Analiza procesów biznesowych (BPEL4WS) za pomocą CWB

Grzegorz Statuch

Streszczenie

- BPEL4WS – *The Business Process Execution Language for Web Services.* Aktualnie rozwijany język, służący do specyfikacji interakcji pomiędzy usługami webowymi (*Web Services*)
- Analiza procesów języka BPEL w aspekcie wykrywania zakleszczeń
- *BPE-calculus* - niewielki język oparty na BPEL4WS pozwalający wyrazić wybrane własności języka BPEL
- Rozbudowa CWB (*The Concurrency Workbench of The New Century*) o możliwość definiowania i analizy procesów za pomocą zaproponowanego języka
- Kompilator algebry procesów – PAC (*Process Algebra Compiler*). Formalnie zdefiniowana składnia i semantyka *BPE-calculus* stanowią dane wejściowe kompilatora PAC, który produkuje moduły włączane do CWB. Tak zmodyfikowane narzędzie wspiera BPE-calculus

BPEL4WS

- Usługi webowe (Web Services) – mogą stać się powszechne
- Zamawianie biletów lotniczych przez Internet
  - klient komunikuje się z agencją turystyczną
  - agencja wysyła zapytania do jednej lub wielu linii lotniczych\
  - oczekuje na potwierdzenia
  - wysyła odpowiedź do klienta
- Proces biznesowy w którym bierz udział kilka stron wymieniających pomiędzy sobą komunikaty przez pewien okres czasu
- Ważne, aby takie procesy biznesowe specyfikować w sposób prosty i niezależny od platform programowo-sprzętowych realizujących usługi webowe
- Firmy BEA, IBM i Microsoft zaproponowały język BPEL oparty na XML. Składnia języka jest opisana za pomocą schematu XML

BPEL4WS - Przykład

- Poniższy fragment kodu w języku BPEL wywołuje serwis webowy rezerwacji lotów

```
<invoke partner="AirCanada"
        portType="reservationsPT"
        operation="makeReservation"
        inputContainer="request"
        outputContainer="confirmation">
</invoke>
```

BPEL4WS - Elementy

- BPEL jest używany do specyfikacji stron (partnerów) zaangażowanych w proces, przepływ i zawartości komunikatów przesyłanych między stronami, logikę procesu i obsługę błędów
- W języku BPEL wyróżnia się **akcje proste** (basic activites), które są elementami składowymi **akcji złożonych** (structured activites)

BPEL4WS - Akcje

- Akcjami podstawowymi są: wywoływanie operacji usługi webowej (ivoke), otrzymywanie i wysyłanie odpowiedzi na zapytania (receive, replay) i dołączanie danych do wiadomości
- Akcjami złożonymi są między innymi następujące konstrukcje: sekwencja (sequence), wykonanie równolegle (flow), while, switch oraz pick – rodzaj niedeterministycznego wyboru

BPEL4WS - Synchronizacja

- Synchronizacja jest zrealizowana za pomocą połączeń skierowanych (directed links). W połączeniu biorą udział: akcja źródłowa (source of the link) i akcja docelowa (target of the link) oraz opcjonalnie warunek przejścia (tranzycji) określony wyrażeniem boolowskim
- Akcja docelowa może się wykonać tylko po zakończeniu swojej aktywności przez akcję źródłową. Każda akcja, która ma połączenia przychodzące ma również warunek wejściowy (join condition), który jest wyrażeniem boolowskim zawierającym stany połączeń oraz operatory boolowskie. Jeśli przyjmie wartość true, do akcja się wykonuje a w przeciwnym przypadku akcja ta jest opuszczana (pomijana)

BPEL4WS - DPE

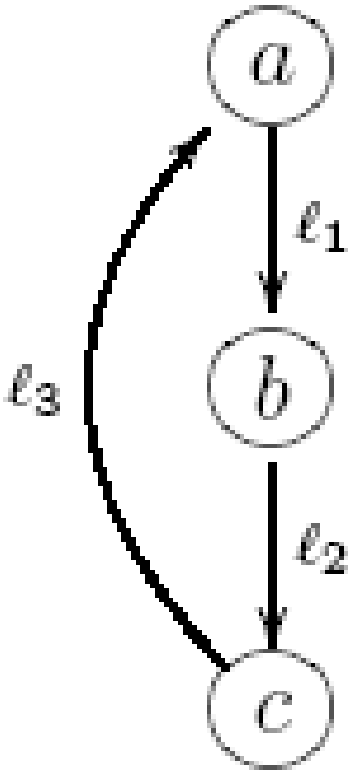
- Co się stanie z połączeniami wychodzącymi z pominiętej akcji?
  - Aby zapobiec oczekiwaniu w nieskończoność przez akcje docelowe tych połączeń, stosowana jest technika DPE (Death Path Elimination), która polega na ustawianiu wartości połączeń wychodzących pominiętych akcji na false

Weryfikacja procesów BPEL

- Równoległość stanowi podstawową cechę procesów jezyka BPEL
- Stworzenie narzędzi do badania ich poprawności jest niezwykle ważne
- Głównym celem przedstawianych prac, jest przygotowanie narzędzia umożliwiającego wykrywanie zakleszczeń

BPEL4WS – Zakleszczenia (I)

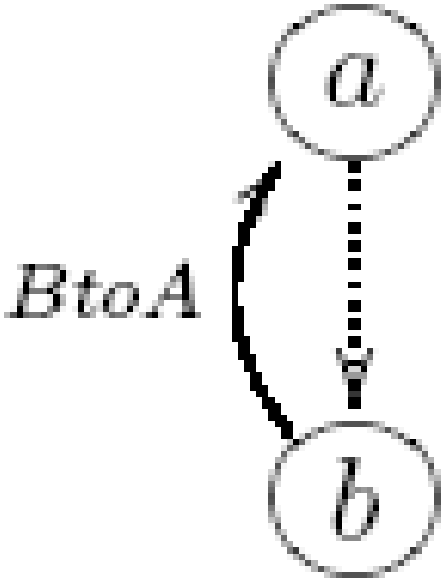
- Zakleszczenia mogą powstawać wtedy, gdy połączenia pomiędzy akcjami stworzą cykl (control cycle) co obrazuje rysunek obok
- Akcja źródłowa nie może się rozpocząć zanim nie zakończy się akcja docelowa



BPEL4WS – Zakleszczenia (II)

Mniej oczywisty przykład

```
<sequence>
    <invoke partner="partnerA"
            operation="a">
            <target linkName="BtoA"/>
    </invoke>
    <invoke partner="partnerB"
            operation="b">
            <source linkName="BtoA"/>
    </invoke>
</sequence>
```



BPE-calculus

- Prosta i zwięzła algebra procesów dla języka BPEL
- Ma modelować przepływ sterowania w procesie biznesowym abstrahując od wielu elementów szczegółowych nieistotnych z punktu widzenia analizy
- Pominięcie następujących elementów języka: przesyłanie danych, obsługa błędów, definicje partnerów, atrybuty procesów
- Rrezygnaqcja z notacji w postaci tagów XML na rzecz bardzie zwięzłej notacji

Akcje proste (basic activities)

- Akcje proste zostały zatem podzielone na dwie grupy:
  - akcje obserwowalne (zewnętrzne)
  - akcje nieobserwowalne (wewnętrzne) oznaczane literą τ
- Akcje invoke, receive i reply komunikują się z otoczeniem są zatem zdarzeniami obserwowalnymi
- Akcja empty nie jest obserwowalna (wewnętrzna). Ponieważ w modelu nie uwzględnia się czasu i danych, zatem akcje wait i assign będą traktowane tak samo jak akcja empty

Akcje proste (basic activities)

- Podczas translacji akcji prostych z języka BPEL do BPE-calculus, akcje wewnętrzne tracą swoje nazwy i stają się τ, natomiast akcje zewnętrzne są identyfikowane przez unikalne nazwy. Mechanizm konwertujący z języka BPEL do BPE-calculus ma swobodę wyboru w traktowaniu akcji jako zewnętrznych bądź wewnętrznych
- Wykrywanie zakleszczeń nie zależy od tej klasyfikacji akcji
- Warto zauważyć, że zdefiniowanie wszystkich akcji jako zewnętrzne daje więcej informacji na temat analizowanego procesu

Akcje złożone (structured activities)

- **Sequence**
  - W BPE-calculus akcja sequence jest reprezentowana za pomocą średnika. Na przykład *buyTicket ; rentCar*
- **Pick** (niedeterministyczny wybór zależny od otoczenia)
  - Instrukcja pick składa się z klauzul onMessage i opcjonalnej klauzuli onAlarm
  - W instrukcji pick wykonuje się tylko jedna z jej klauzul
  - Otrzymanie wiadomości jest modelowane przez akcję zewnętrzną (obserwowalną)
  - Upłynięcie czasu timera jest reprezentowane przez akcję nieobserwowalną.
  - Każda z klauzul jest reprezentowana za pomocą konstrukcji postaci α ; P gdzie α jest akcją prostą lub τ a P jest następującym po niej procesem
  - Wybór jest ozaczany znakiem plus

Akcje złożone (pick - przykład)

```
<pick>
    <onMessage partner="kml" operation="reply" container="klm-reply">
            fly-klm
    </onMessage>
    <onMessage partner="air-canada" operation="reply" container="air-canada-reply">
            fly-air-canada
    </onMessage>
    <onAlarm for="P3DT10H">
            time-out
    </onAlarm>
</pick>
```

- W języku BPE-calculus zapiszemy to w postaci:

  $akml\ ;\ Pklm + aair{-}canada\ ;\ Pair{-}canada + \tau ; Ptime{-}out$

Akcje złożone (switch)

- **Switch** jest kolejną konstrukcją języka BPEL będącą wyborem
- Instrukcja ta zawiera listę przypadków do których są przyporządkowane warunki logiczne wyliczane na podstawie wartości danych
- W języku BPE-calculus akcja switch będzie modelowana jako niedeterministyczny wybór, podobnie jak akcja pick
- Akcja pick zależy od zewnętrznego środowiska natomiast akcja switch nie zależy. Ze względu na tę różnicę akcja switch będzie oznaczana symbolem $\oplus$

Akcje złożone (switch - przykład)

```
<switch>
    <case condition="bpws:getContainerProperty(trip, distance) > 500">
        plane
    </case>
    <case condition="bpws:getContainerProperty(trip, distance) <= 500">
        train
    </case>
</switch>
```

- *W algebrze procesów powyższy fragment będzie wyglądał następująco:*

$$P_{plain} \oplus P_{train}$$

Akcje złożone (while)

<while condition="bpws:getContainerProperty(order, items) > 10">
get-item
</while>

- Warunek pętli nie jest reprezentowany w procesie BPE-calculus
- Uwzględniony jest jedynie fakt, że wnętrze pętli jest wykonywane pewną liczbę razy
- Decyzja zakończenia wykonania pętli jest niedeterministyczna
- Notacja w BPE-calculus:

$$P^*$$

(proces P odpowiada akcji get-item)

Akcje złożone (flow)

- **Flow** jest konstrukcją, która z punktu widzenia proponowanej algebry jest najbardziej interesująca, ponieważ definiuje współbieżność w języku BPEL
- Konstrukcja flow zawiera listę akcji, które są wykonywane równolegle
- W przyjętej notacji flow jest reprezentowany symbolem ‖

  getWeather ‖ *reserveTicket*

- *Synchronizacja pomiędzy akcjami konstrukcji flow jest realizowana za pomocą połączeń*

Połączenia (Links)

- Połączenia (links) są używane wewnątrz instrukcji flow do synchronizacji pomiędzy akcjami
- Połączenia w ramach procesu mają unikalne nazwy
- Każde połączenie ma akcję źródłową i akcję docelową
- W przypadku akcji źródłowej mówimy, że ma połączenie wychodzące (outgoing link)
- Każde połączenie wychodzące na warunek przejścia (transition condition), który jest wyrażeniem logicznym a jego wartość zależy od przetwarzanych danych (domyślną wartością jest true)
- Ponieważ w BPE-calculus nie uwzględnia się danych, więc warunek przejścia może przyjąć jedną z trzech wartości: true, false lub ?
- Notacja:

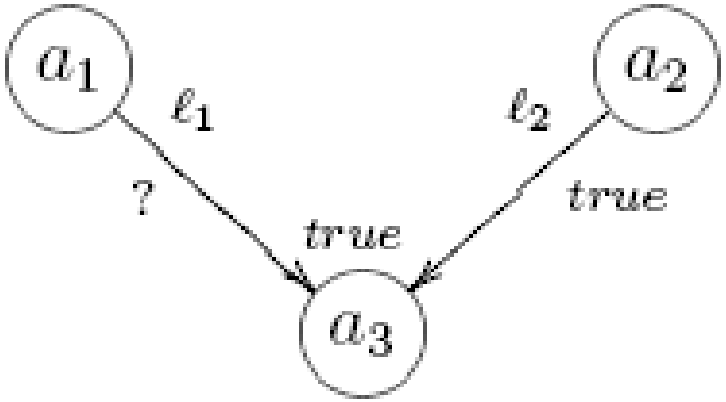
$$l \uparrow c$$

Warunki wejściowe (Join Condition)

- Jeśli akcja jest akcją docelową jakiegoś połączenia, to mówimy, że ma połączenie przychodzące (incoming link)
- Taka akcja posiada warunek wejściowy, który jest wyrażeniem logicznym złożonym z jej połączeń przychodzących i operatorów logicznych
- W języku BPE-calculus warunek wejściowy będzie reprezentowany za pomocą notacji postaci

$$c \Rightarrow P$$

Warunek wejściowy (przykład)



- Warunek przejścia połączenia l1 jest niedeterministyczny natomiast warunek przejścia połączenia l2 ma wartość true. Akcja a3 ma warunek wejściowy ustalony na true
- Jeśli jakieś połączenie przychodzące nie występuje w warunku wejściowym, to warunek wejściowy c zostaje zastąpiony warunkiem
- Przykład w języku BPE-calculus ma postać

$$l_1 \uparrow ? \, a_1 \parallel l_2 \uparrow true \, a_2 \parallel true \wedge (l_1 \vee \neg l_1) \wedge (l_2 \vee \neg l_2) \Rightarrow a_3$$

Składnia BPE-calculus

A_e – zbiór zewnętrznych akcji prostych
(akcje obserwowalne)

τ – wewnętrzna akcja prosta
(nieobserwowalna)

$\mathrm{A} = \mathrm{A}_e \cup \{\tau\}$ – zbiór wszystkich akcji prostych

L – niekończony zbiór połączeń

C – zbiór warunków wejściowych zdefiniowany następująco $c ::= true \mid l \mid \neg c \mid c \wedge c$
gdzie $l \in L$

Składnia BPE-calculus (cd.)

- Zbiór procesów BPE-calculus definiujemy następująco:

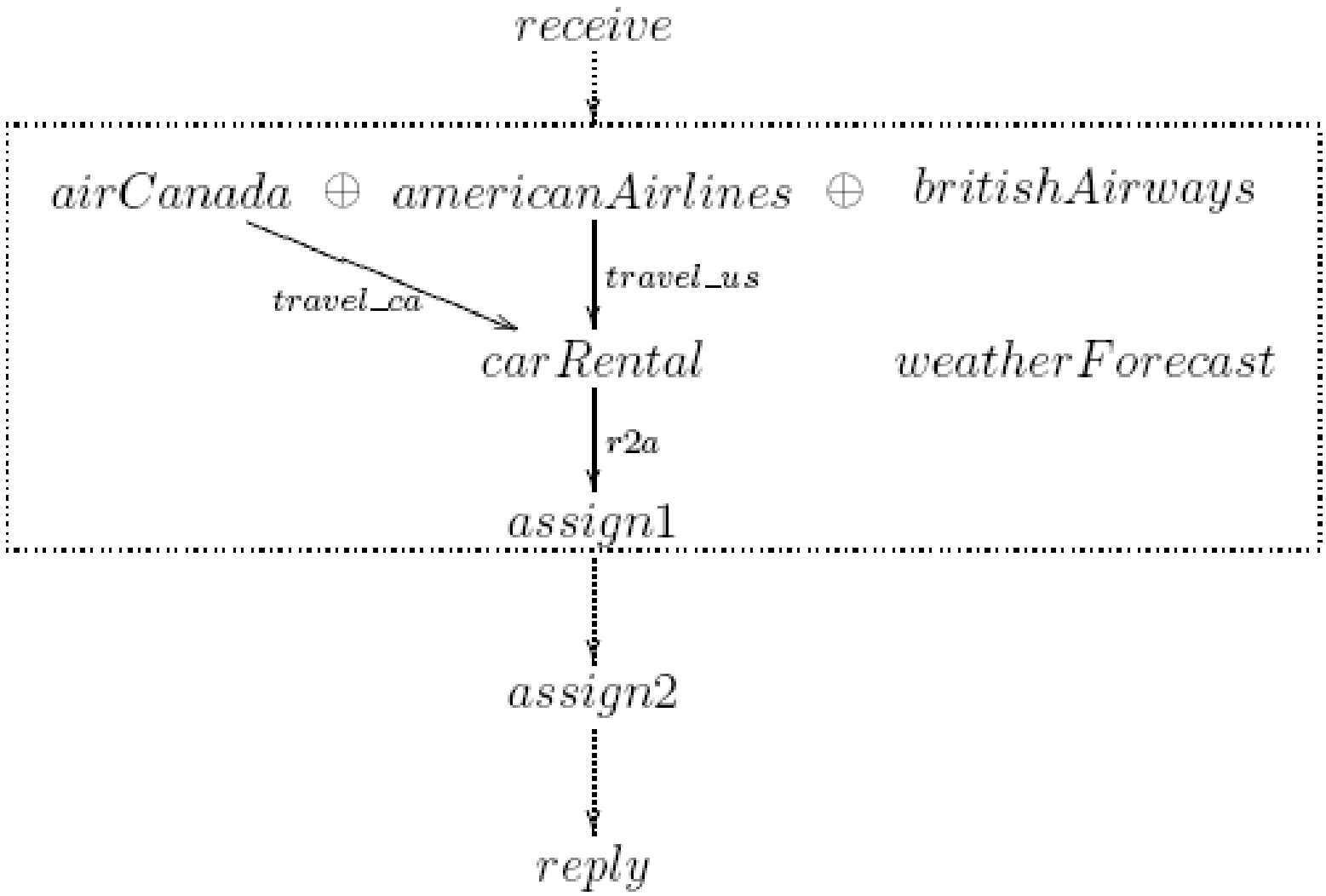
$$P ::= \alpha \mid end \mid (l \uparrow b)P \mid c \Rightarrow P \mid P* \mid P;P \mid P \| P \mid Q+Q \mid P \oplus P$$

$$Q ::= \alpha;P \mid Q+Q \mid$$

gdzie $\alpha \in A \quad l \in L \quad c \in C \quad b \in \{true, false, ?\}$

Widać, że otrzymana w rezultacie algebra procesów jest zdecydowanie prostsza niż BPEL4WS. Algebra odzwierciedla wszystkie aspekty języka związane z przepływem sterowania pomijając przy tym wiele dodatkowych, szczegółowych informacji

BPE-calculus (przykład)



BPE-calculus (przykład –cd.)

- Prostokąt pokazuje akcje współbieżne (flow). Sekwencje akcji są obrazowane za pomocą strzałek z linią przerywaną. Normalne strzałki pokazują połączenia (synchronizację) pomiędzy akcjami
- W języku BPE-calculus można wyrazić przykładowy proces następująco

$$receive\_request;$$

$$ca \uparrow true\ reserveAC \ \oplus \ us \uparrow ?\ reserveAA \ \oplus \ reserveBA$$

$$\| \ getWeatherForecast$$

$$\| \ ca \vee us \Rightarrow r2a \uparrow true \ \ rentCar$$

$$\| \ gr2a \Rightarrow \tau \ ;$$

$$replay\_request$$

BPE-calculus (typowanie)

- W języku BPEL każde połączenie (link) powinno mieć unikalne źródło i cel
- Połączenia nie mogą wykraczać poza pętlę while
- Powyższe ograniczenia zostały odzwierciedlone w prostym systemie typów
- Proces może być typowany, jeśli spełnia pewne zdane warunki odnośnie typu
- Typem procesu P nazwiemy parę uporządkowaną złożoną ze zbioru połączeń wchodzących i zbioru połączeń wychodzących tego procesu

BPE-calculus (definicja typu)

(ACT) $\alpha : (\emptyset, \emptyset)$

(END) $end : (\emptyset, \emptyset)$

(OUT) $$\frac{P : (I, O) \qquad l \notin O}{(l \uparrow b)P : (I, O \cup \{l\})}$$

(JOIN) $$\frac{P : (I, O)}{c \Rightarrow P : (I \cup links(c), O)}$$

(WHILE) $$\frac{P : (I, O) \qquad I = O}{P* : (I, O)}$$

BPE-calculus (definicja typu cd.)

(SEQ) $$\frac{P_1 : (I_1, O_1) \quad P_2 : (I_2, O_2) \quad I_1 \cap I_2 = \emptyset \quad O_1 \cap O_2 = \emptyset}{P_1 ; P_2 : (I_1 \cup I_2, O_1 \cup O_2)}$$

(FLOW) $$\frac{P_1 : (I_1, O_1) \quad P_2 : (I_2, O_2) \quad I_1 \cap I_2 = \emptyset \quad O_1 \cap O_2 = \emptyset}{P_1 \parallel P_2 : (I_1 \cup I_2, O_1 \cup O_2)}$$

(PICK) $$\frac{P_1 : (I_1, O_1) \quad P_2 : (I_2, O_2) \quad I_1 \cap I_2 = \emptyset \quad O_1 \cap O_2 = \emptyset}{P_1 + P_2 : (I_1 \cup I_2, O_1 \cup O_2)}$$

(SWITCH) $$\frac{P_1 : (I_1, O_1) \quad P_2 : (I_2, O_2) \quad I_1 \cap I_2 = \emptyset \quad O_1 \cap O_2 = \emptyset}{P_1 \oplus P_2 : (I_1 \cup I_2, O_1 \cup O_2)}$$

Semantyka BPE-calculus

- Semantyka języka BPE-calculus została określona za pomocą strukturalnej semantyki operacyjnej (SOS - Structured Operational Semantics)
- Semantyka procesu jest określana w terminach przejść (tranzycji) jakie proces może wykonać
- proces P wykonuje akcję i przechodzi do stanu P’

$$P \xrightarrow{\alpha} P'$$

Semantyka BPE-calculus c.d.

- Reguły SOS postaci

$$\frac{premises}{conclusion}(side\ condition)$$

określają etykietowany system przejść $< \Gamma, A, \rightarrow >$

gdzie Γ jest zbiorem konfiguracji

A to zbiór akcji prostych a

$\rightarrow \in \Gamma \times A \times \Gamma$ jest relacją przejścia

Semantyka BPE-calculus c.d.

- W czasie wykonywania się procesu BPE-calculus jego stany zależą od struktury procesu oraz wartości połączeń występujących w danym procesie. W celu określenia czym jest konfiguracja procesu
- Wprowadzimy pojęcie zbioru stanów połączeń $\Sigma = \mathrm{L} \rightarrow \{true, false, ?\}$
- oraz proces pusty (nil process) oznaczany przez 0.
- Zdefiniujemy teraz zbiór konfiguracji jako $\Gamma = P \times \Sigma$ gdzie P jest zbiorem procesów BPE wraz z procesem 0.

Reguły przejść języka BPE-calculus 1

(ACT) $$< \alpha, \sigma > \xrightarrow{\alpha} < 0, \sigma >$$

(OUT1) $$\frac{< P, \sigma > \xrightarrow{\alpha} < 0, \sigma' >}{< (l \uparrow true)P, \sigma > \xrightarrow{\alpha} < 0, \sigma'[true/l] >}$$

(OUT2) $$\frac{< P, \sigma > \xrightarrow{\alpha} < 0, \sigma' >}{< (l \uparrow false)P, \sigma > \xrightarrow{\alpha} < 0, \sigma'[false/l] >}$$

(OUT3) $$\frac{< P, \sigma > \xrightarrow{\alpha} < 0, \sigma' >}{< (l \uparrow ?)P, \sigma > \xrightarrow{\alpha} < 0, \sigma'[true/l] >} \quad \frac{< P, \sigma > \xrightarrow{\alpha} < 0, \sigma' >}{< (l \uparrow ?)P, \sigma > \xrightarrow{\alpha} < 0, \sigma'[false] >}$$

(OUT4) $$\frac{< P, \sigma > \xrightarrow{\alpha} < P', \sigma' >}{< (l \uparrow b)P, \sigma > \xrightarrow{\alpha} < (l \uparrow b)P', \sigma' >}$$

Reguły przejść języka BPE-calculus 2

(JOIN1) $$\frac{C(c)(\sigma) = true}{< c \Rightarrow P, \sigma > \xrightarrow{\tau} < P, \sigma >}$$

(JOIN2) $$\frac{C(c)(\sigma) = false \quad P:(I,O)}{< c \Rightarrow P, \sigma > \xrightarrow{\tau} < 0, \sigma[false/O] >}$$

(WHILE) $$< P*, \sigma > \xrightarrow{\tau} < 0, \sigma >$$ $$\frac{P:(I,O)}{< P*, \sigma > \xrightarrow{\tau} < P; P*, \sigma[?/O] >}$$

(SEQ) $$\frac{< P_1, \sigma > \xrightarrow{\alpha} < 0, \sigma' >}{< P_1; P_2, \sigma > \xrightarrow{\alpha} < P_2, \sigma' >}$$ $$\frac{< P_1, \sigma > \xrightarrow{\alpha} < P'_1, \sigma' > \quad P_1' \neq 0}{< P_1; P_2, \sigma > \xrightarrow{\alpha} < P_1'; P_2, \sigma' >}$$

(FLOW) – podobnie jak SEQ

Reguły przejść języka BPE-calculus 3

(PICK)

$$\frac{< P_1, \sigma > \xrightarrow{\alpha} < P'_1, \sigma' > \quad P_2 : (I_2, O_2)}{< P_1 + P_2, \sigma > \xrightarrow{\alpha} < P_1', \sigma'[\, false / O_2\,] >}$$

$$\frac{< P_2, \sigma > \xrightarrow{\alpha} < P'_2, \sigma' > \quad P_1 : (I_1, O_1)}{< P_1 + P_2, \sigma > \xrightarrow{\alpha} < P_2', \sigma'[\, false / O_1\,] >}$$

(SWITCH)

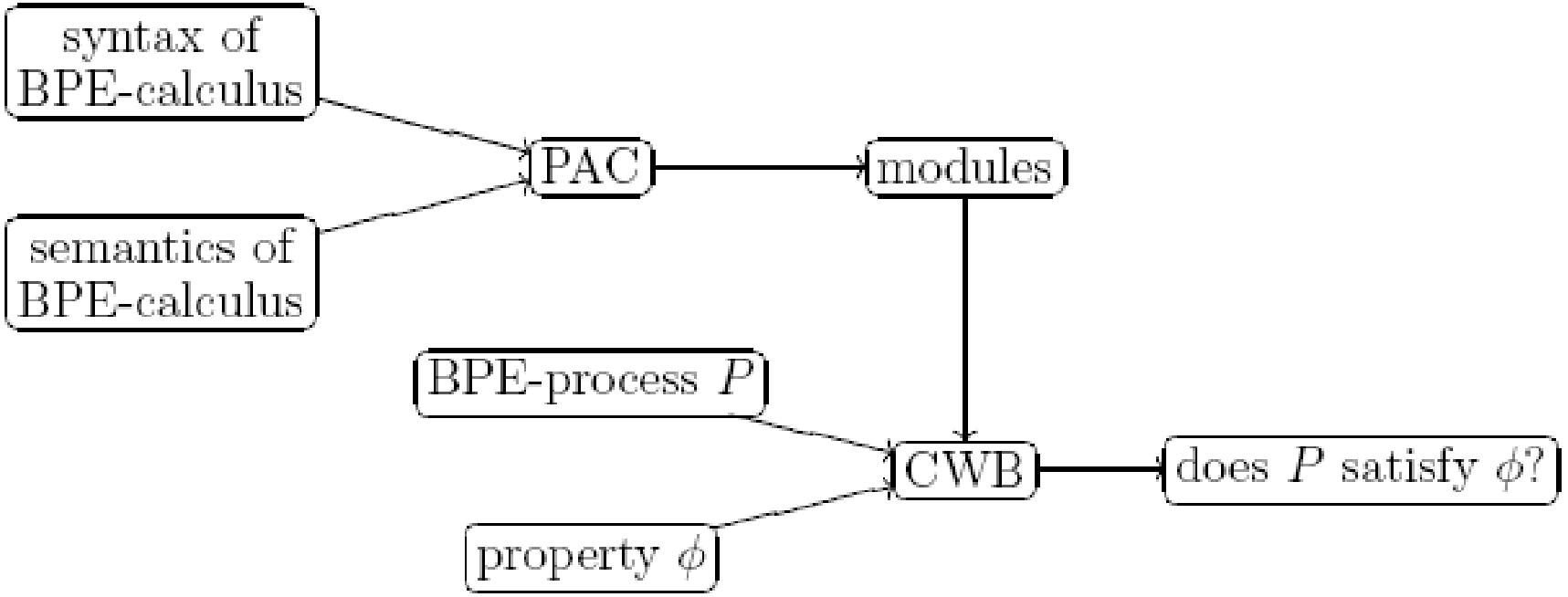
$$\frac{P_2 : (I_2, O_2)}{< P_1 \oplus P_2, \sigma > \xrightarrow{\tau} < P_1, \sigma'[\, false / O_2\,] >}$$

$$\frac{P_1 : (I_1, O_1)}{< P_1 \oplus P_2, \sigma > \xrightarrow{\tau} < P_2, \sigma'[\, false / O_1\,] >}$$

Process Algebra Compiler (PAC)

- PAC jest narzędziem pozwalającym rozszerzyć CWB o możliwość analizy procesów języka BPEL w zaproponowanej algebrze BPE-calculus
- PAC oczekuje na wejściu dwóch plików. Jeden jest opisem składni a drugi semantyki nowego języka
- Na wyjściu PAC generuje kod źródłowy w języku Standard ML
- Wygenerowany kod można włączyć do CWB. Od tego momentu CWB jest w stanie analizować procesy w specyfikacji BPE-calculus

Process Algebra Compiler (PAC)



Do zrobienia

- Przygotować pliki dla kompilatora PAC
- Skompilować CWB z nowymi modułami
- Przeprowadzić weryfikację procesów napisanych w BPE-calculus
- …

Dziękuję za uwagę

- Po zakończeniu prac materiały zostaną udostępnione na stronie

  http://www.mimuw.edu.pl/~gstatuch/