Coverage Analysis of Net Inscriptions in Coloured Petri Net Models

Faustin Ahishakiye   José Ignacio Requeno Jarabo
Volker Stolz   Lars Michael Kristensen

Western Norway University of Applied Sciences

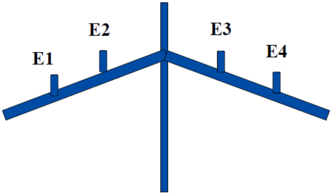
ICT-10-2016
COEMS
Continuous Observation of Embedded Multicore Systems
coems.eu

VECoS 2020
26-27 Oct. 2020

Western Norway University of Applied Sciences

Motivation

```
int isReadyToTakeOff(int E1,
   int E2, int E3, int E4){
if(((E1 == 1)||(E2 == 1)) &&
   ((E3 == 1)||(E4 == 1))){
  return 1;
  }else {
  return 0;
} }
```



- Programs and models contain conditions $\Longrightarrow$ coverage analysis
- What is "enough" is defined in terms of coverage
- More coverage requirements for safety-critical software
  - Modified Condition/Decision Coverage (MC/DC) criterion
- Model based testing and generating test cases:
  - non trivial (complex) and time consuming task for testers

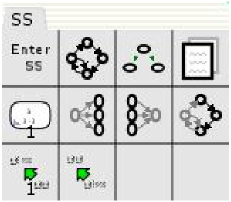
Motivation for coverage analysis of CPN Models

Coloured Petri Nets (CPNs) combine:

- **Petri nets**:
  - formal foundation for modelling concurrency and synchronization
- **Programming language**:
  - provides the primitives for modeling data manipulation
  - functional programming: standard meta-language (SML)

Traditional coverage analysis for CPN model

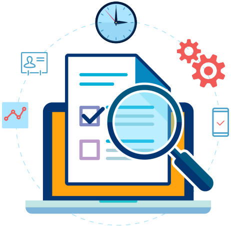
- behavioural properties related to net structure
  - dead markings, dead/live transition instances
- the net inscriptions are only implicitly validated
- coverage of net inscriptions is not made explicit
  - similar to branch/statement coverage



Our coverage analysis approach for CPN models

```
create or replace procedure p is
begin
  if f(1) = 1 or f(2) = 2 then
    dbms_output.put_line('covered');
  else
    dbms_output.put_line('not covered');
  end if;

end p;
/
```



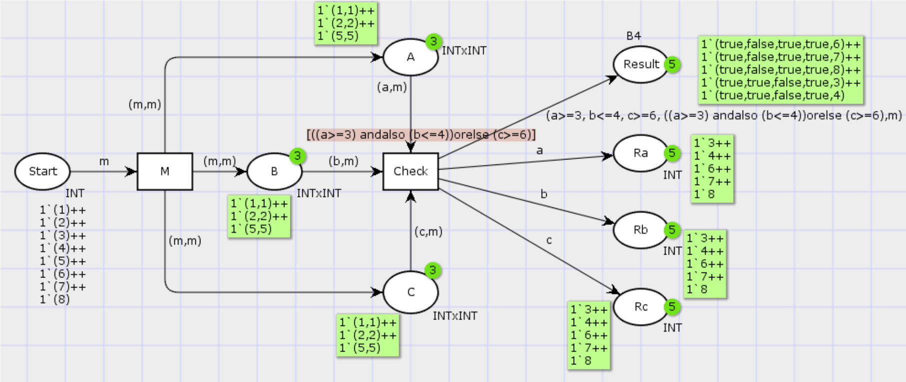
- CPN models contain conditions in SML expressions
  - establish a link between coverage analysis known from programming languages and net inscriptions of CPN models
  - are all conditions in SML expressions in a CPN model covered?
- Better than transition coverage

Research questions

- How can coverage criteria such as MC/DC normally used for programming languages be applied on CPN models?
  - what does MC/DC mean in the context of CPN models?
  - CPN SSE or CTL model checker show only truth evaluations
  - need evidence that each condition contributed to the outcome
- How to collect MC/DC coverage statistics in a CPN model?
- How well are SML conditions covered in existing CPN models?

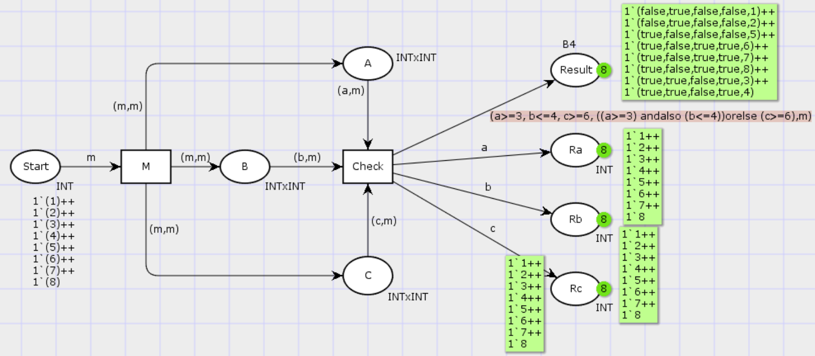
Outline

- Guard SML expression: $((a \geq 3)$ *andalso* $(b \leq 4))$ *orelse* $(c \geq 6)$



Introduction to CPN model

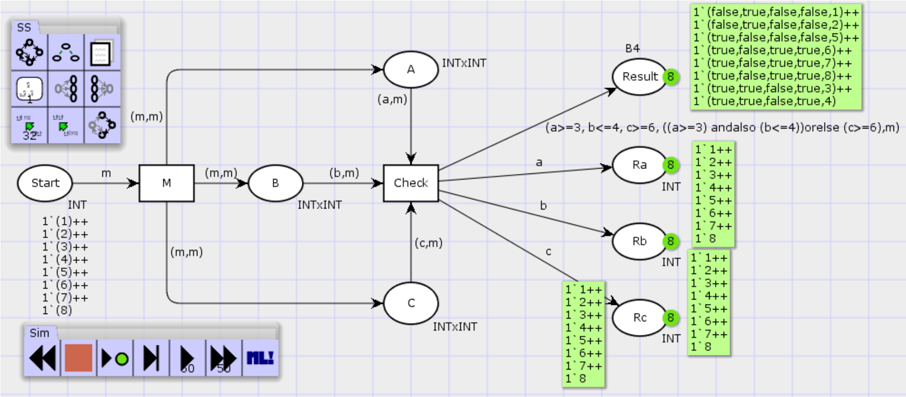
- Arc SML expression: tuple of
  $a \geq 3, b \leq 4, c \geq 6, ((a \geq 3)$ *andalso* $(b \leq 4))$ *orelse* $(c \geq 6), m$



State space & simulation analysis for CPN models

Two ways of running the CPN model:

- Simulation
- State space exploration (SSE)



State space & simulation analysis for CPN models

```
CPN Tools state space report for:  coverageexample2.cpn
Report generated:  Tue Sep 22 17:05:36 2020
Statistics
------------------------------------------------------------------------
State Space Nodes:  1944 Arcs:  9396 Secs:  1 Status:  Full
Scc Graph Nodes:  1944 Arcs:  9396 Secs:  0
Boundedness Properties
------------------------------------------------------------------------
Best Integer Bounds Upper Lower Guard'A 1 8 0 Guard'B 1 8 0 Guard'C 1 8 0 Guard'Ra 1 5 0 Guard'Rb 1 5 0
Guard'Rc 1 5 0 Guard'Result 1 5 0 Guard'Start 1 8 0
Best Upper Multi-set Bounds Guard'A 1 1`(1,1)++ 1`(2,2)++ 1`(3,3)++ 1`(4,4)++ 1`(5,5)++ 1`(6,6)++
1`(7,7)++ 1`(8,8) Guard'B 1 1`(1,1)++ 1`(2,2)++ 1`(3,3)++ 1`(4,4)++ 1`(5,5)++ 1`(6,6)++ 1`(7,7)++ 1`(8,8)
Guard'C 1 1`(1,1)++ 1`(2,2)++ 1`(3,3)++ 1`(4,4)++ 1`(5,5)++ 1`(6,6)++ 1`(7,7)++ 1`(8,8) Guard'Ra 1 1`3++
1`4++ 1`6++ 1`7++ 1`8 Guard'Rb 1 1`3++ 1`4++ 1`6++ 1`7++ 1`8 Guard'Rc 1 1`3++ 1`4++ 1`6++ 1`7++ 1`8
Guard'Result 1 1`(true,false,true,true,6)++ 1`(true,false,true,true,7)++ 1`(true,false,true,true,8)++
1`(true,true,false,true,3)++ 1`(true,true,false,true,4) Guard'Start 1 1`1++ 1`2++ 1`3++ 1`4++ 1`5++ 1`6++
1`7++ 1`8
Best Lower Multi-set Bounds Guard'A 1 empty Guard'B 1 empty Guard'C 1 empty Guard'Ra 1 empty Guard'Rb 1
empty Guard'Rc 1 empty Guard'Result 1 empty Guard'Start 1 empty
Home Properties
------------------------------------------------------------------------
Home Markings [1944]
Liveness Properties
------------------------------------------------------------------------
Dead Markings [1944]
Dead Transition Instances None
Live Transition Instances None
Fairness Properties
------------------------------------------------------------------------
No infinite occurrence sequences.
```

Overview on MC/DC coverage criterion

- **Definition of MC/DC by DO-178C:**
  each condition in a decision has shown to independently affect that decision's outcome by:
  - (1) varying just that condition while holding fixed all other possible conditions (**Unique cause MC/DC**), or
  - (2) varying just that condition while holding fixed all other possible conditions that could affect the outcome (**Masking MC/DC**)
- **Advantage of MC/DC:**
  - requires less test cases (only $n+1$ for n conditions)
  - only MC/DC checks independence effect of each condition

MC/DC test cases for $(A \wedge B) \vee C$

All possible pairs:
3 possible pairs for C

Nr	A	B	C	$(A \wedge B) \vee C$	MC/DC pairs
1	0	0	0	0	
2	0	0	1	1	C(1,2)
3	0	1	0	0	A(3,7)
4	0	1	1	1	C(3,4)
5	1	0	0	0	
6	1	0	1	1	C(5,6)
7	1	1	0	1	B(5,7)
8	1	1	1	1	

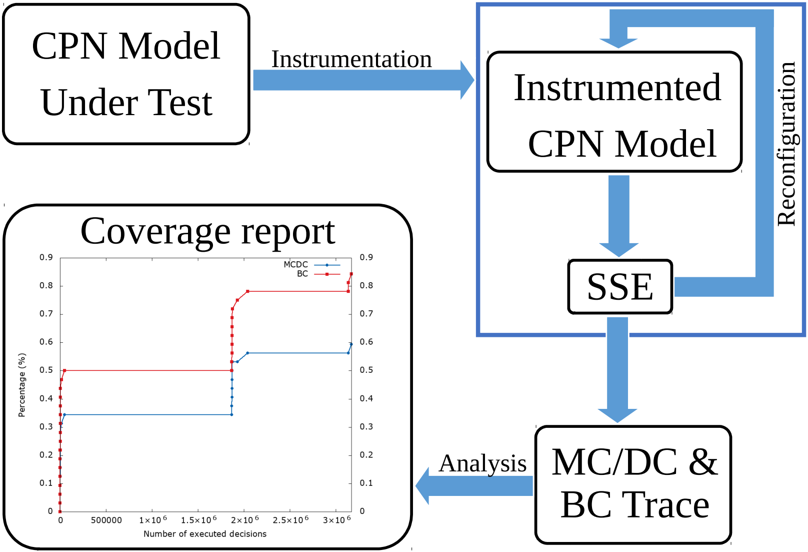
Required $n + 1$ MC/DC pairs:
Only one pair for C contribute
to the minimal set

Nr	A	B	C	$(A \wedge B) \vee C$	MC/DC pairs
	0	1	0	0	
	1	1	0	1	A(3,7)
	1	0	0	0	B(5,7)
	1	0	1	1	C(5,6)

Our solutions/contributions

- Implementation of a CPN Tools library
  - annotation and instrumentation mechanism
  - intercept and collect evaluations of boolean conditions
- A post-processing tool
  - determines whether each decision is MC/DC and branch-covered
- Evaluate our approach on four large public CPN models:
  - Paxos distributed-consensus algorithm
  - MQTT publish-subscribe protocol
  - distributed constraint satisfaction problem (DisCSP) algorithms
  - model of the runtime environment of an actor-based (CPNABS)

Experiment setup



CPN Tools library

- Our library contains different configuration modules:
  - instrumentation: interpretation of guards and arc expressions
  - logging : emit a log-entry that we can later collect and analyse
- Include the library into CPN model

```
val cpnmcdclibpath = "path/to/library";
use (cpnmcdclibpath^"config/logging.sml");
use (cpnmcdclibpath^"config/instrumentation.sml"
use (cpnmcdclibpath^"boot.sml");
use (cpnmcdclibpath^"config/simrun.sml");
```

MC/DC tool invocation

- User imports our library
- Invokes the central `mcdcgen()` function in line with how SSE works in CPN Tools
- Invocation with default settings (no timeout)

```
mcdcgen("path/to/mqtt.log");
```

- Invocation without timeout; base model + 2 configurations

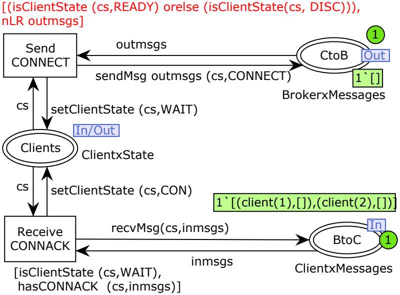
```
mcdcgenConfig(0, applyConfig,[co1,co2],"path/to/
```

Instrumentation of Net Inscriptions

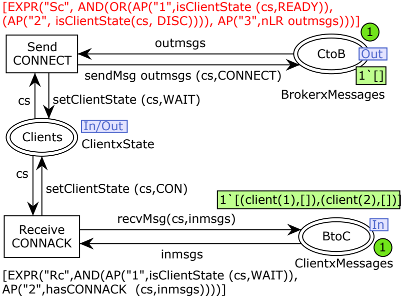
Instrument the guards on transitions:

A guard `a>0 andalso (b orelse (c=42));` $\iff$

`EXPR("Gid", AND(AP("1", a>0), OR(AP("2",b), AP("3", c=42))))`



(a) MQTT original model

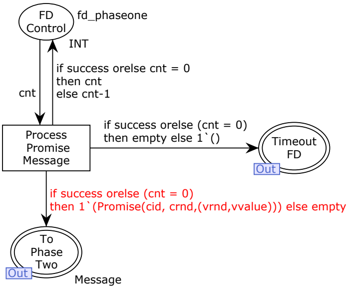


(b) Instrumented model

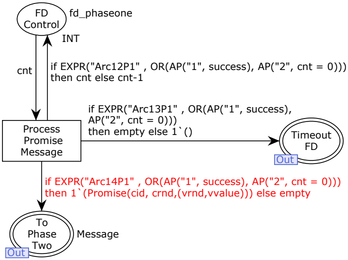
Instrumentation of Net Inscriptions

Instrument the arc expressions:

```
if bexp1 orelse bexp2 then e1 else e2; ⟺
if EXPR("A1",OR(AP("1", bexp1), AP("2",bexp2)) then e1 else e2
```



(c) Paxos original model



(d) Instrumented model

Post-processing tool of coverage data

Log decisions

```
...
a3:01:0
t42:01110:0
t42:01011:1
...
...
...
```

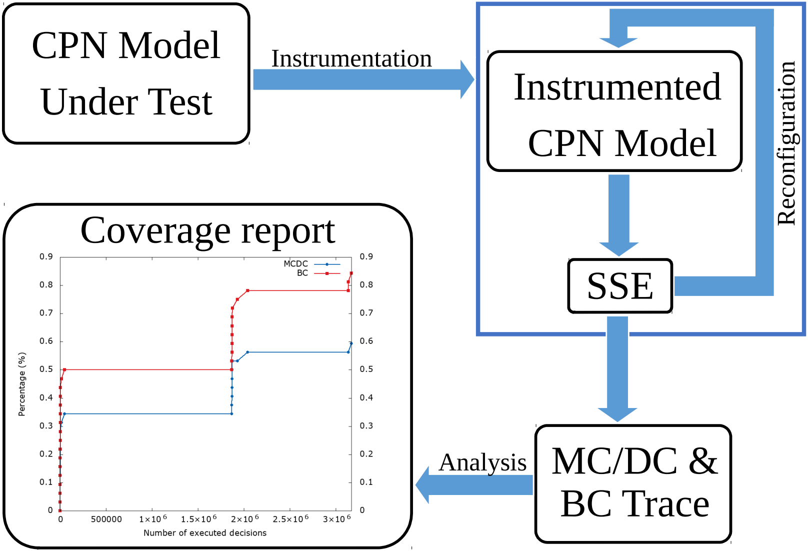
- Guard is evaluated multiple times with varying bindings
- Coverage data from multiple runs are combined

Decisions evaluation table

```
...
Returna19
0001      0
0010      0
0101      0
0110      0
1001      1
1101      1
1110      1
...
```

```
MCDC covered?  False
R{1:[(0001, 1001), (0101, 1101),
(0110, 1110)], 2:[], 3:[], 4:[]}
```

Experiment setup



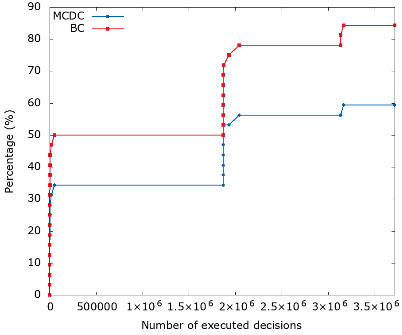
Experimental Results

- Evaluate our approach on four large public CPN models
- We record both MC/DC and BC as the ratio of covered decisions over the total number of decisions
- For the models with an infinite state space:
  - we aborted the SSE after two days
  - execution no longer seemed to increase the coverage metrics

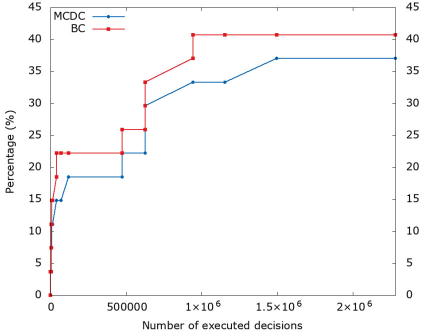
CPN Model	Executed decisions	Model decisions (m)	Non-trivial decisions	MC/DC (%)	BC (%)	State space
Paxos	2,281,466	27	11	37.03	40.74	infinite
MQTT	3,870	18	14	11.11	22.22	finite
CPNABS	3,716,896	32	13	59.37	84.37	infinite
DisCSP	233,819	12 (10)	5	45.45	45.45	finite

Experimental Results

MC/DC and BC coverage versus number of executed decisions



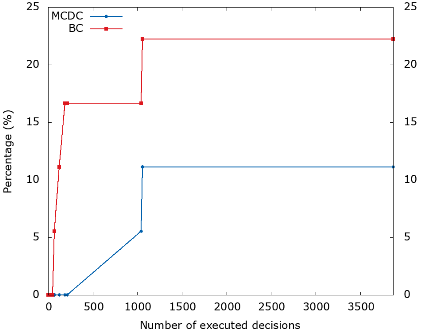
(e) CPN ABS model



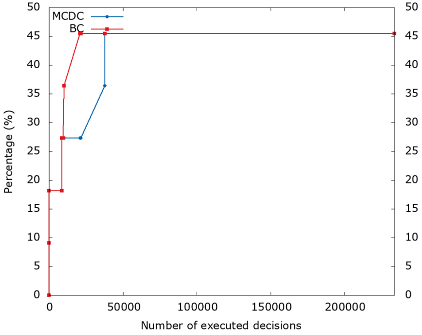
(f) Paxos model

Experimental Results

MC/DC and BC coverage versus number of executed decisions



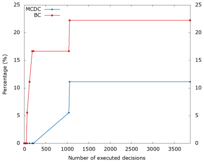
(g) MQTT model

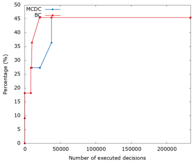


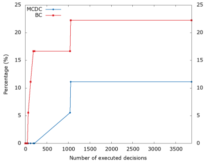
(h) DisCSP WCS model

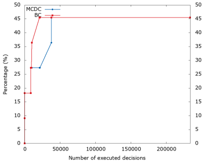
Experimental Results

The curves show low MC/DC & BC percentage









Discussion of Results

- This should attract the attention of the developer:
  - assess whether they have tested their models enough
  - require to revisit their test-suite
- Two factors affect the coverage results presented:
  1. the tested models had no clear test suites
  2. the models might be erroneous in the sense that some decisions are never or partially executed (modelling issue)

Discussion of Results

Our instrumentation:

- No significant impact on the execution time of the model
- Consider the time taken for the full SSE (finite state models):
  - it takes 212.346 sec vs 214.922 sec without and with instrumentation $\Longleftrightarrow$ 1% of overhead: cost for instrumentation.

Conclusion

- A new approach and a tool to measure MC/DC and BC of SML decisions in CPN models
  - a library and annotation mechanism that evaluate conditions
  - post-processing tool that computes and checks coverage
  - collect coverage data from publicly available CPN models
- MC/DC & BC percentage is quite low for all the four CPN models tested, action need to be taken accordingly
- Coverage analysis is interesting and useful for CPN models
  - shows the covered guard and arcs SML decisions
  - shows how conditions contributed on the decisions' outcome
  - reveals related potential problems

Future work

- Simulation guided by the results to test case generation
  - try to achieve higher coverage for unexplored parts of the model
  - suggest test cases for uncovered conditions
- Visualising coverage information in the graphical user interface
- Discuss with the original developers of the tested models:
  - to see if the measured coverage is within their expectations
  - they may have used their model in various configurations