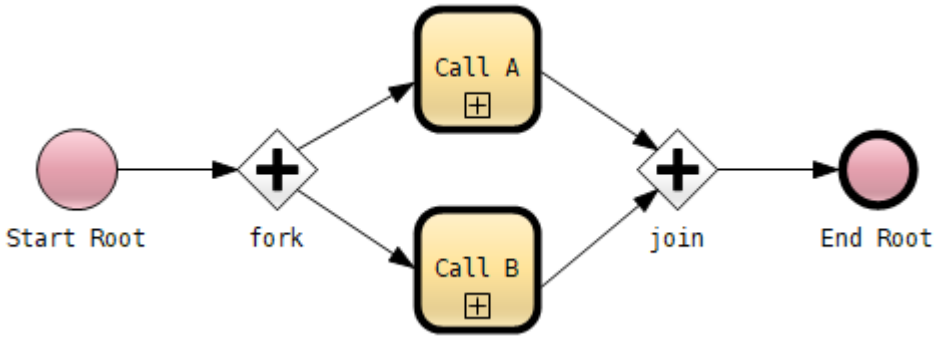
BPMN2 Semantics & OBP2

A demonstration by LE ROUX Luka

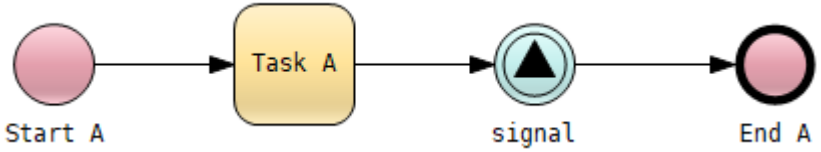
Demonstrations

- 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)

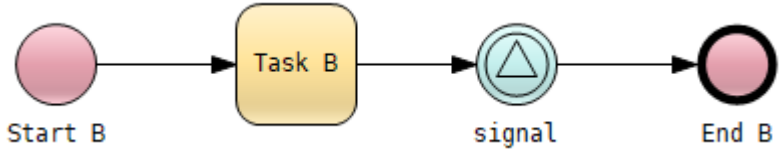
Part 1 : A simple BPMN2 model



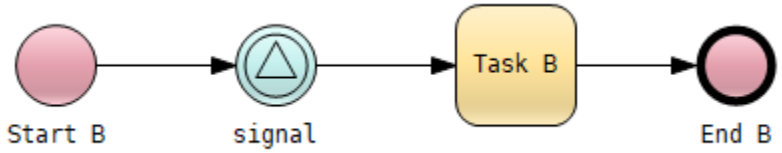
Process Root



Process A



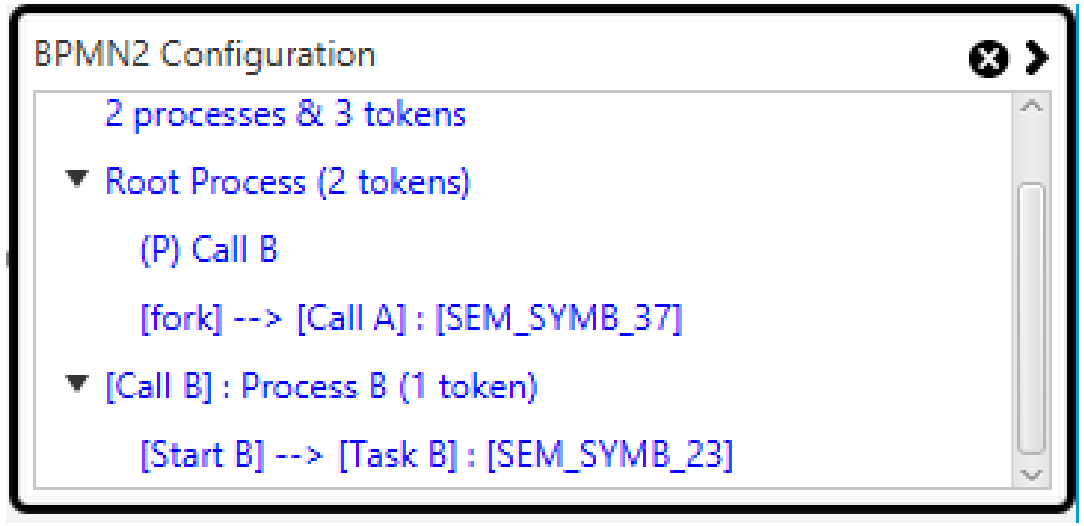
Process B (V1)

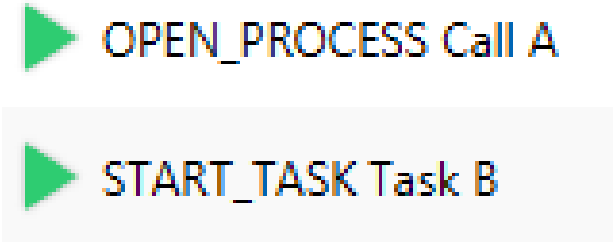


Process B (V2)

Part 1 : Simulation

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





Part 1 : Deadlock properties

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

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

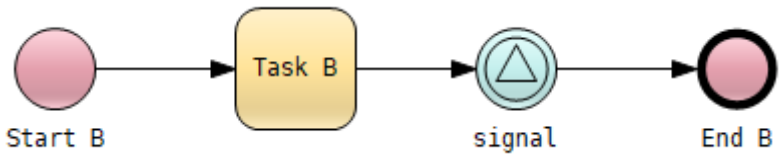
- We can filter out “expected deadlocks”

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

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

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

Part 1 : Deadlock verification



Process B (V1)

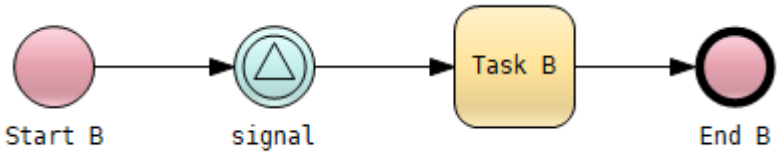
- The demonstration will show this is not verified on V1
- This is due to “process B” potentially not being ready to “catch”
- But if “Task B” ends before “signal” is thrown, the process terminates
- Given the following atomic propositions :

```
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 -> [](S -> <>T))`

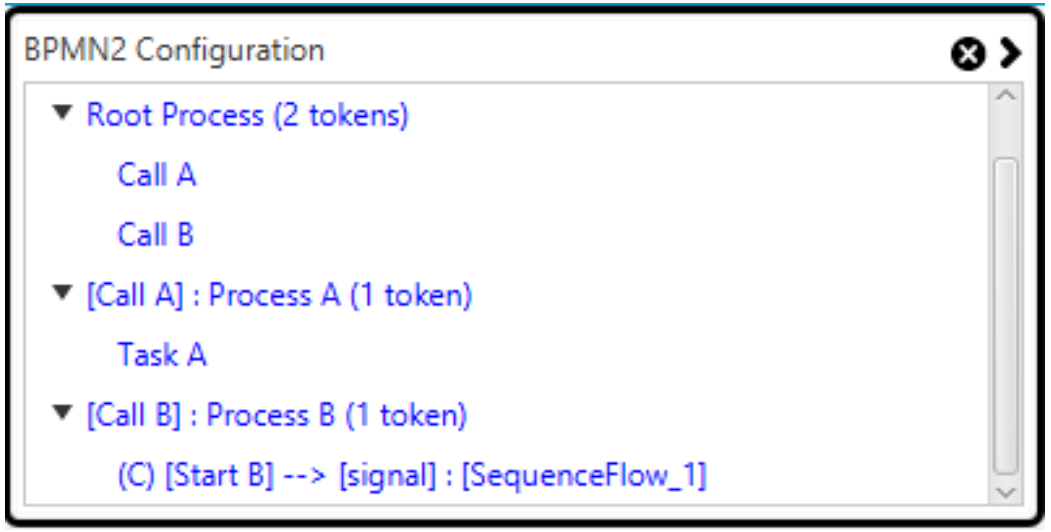
- Demo verification (v1) : only “noDeadlock_iff” is verified

Part 1 : “Flow completion”



Process B (V2)

- 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

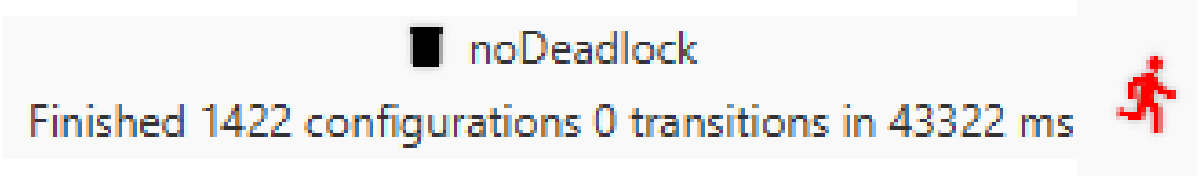


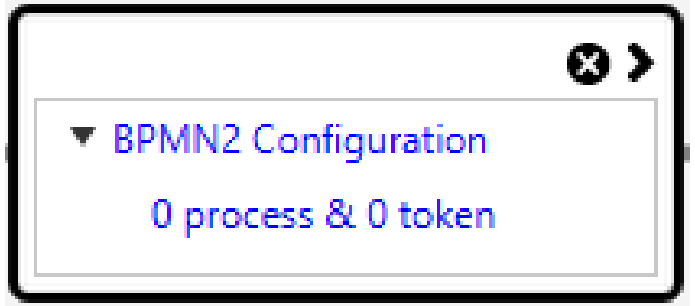
Initial configuration with flow completion activated

Part 2 : One Way representative model

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

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





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

```
noDeadlock = [] not |deadlock|

noDeadlock_bpmn = [] (|deadlock| implies |state.isTerminated()|)

alwaysTerminates = []<>|state.isTerminated()|

// If "Task B" ends before "signal" is thrown, process always properly ends
// Given the following atomic propositions :
B = |action.endsTask("Task B")|
S = |action.sendsSignal("signal")|
T = |state.isTerminated()|
// Dwyers Pattern : After B, T responds to S
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

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