

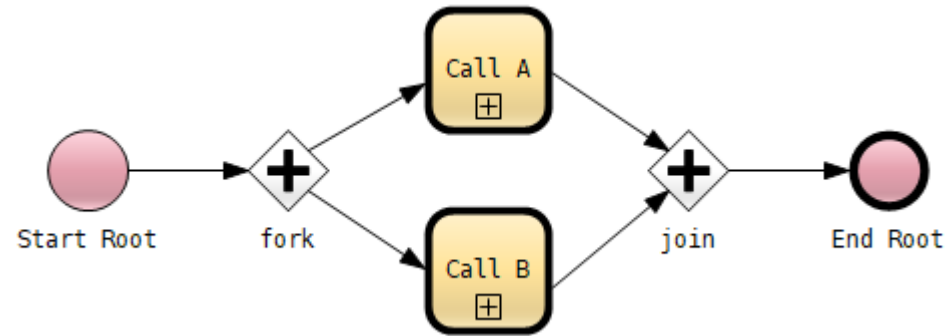
BPMN2 Semantics & OBP2

A demonstration by LE ROUX Luka

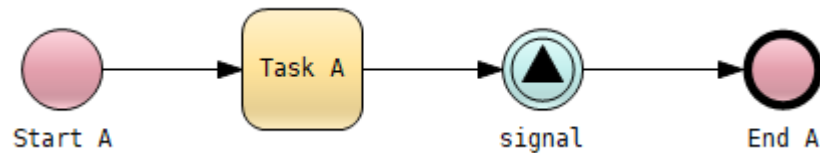
Demonstrations

- Part 1 : A simple BPMN2 model
 - Simulation
 - Verification
 - Simulation & verification with “flow completion” (symmetry reduction)
- Part 2 : One Way representative model
 - Simulation & deadlock
 - With “flow completion”
 - “Process 0” isolated
- Ongoing work (MSP3 as a target)

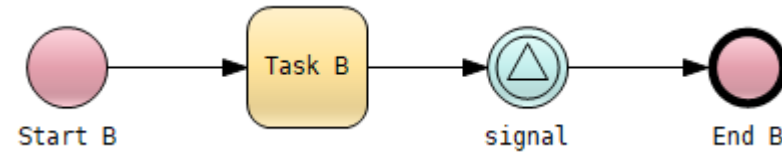
Part 1 : A simple BPMN2 model



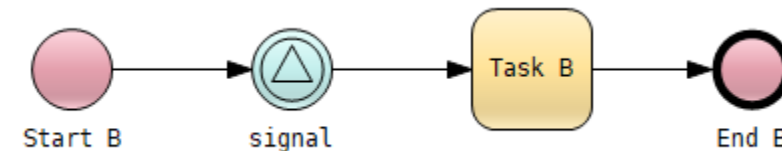
Process Root



Process A



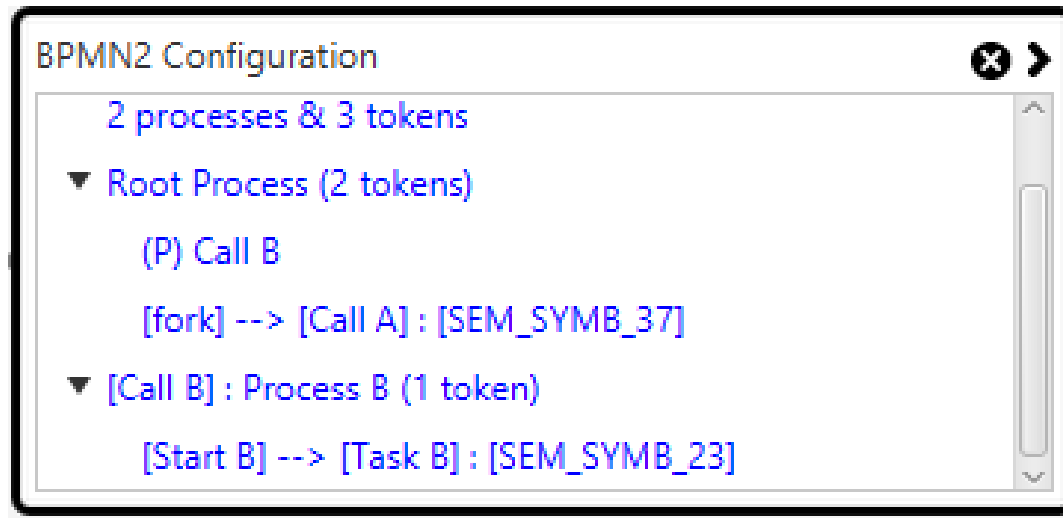
Process B (V1)



Process B (V2)

Part 1 : Simulation

- Demo simulation (V1)
- Example of a BPMN2 execution state and its associated actions:



▶ OPEN_PROCESS Call A

▶ START_TASK Task B

Part 1 : Deadlock properties

- All models that terminates have a “deadlock”, the following fails

```
noDeadlock = [] not |deadlock|
```

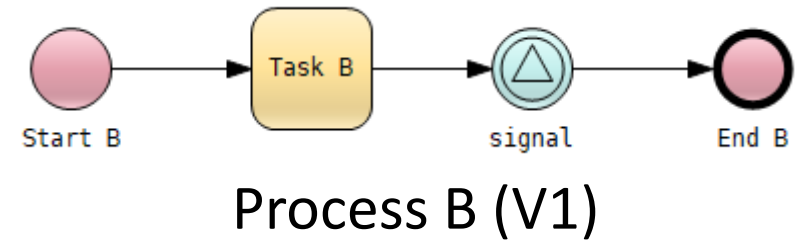
- We can filter out “expected deadlocks”

```
noDeadlock_bpmn = [] (|deadlock| implies |source.isTerminated()|)
```

- Alternatively, we can check that a process **always** terminates

```
alwaysTerminates = []<>|source.isTerminated()|
```

Part 1 : Deadlock verification

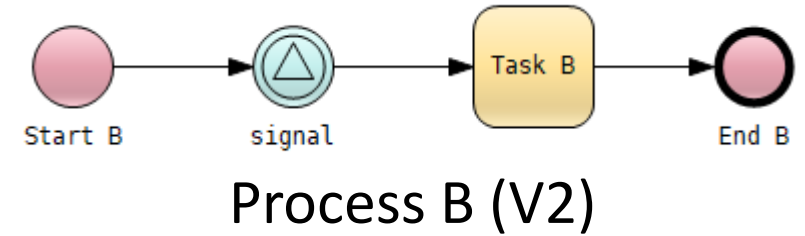


- The demonstration will show this is not verified on V1
- This is due to “process B” potentially not being ready to “catch”
- But if “Task B” ends before “signal” is thrown, the process terminates
- Given the following atomic propositions :

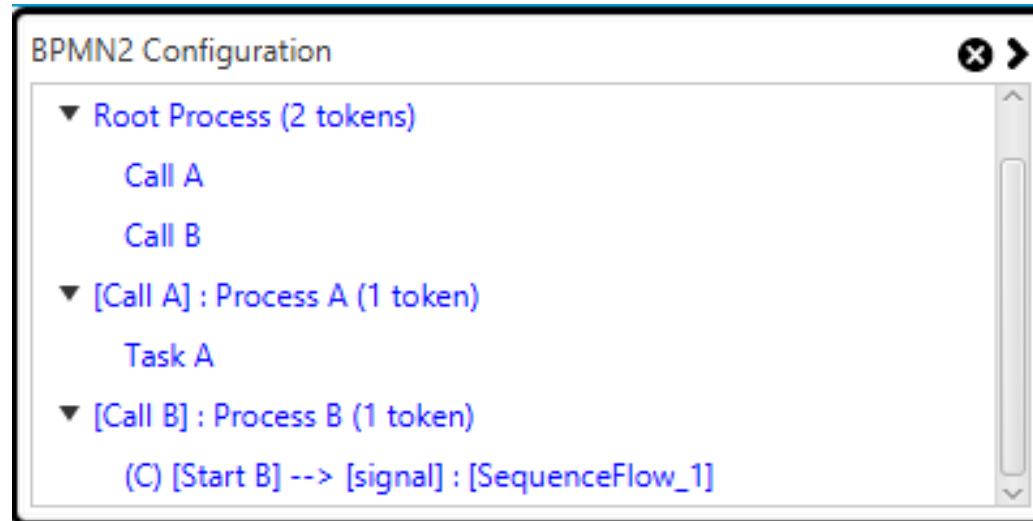
```
B = |action.endsTask("Task B")|  
S = |action.sendsSignal("signal")|  
T = |source.isTerminated()|
```

- Dwyers Pattern : After B, T responds to S
- Translates in LTL to : $[] (B \rightarrow [] (S \rightarrow \langle \rangle T))$
- Demo verification (v1) : only “noDeadlock_iff” is verified

Part 1 : “Flow completion”



- The symmetry reduction developed for One Way
- One implication is to give priority to sequence flows over task actions
- As such, in V2, process B is always ready to “catch”
- Demo simulation & verification (V2) : “alwaysTerminates” now holds



Initial configuration with flow completion activated

Part 2 : One Way representative model

- Demo simulation & deadlock, with and without “flow completion”
- Demo isolated process 0

▼ BPMN2 Configuration

30 processes & 620 tokens

▶ process 0 (12 tokens)

▼ [callActivity 7] -> [callActivity 33] : process 28 (5 tokens)

(C) [parallelGateway 249] --> [Event 214] : [Megald-037E13D96101840E]

(C) [parallelGateway 249] --> [MG-04 passed notification 49] : [Megald-037E13E26101845E]

(C) [parallelGateway 249] --> [Event 213] : [Megald-037E140D61018551]

(C) [parallelGateway 249] --> [Event 166] : [Megald-037E14606101863A]

(C) [parallelGateway 249] --> [Event 165] : [Megald-037E14646101868A]

▶ [callActivity 9] : process 30 (12 tokens)

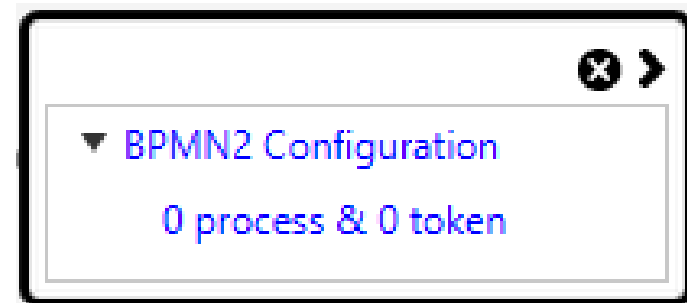
▶ [callActivity 7] : process 22 (6 tokens)

▶ [callActivity 0] : process 1 (16 tokens)

Deadlock found in the model **without** “flow completion”

■ noDeadlock

Finished 1422 configurations 0 transitions in 43322 ms



Deadlock found in the model **with** “flow completion”
(a proper termination)









Ongoing work (MSP3 as a target)

- Timed integration : BPSIM & atomic propositions
- Algorithmics (bitstate hashing)
- More “options” (isolate other processes ?)

deadlock.gpsl

```
1 noDeadlock = [] not |deadlock|
2
3 noDeadlock_bpmn = [] (|deadlock| implies |state.isTerminated()|)
4
5 alwaysTerminates = []<>|state.isTerminated()|
6
7 // If "Task B" ends before "signal" is thrown, process always properly ends
8 // Given the following atomic propositions :
9 B = |action.endsTask("Task B")|
10 S = |action.sendsSignal("signal")|
11 T = |state.isTerminated()|
12 // Dwyers Pattern : After B, T responds to S
13 noDeadlock_iff = [] (B -> [](S -> <>T))
```

Files

-  deadlocks.gpsl
-  oneWay.bpmn
-  oneWay_flowCompletion.bpsli
-  oneWay_process0.bpsli
-  simpleCallsV1.bpmn
-  simpleCallsV1_flowCompletion.bpsli
-  simpleCallsV2.bpmn
-  simpleCallsV2_flowCompletion.bpsli

```
{  
  "modelName" : "oneWay.bpmn",  
  "flowCompletion" : true,  
  "includeCalledProcesses" : false  
}
```