateria = max(0,bateria-1)
Off
turnRecharge [bateria<=20]
TurnOn [bateria > 0]
TurnOff [bateria > 0]
On
entry / bateria = max(0,bateria-10)
do / bateria = max(0,bateria-10)
BatteryDepleted [bateria == 0]
TurnOff
TurnSOS
[bateria>=25]
[bateria > 10]
SOS
entry / bateria = max(0, bateria-25)
TurnRecharge

MSDL

When a counterexample is detected

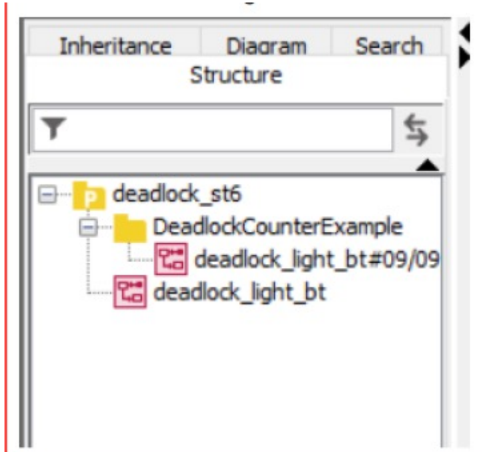


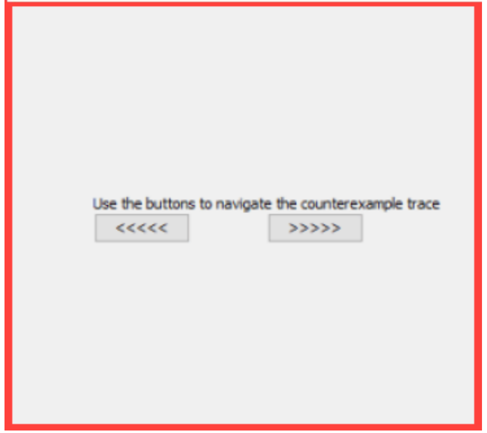

Use the buttons to navigate the counterexample trace

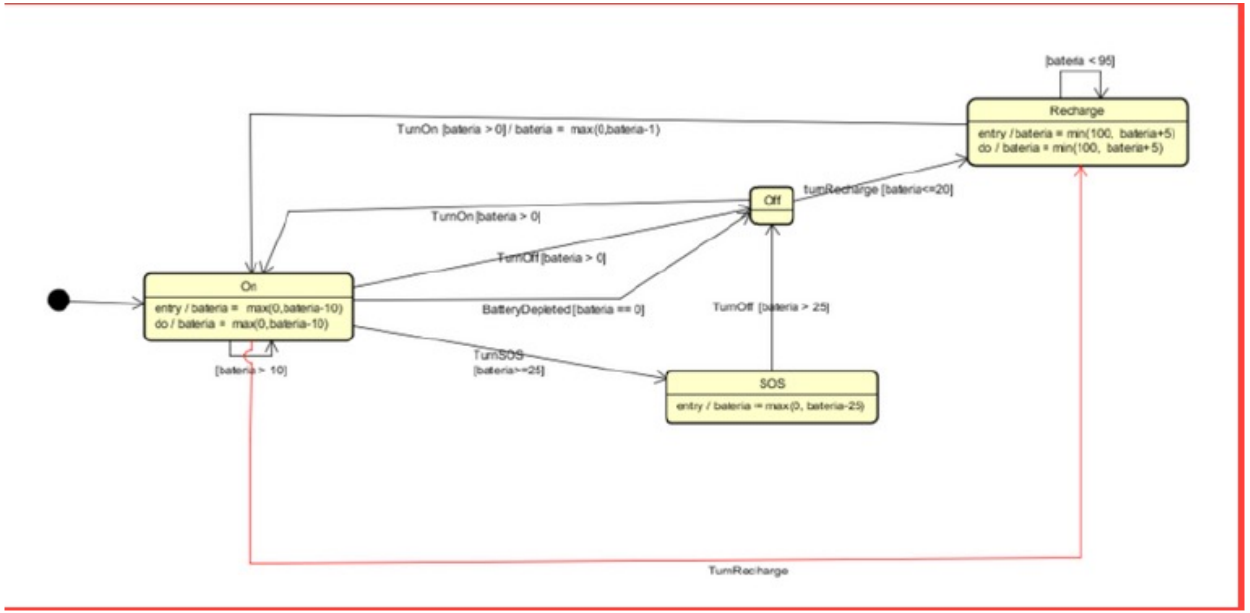

[bateria < 95]
Recharge
entry / bateria = min(100, bateria+5)
do / bateria = min(100, bateria+5)
TurnOn [bateria > 0] / bateria = max(0,bateria-1)
Off
turnRecharge [bateria<=20]
TurnOn [bateria > 0]
TurnOff [bateria > 0]
On
entry / bateria = max(0,bateria-10)
do / bateria = max(0,bateria-10)
BatteryDepleted [bateria == 0]
TurnOff [bateria > 25]
[bateria > 10]
TurnSOS
[bateria>=25]
SOS
entry / bateria = max(0, bateria-25)
TurnRecharge

MSDL

When a counterexample is detected



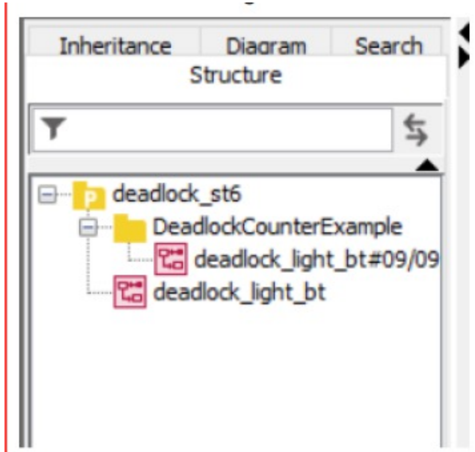

Use the buttons to navigate the counterexample trace

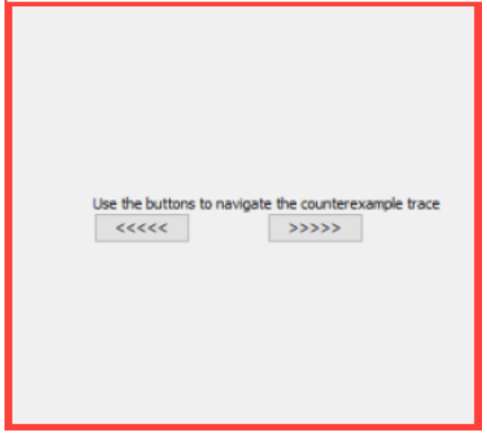


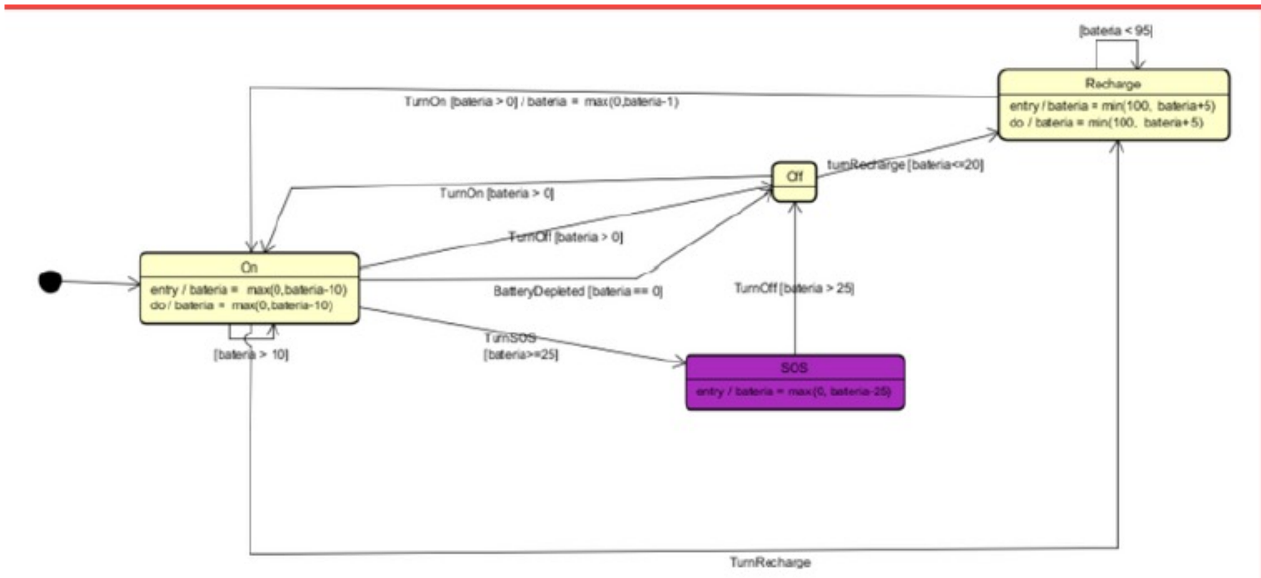
Recharge
entry / bateria = min(100, bateria+5)
do / bateria = min(100, bateria+5)
[bateria < 95]
TurnOn [bateria > 0] / bateria = max(0,bateria-1)
Off
turnRecharge [bateria<=20]
TurnOn [bateria > 0]
TurnOff [bateria > 0]
On
entry / bateria = max(0,bateria-10)
do / bateria = max(0,bateria-10)
BatteryDepleted [bateria == 0]
TurnOff [bateria > 25]
TurnSOS [bateria>=25]
[bateria > 10]
SOS
entry / bateria = max(0, bateria-25)
TurnRecharge

MSDL

When a counterexample is detected




Use the buttons to navigate the counterexample trace


[bateria < 95]
Recharge
entry / bateria = min(100, bateria+5)
do / bateria = min(100, bateria+5)
TurnOn [bateria > 0] / bateria = max(0,bateria-1)
Off
turnRecharge [bateria<=20]
TurnOn [bateria > 0]
TurnOff [bateria > 0]
On
entry / bateria = max(0,bateria-10)
do / bateria = max(0,bateria-10)
BatteryDepleted [bateria == 0]
TurnOff [bateria > 25]
[bateria > 10]
TurnSOS
[bateria>=25]
SOS
entry / bateria = max(0, bateria-25)
TurnRecharge

MSDL

Application 4

Visual Specification of Properties for Robotic Designs

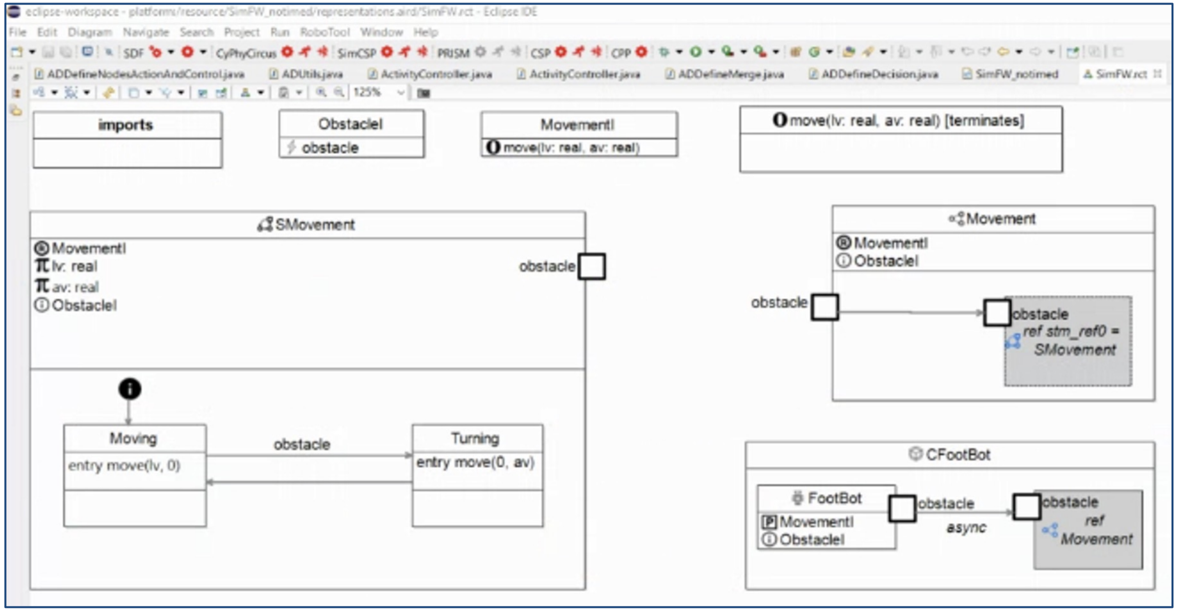
https://link.springer.com/chapter/10.1007/978-3-030-92137-8_3

MSDL

RoboStar Project

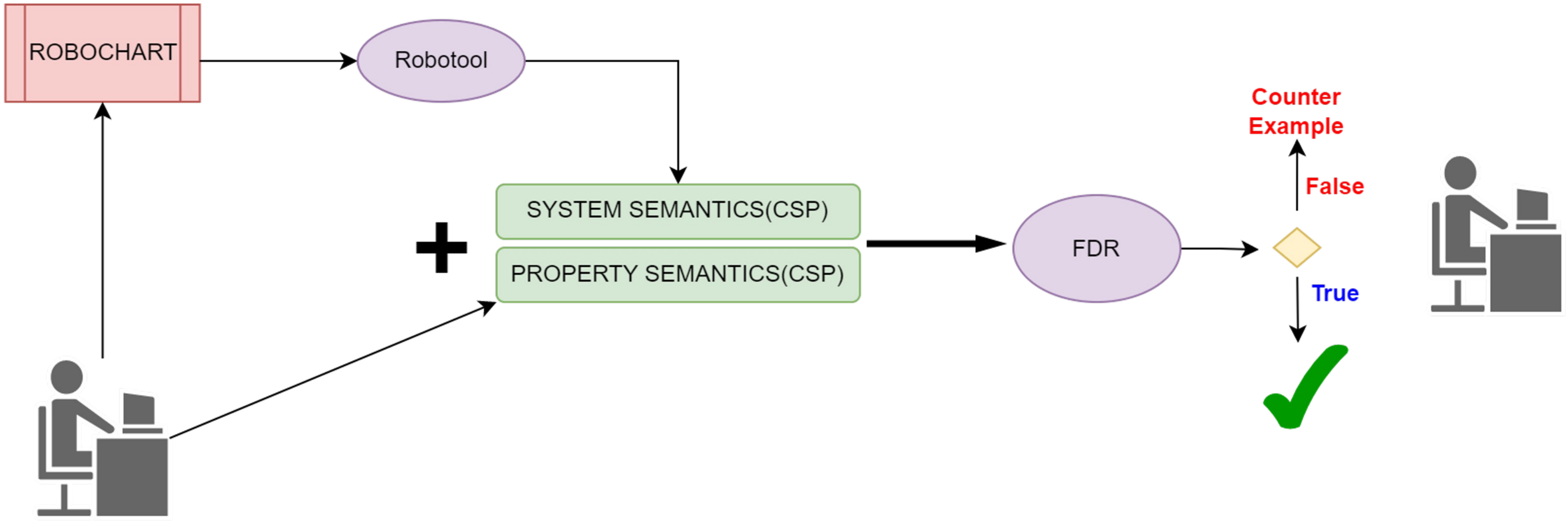
RoboStar*

UNIVERSITY of York

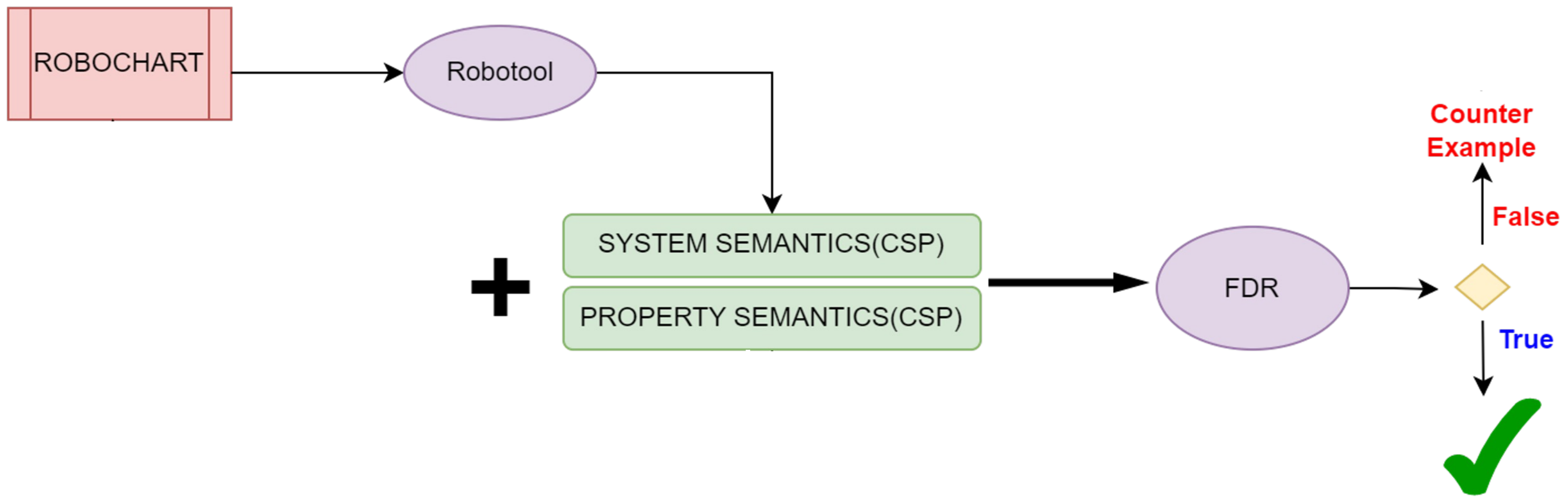


RoboTool

Verifying properties using RoboChart



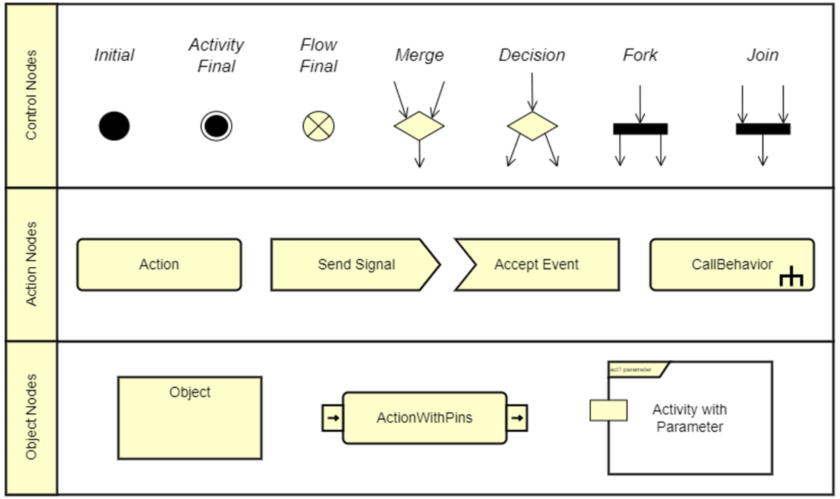
Our approach



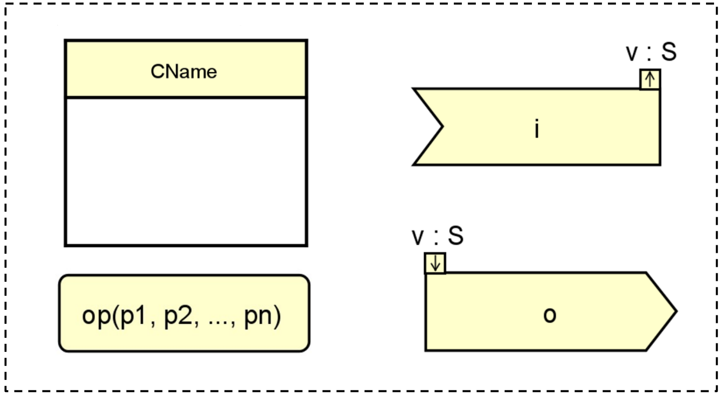
MSDL

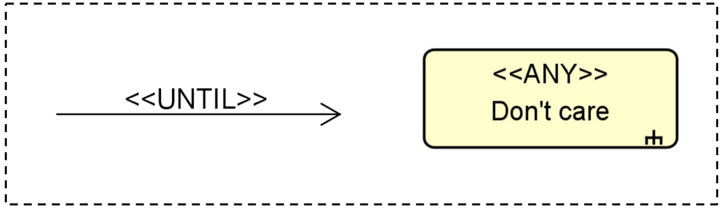
DSL to specify properties based on UML activity diagrams

Activity Nodes



Events and operations

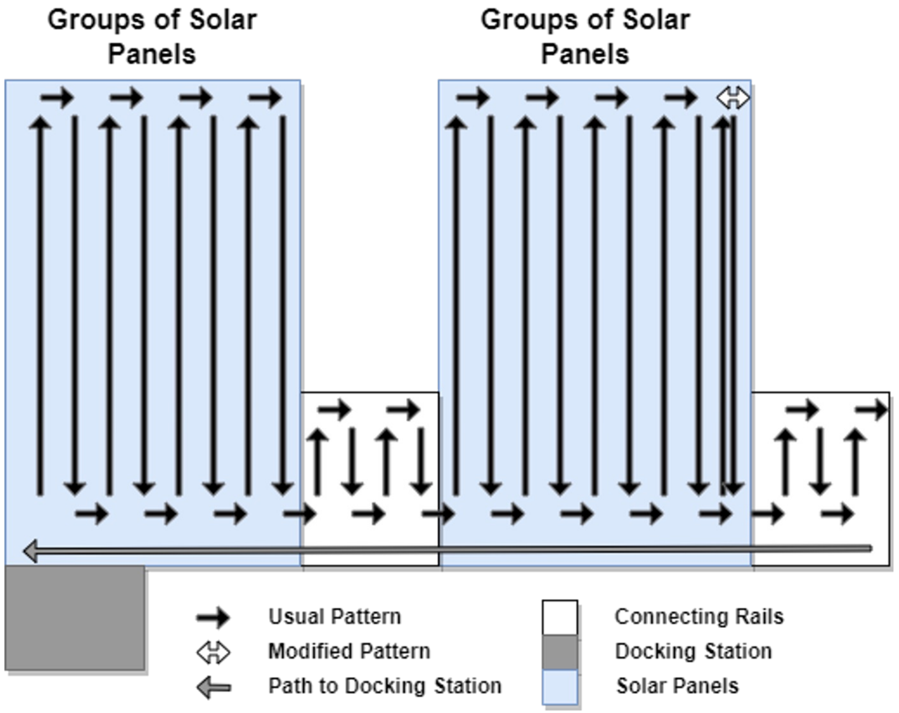


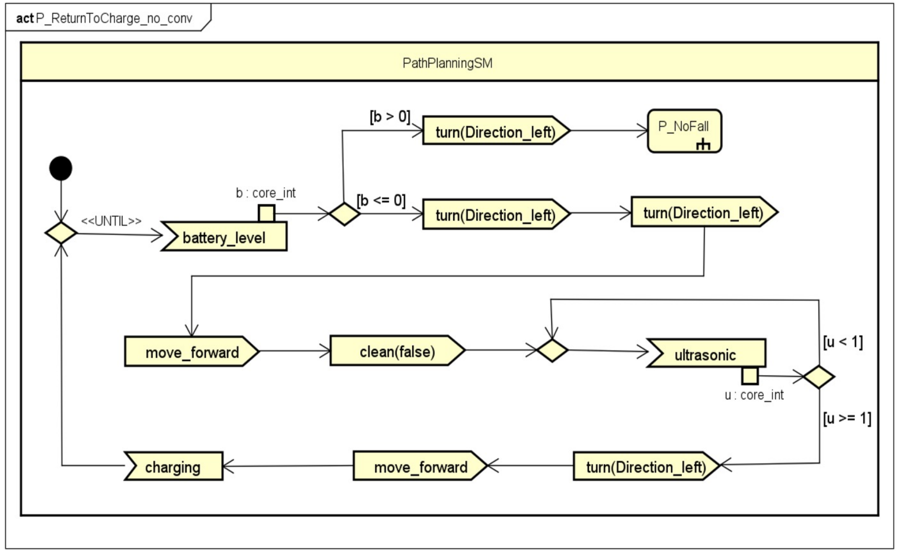


Abstraction patterns

…with a formal semantics defined in CSP

Solar Panel Vacuum Cleaner



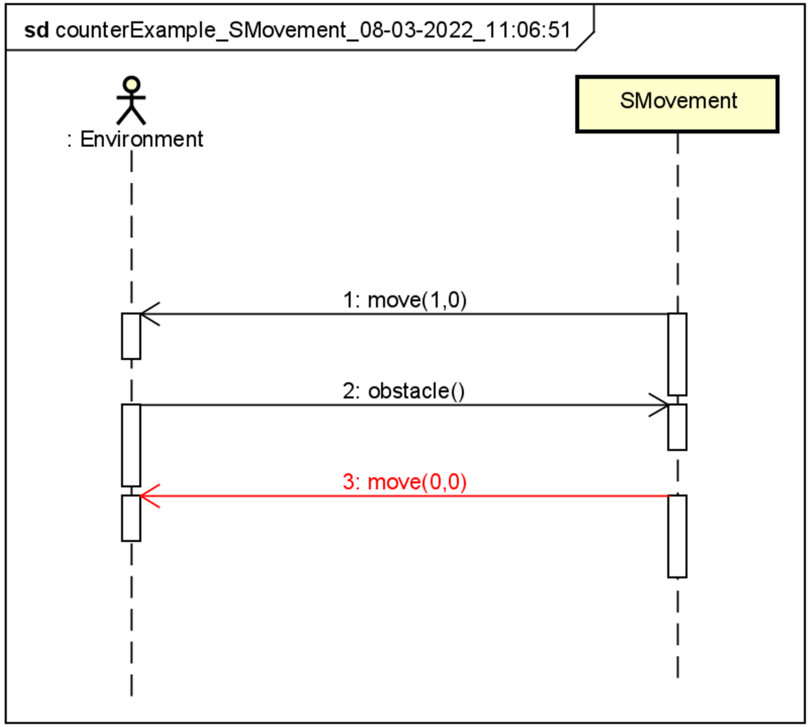


Counterexample as Sequence Diagram

FDR is called in the background

```
Property [T= RoboChart
```

The counterexample is presented as a sequence diagram



Application 5

Safe and constructive design with UML components

https://link.springer.com/chapter/10.1007/978-3-030-03044-5_15

MSDL

Motivation

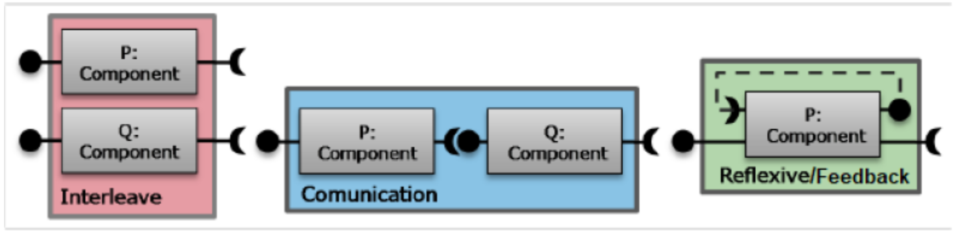
- Component Based Software Development (CBSD):
  - a widely disseminated paradigm
  - focus on component design and integration
  - modelling and design in UML or other graphical notations

- Existing approaches to verification:
  - typically uses formal notation
  - no traceability to the modelling notation
  - perform a posteriori verification: often costly and infeasible

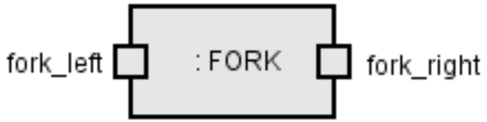
BRIC in a nutshell

Ctr = <B,R,I,C>

B : Behaviour (CSP Process)
R: Channel <-> Interface (relationship)
I: Set of interfaces (datatype)
C: Communication channels (channels)



{ picksup_I,picksup_O, putsdown_I,putsdown_O }

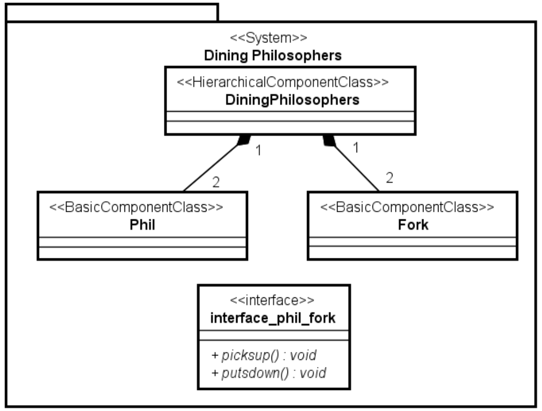


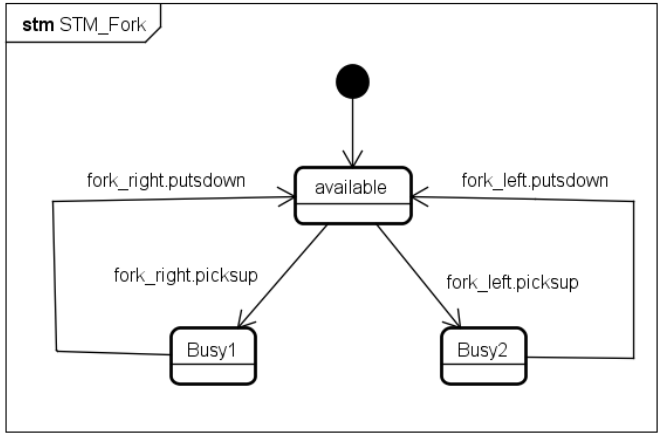
{ picksup_I,picksup_O, putsdown_I,putsdown_O }

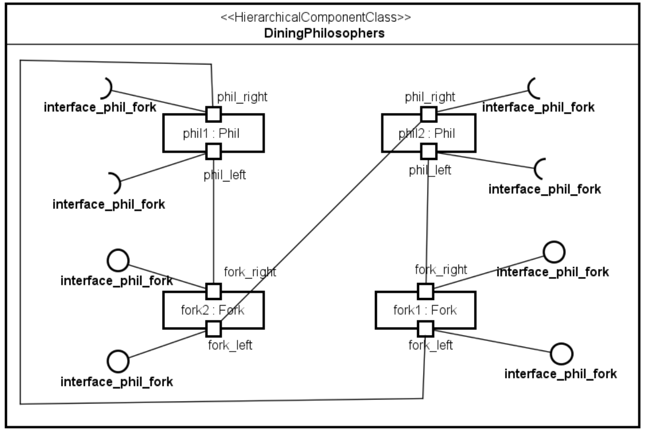
https://repositorio.ufpe.br/bitstream/123456789/2073/1/arquivo6881_1.pdf

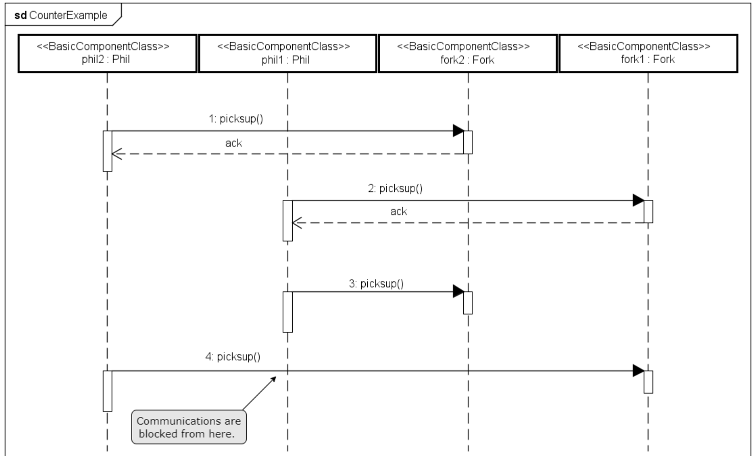
MSDL

Leveraging BRIC to UML

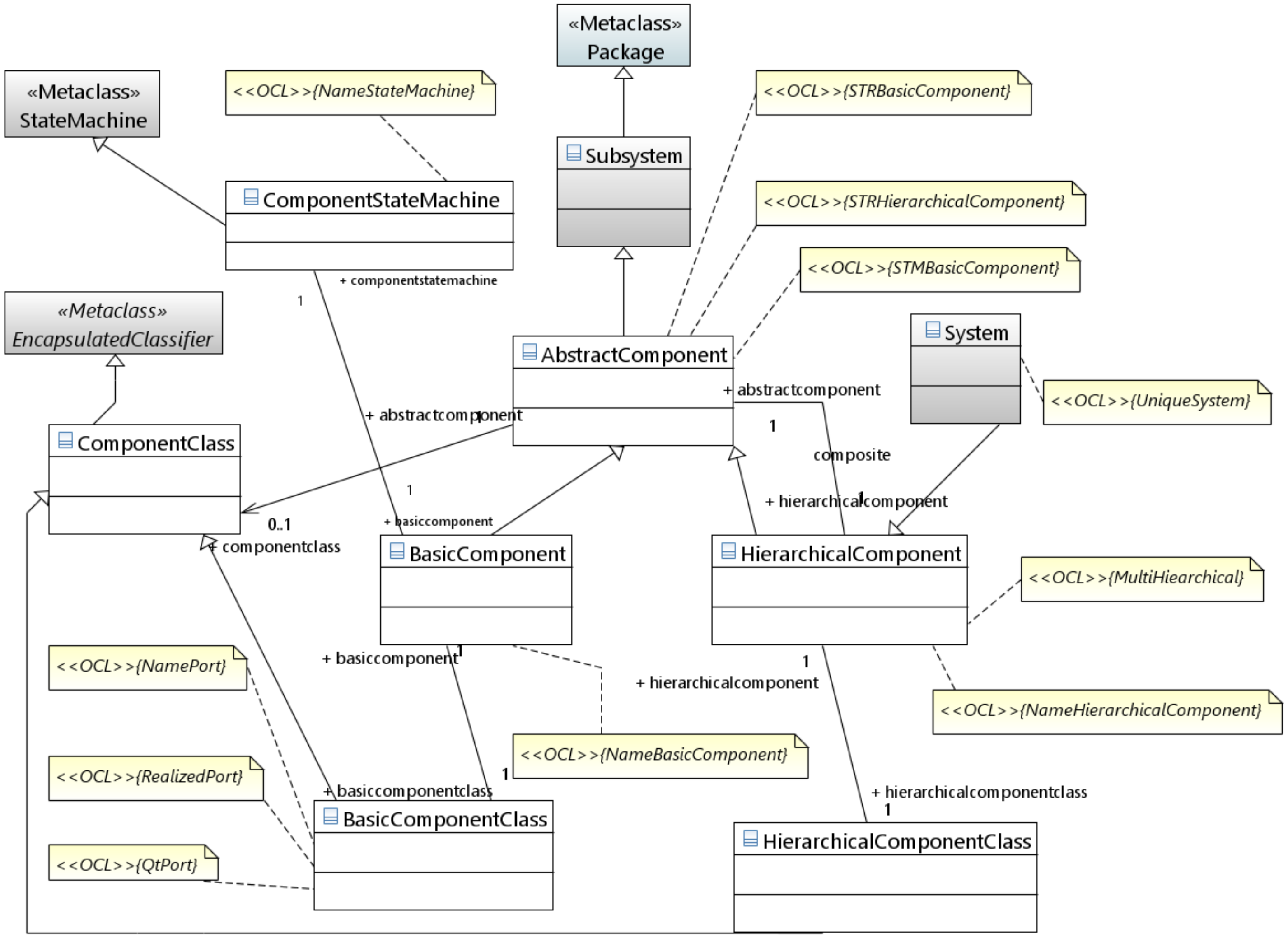






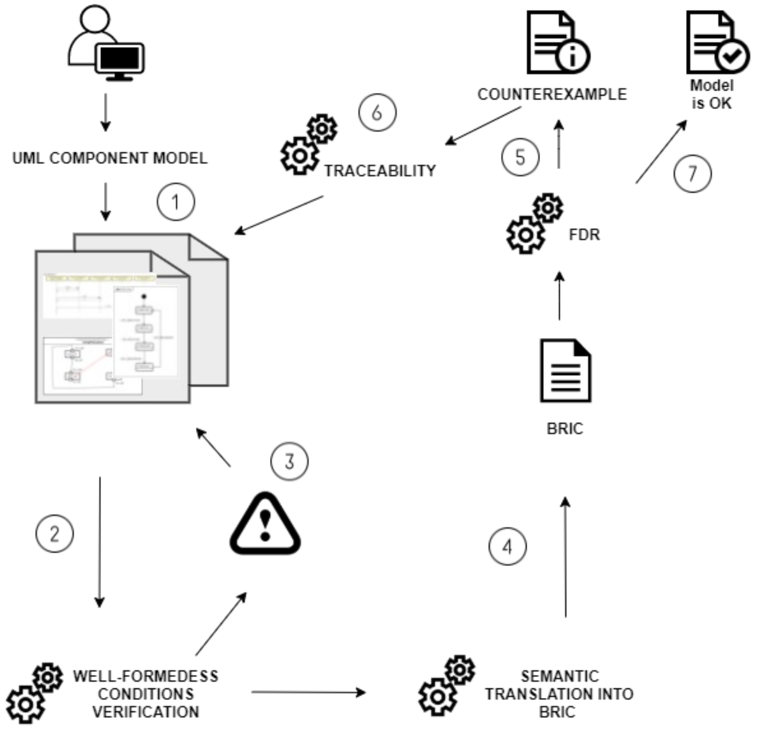


sd CounterExample
<<BasicComponentClass>> phil2 : Phil
<<BasicComponentClass>> phil1 : Phil
<<BasicComponentClass>> fork2 : Fork
<<BasicComponentClass>> fork1 : Fork
1: picksup()
ack
2: picksup()
ack
3: picksup()
4: picksup()
Communications are blocked from here.


«Metaclass» Package
«Metaclass» StateMachine
<<OCL>>{NameStateMachine}
Subsystem
<<OCL>>{STRBasicComponent}
<<OCL>>{STRHierarchicalComponent}
<<OCL>>{STMBasicComponent}
ComponentStateMachine
+ componentstatemachine
1
«Metaclass» EncapsulatedClassifier
AbstractComponent
+ abstractcomponent
1
composite
System
<<OCL>>{UniqueSystem}
+ abstractcomponent
1
ComponentClass
0..1
+ componentclass
+ basiccomponent
BasicComponent
+ hierarchicalcomponent
1
HierarchicalComponent
<<OCL>>{MultiHiearchical}
<<OCL>>{NamePort}
+ basiccomponent
1
1
+ hierarchicalcomponent
<<OCL>>{NameHierarchicalComponent}
<<OCL>>{NameBasicComponent}
<<OCL>>{RealizedPort}
1
+ basiccomponentclass
+ hierarchicalcomponentclass
1
BasicComponentClass
HierarchicalComponentClass
<<OCL>>{QtPort}

MSDL

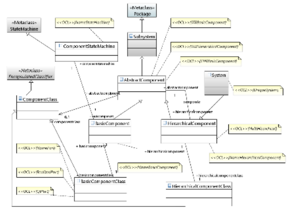
Overview



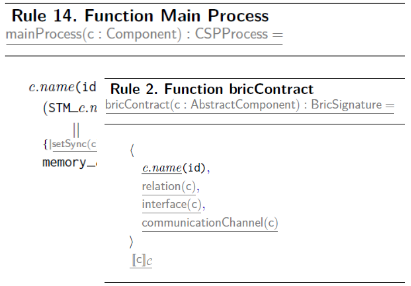
MSDL

Contributions

UML component Model



Formal Semantics



Rule 14. Function Main Process

mainProcess(c : Component) : CSPProcess =

$c.name$(id

(STM_$c.n$

||

{|setSync(c

memory_

Rule 2. Function bricContract

bricContract(c : AbstractComponent) : BricSignature =

⟨

$c.name$(id),

relation(c),

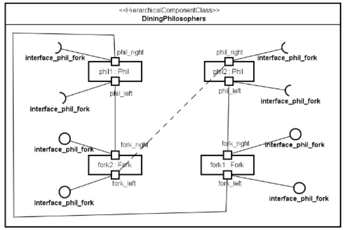
interface(c),

communicationChannel(c)

⟩

[c]c

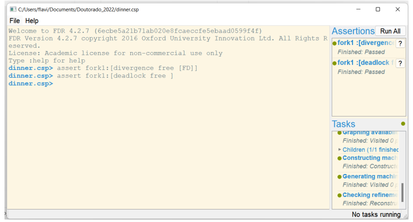
Deadlock Analysis



Well-formedness conditions

```
context BasicComponentClass
inv gtPortBC :
self. ownedPort->size ( )>=1
```

Verifications



C:/Users/flavi/Documents/Doutorado_2022/dinner.csp

```
Welcome to FDR 4.2.7 (6ecbe5a21b71ab020e8fcaeccfe5ebaad0599f4f)
FDR Version 4.2.7 copyright 2016 Oxford University Innovation Ltd. All Rights R
eserved.
License: Academic license for non-commercial use only
Type :help for help
dinner.csp> assert fork1:[divergence free [FD]]
dinner.csp> assert fork1:[deadlock free ]
dinner.csp>
```

Assertions

fork1 :[divergence
Finished: Passed

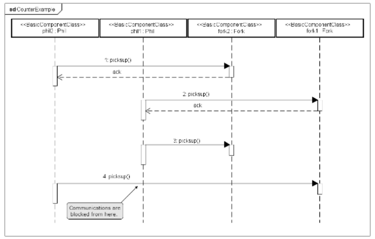
fork1 :[deadlock f
Finished: Passed

Tasks

Graphing availabi
Finished: Visited 0
Children (1/1 finished
Constructing mach
Finished: Construct
Generating machin
Finished: Visited 0
Checking refineme
Finished: Reconstru

No tasks running

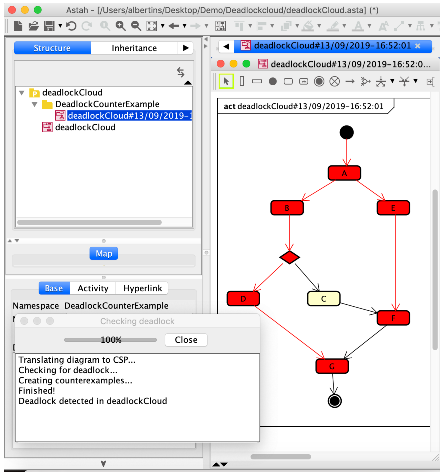
Traceability



sd CounterExample
<<BasicComponentClass>> phil2 : Phil
<<BasicComponentClass>> phil1 : Phil
<<BasicComponentClass>> fork2 : Fork
<<BasicComponentClass>> fork1 : Fork
1: pickup()
ack
2: pickup()
ack
3: pickup()
4: pickup()
Communications are blocked from here.

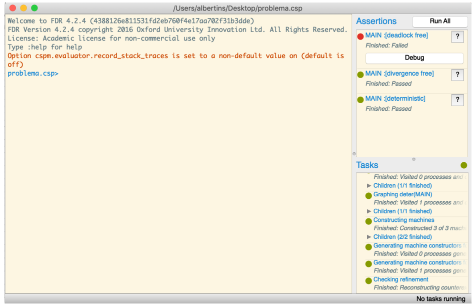
MSDL

Conclusions


Astah - [/Users/albertins/Desktop/Demo/Deadlockcloud/deadlockCloud.asta] (*)
Structure
Inheritance
deadlockCloud
DeadlockCounterExample
deadlockCloud#13/09/2019-1
deadlockCloud
Map
Base
Activity
Hyperlink
deadlockCloud#13/09/2019-16:52:01
act deadlockCloud#13/09/2019-16:52:01
A
B
E
D
C
F
G
Checking deadlock
100%
Close

Namespace DeadlockCounterExample

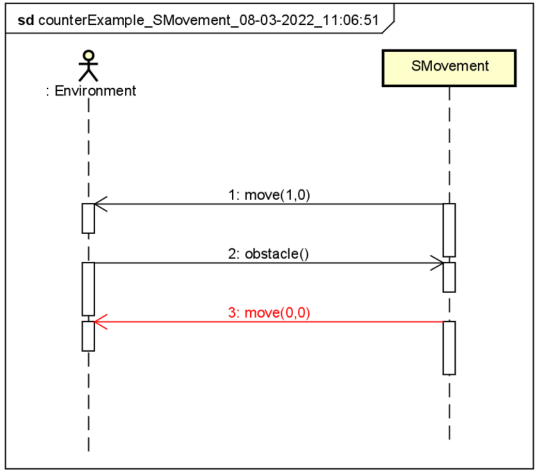
```
Translating diagram to CSP...
Checking for deadlock...
Creating counterexamples...
Finished!
Deadlock detected in deadlockCloud
```

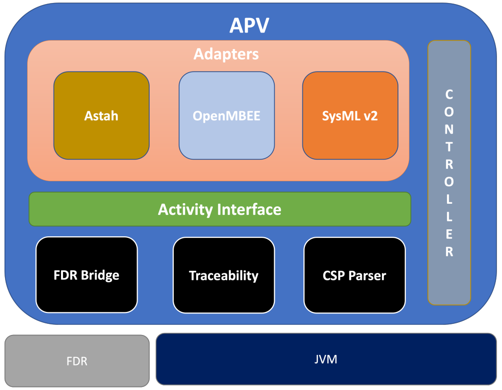


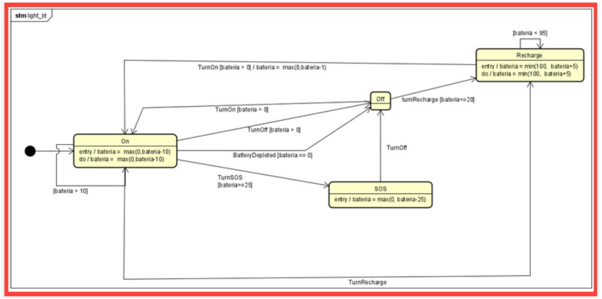
```
Welcome to FDR 4.2.4 (4388126e811531fd2eb760f4e17aa702f31b3dde)
FDR Version 4.2.4 copyright 2016 Oxford University Innovation Ltd. All Rights Reserved.
License: Academic license for non-commercial use only
Type :help for help
Option cspm.evaluator.record_stack_traces is set to a non-default value on (default is
off)
problema.csp>
```

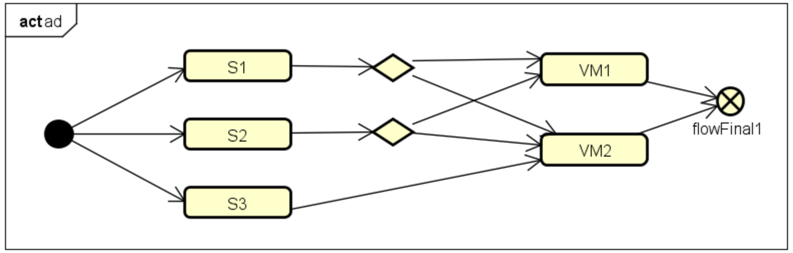
```
MAIN :[deadlock free]
Finished: Failed
MAIN :[divergence free]
Finished: Passed
MAIN :[deterministic]
Finished: Passed
```

```
Finished: Visited 0 processes and
Children (1/1 finished)
Graphing deter(MAIN)
Finished: Visited 1 processes and
Children (1/1 finished)
Constructing machines
Finished: Constructed 3 of 3 mach
Children (2/2 finished)
Generating machine constructors
Finished: Visited 0 processes gene
Generating machine constructors
Finished: Visited 1 processes gene
Checking refinement
Finished: Reconstructing countere
No tasks running
```






stm light_bt
[bateria < 95]
Recharge
entry / bateria = min(100, bateria+5)
do / bateria = min(100, bateria+5)
TurnOn [bateria > 0] / bateria = max(0,bateria-1)
Off
turnRecharge [bateria<=20]
TurnOn [bateria > 0]
TurnOff [bateria > 0]
On
entry / bateria = max(0,bateria-10)
do / bateria = max(0,bateria-10)
BatteryDepleted [bateria == 0]
TurnOff
TurnSOS [bateria>=25]
[bateria > 10]
SOS
entry / bateria = max(0, bateria-25)
TurnRecharge



MSDL

Verifying SysML/UML (Behavioural) Diagrams

Lucas Lima

MSDL Summer workshop

01 September 2023