Modelling and Verifying Web Service Orchestration by means of the Concurrency Workbench

Mariya Koshkina / Franck van Breugel

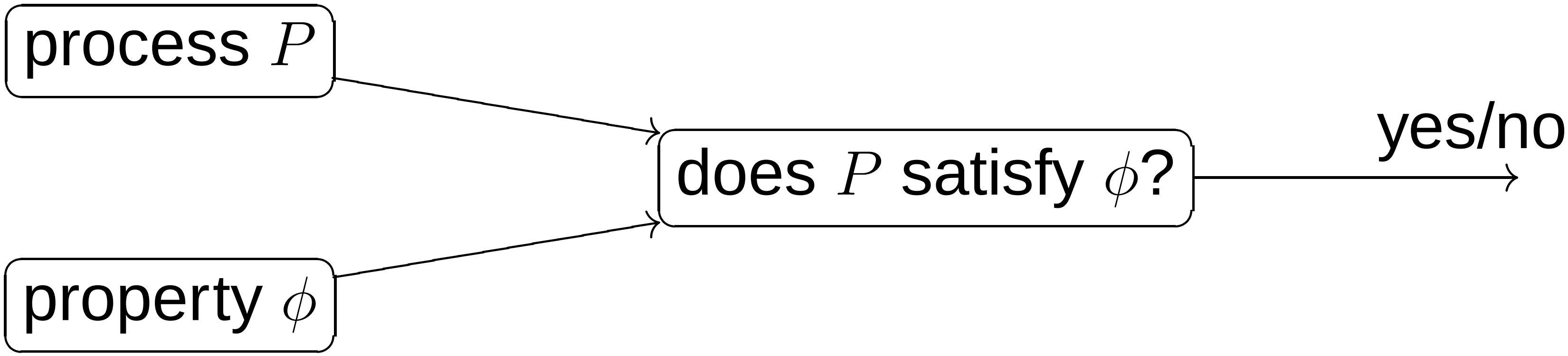
IBM, Toronto / York University, Toronto

Concurrency Workbench (CWB)

- Verification tool originally developed at North Carolina State University and currently maintained at SUNY Stony Brook
- Originally intended for verification of CCS (Calculus of Communicating Systems), but can be extended to support other languages
- Supports several verification methods

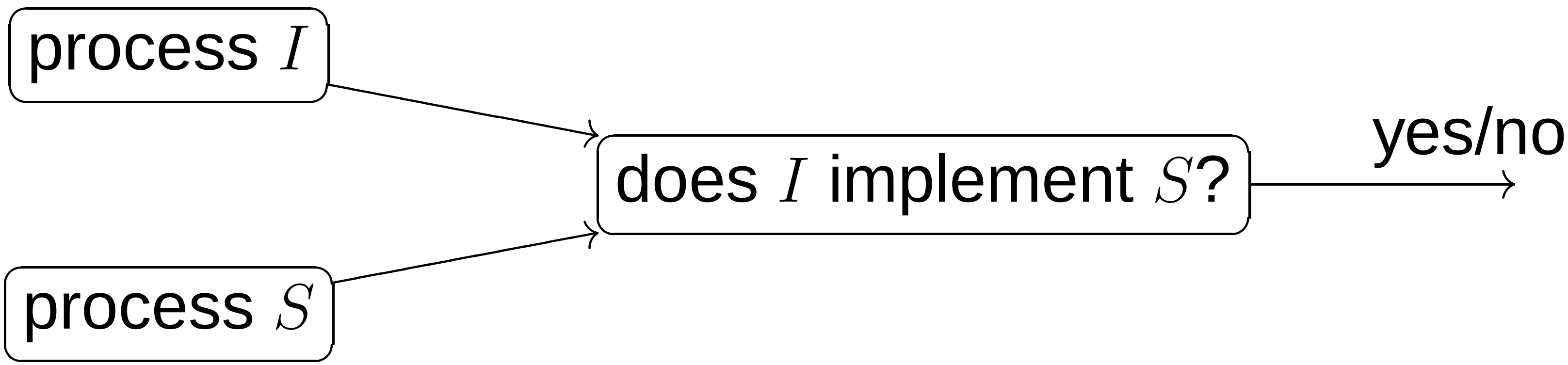
Supported Verification Methods

Model checking



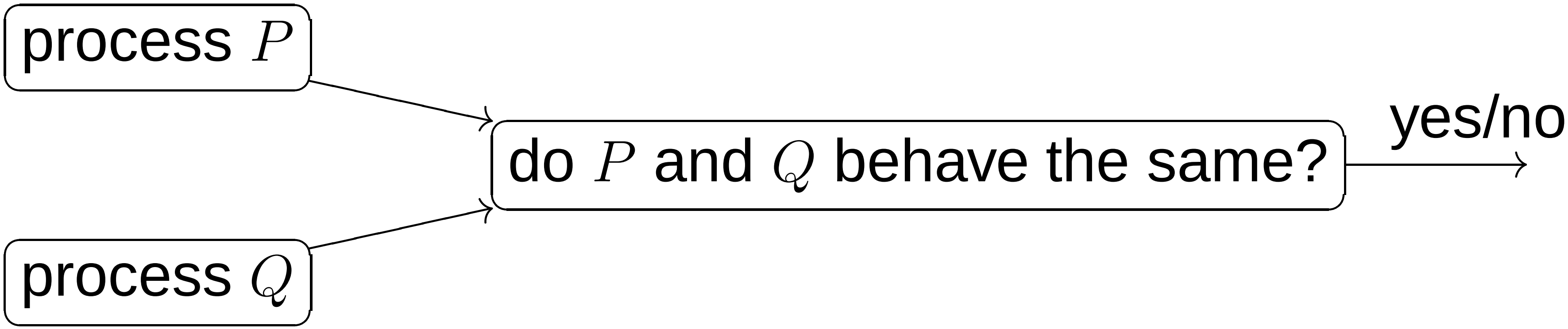
Supported Verification Methods

Preorder checking



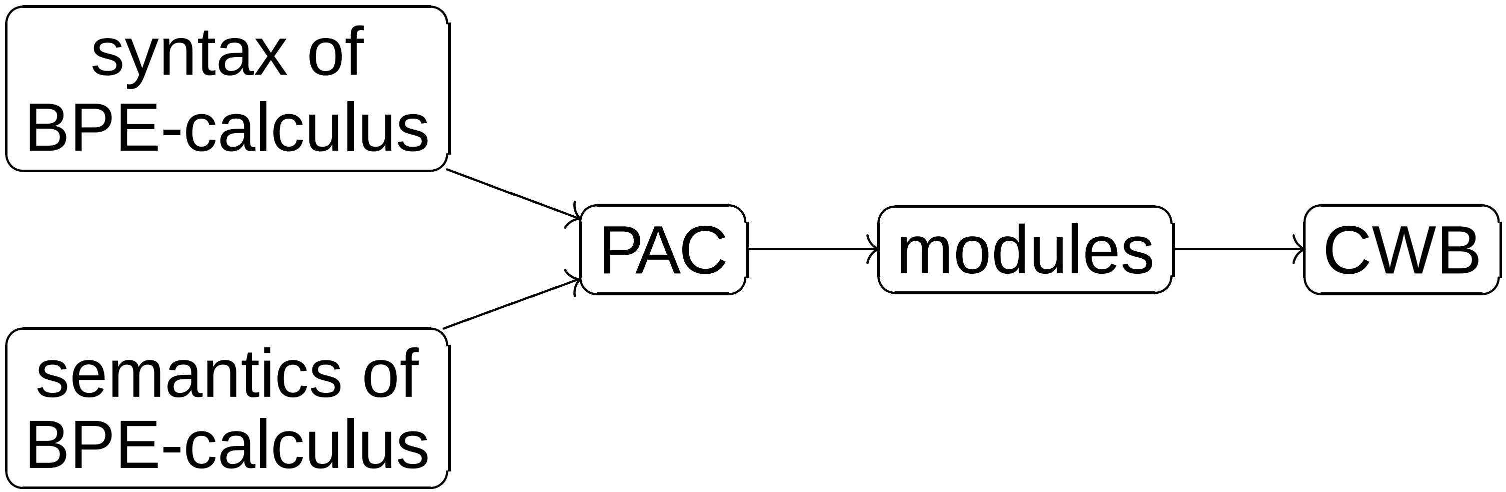
Supported Verification Methods

Equivalence checking



Overview

- BPE-calculus – small language based on BPEL4WS (Business Process Execution Language for Web Services)
- We extend CWB to support BPE-calculus
- Process Algebra Compiler (PAC) – tool used to extend CWB



BPE-Calculus

- Small language that captures the flow of control of BPEL4WS
- Abstracts from some of the details:
  - Abstracts from data
  - Abstracts from compensation/fault handlers
  - Does not model time

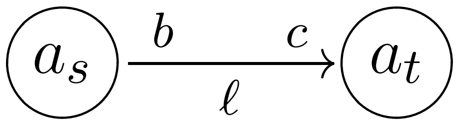
BPE-Calculus Syntax

Models basic constructs of BPEL4WS:

- Basic activities:
  - External (α)
  - Internal (τ)
- Prefixing ($\alpha.P$)
- Choice ($P + P$)
- Concurrency ($P \parallel P$)

BPE-Calculus Syntax

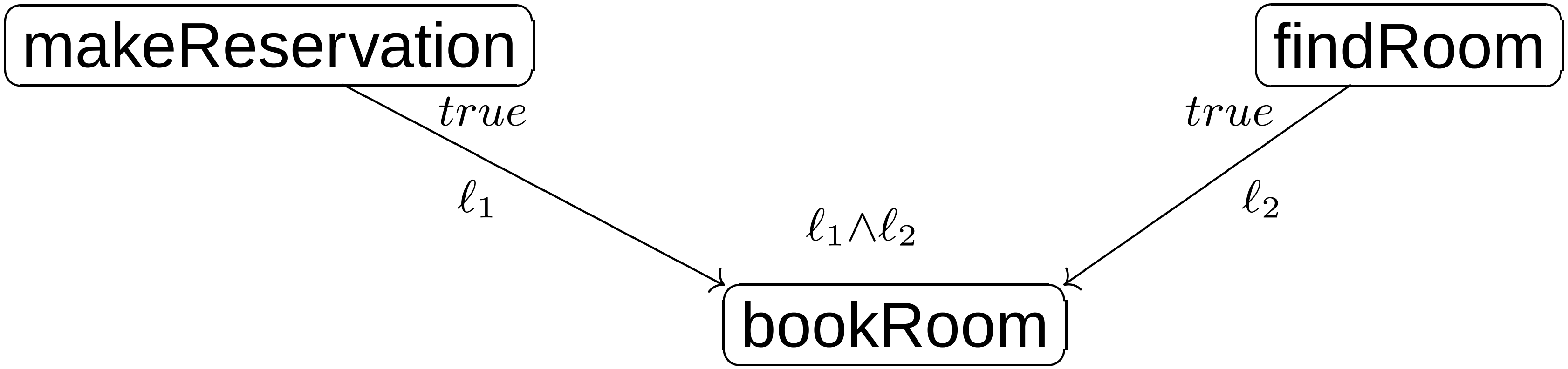
Synchronization is provided by links



Outgoing link: $\ell \uparrow b.P$

Join condition: $c \Rightarrow P$

BPE-Calculus Syntax



The corresponding BPE-process:

$$\text{makeReservation}.\ell_1 \uparrow \text{true}.0 \parallel$$
$$\text{findRoom}.\ell_2 \uparrow \text{true}.0 \parallel$$
$$(\ell_1 \wedge \ell_2) \Rightarrow \text{bookRoom}.0$$

BPE-Calculus Semantics

- Semantics of BPE-calculus is modeled by means of structural operational semantics (Plotkin), which describes the semantics of the process in terms of all possible transitions that the process can make
- Transition: $P \xrightarrow{\text{action}} P'$
- Rules: $\dfrac{\text{premises}}{\text{conclusion}}$ (side conditions)

BPE-Calculus Semantics

A state is a pair $\langle P, \lambda \rangle$, where λ contains the values of the links (true, false, undefined)

Sample rules:

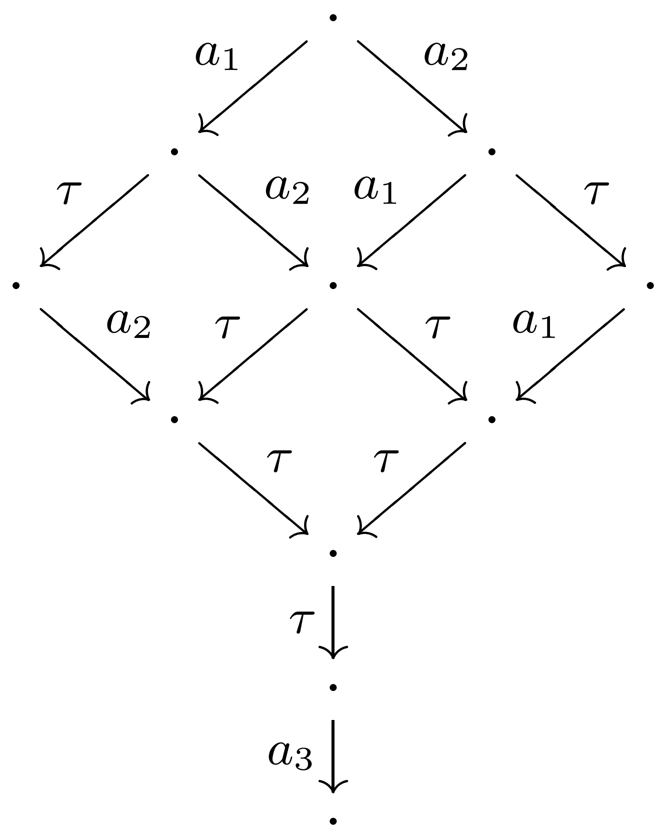
$$(\textsc{Act}) \quad \langle \alpha.P, \lambda \rangle \xrightarrow{\alpha} \langle P, \lambda \rangle$$

$$(\textsc{Flow}_\ell) \quad \frac{\langle P_1, \lambda \rangle \xrightarrow{\alpha} \langle P_1', \lambda' \rangle}{\langle P_1 \parallel P_2, \lambda \rangle \xrightarrow{\alpha} \langle P_1' \parallel P_2, \lambda' \rangle}$$

$$(\textsc{Flow}_r) \quad \frac{\langle P_2, \lambda \rangle \xrightarrow{\alpha} \langle P_2', \lambda' \rangle}{\langle P_1 \parallel P_2, \lambda \rangle \xrightarrow{\alpha} \langle P_1 \parallel P_2', \lambda' \rangle}$$

BPE-Calculus Semantics

$$a_1.\ell_1 \uparrow \textsf{true}.0 \parallel a_2.\ell_2 \uparrow \textsf{true}.0 \parallel \ell_1 \wedge \ell_2 \Rightarrow a_3.0$$



Concurrency Workbench (CWB)

- We use Process Algebra Compiler (PAC) to extend CWB
- PAC takes as input:
  - Syntax description file (Yacc-like grammar)
  - Semantics description file (SOS rules)
- PAC generates:
  - Modules to plug into CWB
- Resulting version of CWB supports verification of BPE-calculus

CWB: Verification

Model Checking: verify that a process satisfies a given property

- Deadlock-freedom
- Other process-specific properties
- Example:

$$\ell_3 \Rightarrow a_1.\ell_1 \uparrow \textsf{true}.0 \parallel$$
$$\ell_1 \Rightarrow a_2.\ell_2 \uparrow \textsf{true}.0 \parallel$$
$$\ell_2 \Rightarrow a_3.\ell_3 \uparrow \textsf{true}.0 \parallel$$
$$a_4.0 \parallel a_5.0 \parallel a_6.0$$

CWB: Verification

Preorder Checking: verify that an implementation satisfies its specification

- Example:
  - Implementation:

    $$\text{receive}.\tau.\ell_1 \uparrow \text{true}.0 + \text{receive}.\tau.\ell_2 \uparrow \text{true}.0 \parallel \ell 1 \vee \ell_2 \Rightarrow \text{reply}.0$$

  - Specification: receive.reply.0

CWB: Verification

Equivalence Checking: check behavioral equivalence

- Can be used to minimize a process
- Example:
  - Process

$$\text{receive}.\tau.\ell_1 \uparrow \text{true}.0 + \text{receive}.\tau.\ell_2 \uparrow \text{true}.0 \parallel \ell_1 \vee \ell_2 \Rightarrow \text{reply}.0$$

  - is observationally equivalent to receive.reply.0

Conclusion

- Introduced BPE-calculus that models BPEL4WS
- Used BPE-calculus syntax and semantics as input to PAC
- Extended CWB to support BPE-calculus

Future Work

- Extend BPE-calculus to incorporate other features of BPEL4WS:
  - Compensation and fault handlers
  - Time
  - Data