, ,

1 Wprowadzenie

W rozdziale tym wprowadzamy język do opisu systemów współbieżnych. Uwaga! Głównym celem jest wyrobienie odpowiedniej intuicji u Czytelnika, zatem nie podajemy tutaj *żadnych* ścisłych definicji. Tych znajdziecie Państwo aż nadto w kolejnych rozdziałach (o ile one zaistnieją :). Wszelkie wyjaśnienia przeprowadzamy za pomocą przykładów, chcąc odwołać się raczej do intuicji Czytelnika niż do Jego matematycznej biegłości.

Nasz język chcemy oprzeć na jak najmniejszej liczbie konstrukcji syntaktycznych i pojęć pierwotnych. Inymi słowy, chcemy aby był on jak najprostszy. Z drugiej strony, ma on być wystarczająco expresywny, aby było możliwe opisanie w nim systemów spotykanych w praktyce.

Systemy będziemy opisywać w sposób kompozycjonalny, tzn. system najczęściej będzie składał się z pewnej liczby mniejszych składowych. Zachowanie systemu składa się z niepodzielnych *zdarzeń*. Zdarzenie to

- albo komunikacja systemu z otoczeniem
- albo zdarzenie wewnętrze, zachodzące wewnątrz systemu, nieobserwowalne z zewnątrz.

W tym drugim przypadku, jest to znów komunikacja, tym razem pomiędzy składowymi systemu. Zatem, każde zdarzenie jest komunikacją! Komunikacja jest jedyną aktywnością procesów!

Z punktu widzenia semantyki, interesować nas będzie *obserwowalne zachowanie* procesu. Będziemy przez to rozumieć zdolność procesu do komunikacji z otoczeniem. Czyli obserwator zewnętrzy może obserwować dany system tylko komunikując się z nim.

Poczynimy pewną liczbę założeń. Po pierwsze, *ignorujemy czas*. Jeśli zatem chcemy opisać długotrwałą aktywność procesu, powinniśmy odróżnić co najmniej dwa zdarzenia: rozpoczęcie aktywności oraz zakończenie tej aktywności. Ponadto, dla uproszczenia nie będziemy dopuszczać jednoczesności (niezależności) zdarzeń. Zamiast jednoczesności, zdarzenia będą mogły być wykonane w dowolnej kolejności.

Po drugie, odrzucamy często spotykany podział na aktywne procesy obliczeniowe i pasywne media komunikacyjne. Medium komunikacyjne (np. bufor, pamięć dzielona, itp.) będzie też aktywnym procesem, pełnoprawnym uczestnikiem obliczeń. Ponadto, dla uproszczenia przyjmujemy iż w zdarzenie komunikacyjne zaangażowane są zawsze dokładnie dwa procesy. Wybieramy najprostszy model komunikacji:

synchroniczny, niebuforowany. Np. komunikację buforowaną można zaimplementować jako osobny proces.

Popatrzmy na kilka prostych przykładów, mających za cel wprowadzenie poszczególnych elementów naszego języka. Zacznijmy od bardzo prostego procesu C ktory naprzemienne odbiera wartość x i wysyła ją dalej:

$$C \stackrel{\text{def}}{=} in(x).C'(x) \tag{1}$$

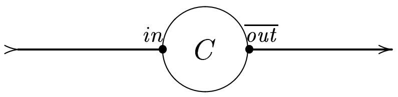
$$C'(x) \stackrel{\text{def}}{=} \overline{out}(x).C \tag{2}$$

albo krócej

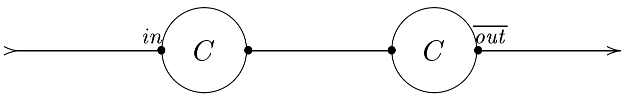
$$C \stackrel{\text{def}}{=} in(x).\overline{out}(x).C$$

Zwróćmy uwage na wiązanie identyfikatorów: $in(x)$ wiąże x w (1) oraz $C'(x)$ wiąże x w (2). Załóżmy tu, że x jest parametrem oznaczjącym dowolną liczbę całkowitą. Konstrukcja postaci $\alpha.P$, oznaczająca: wykonaj α, a następnie kontynuuj wg P, będzie jedynym sposobem szeregowania zdarzeń w naszym języku. Na przykład, proces C może wykonać $in(7)$ i stać się procesem $C'(7)$. Czyli ze *stanu* C, po wykonaniu $in(7)$, przechodzimy do stanu $C'(7)$. Taką zmianę stanu będziemy oznaczać $C \xrightarrow{in(7)} C'(7)$ i nazywać *zdarzeniem wykonanym przez* C.

Będziemy też posługiwać się językiem graficznym, odzwierciedlającym strukturę połączeń komunikacyjnych pomiędzy procesami:



Nazwy in i out możemy intuicyjnie uważać za nazwy portów obsługiwanych przez proces C; $\overline{in}$ jest nazwą *komplementarną* do in. Przyjmujemy chwilowo konwencję, iż porty wyjściowe mają nazwy z kreską u góry, a porty wejściowe nie. Wyobraźmy sobie przez chwile, iż mamy do dospozycji operator dwuargumentowy $\_\frown\_$ który zastosowany do dwóch kopii procesu C daje w wyniku $C\frown C$:



Dwa egzamplarze procesu C są teraz połączone kanałem komunikacyjnym. Zauważmy, iż nazwy portów przyłączonych do tego kanału są nieistotne a zatem nie są uwidocznione na rysunku. Niech $C^{(n)}$ oznacza $\overbrace{C\frown\ldots\frown C}^{n \text{ razy}}$. Zaniedbując wewnętrzną komunikację pomiędzy sąsiednimi procesami C, proces $C^{(n)}$ zachowuje się jak bufor

n-elementowy. Bufor taki można również opisać bezpośrednio, np. tak:

$$\begin{aligned}
\mathrm{Buff}_n(\epsilon) &\stackrel{\mathrm{def}}{=} in(x).\mathrm{Buff}_n(x)\\
\mathrm{Buff}_n(v_1 \dots v_n) &\stackrel{\mathrm{def}}{=} \overline{out}(v_n).\mathrm{Buff}_n(v_1 \dots n_{n-1})\\
\mathrm{Buff}_n(v_1 \dots v_k) &\stackrel{\mathrm{def}}{=} in(x).\mathrm{Buff}_n(xv_1 \dots v_k) + \overline{out}(v_k).\mathrm{Buff}_n(v_1 \dots v_{k-1}) \quad \text{o ile } 0<k<n
\end{aligned}$$

Operator $\_.\_$ ma tutaj wyższy priorytet niż $\_+\_$. Proces Buff_n posiada parametr, podobnie jak proces C' powyżej. Zawsze będziemy zakładać, iż parametry procesów należą do pewnego danego typu danych, w tym przypadku są to skończone listy liczb całkowitych.

Zauważmy, że proces $\mathrm{Buff}_n(\epsilon)$ wyraża tylko zewnętrzne zachowanie bufora, podczas gdy proces $C^{(n)}$ stanowi pewną implementację. Zatem, nasz język można uważać zarówno za język specyfikacji jak i programomowania. Myśląc ogólniej, będziemy w stanie opisać dany system na różnych poziomach uszczegółowienia, pomiędzy specyfikacją wymagań a końcową implementacją.

Czy procesy $C^{(n)}$ i $\mathrm{Buff}_n(\epsilon)$ są równoważne?

$$C^{(n)} = \mathrm{Buff}_n(\epsilon)?$$

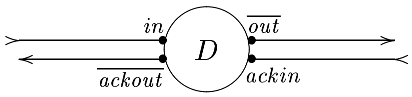
Aby umieć odpowiedzieć na to pytanie, musimy określić dokladniej o jaką równoważność nam chodzi. Zaproponowano wiele pojęć równoważności behawioralnej (obserwacyjnej) dla procesów współbieżnych. Z kilkoma z nich zapoznamy się poźniej, natomiast najważniejszą rolę odegra tzw. *równoważność bisymulacyjna.*

Załóżmy, iż mamy dane dwa systemy, np. $C^{(n)}$ i $\mathrm{Buff}_n(\epsilon)$. Wyobraźmy sobie taraz grę rozgrywaną przez dwóch graczy, Kowalskiego i Nowaka. Celem Kowalskiego jest wykazanie, iż dwa procesy są równowazne. Celem jego oponenta jest dowiedzenie, iż sie różnią. Konfiguracją (stanem) gry jest para stanów: stan procesu $C^{(n)}$ oraz stan procesu $\mathrm{Buff}_n(\epsilon)$. Gracze wykonują ruchy naprzemiennie. Najpierw Nowak wybiera dowolny z dwóch systemów i wykonuje w nim jedno z możliwych zdarzeń. Teraz następuje ruch Kowalskiego: musi on odpowiedzieć w drugim systemie za pomocą takiego samego zdarzenia. Rozgrywka toczy się dalej aż do momentu gdy jeden z graczy nie ma żadnego dozwolonego ruchu – w takiej sytuacji gracz ten przegrywa. W przeciwnym przypadku, tzn. w przypadku gdy rozgrywka sie nie kończy i jest nieskończona, wygrywa Kowalski, gdyż Nowakowi nie udało się odróżnić systemów. Mówimy, iż dwa systemy są równoważne, gdy Kowalski zawsze wygrywa, tzn. gdy ma strategię wygrywającą.[1]

Ćwiczenie 1 *Czy procesy $C^{(n)}$ i $\mathit{Buff}_n(\epsilon)$ są równoważne?*

[1] Wszelkie wątpliwości nasuwające się uważnemu Czytelnikowi w tym miejscu zostaną rozwiane poźniej.

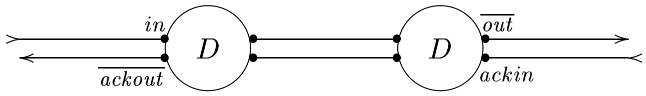
Rozważmy teraz pewną odmianę procesu C:



zdefiniowaną następująco:

$$D \stackrel{\text{def}}{=} in(x).\overline{out}(x).ackout.\overline{ackin}.D$$

Zdarzenia $\overline{ackin}$ i $ackout$ nie są sparametryzowane, stanowią więc one tylko synchronizację procesów. Operator $\_\frown\_$ modyfikujemy tak aby port $\overline{out}$ został połączony z portem in a port $\overline{ackin}$ z portem $ackout$. Czyli $D\frown D$ wygląda następująco:



Ćwiczenie 2 *Czy $D^{(n)} = Buff_n(\epsilon)$? Podpowiedź: rozważ $D\frown D$.*

Zmodyfikujmy D następująco:

$$D \stackrel{\text{def}}{=} in(x).\overline{ackin}.\overline{out}(x).ackout.D$$

Czy teraz $D^{(n)}$ zachowuje się znów jak bufor?

Ćwiczenie 3 *Zmodyfikuj definicje procesu $Buff_n$ tak, aby zachodziła równoważność $D^{(n)} = Buff_n$.*

Nasz język jest już wystarczająco bogaty aby opisać maszyny skończeniestanowe. Wyobraźmy sobie maszynę sprzedającą dwa rodzaje czekoladek: duże kosztujące 2zł i małe kosztujące 1zł. Posiada ona dwa otwory wrzutowe, jeden dla monet 1zł a drugi dla monet 2zł. Po wrzuceniu jednej monety, użytkownik wciska jeden z dwóch guzików: guzik `duża` jeśli chce kupić dużą czekoladkę a guzik `mała`, jeśli małą. Następnie powinien on wyciągnąć czekoladkę z podajnika – jego opróżnienie jest niezbędne dla wznowienia działania maszyny.

Tak proste zachowanie można z łatwością opisać następująco:

$$\text{Maszyna} \stackrel{\text{def}}{=} \texttt{2zł.duża.weź}.\text{Maszyna} + \texttt{1zł.mała.weź}.\text{Maszyna}$$

Ćwiczenie 4 *Zmodyfikuj proces* `Maszyna` *tak, aby po wrzuceniu monety 1zł było wciąż możliwe kupienie dużej czekoladki.*

Ćwiczenie 5 *Czy poprawnym rozwiązaniem poprzedniego zadania jest proces:*

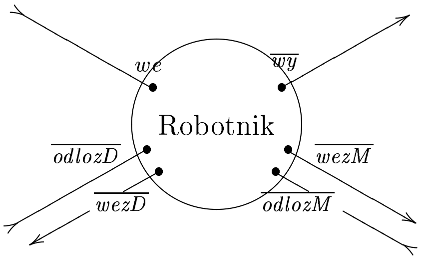
$$M \stackrel{\text{def}}{=} \texttt{2zł.duża.weź}.M + \texttt{1zł.mała.weź}.M + \texttt{1zł.1zł.duża.weź}.M?$$

Jeśli nie, to wskaż strategię Nowaka w odpowiedniej grze.

Ćwiczenie 6 *Zmodyfikuj proces* `Maszyna` *tak, aby maszyna pamiętała kwotę wrzuconą dotychczas, z przedziału 0zł ... 4zł. Maszyna nie musi wydawać reszty.*

Zwróćmy uwagę na rzecz oczywistą ale zarazem bardzo ważną: opisując maszyne, wzięliśmy pod uwagę tylko te aspekty, które nas interesowały. Inaczej mówiąc, abstrahowaliśmy od nieistotnych dla nas w danym momencie szczegółów. I tak będzie zawsze: w naszym języku będziemy w stanie wyraziż tylko pewien *model* systemu rzeczywistego.

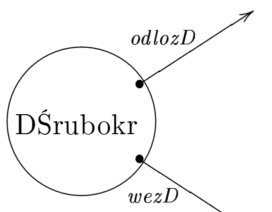
Ostatni przykład będzie miał za cel wprowadzenie wszystkich pozostałych konstrukcji syntaktycznych naszego języka. Opiszemy bardzo prostą linię produkcyją w pewnej fabryce, zatrudniającej dwóch robotników:



Zauważmy, iż graficzny zapis powyżej zawiera pełną informację o nazwach kanałów obsługiwanych przez process Robotnik. W przyszłości będziemy również używać zapisu postaci:

$$\text{Robotnik} : \{we, \overline{wy}, \overline{wezD}, \overline{odlozD}, \overline{wezM}, \overline{odlozM}\};$$

zbiór nazw po prawej stronie będziemy nazywać *rodzajem* procesu Robotnik. Każdy z robotników używa podczas pracy dwóch narzędzi: dużego i małego śrubokręta.

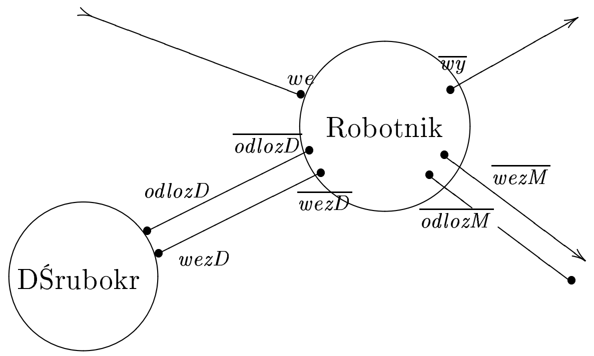


$$\text{DSrubokr} \stackrel{\text{def}}{=} wezD.odlozD.\text{DŚrubokr}$$

Zachowanie procesu Robotnik jest następujące:

$$
\begin{aligned}
\text{Robotnik} &\stackrel{\text{def}}{=} we(\text{część}).\text{Start}(\text{część})\\
\text{Start}(\text{część}) &\stackrel{\text{def}}{=} \textbf{if}\ \ \text{łatwa}(\text{część})\ \textbf{then}\ \ \text{Zakończ}(\text{część})\\
&\quad\ \textbf{else}\ \ \textbf{if}\ \ \text{trudna}(\text{część})\ \textbf{then}\ \ \text{UżyjDŚrub}(\text{część})\\
&\quad\ \textbf{else}\ \ (\text{UżyjDŚrub}(\text{część}) + \text{UżyjMŚrub}(\text{część}))\\
\text{UżyjDŚrub}(\text{część}) &\stackrel{\text{def}}{=} \overline{wezD}.\overline{odlozD}.\text{Zakończ}(\text{część})\\
\text{UżyjMŚrub}(\text{część}) &\stackrel{\text{def}}{=} \overline{wezM}.\overline{odlozM}.\text{Zakończ}(\text{część})\\
\text{Zakończ}(\text{część}) &\stackrel{\text{def}}{=} \overline{wy}(gotowa(\text{część})).\text{Robotnik}
\end{aligned}
$$

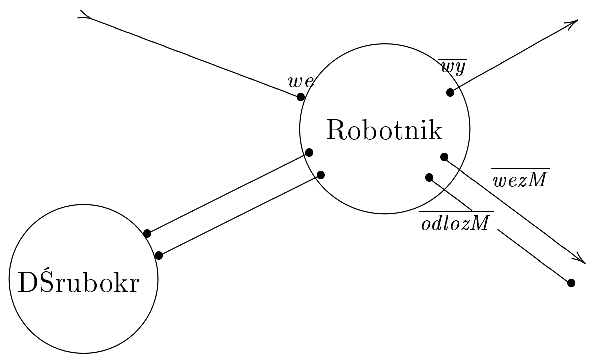
Robotnik komunikuje się z procesem Dśrubokr, zatem fragment naszego systemu będzie stanowiło *złożenie równoległe* procesów Robotnik i DSrubokr, które będziemy zapisywać Robotnik|DSrubokr. Graficznie:



Komplementarne porty obydwu procesów zostały połączone. Oznacza to, iż dwa procesy mogą wspólnie dokonać komunikacji (przypomnijmy, komunikacja jest synchroniczna). Tym niemniej, każdy z procesów może wciąż komunikować się przez te porty z innymi procesami lub ze światem otaczającym. Gdybyśmy chcieli tę dodatkową zdolność do komunikacji wykluczyć, możemy użyć kolejnego operatora. Proces

$$(\text{Robotnik}|\text{DSrubokr})\backslash\{wezD, odlozD\}$$

wygląda następująco:



jedyna różnica to wymazanie nazw portów ze zbioru $\{wezD, odlozD\}$ oraz nazw komplementarnych. Pamiętajmy jednak, iż opisujemy fabrykę zatrudniającą dwóch robotników korzystających z narzędzi. Rozważmy proces

$$((\text{Robotnik}|\text{DŚrubokr})\backslash\{wezD, odlozD\})|\text{Robotnik} \tag{3}$$

i zauważmy, iż drugi robotnik nie ma dostępu do dużego śrubokręta! Zatem poprawna kolejność składania systemu ze składowych jest następująca:

$$(\text{Robotnik}|\text{DŚrubokr}|\text{Robotnik})\backslash\{wezD, odlozD\} \tag{4}$$

a cały system może zostać zdefiniowany następująco:

$$\text{Fabryka} \stackrel{\text{def}}{=} (\text{Robotnik}|\text{DŚrubokr}|\text{MŚrubokr}|\text{Robotnik})\backslash\{wezD, odlozD, wezM, odlozM\} \tag{5}$$

Ćwiczenie 7 *Narysuj reprezentacje graficzne procesów (3), (4) oraz (5). Pamiętaj, że złożenie równoległe nie wymazuje nazw połączonych portów.*

Ćwiczenie 8 *Uzasadnij, że operator złożenia równoległego jest symetryczny i łączny, tzn. złożenie równoległe wielu procesów zachowuje się tak samo, niezależnie od kolejności składania.*

Ćwiczenie 9 *Podaj wszystkie możliwe wyrażenia inne niż (5) definiujące poprawnie fabrykę. Nie odróżniamy wyrażeń różniących się tylko kolejnością złożenia równoległego.*

Ostatnią konstrukcją syntaktyczną jest możliwośc przemianowania nazw portów. Np. proces MŚrubokr, którego definicję (rozmyślnie) dotychczas pominęliśmy, zachowuje się dokladnie tak samo jak DŚrubokr, aczkolwiek nazwy portów różnią się nieznacznie. Zdefiniujemy go następujaco:

$$\text{MŚrubokr} \stackrel{\text{def}}{=} \text{DŚrubokr}[wezM/wezD, odlozM/odlozD]$$

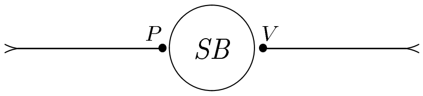
Ćwiczenie 10 *Rozważmy następujacą specyfikację fabryki:*

$$SpecFabr \stackrel{def}{=} Rob|Rob$$
$$Rob \stackrel{def}{=} in(część).\overline{out}(część).Rob$$

Czy proces Fabryka jest poprawną implementacją specyfikacji SpecFabr? Tzn., czy Fabryka = SpecFabr?

Ćwiczenie 11 *Rozważ zmodyfikowane zachowanie robotnika: odkłada on narzędzie dopieto po wysłaniu gotowej części. Zmodyfikuj odpowiednio opis fabryki. Czy wciąż zachodzi równoważność Fabryka = SpecFabr?*

Ćwiczenie 12 *Zdefiniuj semafor binarny:*



oraz semafor ogólny SO_n, dla $n \geq 1$. Czy zachodzi

$$SO_n = \overbrace{SB|\ldots|SB}^{n\ razy}\ ?$$

2 Język CCS

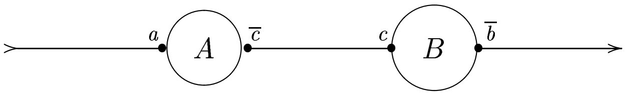
Czas na precyzyjną definicję składni i semantyki naszego języka.

Przyjmiemy następujące założenia i umowy notacyjne. Zakładamy, iż dany jest zbiór *nazw* $\mathcal{A}$; $a, b, c, \ldots \in \mathcal{A}$. $\overline{\mathcal{A}}$ zawiera ko-nazwy, czyli nazwy komplementarne, $\overline{a}, \overline{b}, \ldots \in \overline{\mathcal{A}}$. Zbiór *etykiet* $\mathcal{L}$ określimy jako $\mathcal{L} := \mathcal{A} \cup \overline{\mathcal{A}}$; $l, l', \overline{l}, \ldots \in \mathcal{L}$. Zakładamy iż $\overline{\overline{l}} = l$, dla każdej etykiety $l \in \mathcal{L}$. Wreszcie, zbiór *zdarzeń* Act to Act $:= \mathcal{L} \cup \{\tau\}$; $\alpha, \beta, \ldots \in$ Act. Zdarzeniu τ przypiszemy specjalne znaczenie. Będzie to *zdarzenie nieobserwowalne*, oznaczające aktywność wewnętrzną procesu, np. komunikację pomiędzy jego składowymi. Nie ma ono zdarzenia komplementarnego.

Semantykę języka określimy za pomocą etykietowanych tranzycji postaci P $\xrightarrow{\alpha}$ Q, oznaczających iż proces będący w stanie P może wykonać zdarzenie α i znaleźć się w stanie Q.

2.1 Przykłady

Często zdolność procesu P do wykonania zdarzenia będzie wynikać z takiej zdolności ktorejś ze składowych tego procesu. Rozważmy prosty przykład:



$$\text{A} \stackrel{\text{def}}{=} a.\text{A'} \qquad \text{B} \stackrel{\text{def}}{=} c.\text{B'}$$

$$\text{A'} \stackrel{\text{def}}{=} \overline{c}.\text{A} \qquad \text{B'} \stackrel{\text{def}}{=} \overline{b}.\text{B}$$

Ponieważ A $\xrightarrow{a}$ A', więc wnioskujemy, że A|B $\xrightarrow{a}$ A'|B. Podobnie, z A' $\xrightarrow{\overline{c}}$ A wnioskujemy

$$\text{A'|B} \xrightarrow{\overline{c}} \text{A|B}. \tag{6}$$

Uwaga! ta ostatnia tranzycja nie oznacza komunikacji pomiędzy A' i B, ponieważ jedynym uczestnikiem jest A' ! Wyraża ona jedynie zdolność procesu A'|B do komunikacji ze światem otaczającym przez port o nazwie $\overline{c}$. Natomiast komunikacja pomiędzy A' i B jest oczywiście możliwa: ponieważ A' $\xrightarrow{\overline{c}}$ A oraz B $\xrightarrow{c}$ B', mamy prawo wywnioskować

$$\text{A|B} \xrightarrow{?} \text{A'|B'}.$$

Jakie powinno być zdarzenie towarzyszące komunikacji pomiędzy A i B? Co powinien obserwować obserwator zewnętrzny? Ustalamy, iż obserwator zobaczy τ:

$$\text{A|B} \xrightarrow{\tau} \text{A'|B'}.$$

Gdybyśmy chcieli wykluczyć komunikację przez port $\overline{c}$ z otoczeniem, możemy użyć operatora zakazującego: proces (A|B)$\backslash c$ nie jest już w stanie wykonać zdarzenia $\overline{c}$. Ogólniej, z P $\xrightarrow{\alpha}$ P' potrafimy wywieść P$\backslash L \xrightarrow{\alpha} P'\backslash L$, dla $L \subseteq \mathcal{L}$, o ile $\alpha, \overline{\alpha} \notin L$. Wszystkie tranzycje, które potrafimy wywieść tworzą tzw. drzewo tranzycji (po lewej) albo graf tranzycji (po prawej):

$$
\begin{tikzcd}
 & (\mathrm{A|B})\backslash c \arrow[d, "a"] & \\
 & (\mathrm{A'|B})\backslash c \arrow[d, "\tau"] & \\
 & (\mathrm{A|B'})\backslash c \arrow[dl, "a"'] \arrow[dr, "\overline{b}"] & \\
(\mathrm{A'|B'})\backslash c \arrow[d, "\overline{b}"] & & (\mathrm{A|B})\backslash c \arrow[d, "a"] \\
(\mathrm{A'|B})\backslash c & & (\mathrm{A'|B})\backslash c \\
\ldots & & \ldots
\end{tikzcd}
\qquad
\begin{tikzcd}
 & (\mathrm{A|B'})\backslash c \arrow[dl, "a"'] \arrow[dr, "\overline{b}"] & \\
(\mathrm{A'|B'})\backslash c \arrow[dr, "\overline{b}"'] & & (\mathrm{A|B})\backslash c \arrow[dl, "a"] \\
 & (\mathrm{A'|B})\backslash c \arrow[uu, "\tau"] &
\end{tikzcd}
$$

Rozważając semantyczną równoważność procesów, nie będzie nas interesować zawartość wierzchołków tego grafu; np. równoważny procesowi (A|B')$\backslash c$ jest proces C_0 określony następująco:

$$
\begin{aligned}
\mathrm{C}_0 &\stackrel{\mathrm{def}}{=} \overline{b}.\mathrm{C}_1 + a.\mathrm{C}_2 \\
\mathrm{C}_1 &\stackrel{\mathrm{def}}{=} a.\mathrm{C}_3 \\
\mathrm{C}_2 &\stackrel{\mathrm{def}}{=} \overline{b}.\mathrm{C}_3 \\
\mathrm{C}_3 &\stackrel{\mathrm{def}}{=} \tau.\mathrm{C}_0
\end{aligned}
$$

gdyż grafy tranzycji procesów (A|B')$\backslash c$ i C_0 są izomorficzne. Albo (A|B)$\backslash c = a.\tau.C$, gdzie

$$\mathrm{C} \stackrel{\mathrm{def}}{=} \overline{b}.a.\tau.\mathrm{C} + a.\overline{b}.\tau.\mathrm{C}.$$

Czyli znaczenie procesu zależy tylko od jego grafu tranzycji.

Uważny Czytelnik mógłby spytać, czy jest sens przepisywać τ we wszystkich równaniach powyżej? Czy też można po prostu je skreślić, nie zmieniając znaczenia (zachowania) procesów? Okazuje się, iż wszędzie powyżej można skreślić τ tak, by wszystkie równoważności wspomniane w tekście pozostały prawdziwe. Jest tak dzięki prawu:

$$\alpha.\tau.P = \alpha.P.$$

Wynika z niego na przykład iż

$$(\mathrm{A}|\mathrm{B})\backslash c = a.\mathrm{D}, \tag{7}$$

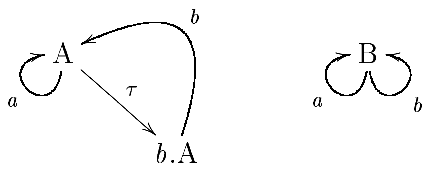
gdzie

$$\mathrm{D} \stackrel{\mathrm{def}}{=} \overline{b}.a.\mathrm{D} + a.\overline{b}.\mathrm{D}.$$

Może zatem τ można usunąć zawsze i jest w związku z tym niepotrzebne? Tak nie jest, a oto kontrprzykład. Rozważmy procesy A i B zdefiniowane tak:

$$\mathrm{A} \stackrel{\mathrm{def}}{=} a.\mathrm{A} + \tau.b.\mathrm{A}$$
$$\mathrm{B} \stackrel{\mathrm{def}}{=} a.\mathrm{B} + b.\mathrm{B}$$

Oto ich grafy tranzycji:



Ćwiczenie 13 *Czy $A = B$?*

Odpowiedź brzmi: nie, ponieważ proces A potrafi „po cichu” przejść do stanu b.A, w którym nie jest możliwe wykonanie a, a proces B nie potrafi znaleźć się w takim stanie.

2.2 Składnia i semantyka języka podstawowego

Najpierw zdefiniujemy *język podstawowy* $\mathcal{E}$, bez przekazywania informacji podczas komunikacji. Jest on prostszy a zatem lepiej przystosowany do rozważań teoretycznych. Następnie na jego podstawie zbudujemy *język pełny* $\mathcal{E}^+$, poprzez translację do języka podstawowego.

Definicja 1 *Zdefiniujemy teraz składnię wyrażeń procesowych $E \in \mathcal{E}$. Niech $K, L \subseteq \mathcal{L}$ oznaczają podzbiory etykiet. Przez funkcję przemianowującą f będziemy rozumieć $f : \mathrm{Act} \to \mathrm{Act}$ taką, że:*

$$f(\overline{l}) = \overline{f(l)}$$
$$f(\mathcal{L}) \subseteq \mathcal{L}$$
$$f(\tau) = \tau$$

Niech $\mathcal{X}$ *oznacza dany zbiór zmiennych;* $X, Y, \ldots \in \mathcal{X}$. *Niech* $\mathcal{K}$ *oznacza zbiór stałych;* $A.B, \ldots \in \mathcal{K}$. *Przez* $\{E_i : i \in I\}$ *będziemy rozumieć indeksowany zbiór wyrażeń, czyli po prostu funkcję ze zbioru* I *w zbiór wyrażeń. Zbiory indeksujące będziemy oznaczać literami* $I, J, \ldots$.

Składnia języka podstawowego $\mathcal{E}$ *jest następująca:*

$$
\begin{array}{lll}
E & ::= & X & \textit{zmienne} \\
 & & A & \textit{stałe} \\
 & & \alpha.E & \textit{prefiks} \\
 & & E|E & \textit{złożenie równoległe} \\
 & & \sum_{i\in I} E_i & \textit{wybór niedeterministyczny} \\
 & & E\backslash L & \textit{zakaz} \\
 & & E[f] & \textit{przemianowanie.}
\end{array}
$$

Procesami nazywamy wyrażenia procesowe bez zmiennych; będziemy je oznaczać $P, Q, \ldots$.

Teraz i w przyszłości zawsze będziemy zakładać, iż wszystkie stałe występujące w danym wyrażeniu są zdefiniowane za pomocą równości:

$$\mathrm{A} \stackrel{\mathrm{def}}{=} E.$$

Definicje te mogą być wzajemnie rekurencyjne, ale wszystkie stałe występujące po prawej stronie, w wyrażeniu E, muszą mieć definicje.

Podstawową nowością jest brak operatora $\_ + \_$; zamiast niego mamy znacznie ogólniejszy operator wyboru niedeterministycznego $\sum_{i\in \mathrm{I}} E_i$. Dla $\mathrm{I} = \{1, 2\}$, otrzymujemy $E_1 + E_2$. Ponadto dla $\mathrm{I} = \emptyset$ otrzymujemy proces niezdolny do wykonania żadnego zdarzenia, który będziemy oznaczać jako $0 \stackrel{\mathrm{def}}{=} \sum_{i\in\emptyset} E_i$.

Dla ułatwienia zapisu, ustalamy kolejność priorytetów poszczególnych operatorów jak następuje, w kolejności od najwyższego:

- zakaz $\_\backslash L$ i przemianowanie $\_[f]$
- prefiks $\alpha.\_$
- złożenie równoległe $\_|\_$
- suma (wybór)

Zamiast $E\backslash\{l\}$ będziemy pisać skrótowo $E\backslash l$. Semantykę języka $\mathcal{E}$ zadamy za pomocą kilku reguł:

$$\frac{}{\alpha.E \xrightarrow{\alpha} E} \qquad \frac{E_j \xrightarrow{\alpha} E'}{\sum_{i\in \mathrm{I}} E_i \xrightarrow{\alpha} E'}\ (j \in \mathrm{I}) \qquad \frac{(\mathrm{A} \overset{\mathrm{def}}{=} E) \qquad E \xrightarrow{\alpha} E'}{\mathrm{A} \xrightarrow{\alpha} E'}$$

$$\frac{E \xrightarrow{\alpha} E'}{E|F \xrightarrow{\alpha} E'|F} \qquad \frac{F \xrightarrow{\alpha} F'}{E|F \xrightarrow{\alpha} E|F'} \qquad \frac{E \xrightarrow{a} E' \qquad F \xrightarrow{\overline{a}} F'}{E|F \xrightarrow{\tau} E'|F'}$$

$$\frac{E \xrightarrow{\alpha} E'}{E\backslash L \xrightarrow{\alpha} E'\backslash L}\ (\alpha, \overline{\alpha} \notin L) \qquad \frac{E \xrightarrow{\alpha} E'}{E[f] \xrightarrow{f(\alpha)} E'[f]}$$

Ćwiczenie 14 *Zdefiniuj proces $C\frown C$ z poprzedniego rozdziału.*

Tranzycja $E \xrightarrow{\alpha} E'$ zachodzi o ile da sie ją wywieść za pomocą powyższych reguł. Na przykład, poniżej znajduje się wywód tranzycji $(a.E + b.0|\overline{a}.F)\backslash a \xrightarrow{\tau} (E|F)\backslash a$:

$$\begin{tikzcd}
a.E \xrightarrow{a} E \arrow[d, Rightarrow] & & \\
a.E + b.0 \xrightarrow{a} E \arrow[dr, Rightarrow] & & \overline{a}.F \xrightarrow{\overline{a}} F \arrow[dl, Rightarrow] \\
& (a.E + b.0|\overline{a}.F) \xrightarrow{\tau} E|F \arrow[d, Rightarrow] & \\
& (a.E + b.0|\overline{a}.F)\backslash a \xrightarrow{\tau} (E|F)\backslash a &
\end{tikzcd}$$

Taki zapis nazywać będziemy *drzewem wywodu*. Jest to pojęcie o tyle ważne, iż dostarcza indukcyjnej techniki dowodowej w sytuacji, gdy chcemy dowieść jakiejś własności wszystkich tranzycji. Ponieważ *wszystkich* oznacza tak naprawdę wszystkie te, które mają co najmniej jeden wywód, możemy poprowadzić indukcję względem rozmiaru drzewa wywodu.

Ćwiczenie 15 *Czy każda tranzycja ma co najwyżej jedno drzewo wywodu?*

Gdy rozważamy wszystkie zdarzenia które może wykonać dany proces P, otrzymujemy pewną liczbę *bezpośrednich potomków*, tzn. procesów P' takich, że dla pewnego α zachodzi $P \xrightarrow{\alpha} P'$. Jeśli następnie będziemy kontynuować, tzn. rozważymy wszystkich bezpośrednich potomków bezpośrednich potomków procesu P, itd., otrzymamy drzewo potomków procesu P, ewentualne graf potomków. Np. drzewo potomków rozważanego wcześniej przykładowego procesu $(a.E + b.0|\overline{a}.F)\backslash a$

rozpoczyna się następująco:

$$
\begin{tikzcd}[row sep=small]
 & (E|F)\backslash a \quad \cdots \\
(a.E + b.0|\overline{a}.F)\backslash a \arrow[ur, "\tau"] \arrow[dr, "b"'] & \\
 & (0|\overline{a}.F)\backslash a
\end{tikzcd}
$$

Drzewo potomków będzie miało fundamentalne znaczenie gdy zajmiemy się porównywaniem zachowania procesów. Jak już wspomnieliśmy, znaczenie procesu powinno zależeć tylko od jego drzewa potomków, ale gdy wymaże się z niego wyrażenia procesowe znajdujące się w węzłach – nie powinno ono zupełnie zależeć od tych wyrażeń. Zastanówmy się na przykład, kiedy dwa poniższe procesy powinny być uważane za równoważne, czyli semantycznie równe:

$$
\begin{tikzcd}[row sep=small]
 & P_1 & & & Q_1 \\
P \arrow[ur, "\alpha_1"] \arrow[r, "\alpha_2"] \arrow[ddr, "\alpha_n"'] & P_2 & & Q \arrow[ur, "\alpha_1"] \arrow[r, "\alpha_2"] \arrow[ddr, "\alpha_n"'] & P_Q \\
 & \vdots & & & \vdots \\
 & P_n & & & Q_n
\end{tikzcd}
$$

Zauważmy, iż założyliśmy tutaj dla uproszczenia iż procesy mają skończoną liczbę bezpośrednich potomków oraz te same zdarzenia są możliwe w obydwu procesach, a w dodatku w tej samej liczbie. Pierwsza propozycja mogłaby być taka: są równoważne o ile $P_i \equiv Q_i$, dla wszystkich i. Symbolem $\equiv$ oznaczamy tutaj *syntaktyczną równość*, czyli identycznośc wyrażeń. Propozycja ta nie jest jednak zbyt udana, w szczególności nie spełnia wymagania postawionego przed momentem. Zatem moglibyśmy ją zmodyfikować, zastępując równośc syntaktyczną przez równośc semantyczną: P i Q są semantycznie równe jeśli P_i i Q_i są semantycznie równe, dla wszystkich i. Czyli zredukowaliśmy równość procesów do równości ich bezpośrednich potomków. Niestety, nie jest to definicja indukcyjna, gdyż nie zawsze ciąg redukcji kończy się. Tym niemniej, definicja tego typu ma dobrze określony sens matematyczny. Jest to definicja *koindukcyjna*, oparta na największym punkcie stałym pewnego operatora zamiast na najmniejszym. Podstawowe pojęcie naszego wykładu, które poznamy wkrótce, *równoważność bisymulacyjna*, będzie zdefiniowane właśnie koindukcyjnie.

Ćwiczenie 16 *Zapisz w języku podstawowym jedno z rozwiązań problemu pięciu filozofów.*

Ćwiczenie 17 *Jak zdefiniować operator _ ; _ złożenia sekwencyjnego w naszym języku. (Wskazówka: Procesy P i Q uszeregowane sekwencyjnie w $P;Q$ powinny komunikować się przez „prywatny” kanał komunikacyjny.)*

2.3 Język pełny

Teraz zajmiemy się językiem pełnym, zawierającym wszystkie konstrukcje językowe których używaliśmy w rozdziale 1.

Definicja 2 *Składnia języka pełnego $\mathcal{E}^+$ jest następująca:*

$$
\begin{array}{lll}
E & ::= & X & \textit{zmienne} \\
 & & A(e_1, \ldots, e_k) & \textbf{stałe} \\
 & & a(x).E & \textbf{prefiks wejściowy} \\
 & & \overline{a}(e).E & \textbf{prefiks wyjściowy} \\
 & & \tau.E & \tau\textit{-prefiks} \\
 & & \textbf{if } \ b \textbf{ then } \ E & \textbf{wyrażenie warunkowe} \\
 & & E|E & \textit{złożenie równoległe} \\
 & & \sum_{i \in I} E_i & \textit{wybór niedeterministyczny} \\
 & & E \backslash L & \textit{zakaz} \\
 & & E[f] & \textit{przemianowanie.}
\end{array}
$$

Nowe lub rozszerzone konstrukcje językowe zostały wytłuszczone i teraz je omówimy. Po pierwsze, zakładamy, iż wszystkie stałe są wyposażone w *arność*, determinującą liczbę parametrów. Parametry te należą do ustalonego typu danych, $\mathcal{V}$. W wyrażeniu $\mathrm{A}(e_1, \ldots, e_k)$, wyrażenia $e_1, \ldots, e_k$ są zbudowane z operacji dostarczanych przez $\mathcal{V}$. Oczywiście definicje stałych powinny mieć też nieznacznie ogólniejszą postać niż w $\mathcal{E}$:

$$\mathrm{A}(x_1, \ldots, x_k) \stackrel{\text{def}}{=} E, \tag{8}$$

gdzie zmienne $x_1, \ldots, x_k$ mogą pojawiać się w E. Należy odróżnić zmienne $x_1, \ldots, x_k$ od zmiennych procesowych $X \in \mathrm{X}$. Nazwijmy je zatem *zmiennymi wartości*. Definicja (8) wiąże zmienną x a jej zasięgiem jest E.

Wyróżniamy prefiks wejściowy $a(x).E$, $a \in \mathcal{A}$, odbierający wartośc typu $\mathcal{V}$ i przypisujący ją na zmienną x w wyrażeniu procesowym E. Prefiks wejściowy jest drugą i ostatnią konstrukcją wiążącą zmienną x, a jej zasięgiem jest znów E. Wyróżniamy ponadto prefiks wyjściowy $\overline{a}(e).E$, wysyłający wartość wyrażenia e. Nową konstrukcją jest wyrażenie warunkowe **if** b **then** E, które w zależności od ewaluacji wyrażenia logicznego b równe jest albo wyrażeniu procesowemu E albo 0.

Ćwiczenie 18 *Dlaczego w języku nie ma ogólnego wyrażenia warunkowego*

$$\textbf{if } \ b \textbf{ then } \ E_1 \textbf{ else } \ E_2?$$

Czy można je wyrazić za pomocą konstrukcji obecnych w $\mathcal{E}^+$?

Semantyka języka $\mathcal{E}^+$ powstanie poprzez translację do języka podstawowego. Dla $E \in \mathcal{E}^+$, niech $\widehat{E} \in \mathcal{E}$ oznacza odpowiadające mu wyrażenie procesowe w $\mathcal{E}$. Translacja wygląda następująco (zakładamy, iż w wyrażeniu translowanym wszystkie zmienne wartości są związane):

F	$\longmapsto$	$\widehat{F}$
$\mathrm{A}(e_1, \ldots, e_k)$		$\mathrm{A}_{e_1,\ldots,e_k}$
$a(x).E$		$\sum_{v\in\mathcal{V}} a_v.\widehat{E\{v/x\}}$
$\overline{a}(e).E$		$\overline{a}_e.\widehat{E}$
$\tau.E$		$\tau.\widehat{E}$
$E_1 \vert E_2$		$\widehat{E_1}\vert\widehat{E_2}$
$\sum_{i\in I} E_i$		$\sum_{i\in I}\widehat{E_i}$
$E\backslash L$		$\widehat{E}\backslash\{l_v : l \in L, v \in \mathcal{V}\}$
$E[f]$		$\widehat{E}[\widehat{f}]$, gdzie $\widehat{f}(l_v) = f(l)_v$
if b **then** E		$\begin{cases} \widehat{E} & \text{o ile } b \text{ jest prawdziwe} \\ 0 & \text{w.p.p.} \end{cases}$
X		X

Podstawowa sztuczka, która tutaj się dzieje to pozbycie się parametrów w stałych procesowych $\mathrm{A}(e_1, \ldots, e_k)$ poprzez wkomponowanie wartości wyrażeń $e_1, \ldots, e_k$ w nazwę stałej. Czyli, wyrażenia $\widehat{E}$ odwołuja się do stałych postaci $\mathrm{A}_{v_1,\ldots,v_k}$, gdzie $v_1, \ldots, v_k \in \mathcal{V}$ a k jest arnością stałej A w $\mathcal{E}^+$. Zauważmy, iż wyrażenia $e_1, \ldots, e_k$ nie mogą zawierać zmiennych, gdyż założyliśmy, iż argument translacji $\widehat{\_}$ ma wszystkie zmienne wartości związane. Oznacza to, że można zawsze wyliczyć wartości tych wyrażeń gdy to jest potrzebne. Co pozwala nam, dla uproszczenia, nie odróżniać wyrażenia e_i od jego wartości:

$$\mathrm{A}(e_1, \ldots, e_k) \quad \longmapsto \quad \mathrm{A}_{\text{wartość}(e_1),\ldots,\text{wartość}(e_k)}.$$

Podobna sztuczka odbywa się również dla prefiksu wyjściowego $\overline{a}(e).E$. Nazwy portów używane przez wyrażenia $\widehat{E}$ są postaci a_v, gdzie a jest nazwą z języka $\mathcal{E}$ a $v \in \mathcal{V}$.

Ćwiczenie 19 *Rozważ następujący prościutki język programowania współbieżnego:*

$$
\begin{array}{lll}
I & ::= & X := E & \textit{przypisanie} \\
 & & \textbf{skip} & \textit{nic nie rób} \\
 & & I_1; I_2 & \textit{złożenie sekwencyjne} \\
 & & \textbf{if}\ \ E\ \textbf{then}\ \ I_1\ \textbf{else}\ \ I_2 & \textit{instrukcja warunkowa} \\
 & & \textbf{begin}\ \ D; I\ \ \textbf{end} & \textit{blok} \\
 & & I_1\ \ \textbf{par}\ \ I_2 & \textit{złożenie równoległe}
\end{array}
$$

$$
D \quad ::= \quad \textbf{var}\ \ X \qquad\qquad \textit{deklaracja zmiennej}
$$

$$
\begin{array}{llll}
E & ::= & X & \textit{wystąpienie zmiennej} \\
 & & F(E_1, \ldots, E_K) & \textit{aplikacja funkcji}
\end{array}
$$

Zdefiniuj jego semantykę poprzez translację do języka pełnego. (Jednym z pierwszych kroków jest rozwiązanie ćwiczenia 17).

Ćwiczenie 20 *Rozszerz język programowania o deklarację procedury* **proc** $X(Y)$ *i instrukcję wywołania procedury* **call** $X(E)$ *(przekazywanie parametru przez wartość).*

2.4 Stałe procesowe, rekurencja a punkty stałe

Wzajemnie rekurencyjne definicje stałych to jedyny mechanizm w naszym języku pozwalający opisać procesy o nieskończonym działaniu. Taki sposób definiowania nieskończonych zachowań jest stosunkowo łatwy w użyciu i czytelny dla użytkownika. Natomiast ma pewną wadę: definicje stałych pojawiających się w jakimś wyrażeniu nie są składową tego wyrażenia, lecz obiektem zewnętrznym. Równoważnie, można umieścić w pewien sposób definicje używanych stałych procesowych wewnątrz wyrażeń procesowych, np. zamiast $0 + b.\mathrm{A}$, gdzie A ma definicję: $\mathrm{A} \stackrel{\mathrm{def}}{=} a.\mathrm{A}$, możemy napisać $0 + b.\mathrm{fix}(X = a.X)$. W tym celu wprowadzamy do języka dodatkowa konstrukcję:

$$E \ ::= \ \ldots \quad | \quad \mathrm{fix}_j(\{X_i = E_i : i \in I\})$$

Zbiór $\{X_i = E_i : i \in \mathrm{I}\}$ zawiera równania definiujące rodzinę procesów indeksowaną zbiorem I. Wyrażenie $\mathrm{fix}_j(\{X_i = E_i : i \in I\})$ oznacza element tego zbioru odpowiadający indeksowi $j \in \mathrm{I}$. (Uwaga: po raz pierwszy potrzebujemy tutaj zmiennych procesowych.) W skrócie, zamiast $\mathrm{fix}_j(\{X_i = E_i : i \in I\})$ będziemy pisać $\mathrm{fix}_j(\{\widetilde{X} = \widetilde{E}\})$, nie wymieniając zbioru I explicite. Dadatkowo, niech $F\{\widetilde{E}/\widetilde{X}\}$

oznacza wyrażenie procesowe F, w którym równolegle podstawiono wyrażenie E_i za zmienną X_i, dla wszystkich $i \in \mathrm{I}$.

Oto reguła dla operatora fix, pokazująca iż jest to w rzeczywistości operator najmniejszego punktu stałego:

$$\frac{E_j\{\widetilde{\mathrm{fix}}(\widetilde{X} = \widetilde{E})/\widetilde{X}\} \xrightarrow{\alpha} E'}{\mathrm{fix}_j(\{\widetilde{X} = \widetilde{E}\}) \xrightarrow{\alpha} E'}$$

$\widetilde{\mathrm{fix}}(\widetilde{X} = \widetilde{E})$ oznacza rodzinę $\{\mathrm{fix}_j((\widetilde{X} = \widetilde{E}) : j \in \mathrm{I}\}$. Operator fix ma również pewną poważną wadę: wyrażenia są znacznie mniej czytelne, jak również powyższa reguła wygląda mniej przyjaźnie. Dlatego najczęściej pozostaniemy przy stałych procesowych.

Ćwiczenie 21 *Rozważ proces $A|B$ zadany jak następuje:*

$$A \stackrel{def}{=} \overline{a}.0 + A|B \qquad\qquad B \stackrel{def}{=} (a.B + \overline{b}.A + a.0)\backslash a$$

Podaj równoważne mu wyrażenie procesowe E z uzyciem operatora fix *zamiast stałych. Znajdź wywód dla tranzycji $E \xrightarrow{\tau} 0|0$.*

Ćwiczenie 22 *Zastanów się, czy stałe procesowe oraz operator* fix *są wzajemnie równoważne.*

3 Prawa równościowe

Zanim formalnie zdefiniujemy pojęcie semantycznej równości procesów, co nastąpi w następnym rozdziale, teraz rozważymy pewne równości zachodzące pomiędzy procesami. Równości te odnoszą się do semantycznej równości procesów $=$ wspomnianej w poprzednim rozdziale. Możemy zatem sobie wyobrazić *algebrę procesów*. Jej elementami są procesy (wyrażenia procesowe bez zmiennych) modulo równość semantyczna. Tak więc, semantycznie, proces to nie jest wyrażenie, ale cała klasa abstrakcji równoważnych wyrażeń. Operacjami w algebrze procesów są znane nam już operacje do budowania procesów: prefiks, suma (wybór), przemianowanie, zakaz oraz złożenie równoległe. Aby to wszystko miało sens, równość semantyczna musi być kongruencją względem operacji w języku (podstawowym), tzn. musi ona być zachowana przez te operacje. Dowiedziemy tego w którymś z następnych rozdziałów. Dzięki temu, można w dowolnym wyrażeniu podmienić proces na inny, o ile jest on równoważny. Dodatkowo, pewne równania mają jednoznaczne rozwiązanie w algebrze procesów (patrz rozdział 3.2).

Wyróżnimy 3 grupy równości:

- *równości statyczne*, dotyczące operatorów statycznych, do których zaliczymy $\_|\_$, $\_\backslash L$ oraz $\_[f]$;
- *równości dynamiczne*, dotyczące operatorów dynamicznych: $\_ + \_$, $\alpha.\_$ oraz rekurencji;
- *prawo ekspansji*, łączące dwie grupy operatorów.

Operatory statyczne to te konstrukcje językowe, które nie znikają podczas tranzycji. Determinują one zatem statyczną strukturę systemu. Ogólnie, posiadają one tylko reguły następującej postaci:

$$\frac{E_{i_1} \xrightarrow{\alpha_1} E'_{i_1} \quad \ldots \quad E_{i_m} \xrightarrow{\alpha_m} E'_{i_m}}{op(E_1, \ldots, E_n) \xrightarrow{\alpha} op(E'_1, \ldots, E'_n)}$$

gdzie

- $\{i_1, \ldots, i_m\} \subseteq \{1, \ldots, n\}$;
- $i_j \neq i_k$, gdy $j \neq k$;
- $E'_i \equiv E_i$, gdy $i \notin \{i_1, \ldots, i_m\}$.

3.1 Równości dynamiczne

Procesy wraz z operatorem binarnego wyboru stanowią idempotentny monoid przemienny, którego elementem neutralnym jest 0:

$$\begin{array}{lrcl}
(+1) & P+Q & = & Q+P \\
(+2) & P+(Q+R) & = & (P+Q)+R \\
(+3) & P+P & = & P \\
(+4) & P+0 & = & P
\end{array}$$

Dla uzasadnienia wystarczy przypomnieć sobie zgrubne przybliżenie równości semantycznej opisane na końcu rozdziału 2.2. Ponieważ $P+Q \xrightarrow{\alpha} P'$ wtw. gdy $Q+P \xrightarrow{\alpha} P'$, więc $P+Q=Q+P$. Podobnie pozostałe równości.

Ćwiczenie 23 *Zaproponuj uogólnienie praw* (+1) – (+4) *dla sum nieskończonych.*

Drugą grupę równości stanowią prawa dotyczące τ:

$$\begin{array}{lrcl}
(\tau_1) & \alpha.\tau.P & = & \alpha.P \\
(\tau_2) & P+\tau.P & = & \tau.P \\
(\tau_3) & \alpha.(P+\tau.Q)+\alpha.Q & = & \alpha.(P+\tau.Q)
\end{array}$$

Uzasadnienie dla (τ_2) i (τ_3) jest podobne jak poprzednio, z tą różnicą, iż musimy wziąć pod uwagę nieobserwowalność zdarzenia τ. Zatem, zamiast relacji $\xrightarrow{\alpha}$ rozważmy relację $\stackrel{\alpha}{\Longrightarrow}$ określoną następująco:

$$\stackrel{\alpha}{\Longrightarrow} \; = \; (\xrightarrow{\tau})^* \circ \xrightarrow{\alpha} \circ (\xrightarrow{\tau})^*, \tag{9}$$

przy czym $(\xrightarrow{\tau})^*$ oznacza zwrotno-tranzytywne domknięcie relacji $\xrightarrow{\tau}$. Czyli zdarzenie α może być poprzedzone przez dowolną liczbę zdarzeń τ (być może 0) a także dowolna liczba τ może nastąpic po α. Tranzycje $P \stackrel{\alpha}{\Longrightarrow} Q$ będziemy nazywać *słabymi tranzycjami* a proces Q *słabym bezpośrednim potomkiem* procesu P.

Wracając do uzasadnienia (τ_2), mamy $P+\tau.P \stackrel{\alpha}{\Longrightarrow} Q$ wtw gdy $\tau.P \stackrel{\alpha}{\Longrightarrow} Q$. Równość (τ_3) można uzasadnić podobnie.

Oto kilka nierówności, które nie będą zachodzić w ogólności:

$$\tau.P \neq P,$$

czyli τ nie zawsze może być wymazane. W naszej algebrze procesów nie ma również rozdzielności prefiksu względem sumy:

$$\alpha.(P+Q) \;\neq\; \alpha.P+\alpha.Q$$

Ćwiczenie 24 *Pokaż*

$$\begin{aligned} P+\tau.(P+Q) &= \tau.(P+Q) \\ \alpha.(P+\tau.\tau.P) &= \alpha.P, \end{aligned}$$

korzystając z (τ_1) - (τ_3).

Ćwiczenie 25 *Pokaż* $\tau.(P+\alpha.(Q+\tau.R)) \;=\; \tau.(P+\alpha.(Q+\tau.R))+\alpha.R$.

Ćwiczenie 26 *Ogólniej niż w ćwiczeniu 25: pokaż że jeśli* $P \overset{\alpha}{\Longrightarrow} P'$, *to* $P = P+\alpha.P'$, *dla dowolnego procesu* P *zbudowanego wyłącznie z operatorów dynamicznych.*

3.2 Jednoznaczność rozwiązań układów równań

Rozważmy stałą procesową zdefiniowaną przez

$$\mathrm{A} \stackrel{\mathrm{def}}{=} E\{\mathrm{A}/X\};$$

po prawej stronie mamy wyrażenie procesowe E, w którym za wszystkie wystąpienia zmiennej X podstawiono A. Pomyślmy o A jako o rozwiązaniu równania:

$$X \;=\; E \tag{10}$$

Byłoby idealnie, gdyby równanie to miało dokładnie jedno rozwiązanie, właśnie A. Oczywiście, dokładnie jedno *modulo równość semantyczna* $=$. Czyli chcielibyśmy żeby zachodziło:

- $\mathrm{A} = E\{\mathrm{A}/X\}$ (proces A jest rozwiązaniem równania (10)); oraz
- Jeśli $P = E\{P/X\}$ oraz $Q = E\{Q/X\}$, to $P = Q$ (proces A jest *jedynym* rozwiązaniem).

Po co nam jednoznaczność? Przypomnijmy sobie równość (7) z rozdziału 2.1:

$$(\mathrm{A}|\mathrm{B})\backslash c \;=\; a.\mathrm{D},$$

gdzie stała D określona jest następująco:

$$\mathrm{D} \stackrel{\mathrm{def}}{=} \overline{b}.a.\mathrm{D}+a.\overline{b}.\mathrm{D}.$$

Mamy prawo wywnioskować równość (7) ponieważ (A|B)$\backslash c = a.$(A'|B)$\backslash c$ oraz proces (A'|B)$\backslash c$ jest rozwiązaniem równania

$$X \;=\; \overline{b}.a.X + a.\overline{b}.X. \tag{11}$$

I tutaj potrzebna jest jednoznaczność, aby wywnioskować D = (A'|B)$\backslash c$!

Definicja 3 *Zmienna X jest* <u>*sekwencyjna*</u> *w wyrażeniu E jeśli X występuje w E tylko w podwyrażeniach zbudowanych włącznie z prefiksu i sumy. Zmienna X jest* <u>*strzeżona*</u> *w wyrażeniu E jeśli każde wystąpienie X w E jest wewnątrz pewnego podwyrażenia $l.E'$ w E.*

Przykład 4 *Aby wyjaśnić niejasności, popatrzmy na kilka wyrażeń:*

- *X jest strzeżona ale nie sekwencyjna w $a.X|b.0$,*
- *X jest sekwencyjna ale nie strzeżona w $\tau.X + b.0|c.Y$ oraz w $X + c.Y$,*
- *Y jest i sekwencyjna i strzeżona w $X + c.Y$.*

Poniższe twierdzenie udowodnimy później:

Twierdzenie 5 (jednoznaczność rozwiązań)

- *Jeśli $A \overset{def}{=} E\{A/X\}$ to $A = E\{A/X\}$.*
- *Rozważmy układ równań $A_i \overset{def}{=} E_i\{\widetilde{A}/\widetilde{X}\}$, indeksowany przez $i \in I$. Przez $E_i\{\widetilde{A}/\widetilde{X}\}$ rozumiemy wyrażenie procesowe E_i, w którym za wszystkie wystąpienia zmiennej X_j podstawiono A_j, dla wszystkich $j \in I$. Zakładamy iż zmienne pojawiające się w wyrażeniach E_i należą do zbioru $\{X_j : j \in I\}$ oraz że zmienna X_i jest sekwencyjna i strzeżona w E_j, dla każdego $i, j \in I$. Niech $\widetilde{P}$ i $\widetilde{Q}$ będą dwiema rodzinami procesów indeksowanych przez $i \in I$. Jeśli $\widetilde{P} = \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$ (przez co rozumiemy: $P_i = E_i\{\widetilde{P}/\widetilde{X}\}$, dla wszystkich i) oraz $\widetilde{Q} = \widetilde{E}\{\widetilde{Q}/\widetilde{X}\}$, to $\widetilde{P} = \widetilde{Q}$ (tzn. $P_i = Q_i$, dla wszystkich i).*

W drugiej części twierdzenia zakładamy, iż wszystkie zmienne są strzeżone i sekwencyjne. Łatwo się zatem domyślić, iż nie zawsze mamy jednoznaczność. Oto kilka przykładów:

Ćwiczenie 27 *Wskaż nieskończone zbiory rozwiązań następujących dwóch równań:*

$$\begin{aligned} X &= \tau.X \\ Y &= (a.Y|\overline{a}.0)\backslash a \end{aligned}$$

Ćwiczenie 28 *Pokaż, że dla każdego P, proces $\tau.(a.0 + \tau.P)$ jest rozwiązaniem równania $X = a.0 + \tau.X$.*

Ćwiczenie 29 *Znajdź nieskończony zbiór rozwiązań układu równań:*

$$\begin{aligned} X &= a.0 + \tau.Y \\ Y &= b.0 + \tau.X \end{aligned}$$

Jak wspomnieliśmy, twierdzenie 5 jest przydatne w rachunkach:

Ćwiczenie 30 *Zakładając $P = a.b.P$ oraz $Q = b.a.Q$, wykaż $P = a.Q \quad i \quad Q = b.P$.*

W rzeczywistości wymaganie, aby wszystkie zmienne były strzeżone i sekwencyjne jest zbyt mocne i można je osłabić. Oto kilka przykładów:

Ćwiczenie 31 *Rozważ układ równań, w którym zmienna X nie jest strzeżona:*

$$\begin{aligned} X &= a.X + \tau.Y \\ Y &= a.X + b.Y \end{aligned}$$

i wykaż, iż jego jedyne rozwiązanie jest następujące: $X = \tau.A, Y = A$, gdzie $A \stackrel{def}{=} a.A + a.A$.

Ćwiczenie 32 *Znajdź układ równań taki, że przynajmniej jedna ze zmiennych nie jest sekwencyjna w przynajmniej jednym równaniu, a pomimo tego ma on dokładnie jedno rozwiązanie.*

3.3 Prawo ekspansji

W praktyce często opis systemu współbieżnego ma następującą postać:

$$(P_1[f_1]|\dots|P_n[f_n])\backslash L,$$

$n \geq 1$.[2] Poniższe twierdzenie opisuje wszystkie tranzycje procesu w takiej postaci:

Twierdzenie 6

$$\begin{aligned} &(P_1[f_1]|\dots|P_n[f_n])\backslash L \quad = \\ &\sum\{\ f_i(\alpha).(P_1[f_1]|\dots|P_i'[f_i]|\dots|P_n[f_n])\backslash L \ :\ P_i \xrightarrow{\alpha} P_i', f_i(\alpha) \notin L \cup \overline{L}\ \} \ + \\ &\sum\{\ \tau.(P_1[f_1]|\dots|P_i'[f_i]|\dots|P_j'[f_j]|\dots|P_n[f_n])\backslash L \ :\ P_i \xrightarrow{l_1} P_i', P_j \xrightarrow{l_2} P_j', f_i(l_1) = \overline{f_j(l_2)}, i < j\ \} \end{aligned}$$

[2]Dodatkowo, czasem procesy P_i będą *sekwencyjne*, tzn. zbudowane tylko z operatorów dynamicznych.

Twierdzenie 6 jest sformułowane możliwie najogólniej, stąd może ono być niezbyt czytelne. Tym niemniej, jego dowód jest bardzo prosty – wystarczy skorzystać z przybliżenia równości semantycznej opisanego na końcu rozdziału 2.2.

Ćwiczenie 33 *Sformułuj Twierdzenie 6 w przypadku, gdy wszystkie funkcje przemianowujące f_j są identycznościami, a zatem można je pominąć.*

Ćwiczenie 34 *Pokaż że proces $(A'|B)\backslash c$ z rozdziału 3.2 jest faktycznie rozwiązaniem równania (11).*

Oto ważne i ciekawe wnioski z Twierdzenia 6 dla $n = 1$:

Wniosek 7

$$(\alpha.P)\backslash L = \begin{cases} 0, & \textit{gdy} \quad \alpha \in L \cup \overline{L} \\ \alpha.P\backslash L, & \textit{w.p.p.} \end{cases} \tag{12}$$

$$(\alpha.P)[f] = f(\alpha).P[f] \tag{13}$$

$$(P+Q)\backslash L = P\backslash L + Q\backslash L \tag{14}$$

$$(P+Q)[f] = P[f] + Q[f] \tag{15}$$

Ćwiczenie 35 *Przypomnij sobie specyfikację i implementację bufora z rozdziału 1. Korzystając z Twierdzenia 5 oraz z Wniosku 7(13), wykaż $C \frown C = \mathit{Buff}_2(\epsilon)$, przy czym:*

$$\begin{aligned}
C &\stackrel{def}{=} in.C' \\
C' &\stackrel{def}{=} \overline{out}.C \\
P \frown Q &\stackrel{def}{=} (P[c/out]|Q[c/in])\backslash c \\
\mathit{Buff}_2(0) &\stackrel{def}{=} in.\mathit{Buff}_2(1) \\
\mathit{Buff}_2(2) &\stackrel{def}{=} \overline{out}.\mathit{Buff}_2(1) \\
\mathit{Buff}_2(1) &\stackrel{def}{=} in.\mathit{Buff}_2(2) + \overline{out}.\mathit{Buff}_2(0)
\end{aligned}$$

$P[c/out]$ oznacza przemianowanie procesu P zmieniające tylko jedną nazwę, out. Tzn. $out \mapsto c, \overline{out} \mapsto \overline{c}$.

Ćwiczenie 36 *Ostatnie dwie równości wniosku 7 to prawa rozdzielności dwóch operatorów statycznych względem sumy. Czy rozdzielność taka zachodzi również dla _|_, tzn. czy prawdą jest*

$$(P+Q)|R \;=\; P|R + Q|R\ ?$$

3.4 Rodzaj syntaktyczny

Zanim przejdziemy do praw statycznych, musimy wprowadzić jedno proste pojęcie.

Definicja 8 *Mówimy, że zbiór $L \subseteq \mathcal{L}$ jest rodzajem procesu P, co oznaczamy przez $P : L$, o ile wszystkie zdarzenia procesu P oraz jego potomków należą do zbioru $L \cup \{\tau\}$.*

Zauważmy iż rodzaj procesu nie jest wyznaczony jednoznacznie natomiast istnieje zawsze rodzaj najmniejszy. Rodzaj procesu będzie nam potrzebny dla *warunkowego* sformułowania pewnych praw, np.

$$(P|Q)\backslash a \;=\; (P\backslash a)|Q, \qquad \text{o ile} \;\; a, \overline{a} \notin L, \quad Q : L.$$

Chcielibyśmy umieć efektywnie wyliczyć rodzaj procesu. Poniżej definiujemy *rodzaj syntaktyczny*, ozn. $\mathcal{L}(P)$.

Załóżmy przez chwilę, iż $\mathcal{L}(\mathrm{A})$ jest dane, dla każdej stałej A. Rozważmy następujący oczywisty przepis na wyliczenie rodzaju syntaktycznego:

$$\begin{aligned} \mathcal{L}(l.P) &= \{l\} \cup \mathcal{L}(P) \\ \mathcal{L}(\tau.P) &= \mathcal{L}(P) \\ \mathcal{L}(\textstyle\sum_{i\in \mathrm{I}} P_i) &= \bigcup_{i\in \mathrm{I}} \mathcal{L}(P_i) \\ \mathcal{L}(P|Q) &= \mathcal{L}(P) \cup \mathcal{L}(Q) \\ \mathcal{L}(P\backslash L) &= \mathcal{L}(P) \setminus (L \cup \overline{L}) \\ \mathcal{L}(P[f]) &= f(\mathcal{L}(P)) \end{aligned} \tag{16}$$

Przyporządkowanie rodzajów $\mathcal{L}(\mathrm{A})$ stałym A nazwiemy *poprawnym*, o ile $\mathcal{L}(P) \subseteq \mathcal{L}(\mathrm{A})$, gdy $\mathrm{A} \stackrel{\mathrm{def}}{=} P$.

Ćwiczenie 37 *Pokaż że istnieje najmniejsze poprawne przyporządkowanie rodzajów stałym procesowym, tzn. takie, które każdej stałej przyporządkowuje możliwie najmniejszy zbiór etykiet. (Wskazówka: użyj twierdzenia Knastera-Tarskiego o najmniejszym punkcie stałym w kracie zupełnej.)*

Definicja 9 *Przez rodzaj syntaktyczny procesu P, ozn. $\mathcal{L}(P)$, rozumiemy zbiór otrzymany za pomocą reguł (16), dla najmniejszego poprawnego przyporządkowania rodzajów stałym procesowym.*

Ćwiczenie 38 *Pokaż, za pomocą przykładu, że rodzaj syntaktyczny nie jest w ogólności najmniejszym rodzajem procesu.*

Fakt 10 $P : \mathcal{L}(P)$.

Fakt ten można łatwo dowieść na używając poniższego lematu:

Lemat 11 *Niech* $P \xrightarrow{\alpha} P'$. *Wtedy*

- $\alpha \in \mathcal{L}(P) \cup \{\tau\}$,
- $\mathcal{L}(P') \subseteq \mathcal{L}(P)$.

Ćwiczenie 39 *Dowiedż lematu przez indukcje po rozmiarze drzewa wywodu. Następnie udowodnij fakt.*

3.5 Równości statyczne

Równości statyczne to (z grubsza) dokładnie te równości pomiędzy procesami które zachodzą gdy interpretujemy proces jako *graf statyczny*. Tzn. są to równości zachodzące w algebrze grafów statycznych, którą opisujemy poniżej. Poniższa definicja uściśla pojęcie znane nam już z przykładów w rozdziale 1.

Definicja 12 *Graf statyczny to graf skończony o następujących właściwościach:*

- *Każdy wierzchołek jest etykietowany procesem.*
- *Każdy wierzchołek, z etykietą powiedzmy* P, *dla każdej* $l \in \mathcal{L}(P)$, *posiada port o etykiecie wewnętrznej* l.
- *Każda krawędź grafu łączy dwa porty w różnych wierzchołkach.*
- *Każdy port może posiadać etykietę zewnętrzną; jeśli ją ma, może ona być rożna od jego etykiety wewnętrznej.*
- *Porty połączone krawędzią albo nie posiadają etykiet zewnętrznych albo ich etykiety zewnętrzne są komplementarne.*
- *Wszystkie pary portów o komplementarnych etykietach zewnętrznych są połączone.*

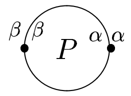
Algebra którą teraz definiujemy zawiera wszystkie grafy statyczne spełniające powyższą definicję. Posiada ona operacje odpowiadające trzem operatorom statycznym, których działanie jest następujące:

- Graf $G_1|G_2$ tworzymy z grafów G_1 i G_2 przez połączenie krawędzią wszystkich par portów o komplementarnych etykietach zewnętrznych, jeden z G_1 a drugi z G_2.

- Graf $G\backslash L$ tworzymy wymazując w G wszystkie etykiety zewnętrzne ze zbioru $L \cup \overline{L}$.
- Graf $G[f]$ tworzymy zastępując w G każdą etykietę zewnętrzną l przez $f(l)$; następnie łączymy krawędzią wszystkie pary portów o komplementarnych etykietach zewnętrznych.

Uwaga dla Czytelnika myślącego bardzo formalnie: w rzeczywistości w algebrze grafów statycznych mamy jedną operację dwuargumentową oraz dwie rodziny operacji jednoargumentowych: $\_\backslash L$ dla każdego $L \subseteq \mathcal{L}$ oraz $\_[f]$ dla każdej funkcji przemianowującej f. Ta sama uwaga jest prawdziwa dla algebry procesów którą rozważamy w tym rozdziale, z tym, że mamy w niej więcej operacji: jednoargumentową $\alpha.\_$ dla każdego $\alpha \in \mathrm{Act}$ oraz rodzinę operacji o różnych krotnościach, $\sum_{i \in \mathrm{I}} \_{}_i$, w zależności od mocy zbioru I.

Dla każdego procesu P, w algebrze grafów statycznych mamy następujący graf jednowierzchołkowy:



posiadający port dla każdej etykiety $\alpha, \beta, \ldots \in \mathcal{L}(P)$. (Jeśli Czytelnik usiłuje wciąż myśleć bardzo formalnie, to wszystkie takie grafy należy uważać za stałe w algebrze grafów statycznych. :)

Ćwiczenie 40 *Etykiety wewnętrzne pozostają niezmienione podczas budowania większych grafów z mniejszych. Czy są one niezbędne? Co by się zmieniło, gdyby ich nie było?*

Przedstawimy teraz trzy grupy równości, dotyczących każdego z operatorów statycznych. Wszystkie one są prawdziwe zarówno w algebrze grafów jak i w algebrze procesów. Ponadto, jeśli ograniczyć się do różnowartościowych funkcji przemianowujących (czego nie chcemy oczywiście robić), prawa te pozwalają wywieść *wszystkie* równości prawdziwe w algebrze grafów, stanowią zatem jej pełną aksjomatyzację. Oczywiście, w algebrze procesów prawdziwych jest znacznie więcej równości niż w algebrze grafów: w tej ostatniej możemy wnioskować tylko o *statycznych* własnościach procesów.

Ćwiczenie 41 *Podaj prosty przykład uzasadniający ostatnie zdanie.*

Po pierwsze, przemienność i łączność złożenia równoległego oraz neutralność procesu 0, interpretowanego jako graf pusty:

$$
\begin{array}{lrcl}
(|1) & P|Q & = & Q|P \\
(|2) & P|(Q|R) & = & (P|Q)|R \\
(|3) & P|0 & = & P
\end{array}
$$

Po drugie, cztery prawa dla operatora zakazu:

$$
\begin{array}{lrcll}
(\backslash 1) & P\backslash L & = & P & \text{o ile } \mathcal{L}(P) \cap (L \cup \overline{L}) = \emptyset \\
(\backslash 2) & P\backslash K\backslash L & = & P\backslash (K \cup L) & \\
(\backslash 3) & P[f]\backslash L & = & P\backslash f^{-1}(L)[f] & \\
(\backslash 4) & (P|Q)\backslash L & = & P\backslash L|Q\backslash L & \text{o ile } \mathcal{L}(P) \cap \overline{\mathcal{L}(Q)} \cap (L \cup \overline{L}) = \emptyset
\end{array}
$$

Po trzecie, cztery prawa dla przemianowania:

$$
\begin{array}{lrcll}
([]1) & P[\mathrm{id}] & = & P & \\
([]2) & P[f] & = & P[f'] & \text{o ile } f \text{ i } f' \text{ są identyczne na } \mathcal{L}(P) \\
([]3) & P[f][f'] & = & P[f' \circ f] & \\
([]4) & (P|Q)[f] & = & P[f]|Q[f] & \text{o ile } f \text{ jest różnowartościowe na} \\
 & & & & \mathcal{L}(P|Q) \cup \overline{\mathcal{L}(P|Q)}
\end{array}
$$

Prawa ($\backslash$4) oraz ([]4) są to *warunkowe* prawa rozłączności dwóch operatorów statycznych względem trzeciego, czyli złożenia równoległego. Oto kilka prostych wniosków:

Wniosek 13

$$
\begin{array}{rcll}
P[b/a] & = & P & \textit{o ile } a, \overline{a} \notin \mathcal{L}(P) \qquad (17) \\
P\backslash a & = & P[b/a]\backslash b & \textit{o ile } b, \overline{b} \notin \mathcal{L}(P) \qquad (18) \\
P\backslash a[b/c] & = & P[b/c]\backslash a & \textit{o ile ?} \qquad (19)
\end{array}
$$

Ćwiczenie 42 *Uzupełnij warunek w (19).*

Warto zwrócić tutaj uwagę, iż (18) przypomina nieco α-konwersję w rachunku λ ($\lambda a.P = \lambda b.P\{b/a\}$).

Ćwiczenie 43 *Dowiedź wniosku 13, korzystając z* ($\backslash$1) - ($\backslash$3) *oraz* ([]1) - ([]3).

Ćwiczenie 44 *Dowiedź wniosku 7.*

Ćwiczenie 45 *Pokaż, iż operator* $\_\frown\_$ *zdefiniowany w ćwiczeniu 35 jest łączny.*

Ćwiczenie 46 *Czy równości podane w tym rozdziale pozostają prawdziwe, gdy zamiast rodzaju syntaktycznego użyjemu najmniejszego rodzaju? Rozważ prawdziwość tych równości w algebrze grafów oraz w algebrze procesów.*

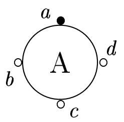
4 Silna równoważność bisymulacyjna

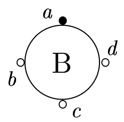
Ten rozdział poświęcony jest najważniejszemu pojęciu tego wykładu: pojęciu *bisymulacji*. Wprowadzimy tutaj równoważność procesów, zwaną *silną równoważnością bisymulacyjną*, ozn. $\sim$, opartą na tym właśnie pojęciu. Nie jest to jeszcze równość semantyczna $=$, której używaliśmy w poprzednim rozdziale; tę wprowadzimy w następnym rozdziale. Równoważność bisymulacyjna $\sim$ traktuje τ jak zwykłe zdarzenie, np. $a.\tau.0 \not\sim a.0$. W związku z tym odróżnia ona więcej procesów niż $=$. Mimo to, większość równości z poprzedniego rozdziału jest prawdziwa już dla $\sim$, a przy tym jej definicja jest bardziej przejrzysta niż $=$.

Zacznijmy od prostego przykładu, pozwalającego wyrobić intuicję:

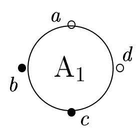
$$
\begin{tikzcd}
\mathrm{A} \arrow[r, "a"] & \mathrm{A}_1 \arrow[r, "b"] \arrow[dr, "c"] & \mathrm{A}_2 \\
 & & \mathrm{A}'_2 \arrow[ull, bend left, "d"]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\mathrm{B} \arrow[r, "a"] \arrow[dr, "a"] & \mathrm{B}_1 \arrow[r, "b"] & \mathrm{B}_2 \\
 & \mathrm{B}'_1 \arrow[r, "c"] & \mathrm{B}'_2 \arrow[ull, bend left, "d"]
\end{tikzcd}
$$

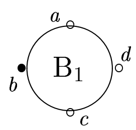
Jeśli na powyższe dwa systemy spojrzymy jak na automaty, to języki rozpoznawane przez nie są identyczne: $(acd)^*ab$, przy założeniu że wszystkie stany z indeksem 2 są akceptujące. Ale wyobraźmy sobie, że mamy możliwość eksperymentowania z tymi systemami. Początkowo, obydwa systemy są skłonne wykonać tylko a, jest to jedyny *aktywny* port:



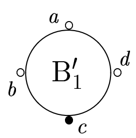


a wszystkie pozostałe porty są nieaktywne. Wykonajmy zdarzenie a w obydwu. W przypadku systemu A rezultat jest jednoznaczny (deterministyczny). W przypadku B mamy dwie możliwości, w każdej z nich inny jest zbiór aktywnych portów:





lub



Eksperyment odróżnia zatem A i B. Naszą intencją jest następująca definicja równoważności $\sim$:

$$P \sim Q \quad \text{gdy} \quad \text{wtw.} \quad \boxed{\begin{array}{l} \forall \alpha \in \mathrm{Act}, \\ (i)\ \ P \xrightarrow{\alpha} P' \implies \exists Q' \text{ t.że } Q \xrightarrow{\alpha} Q' \text{ i } P' \sim Q' \\ (ii)\ \ Q \xrightarrow{\alpha} Q' \implies \exists P' \text{ t.że } P \xrightarrow{\alpha} P' \text{ i } P' \sim Q' \end{array}} \tag{20}$$

Czyli P i Q są równoważne wtw. gdy nie umiemy ich odróżnić w pojedynczym kroku eksperymentu oraz systemy które otrzymamy w rezultacie tego kroku są znów równoważne. Ponadto interesowałaby nas największa możliwie relacja $\sim$. Ale czy powyższe sformułowanie (20) jest poprawne? Zaraz się to wyjaśni.

4.1 Silna bisymulacja

Definicja 14 *Relację binarną R pomiędzy procesami nazywamy <u>silną bisymulacją</u>, jeśli dla każdej pary $(P, Q) \in R$, $\forall \alpha \in \mathrm{Act}$,*

$$(i)\ \ P \xrightarrow{\alpha} P' \implies \exists Q' \ \textit{t.że}\ Q \xrightarrow{\alpha} Q' \ \textit{i}\ (P', Q') \in R$$
$$(ii)\ \ Q \xrightarrow{\alpha} Q' \implies \exists P' \ \textit{t.że}\ P \xrightarrow{\alpha} P' \ \textit{i}\ (P', Q') \in R.$$

Używamy określenia *silna* bisymulacja dla odróżnienia od *słabej* bisymulacji, którą zdefiniujemy w następnym rozdziale. Zanim to jednak nastąpi, będziemy czasem pisać krótko *bisymulacja.*

Ćwiczenie 47 *Znajdź relację bisymulacji pomiędzy specyfikacją semafora ogólnego $Sem_n(0)$ a jego implementacją $\underbrace{Sem|\ldots|Sem}_{n\ razy}$, dla małych n:*

$$Sem_n(0) \stackrel{def}{=} P.Sem_n(1) \qquad\qquad Sem \stackrel{def}{=} P.V.Sem$$
$$Sem_n(i) \stackrel{def}{=} P.Sem_n(i{+}1) + V.Sem_n(i{-}1) \quad dla \ \ 1 \leq i \leq n{-}1$$
$$Sem_n(n) \stackrel{def}{=} V.Sem_n(n{-}1)$$

Fakt 15 *Jeśli R, R' oraz R_i, dla $i \in I$, są silnymi bisymulacjami, to*

(i) *relacja identyczności* $\equiv$ $\qquad (iii)$ $R' \circ R$

(ii) R^{-1} $\qquad (iv)$ $\bigcup_{i \in I} R_i$

są nimi też.

Definicja 16 *Procesy P i Q są silnie równoważne, ozn. $P \sim Q$, jeśli $(P, Q) \in R$, dla pewnej bisymulacji R.*

Innymi słowy, silna równoważność $\sim$ jest sumą wszystkich bisymulacji:

$$\sim \; = \bigcup_{R \text{ silna bisymulacja}} R$$

Relację $\sim$ będziemy nazywać też *silną równoważnością bisymulacyjną.*

Uwaga 17 *Mimo iż implicite rozważamy tu tylko graf tranzycyjny procesów z języka podstawowego, łatwo widać, że definicja bisymulacji, a zatem również silnej równoważności, ma sens dla dowolnego innego grafu tranzycyjnego.*

Fakt 18

(i) $\sim$ *jest największą silną bisymulacją*

(ii) $\sim$ *jest równoważnością*

Ćwiczenie 48 *Udowodnij fakty 15 i 18.*

Fakt 19 *Zachodzi równoważność (20).*

Dowód: Niech $\sim'$ oznacza relację t.że $P \sim' Q$ wtw. gdy procesy P i Q spełniają warunek ujęty w ramce w (20). Należy pokazać równość dwóch relacji: $\sim \; = \; \sim'$. Po pierwsze, mamy $\sim \; \subseteq \; \sim'$, ponieważ $\sim$ jest silną bisymulacją. Po wtóre, mamy również $\sim' \subseteq \; \sim$ ponieważ $\sim'$ jest również silną bisymulacją a $\sim$ jest największą silną bisymulacją. □

Ćwiczenie 49 *Pokaż, iż relacja $\sim'$ użyta w dowodzie faktu 19 jest rzeczywiście silną bisymulacją.*

Fakt 20 $\sim$ *jest największą relacją spełniającą (20).*

Dowód: Każda relacja R spełniająca (20) jest bisymulacją, a zatem $R \subseteq \; \sim$. □

Oczywiście $\sim$ nie jest jeszcze równoważnością semantyczną $=$ obiecywaną w poprzednich rozdziałach. Podstawową przyczyną jest fakt, iż traktuje ona zdarzenie τ tak jak wszystkie inne. W związku z tym bez wątpienia $\sim$ nie spełnia praw (τ_1) – (τ_3) z rozdziału 3, np.

$$\alpha.\tau.P \not\sim \alpha.P$$

Rozważamy najpierw $\sim$, zanim przejdziemy do $=$ z dwóch powodów. Po pierwsze, spełnia ona *wszystkie pozostałe* prawa z rozdziału 3. Zatem silna równoważność uwzględnia wszystkie aspekty semantyczne procesów z wyjątkiem zdarzenia nieobserwowalnego τ. Po drugie, $\sim$ jest prostsza, a $=$ definiuje się w dużym stopniu podobnie jak $\sim$.

Ćwiczenie 50 *Wybierz jedno z praw statycznych i jedno z praw dynamicznych i pokaż, że* $\sim$ *je spełnia. (Wskazówka: podobnie jak w dowodzie faktu 24.)*

Ćwiczenie 51 *Przez indukcję po* n *pokaż, że okrojona wersja prawa ekspansji zachodzi dla* $\sim$ *:*

$$\begin{aligned} P_1|\ldots|P_n \quad \sim \quad & \sum\{\ \alpha.(P_1|\ldots|P_i'|\ldots|P_n)\ :\ P_i \xrightarrow{\alpha} P_i'\ \}\ + \\ & \sum\{\ \tau.(P_1|\ldots|P_i'|\ldots|P_j'|\ldots|P_n)\ :\ P_i \xrightarrow{l} P_i', P_j \xrightarrow{\bar{l}} P_j', i < j\ \}. \end{aligned}$$

Ćwiczenie 52 *Relację* R *nazwijmy s-bisymulacją jeśli* R *jest symetryczna oraz dla każdej pary* $(P,Q) \in R$, $\forall \alpha \in \mathrm{Act}$,

$$P \xrightarrow{\alpha} P' \implies \exists Q' \ \ t.\dot{z}e \ \ Q \xrightarrow{\alpha} Q' \ \ i \ \ (P',Q') \in R.$$

Zaoszczędziliśmy polowę definicji bisymulacji. Czy każda s-bisymulacja jest bisymulacją? Czy każda bisymulacja jest s-bisymulacją. Czy równoważność bisymulacyjna pokrywa się z równoważnością s-bisymulacyjną, tzn. czy zachodzi:

$$\Big(\bigcup_{R \ \ silna\ bisymulacja} R\Big) = \Big(\bigcup_{R \ \ s\text{-}bisymulacja} R\Big)\ ?$$

4.2 Z dokładnością do $\sim$

Aby wykazać, iż para procesów jest silnie równoważne, wystarczy wskazać (jakąkolwiek) bisymulację zawierającą tę parę. Zatem pojęcie bisymulacji daje nam automatycznie metodę dowodzenia równoważności. Niestety, nie zawsze łatwo jest odgadnąć tę właściwą relację bisymulacji. Innym problemem jest rozmiar relacji. Na przykład Czytelnik, który rozwiązał ćwiczenie 47 zauważył zapewne, że wraz ze wzrostem n rozmiar bisymulacji rośnie gwałtownie. Poniżej przedstawiamy modyfikację definicji 14 pozwalającą ograniczyć w znacznym stopniu rozmiar relacji, a więc prowadzącą do bardziej efektywnej techniki dowodowej:

Definicja 21 *Relację binarną* R *pomiędzy procesami nazywamy* *silną bisymulacją z dokładnością do* $\sim$, *jeśli dla każdej pary* $(P,Q) \in R$, $\forall \alpha \in \mathrm{Act}$,

$$(i)\ \ P \xrightarrow{\alpha} P' \implies \exists Q' \ \ t.\dot{z}e \ \ Q \xrightarrow{\alpha} Q' \ \ i \ \ (P',Q') \in\ \sim \circ R \circ \sim$$

$$(ii)\ \ Q \xrightarrow{\alpha} Q' \implies \exists P' \ \ t.\dot{z}e \ \ P \xrightarrow{\alpha} P' \ \ i \ \ (P',Q') \in\ \sim \circ R \circ \sim$$

Jedyna różnica względem definicji 14 to osłabienie wymogu co do pary (P', Q'). Tutaj wystarczy, jeśli $(P', Q') \in\ \sim \circ R \circ \sim$, czyli dla pewnych P'', Q'', $P' \sim P''$, $(P'', Q'') \in R$ i $Q'' \sim Q'$. Możemy to przedstawić tak:

$$\begin{tikzcd}
& P \arrow[dl, "\alpha"'] \arrow[rr, phantom, "R"] & & Q \arrow[dr, "\alpha"] & \\
P' \arrow[r, phantom, "\sim"] & P'' \arrow[rr, phantom, "R"] & & Q'' \arrow[r, phantom, "\sim"] & Q'
\end{tikzcd}$$

Ćwiczenie 53 *Znajdź jak najmniejszą bisymulację z dokładnością do $\sim$ dla pary procesów z ćwiczenia 47.*

We wniosku 23 poniżej pokażemy, iż za pomocą bisymulacji z dokładnością do $\sim$ możemy pokazać dokładnie to samo, co za pomocą bisymulacji, mianowicie silną równoważność procesów. Zatem obydwie metody dowodowe są sobie równoważne.

Lemat 22 *Jeśli R jest silną bisymulacją z dokładnością do $\sim$, to $\sim \circ R \circ \sim$ jest silną bisymulacją.*

Dowód: Aby dowieść że $\sim \circ R \circ \sim$ jest silną bisymulacją, korzystamy (dwukrotnie) z faktu iż $\sim$ jest bisymulacją oraz z tego, że R jest bisymulacją z dokładnością do $\sim$:

$$\begin{tikzcd}
& P \arrow[dl, "\alpha"'] \arrow[r, phantom, "\sim"] & P_1 \arrow[dl, "\alpha"'] \arrow[rr, phantom, "R"] & & Q_1 \arrow[dr, "\alpha"] \arrow[r, phantom, "\sim"] & Q \arrow[dr, "\alpha"] & \\
P' \arrow[r, phantom, "\sim"] & P''' \arrow[r, phantom, "\sim"] & P'' \arrow[rr, phantom, "R"] & & Q'' \arrow[r, phantom, "\sim"] & Q''' \arrow[r, phantom, "\sim"] & Q'
\end{tikzcd}$$

Ale $\sim \circ \sim\ \subseteq\ \sim$, więc $(P', Q') \in\ \sim \circ R \circ \sim$. $\square$

Ćwiczenie 54 *Pokaż, że zachodzi również implikacja przeciwna: jeśli $\sim \circ R \circ \sim$ jest silną bisymulacją, to R jest silną bisymulacją z dokładnością do $\sim$.*

Wniosek 23 *Jeśli R jest silną bisymulacją z dokładnością do $\sim$, to $R \subseteq\ \sim$.*

Dowód: Ponieważ $\equiv\ \subseteq\ \sim$, mamy $R = (\equiv \circ R \circ \equiv) \subseteq\ \sim \circ R \circ \sim$. A z poprzedniego faktu, $\sim \circ R \circ \sim\ \subseteq\ \sim$. $\square$

4.3 Kongruencja

Silna równoważność jest kongruencją względem wszystkich operatorów.

Fakt 24 *Jeśli $P \sim Q$, $P' \sim Q'$ oraz $P_i \sim Q_i$, dla $i \in I$, to:*

$$(i)\ \alpha.P \ \sim\ \alpha.Q \qquad (iii)\ P\backslash L \ \sim\ Q\backslash L$$
$$(ii)\ \sum\nolimits_{i\in I} P_i \ \sim\ \sum\nolimits_{i\in I} Q_i \qquad (iv)\ P[f] \ \sim\ Q[f]$$
$$(v)\ P|P' \ \sim\ Q|Q'$$

Dowód: Punkty (i) i (ii) można pokazać bezpośrednio z (20). Aby udowodnić punkt (v) wystarczy zauważyć, iż relacja

$$\{(P|P', Q|Q') : P \sim Q, P' \sim Q'\}$$

jest silną bisymulacją.

Ćwiczenie 55 *Pokaż, że faktycznie tak jest.*

Punkty (iii) i (iv) podobnie do (v). □

Dzięki temu, że $\sim$ jest kongruencją, możemy swobodnie przekształcać wyrażenia procesowe zastępując podwyrażenie innym silnie mu równoważnym. Chcielibyśmy posiadać również podobną swobodę w odniesieniu do przekształcania wyrażeń definiujących stałe procesowe. Ale w tym celu musimy udowodnić, że zmieniając prawą stronę równania definiującego stałą nie zmieniamy znaczenia tej stałej; np. stałe określone przez poniższe dwie definicje powinny być silnie równoważne:

$$\mathrm{A} \stackrel{\mathrm{def}}{=} b.0 + a.(\mathrm{A}|c.0)$$
$$\mathrm{B} \stackrel{\mathrm{def}}{=} a.(c.0|\mathrm{B}) + 0 + b.0$$

Definicja 25 *Przez* $\mathrm{fv}(E)$ *oznaczamy zbiór wszystkich zmiennych występujących w* E. *Załóżmy, że* $\mathrm{fv}(E), \mathrm{fv}(F) \subseteq \widetilde{X}$. *Mówimy, że wyrażenia E, F są silnie równoważne, ozn. $E \sim F$, jeśli dla każdego zbioru procesów $\widetilde{P}$ (indeksowanego tym samym zbiorem co $\widetilde{X}$), $E\{\widetilde{P}/\widetilde{X}\} \sim F\{\widetilde{P}/\widetilde{X}\}$.*

Twierdzenie 26 *Niech* $\mathrm{fv}(E), \mathrm{fv}(F) \subseteq \{X\}$. *Jeśli*

$$A \stackrel{def}{=} E\{A/X\}$$
$$B \stackrel{def}{=} F\{B/X\}$$
$$E \sim F$$

to $A \sim B$.

Dowód: ... □

Fakt 24 (być może z wyjątkiem punktu (*ii*)) wynika ze znacznie ogólniejszego twierdzenia:

Twierdzenie 27 *Rozważmy dowolny język opisu procesów, zawierający pewną liczbę operatorów. Załóżmy, iż wszystkie operatory są zdefiniowane wyłącznie przy pomocy reguł następującej postaci:*

$$\frac{\{X_i \xrightarrow{\alpha} X'_i : i \in I\}}{\mathrm{op}(X_1, \ldots, X_n) \xrightarrow{\alpha} E}$$

gdzie n jest arnością operatora op, $I \subseteq \{1, \ldots, n\}$, *zmienne X_i, Y_i są parami różne,* $\mathrm{fv}(E) \subseteq \{X_i : i \notin I\} \cup \{Y_i : i \in I\}$ *oraz żadna zmienna nie występuje więcej niż raz w E. Wtedy $\sim$ jest kongruencją względem wszystkich operatorów języka.*

Ćwiczenie 56 *Udowodnij twierdzenie 27.*

4.4 Jednoznaczność rozwiązań układów równań

Definicja 28 *Zmienna X jest <u>słabo strzeżona</u> w wyrażeniu E jeśli każde wystąpienie X w E jest wewnątrz pewnego podwyrażenia $\alpha.E'$ w E.*

Na przykład, X jest słabo strzeżona w $\tau.X$. Poniższe twierdzenie udowodnimy później:

Twierdzenie 29 (jednoznaczność rozwiązań względem $\sim$)

- *Jeśli $A \stackrel{def}{=} E\{A/X\}$ to $A \sim E\{A/X\}$.*
- *Rozważmy układ równań $A_i \stackrel{def}{=} E_i\{\widetilde{A}/\widetilde{X}\}$, indeksowany przez $i \in I$. Zakładamy iż zmienne pojawiające się w wyrażeniach E_i należą do zbioru $\{X_j : j \in I\}$ oraz że zmienna X_i jest słabo strzeżona w E_j, dla każdego $i, j \in I$. Jeśli $\widetilde{P} \sim \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$ oraz $\widetilde{Q} \sim \widetilde{E}\{\widetilde{Q}/\widetilde{X}\}$, to $\widetilde{P} \sim \widetilde{Q}$.*

Jak widać, dla $\sim$ wystarczą słabsze założenia niż dla $=$ w rozdziale 3.2.
Dowód: ... □

4.5 Największy punkt stały

Przypomnijmy sobie równoważność (20). W rzeczywistości napisane tam jest, iż $\sim$ jest największym punktem stałym pewnego przekształcenia. Oto szczegóły:

Definicja 30 *Niech $\mathcal{F}$ będzie przekształceniem w zbiorze relacji binarnych pomiędzy procesami. Dla takiej relacji R, niech relacja $\mathcal{F}(R)$ będzie określona następująco: $(P, Q) \in \mathcal{F}(R)$ wtw. gdy*

$$\boxed{\begin{array}{l} \forall \alpha \in \mathrm{Act}, \\ (i) \;\; P \xrightarrow{\alpha} P' \;\; \Longrightarrow \exists Q' \;\; t.\dot{z}e \;\; Q \xrightarrow{\alpha} Q' \;\; i \;\; (P', Q') \in R \\ (ii) \;\; Q \xrightarrow{\alpha} Q' \;\; \Longrightarrow \exists P' \;\; t.\dot{z}e \;\; P \xrightarrow{\alpha} P' \;\; i \;\; (P', Q') \in R \end{array}}$$

Fakt 31

(i) $\mathcal{F}$ jest monotoniczna.

(ii) R jest silną bisymulacją wtw. gdy $R \subseteq \mathcal{F}(R)$.

Poniższe twierdzenie jest w istocie innym sformułowaniem faktów 19 oraz 20 a jego dowód jest również identyczny jak dowody wspomnianych faktów. Podstawowym pożytkiem z definicji 30 oraz twierdzenia 32 jest znacznie większa czytelność.

Twierdzenie 32 *Silna równoważność jest największym punktem stałym $\mathcal{F}$.*

Dowód: Wiemy, że $\sim$ jest silną bisymulacją, czyli

$$\sim \; \subseteq \mathcal{F}(\sim). \tag{21}$$

Ponieważ $\mathcal{F}$ jest monotoniczna, $\mathcal{F}(\sim) \subseteq \mathcal{F}(\mathcal{F}(\sim))$, tzn. $\mathcal{F}(\sim)$ jest również bisymulacją. Ale $\sim$ jest przecież największą bisymulacją, zatem $\mathcal{F}(\sim) \subseteq \; \sim$. Czyli pokazaliśmy $\sim \; = \mathcal{F}(\sim)$.

Tak naprawdę to wcale nie musimy korzystać z (21). Albo możemy wykazać (21), ponieważ na podstawie definicji silnej równoważności wiemy, iż

$$\sim \; = \bigcup \{R : R \subseteq \mathcal{F}(R)\}. \tag{22}$$

W tym celu weźmy dowolną relację R ze zbioru powyżej, tzn. taką, że $R \subseteq \mathcal{F}(R)$ (czyli, inaczej mówiąc, dowolną bisymulację). Na podstawie (22) mamy $R \subseteq \; \sim$, zatem $\mathcal{F}(R) \subseteq \mathcal{F}(\sim)$. Wnioskujemy więc, że $R \subseteq \mathcal{F}(\sim)$. A ponieważ R była wybrana dowolnie, mamy $\bigcup \{R : R \subseteq \mathcal{F}(R)\} \subseteq \mathcal{F}(\sim)$, czyli $\sim \; \subseteq \mathcal{F}(\sim)$. $\square$

Uważny Czytelnik z pewnością zauważył, że niepotrzebny był trud dowodzenia powyższego twierdzenia, ponieważ jest ono szczególnym przypadkiem sytuacji rozważanej w twierdzeniu Knastera-Tarskiego o punkcie stałym w kracie zupełnej. Co więcej, dowód twierdzenia 32 działa dla dowolnej kraty zupełnej, gdyż nie odwołujemy się w nim do żadnych specyficznych własności naszych relacji.

Twierdzenie 33 (Knaster-Tarski) *Każda funkcja monotoniczna $f : X \to X$ w kracie zupełnej $\langle X, \sqsubseteq \rangle$ posiada największy punkt stały, równy kresowi górnemu*

$$\bigsqcup \{x \in X : x \sqsubseteq f(x)\}.$$

Ćwiczenie 57 *Tak naprawdę to jest to twierdzenie* dualne *do twierdzenia Knastera-Tarskiego, mówiącego o najmniejszym punkcie stałym równym kresowi dolnemu zbioru $\{x \in X : f(x) \sqsubseteq x\}$. Uzasadnij, dlaczego obydwa twierdzenia są sobie równoważne.*

4.6 Indukcja a koindukcja

Metodę dowodzenia za pomocą bisymulacji nazywamy *koindukcją*. Czy jest ona faktycznie *dualna* do znanej nam dobrze indukcji? Tak! Spróbujmy to jakoś uzasadnić, wykorzystując dualność pomiędzy najmniejszym a największym punktem stałym. Zasada koindukcji jest następująca: aby pokazać równoważność dwóch procesów, znajdź bisymulację, która zawiera tę parę. Zatem zbiór par, o których możemy pokazać, że są równoważne, to największa relacja R taka, że

$$R \subseteq \mathcal{F}(R),$$

dla $\mathcal{F}$ z poprzedniego rozdziału.

Popatrzmy teraz na indukcję, na prostym przykładzie. Rozważmy relację równoważności skończonych list zawierających liczby naturalne, określoną następująco: dwie listy są równoważne, jeśli są równej długości a ich, pierwsze, drugie, ..., itd. elementy przystają modulo 7. Oczywiście, równoważności tej można dowieść indukcyjnie: listy są równoważne albo gdy obydwie są puste albo gdy obydwie są niepuste, ich głowy przystają modulo 7 oraz ich ogony są równoważne. Relacja ta jest *najmniejszym* punktem stałym pewnego operatora $\mathcal{G}$ określonego jak następuje. Dla relacji binarnej R zawierającej pary list,

- $\mathcal{G}(R)$ zawiera zawsze parę list pustych, $(\textsc{nil}, \textsc{nil}) \in \mathcal{G}(R)$, niezależne od R;
- jeśli R zawiera parę list postaci (x, y) oraz n i m przystają modulo 7, to $(\textsc{cons}(n, x), \textsc{cons}(m, y)) \in \mathcal{G}(R)$.

Czyli

$$\begin{aligned}\mathcal{G}(R) &= \{(\text{NIL}, \text{NIL})\} \cup \\ &\quad \{(\text{CONS}(n,x), \text{CONS}(m,y)) : (x,y) \in R \ \text{ oraz } n \text{ i } m \text{ przystają modulo } 7\}.\end{aligned}$$

Relacja równoważności, którą rozważamy, to najmniejsza R taka, że

$$\mathcal{G}(R) \subseteq R.$$

Dla lepszego porównania indukcji i koindukcji, rozważmy listy nieskończone oraz zaadaptujmy rozważaną relację równoważności. Teraz rozumowanie indukcyjne zawodzi, ale można dowieść naszej równoważności koindukcyjnie. Okazuje się, że teraz nasza relacja równoważności to największy punkt stały pewnego operatora $\mathcal{F}$; lub innymi słowy, największa R taka, że

$$R \subseteq \mathcal{F}(R).$$

Operator $\mathcal{F}$ definiujemy teraz podobnie do $\mathcal{G}$:

$$\mathcal{F}(R) = \{(\text{CONS}(n,x), \text{CONS}(m,y)) : (x,y) \in R \ \text{ oraz } n \text{ i } m \text{ przystają modulo } 7\}.$$

Ćwiczenie 58 *Rozważ zbiór wszystkich list skończonych i nieskończonych. Zdefiniuj operator $\mathcal{F}$ taki, że równoważność list jest jego największym punktem stałym.*

5 Słaba równoważność bisymulacyjna

Pojęcia słabej bisymulacji, słabej równoważności, słabej bisymulacji z dokładnością do słabej równoważności, itp. są analogiczne do swoich silnych odpowiedników, z tą różnicą, iż oparte one są o inną relację tranzycji. Zamiast tranzycji $\stackrel{\alpha}{\longrightarrow}$ będziemy teraz używać $\stackrel{\varepsilon}{\Longrightarrow}$ oraz $\stackrel{l}{\Longrightarrow}$, $l \in \mathcal{L}$, zdefiniowanych następująco:

$$\stackrel{\varepsilon}{\Longrightarrow} \;=\; (\stackrel{\tau}{\longrightarrow})^* \qquad\qquad \stackrel{l}{\Longrightarrow} \;=\; (\stackrel{\tau}{\longrightarrow})^* \circ \stackrel{l}{\longrightarrow} \circ (\stackrel{\tau}{\longrightarrow})^*.$$

Można jednolicie wyrazić obydwie relacje następująco:

Definicja 34 *Dla* $t = \alpha_1 \dots \alpha_n \in \mathrm{Act}^*$*,* $P \stackrel{t}{\Longrightarrow} P'$ *jeśli*

$$P(\stackrel{\tau}{\longrightarrow})^* \stackrel{\alpha_1}{\Longrightarrow} (\stackrel{\tau}{\longrightarrow})^* \dots (\stackrel{\tau}{\longrightarrow})^* \stackrel{\alpha_n}{\Longrightarrow} (\stackrel{\tau}{\longrightarrow})^* P'.$$

Dla $t \in \mathrm{Act}^*$*, niech* $\widehat{t}$ *oznacza ciąg* t*, z którego usunięto wszystkie wystąpienia* τ*. Powiemy, że* P' *jest słabym* t*-potomkiem procesu* P *jeśli* $P \stackrel{\widehat{t}}{\Longrightarrow} P'$*.*

Zauważmy, iż obydwie relacje wspomniane powyżej, $\stackrel{\varepsilon}{\Longrightarrow}$ oraz $\stackrel{l}{\Longrightarrow}$, są szczególnymi przypadkami relacji $P \stackrel{\widehat{\alpha}}{\Longrightarrow} P'$, dla $\alpha \in \mathrm{Act}$. Mamy zatem, dla $t \in \mathrm{Act}^*$, trzy relacje:

$$\stackrel{t}{\longrightarrow} \;\subsetneq\; \stackrel{t}{\Longrightarrow} \;\subsetneq\; \stackrel{\widehat{t}}{\Longrightarrow}.$$

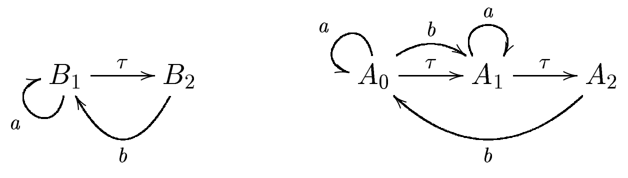
Dla pierwszej z nich, t określa *dokładnie* ile zdarzeń τ ma mieć miejsce. Dla drugiej, t specyfikuje ile *co najmniej* τ ma się odbyć. W przypadku trzeciej, nic nie wiadomo o liczbie τ.

Definicja 35 *Relację binarną* R *pomiędzy procesami nazywamy słabą bisymulacją, jeśli dla każdej pary* $(P, Q) \in R$*,* $\forall \alpha \in \mathrm{Act}$*,*

$$(i)\;\; P \stackrel{\alpha}{\longrightarrow} P' \;\Longrightarrow\; \exists Q' \;\; t.\dot{z}e \;\; Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q' \;\; i \;\; (P', Q') \in R$$

$$(ii)\;\; Q \stackrel{\alpha}{\longrightarrow} Q' \;\Longrightarrow\; \exists P' \;\; t.\dot{z}e \;\; P \stackrel{\widehat{\alpha}}{\Longrightarrow} P' \;\; i \;\; (P', Q') \in R.$$

Ćwiczenie 59 *Znajdź słabą bisymulację pomiędzy procesami:*



Będziemy czasem pisać krótko *bisymulacja* mając na myśli słabą bisymulację. Kilka poniższych faktów i definicji jest zupełnie analogicznych do faktów i definicji z poprzedniego rozdziału. Tym niemniej, dowody nie zawsze są zupełnie identyczne.

Fakt 36 *Jeśli R, R' oraz R_i, dla $i \in I$, są słabymi bisymulacjami, to*

$$
\begin{array}{llll}
(i) & \textit{relacja identyczności} \ \equiv & \qquad (iii) & R' \circ R \\
(ii) & R^{-1} & \qquad (iv) & \bigcup_{i \in I} R_i
\end{array}
$$

są nimi też.

Ćwiczenie 60 *Udowodnij punkt* (iii). *(Wskazówka: ćwiczenie 65.)*

Definicja 37 *Procesy P i Q są <u>słabo równoważne</u>, ozn. $P \approx Q$, jeśli $(P, Q) \in R$, dla pewnej słabej bisymulacji R.*

Innymi słowy, słaba równoważność $\approx$ jest sumą wszystkich bisymulacji:

$$\approx \ = \bigcup_{R \text{ słaba bisymulacja}} R$$

Relację $\approx$ będziemy nazywać też *słabą równoważnością bisymulacyjną.*

Ćwiczenie 61 *Udowodnij, że*

$$
\begin{array}{ll}
(i) & \approx \ \textit{jest największą słabą bisymulacją} \\
(ii) & \approx \ \textit{jest równoważnością.}
\end{array}
$$

Ćwiczenie 62 *Znajdź najmniejszą słabą bisymulację zawierającą wszystkie pary postaci* $(P, \tau.P)$.

Ćwiczenie 63 *Udowodnij fakt analogiczny do (20) z poprzedniego rozdziału:*

$$
P \approx Q \qquad \textit{wtw. gdy} \qquad
\begin{array}{l}
\forall \alpha \in \mathrm{Act}, \\
(i) \ \ P \xrightarrow{\alpha} P' \implies \exists Q' \ \ t.że \ \ Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q' \ \ i \ \ P' \sim Q' \\
(ii) \ \ Q \xrightarrow{\alpha} Q' \implies \exists P' \ \ t.że \ \ P \stackrel{\widehat{\alpha}}{\Longrightarrow} P' \ \ i \ \ P' \sim Q'
\end{array}
$$

Ćwiczenie 64 *Czy słaba równoważność jest również największym punktem stałym pewnego przekształcenia $\mathcal{F}$?*

Ćwiczenie 65 *Uzasadnij, że R jest bisymulacją wtw. gdy dla każdej pary $(P,Q) \in R$, $\forall \alpha \in \mathrm{Act}$,*

$$(i)\;\; P \stackrel{\widehat{\alpha}}{\Longrightarrow} P' \;\;\Longrightarrow\;\; \exists Q' \;\; t.że \;\; Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q' \;\; i \;\; (P',Q') \in R$$
$$(ii)\;\; Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q' \;\;\Longrightarrow\;\; \exists P' \;\; t.że \;\; P \stackrel{\widehat{\alpha}}{\Longrightarrow} P' \;\; i \;\; (P',Q') \in R.$$

Czyli powyższy „usymetryczniony" warunek można równoważnie przyjąć za definicję bisymulacji, zamiast definicji 35. Podstawowa różnica między nimi jest taka, że warunek użyty w definicji 35 jest znacznie łatwiejszy do sprawdzenia, podczas dowodzenia równoważności procesów za pomocą bisymulacji.

Ćwiczenie 66 *Rozważ ponownie dwie równoważne definicje bisymulacji, ale po zastąpieniu zdarzenia α przez skończony ciąg zdarzeń t. Czy dwie nowe definicje są również równoważne definicji 35?*

Jak wynika z ćwiczenia 62, $\approx$ nie jest z pewnością poszukiwaną przez nas równością semantyczną $=$, gdyż w ogólności $P \neq \tau.P$. Co więcej, $\approx$ nie jest kongruencją, gdyż $a.0 \;\approx\; \tau.a.0$ ale $a.0 + b.0 \;\not\approx\; \tau.a.0 + b.0$.

Ćwiczenie 67 *Pokaż ostatnią nierówność.*

Słaba równoważność utożsamia zbyt wiele par procesów, mamy zatem:

$$\sim \;\; \subsetneq \; = \; \subsetneq \;\; \approx . \tag{23}$$

Tym niemniej, relacja $\approx$ jest już bardzo blisko semantycznej równości. Oto kilka argumentów.

Definicja 38 *P jest stabilny jeśli nie ma τ-potomków (ozn. $P \stackrel{\tau}{\nrightarrow}$).*

Fakt 39 *$P \approx Q \implies P = Q$, o ile P i Q są stabilne.*

Fakt ten ma kluczowe znaczenie dla dowodzenia równości procesów. Potwierdza on, iż słaba bisymulacja jest poprawną metodą dowodzenia równości procesów: zamiast dowodzić $P = Q$, w praktyce wystarczy rozważyć $start.P = start.Q$ (czyli zakładamy, iż działanie systemu rozpoczyna się od dodatkowego zdarzenia $start \notin \mathcal{L}(P) \cup \mathcal{L}(Q)$), a ta ostatnia para jest stabilna.

Fakt 40 *Jeśli $P \approx Q$, $P' \approx Q'$ oraz $P_i \approx Q_i$, dla $i \in I$, to:*

$(i)\;\; \alpha.P \;\approx\; \alpha.Q$

$(ii)\;\; \sum_{i \in I} \alpha_1.P_i \;\approx\; \sum_{i \in I} \alpha_1.Q_i$

$(iii)\;\; P \backslash L \;\approx\; Q \backslash L$

$(iv)\;\; P[f] \;\approx\; Q[f]$

$(v)\;\; P|P' \;\approx\; Q|Q'$

Jedyna różnica względem analogicznego faktu dla silnej równoważności to ograniczenie się w punkcie (ii) do sum *słabo strzeżonych*; w szczególności α_i może być równe τ. Z faktu 40 wynika, że $\approx$ jest kongruencją, gdy używamy jedynie sum słabo strzeżonych, co w praktyce jest zupełnie wystarczające.

Ćwiczenie 68 *Udowodnij fakt 40.*

Jako ostatni argument przemawiający za bliskością $=$ i $\approx$ wspomnijmy, że punkt (i) można wzmocnić następująco:

Fakt 41 *Jeśli* $P \approx Q$*, to* $\alpha.P \; = \; \alpha.Q$*.*

Dowody faktów 39 oraz 41 podamy później, gdy poznamy definicję równości semantycznej. Zanim ją jednak podamy, zajmiemy się analizą kilku przykładów praktycznych, ilustrujących przede wszystkim użycie bisymulacji do dowodzenia poprawności. Nasza dotychczasowa wiedza jest w zupełności wystarczająca, gdyż dzięki faktowi 39 będziemy mogli nie odróżniać $=$ od $\approx$.

5.1 Z dokładnością do $\approx$

Podobnie jak w przypadku silnej bisymulacji, chcemy usprawnić nieco naszą metodę dowodową tak, aby dla wykazania $P \approx Q$ wystarczyło znaleźć jak najmniejszą relację zawierającą parę (P, Q). Oto naturalna adaptacja definicji z poprzedniego rozdziału.

Definicja 42 *Relację binarną* R *pomiędzy procesami nazwiemy* <u>*słabą bisymulacją z dokładnością do* $\approx$</u>*, jeśli dla każdej pary* $(P, Q) \in R$, $\forall \alpha \in \mathrm{Act}$,

$$(i) \;\; P \xrightarrow{\alpha} P' \;\; \Longrightarrow \exists Q' \;\; t.że \;\; Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q' \;\; i \;\; (P', Q') \in \; \approx \circ R \circ \approx$$
$$(ii) \;\; Q \xrightarrow{\alpha} Q' \;\; \Longrightarrow \exists P' \;\; t.że \;\; P \stackrel{\widehat{\alpha}}{\Longrightarrow} P' \;\; i \;\; (P', Q') \in \; \approx \circ R \circ \approx$$

Czytelnik znudzony kolejną z pozoru oczywistą wariacją na temat bisymulacji zapewne jest stuprocentowo pewien, że zachodzi:

Fakt 43 *Jeśli* R *jest słabą bisymulacją z dokładnością do* $\approx$ *, to* $\approx \circ R \circ \approx$ *jest słabą bisymulacją. (A zatem* $R \subseteq \; \approx$ *.)*

Ćwiczenie 69 *Spróbuj udowodnić powyższy fakt, zwracając baczną uwagę na szczegóły.*

Dlaczego wymagamy od czytelnika żmudnego dowodzenia *oczywistego* faktu? Dlatego, iż w rzeczywistości jednak fakt 43 *nie* zachodzi!

Ćwiczenie 70 *Spróbuj wykazać, że fakt 43 nie zachodzi. Rozważ $\alpha = \tau$ oraz parę procesów postaci $(\tau.P, 0)$, dla pewnego P. Znajdź relację R zawierającą tę parę taką, że R jest bisymulacją z dokładnością do $\approx$, a mimo to $R \not\subseteq \approx$.*

Cały problem znika, gdy oprzemy pojęcie bisymulacji z dokładnością do $\approx$ na symetrycznej wersji bisymulacji, rozważanej w ćwiczeniu 65. Ale wtedy sprawdzenie, że dana relacja jest bisymulacją z dokładnością do $\approx$ jest zbyt kosztowne!

Jednym ze sposobów rozwiązania problemu jest ograniczenie się do relacji R spełniających następujący warunek: dla każdej pary $(P, Q) \in R$, $\forall \alpha \in \mathrm{Act}$,

$$\begin{array}{ll} (i) & P \xrightarrow{\alpha} P' \implies \exists Q' \text{ t.że } Q \overset{\widehat{\alpha}}{\Longrightarrow} Q' \text{ i } (P', Q') \in \sim \circ R \circ \approx \\ (ii) & Q \xrightarrow{\alpha} Q' \implies \exists P' \text{ t.że } P \overset{\widehat{\alpha}}{\Longrightarrow} P' \text{ i } (P', Q') \in \approx \circ R \circ \sim \end{array} \tag{24}$$

Np. rozważmy relację zawierającą tylko jedną parę $(\tau.a.0, 0)$, będącą jednym z rozwiązań ostatniego ćwiczenia, i zauważmy, iż została ona wyeliminowana przez powyższy warunek. Niestety, wyeliminowaliśmy za dużo, i w praktyce relacje spełniające (24) nie wystarczają (tak jest np. w przykładzie specyfikacji i implementacji systemu `Zarządca`).

Wybierzemy inne rozwiązanie problemu bisymulacji z dokładnością do $\approx$, oparte na pojęciu *pre-bisymulacji*. Różni się ona od słabej bisymulacji tylko tym, że bierze w pewien sposób pod uwagę liczbę zdarzeń τ wykonanych przez procesy: jeśli para (P, Q) jest w relacji pre-bisymulacji, to zawsze proces Q ma prawo wykonać *co najmniej* tyle zdarzeń τ co proces P, i vice versa, proces P ma prawo wykonać *co najwyżej* tyle τ co Q. pre-bisymulacja jest w praktyce zupełnie wystarczająca dla wykazania poprawności implementacji względem specyfikacji, gdyż w przeważającej liczbie przypadków implementacja wykonuje znacznie więcej τ niż specyfikacja. Przypomnijmy, iż dowodzenie poprawności odbywa się zwykle następująco: i specyfikację i implementację wyrażamy jako procesy, `Imp` i `Spec`, odpowiednio. Następnie implementację `Imp` uważa się za poprawną, gdy

$$\texttt{Spec} \approx \texttt{Imp}.$$

Oto precyzyjna definicja nowego pojęcia:

Definicja 44 *Relację binarną R pomiędzy procesami nazywamy <u>pre-bisymulacją</u>, jeśli dla każdej pary $(P, Q) \in R$, $\forall \alpha \in \mathrm{Act}$,*

$$(i) \quad P \xrightarrow{\alpha} P' \implies \exists Q' \textit{ t.że } Q \overset{\alpha}{\Longrightarrow} Q' \textit{ i } (P', Q') \in R$$

$$(ii) \quad Q \xrightarrow{\alpha} Q' \implies \exists P' \textit{ t.że } (P \xrightarrow{\alpha} P' \textit{ lub } P \xrightarrow{\widehat{\alpha}} P') \textit{ i } (P', Q') \in R.$$

Mówimy, że procesy P, Q są w relacji pre-porządku pre-bisymulacyjnego, ozn. $P \lessapprox Q$, jeśli (P, Q) należy do pewnej pre-bisymulacji.

Ćwiczenie 71 *Pokaż, że relacja* $\lessapprox$ *jest pre-porządkiem (zwrotna i przechodnia) oraz, że jest zachowana przez wszystkie operatory z wyjątkiem wyboru.*

Oczywiście:

$$\sim \;\subsetneq\; \lessapprox \;\subsetneq\; \approx .$$

A nawet więcej: każda silna bisymulacja jest pre-bisymulacją, a ta jest słabą bisymulacją.
Zaletą pre-bisymulacji w porównaniu do słabej bisymulacji jest to, że pozwala ona zdefiniować *pre-bisymulację z dokładnością do* $\lessapprox$ w taki sposób, by metoda dowodowa oparta na niej była poprawna (względem $\lessapprox$), patrz wniosek 47 poniżej.

Definicja 45 *Relację binarną R pomiędzy procesami nazywamy pre-bisymulacją z dokładnością do $\lessapprox$, jeśli dla każdej pary $(P,Q) \in R$, $\forall \alpha \in \mathrm{Act}$,*

$$(i)\;\; P \xrightarrow{\alpha} P' \;\Longrightarrow\; \exists Q' \;\; t.że \;\; Q \overset{\alpha}{\Longrightarrow} Q' \;\; i \;\; (P',Q') \in \;\sim \circ R \circ \lessapprox$$

$$(ii)\;\; Q \xrightarrow{\alpha} Q' \;\Longrightarrow\; \exists P' \;\; t.że \;\; (P \xrightarrow{\alpha} P' \;\; lub \;\; P \xrightarrow{\widehat{\alpha}} P') \;\; i \;\; (P',Q') \in \lessapprox \circ R \circ \lessapprox$$

Lemat 46 *Jeśli R jest pre-bisymulacją z dokładnością do $\lessapprox$, to $\lessapprox \circ R \circ \lessapprox$ jest pre-bisymulacją.*

Ćwiczenie 72 *Udowodnij lemat.*

Wniosek 47 *Jeśli R jest pre-bisymulacją z dokładnością do $\lessapprox$, to $R \subseteq \lessapprox$.*

Ćwiczenie 73 *Czy jeśli $P \approx Q$, to $\mathcal{L}(P) = \mathcal{L}(Q)$? Czy jeśli $P \sim Q$, to $\mathcal{L}(P) = \mathcal{L}(Q)$?*

6 Równoważność obserwacyjna

W tym rozdziale zajmiemy się relacją $=$, którą będziemy nazywać różnie: równość semantyczna, równoważność obserwacyjna, kongruencja obserwacyjna, itd. Jedyny problem z $\approx$ to fakt, że nie jest ona kongruencją (względem $\_ + \_$), zatem naszą intencją jest, aby $=$ była największą kongruencją zawartą w $\approx$. Definicja jest jednak inna i niesie więcej informacji:

Definicja 48 *$P = Q$ jeśli dla każdego $\alpha \in \mathrm{Act}$,*

$$(i)\quad P \xrightarrow{\alpha} P' \implies \exists Q' \ \textit{t.że}\ Q \overset{\alpha}{\Longrightarrow} Q' \ \ i\ \ P' \approx Q'$$
$$(ii)\quad Q \xrightarrow{\alpha} Q' \implies \exists P' \ \textit{t.że}\ P \overset{\alpha}{\Longrightarrow} P' \ \ i\ \ P' \approx Q'.$$

Ćwiczenie 74 *Na podstawie definicji dowiedź inkluzji (23) wspomnianych w poprzednim rozdziale.*

Ćwiczenie 75 *Bezpośrednio z definicji wykaż, że zachodzą prawa (τ_1) – (τ_3) z rozdziału 3.*

Oczywiście $P \neq \tau.P$ w ogólności. Zachodzi natomiast poniższy fakt, pokazujący chyba najlepiej bliskość relacji $=$ i $\approx$:

Fakt 49 *$P \approx Q$ wtw. gdy $P = Q$ lub $P = \tau.Q$ lub $\tau.P = Q$.*

Dowód: $\Longleftarrow$: $= \subseteq \approx$.
$\Longrightarrow$: 3 przypadki:

(i) $P \xrightarrow{\tau} P' \approx Q$, dla pewnego P'. Wtedy $P = \tau.Q$.

(ii) Symetrycznie: $Q \xrightarrow{\tau} Q' \approx P$, dla pewnego Q'. Wtedy $\tau.P = Q$.

(iii) Wpp. $P = Q$.

□

Łatwo widać, bezpośrednio z definicji, że fakty 39 i 41 z poprzedniego rozdziału są prawdziwe.

Przypomnijmy, iż mamy jednoznaczność rozwiązań, sformułowaną w twierdzeniu 5 w rozdziale 3. Dowód tego twierdzenia pozostawiamy Czytelnikowi jako ćwiczenie.

6.1 Kongruencja

Fakt 50 *$P = Q$ wtw. gdy dla każdego procesu R, $P + R \approx Q + R$, przy założeniu, że istnieje $l \in \mathcal{L}$ t. że $l \notin \mathcal{L}(P) \cup \mathcal{L}(Q)$.*

Dowód: $\Longrightarrow$: Relacja $\{(P + R, Q + R) : P = Q\} \cup \approx$ jest słabą bisymulacją.
$\Longleftarrow$: Załóżmy, że $P \neq Q$, czyli dla pewnych α i P', mamy $P \xrightarrow{\alpha} P'$ i dla każdego Q', jeśli $Q \stackrel{\alpha}{\Longrightarrow} Q'$, to $P' \not\approx Q'$. Musimy wskazać R taki, że $P + R \not\approx Q + R$. Gdy $\alpha = \tau$, wystarczy $R \equiv 0$. Wpp. niech $R \equiv l.0$. Powinniśmy pokazać, że jeśli $Q + R \stackrel{\widehat{\alpha}}{\Longrightarrow} Q'$, dla jakiegoś Q', to $P' \not\approx Q'$. Jeśli $Q' \equiv Q + R$, to oczywiście $P' \not\approx Q'$, bo Q' ma l-potomka, a P' nie. Wpp. $Q + R \stackrel{\alpha}{\Longrightarrow} Q'$, zatem $Q \stackrel{\alpha}{\Longrightarrow} Q'$ i znów $P' \not\approx Q'$. $\square$

Założenie, że istnieje $l \in \mathcal{L}$ t. że $l \notin \mathcal{L}(P) \cup \mathcal{L}(Q)$, można w rzeczywistości pominąć. Poniżej będziemy używać faktu 50 tak, jakby zachodził bez tego założenia.

Fakt 51 $=$ *jest kongruencją względem* wszystkich *operatorów.*

Ćwiczenie 76 *Korzystając z faktu 50 dowiedź, że $P + R = Q + R$ gdy $P = Q$.*

Ćwiczenie 77 *Korzystając z faktu 50 wykaż, że $=$ jest* największą *kongruencją zawartą w $\approx$.*

Tak jak dla silnej równoważności, chcielibyśmy posiadać swobodę przekształcania wyrażenia definiującego stałą procesową na inne, obserwacyjnie równoważne. Ale w tym celu musimy udowodnić, że zmieniając wyrażenie definiujące stałą nie zmieniamy znaczenia tej stałej. Twierdzenie 53 poniżej jest analogiczne do twierdzenia 26 z rozdziału 4.

Definicja 52 *Niech E, F będą wyrażeniami, t że $\mathrm{fv}(E), \mathrm{fv}(F) \subseteq \widetilde{X}$. Mówimy, że E i F są obserwacyjnie równoważne, ozn. $E = F$, jeśli dla każdego zbioru procesów $\widetilde{P}$ $E\{\widetilde{P}/\widetilde{X}\} = F\{\widetilde{P}/\widetilde{X}\}$.*

Twierdzenie 53 *Niech $\mathrm{fv}(E), \mathrm{fv}(F) \subseteq \{X\}$. Jeśli*

$$A \stackrel{def}{=} E\{A/X\}$$
$$B \stackrel{def}{=} F\{B/X\}$$
$$E = F$$

to $A = B$.

Dowód: ... $\square$

7 Nierozstrzygalność równoważności procesów

W tym rozdziale ograniczamy nieco język, pozwalając tylko na skończoną sumę; równoważnie mówiąc, mamy tylko binarną sumę _ + _. Pozwalamy również tylko na skończoną liczbę stałych procesowych. Dzięki temu, wyrażenia procesowe są obiektami skończonymi. Zakładamy też, że zbiór nazw mogących występować w wyrażeniach procesowym jest skończony. Język tak ograniczony nazwiemy $\mathcal{E}_{\mathrm{fin}}$.

Mimo ograniczenia, równoważność obserwacyjna (wraz z wszystkimi równoważnościami bisymulacyjnymi, które poznaliśmy dotychczas) jest niestety nierozstrzygalna dla $\mathcal{E}_{\mathrm{fin}}$! Pierwsza część tego rozdziału zawiera dowód nierozstrzygalności. Równoważność obserwacyjna staje się rozstrzygalna dla pewnych fragmentów języka $\mathcal{E}_{\mathrm{fin}}$, o czym wspomnimy następnie.

7.1 Dowód nierozstrzygalności

Zaczniemy od wprowadzenia dobrze znanego problemu nierozstrzygalnego, który następnie zredukujemy do problemu równoważności procesów.

Definicja 54 *Maszyną 2-licznikową nazywamy program składający się z m instrukcji, etykietowanych $l_1, \ldots, l_m$:*

$$\begin{aligned} l_1 &: \mathtt{instr}_1 \\ &\vdots \\ l_{m-1} &: \mathtt{instr}_{m-1} \\ l_m &: \mathtt{halt} \end{aligned}$$

Instrukcje $\mathtt{instr}_1 \ldots \mathtt{instr}_{m-1}$ operują na dwóch zmiennych (licznikach) c_1, c_2 o wartościach naturalnych. Każda z nich należy do jednego z dwóch typów:

$$\textit{typ } I: \quad c_j := c_j + 1;\ \mathtt{goto}\ l_k$$

$$\begin{aligned} \textit{typ } II: \quad & \mathtt{if}\ c_j = 0\ \mathtt{then\ goto}\ l_{k_1} \\ & \mathtt{else}\ c_j := c_j - 1;\ \mathtt{goto}\ l_{k_2} \end{aligned}$$

Działanie programu zaczyna się od instrukcji l_1 : $\mathtt{instr}_1$, przy wartości obydwu zmiennych równych zero: $c_1 = 0, c_2 = 0$. Mówimy, że maszyna zatrzymuje się, jeśli sterowanie znajdzie się po pewnej liczbie kroków przy instrukcji l_m : $\mathtt{halt}$.

Definicja 55 *Problem stopu dla maszyny 2-licznikowej definiuje się nastepująco: dla danej maszyny 2-licznikowej M stwierdzić, czy maszyna ta się zatrzymuje.*

Twierdzenie 56 *Problem stopu dla maszyny* 2*-licznikowej jest nierozstrzygalny.*

Zajmiemy się teraz redukcją powyższego problemu do następującego: dla danych dwóch procesów P i Q stwierdzić, czy $P = Q$? W tym celu musimy zasymulować wystarczająco wiernie działanie maszyny M w procesie P lub Q lub obydwu. Skończone sterowanie nie nastręcza żadnego kłopotu, natomiast cała trudność jest w symulacji działania licznika. Spójrzmy na narzucające się rozwiązanie (i oznacza inkrementację, d dekrementację a z test na zero):

$$
\begin{aligned}
\mathrm{A}_0 &\stackrel{\mathrm{def}}{=} i.\mathrm{A}_1 + z.\mathrm{A}_0 \\
\mathrm{A}_1 &\stackrel{\mathrm{def}}{=} i.\mathrm{A}_2 + d.\mathrm{A}_0 \\
&\ \vdots \\
\mathrm{A}_{n+1} &\stackrel{\mathrm{def}}{=} i.\mathrm{A}_{n+2} + d.\mathrm{A}_n \\
&\ \vdots
\end{aligned}
$$

Graf tranzycyjny procesu A_0 wygląda następująco:

$$
\begin{tikzcd}
\mathrm{A}_0 \arrow[loop left, "z"] \arrow[r, bend left, "i"] & \mathrm{A}_1 \arrow[l, bend left, "d"] \arrow[r, bend left, "i"] & \mathrm{A}_2 \arrow[l, bend left, "d"] \arrow[r, bend left, "i"] & \cdots \arrow[l, bend left, "d"]
\end{tikzcd}
\tag{25}
$$

Oczywiście jest ono nieakceptowalne, ponieważ nie należy ono do naszego języka $\mathcal{E}_{\mathrm{fin}}$, skoro wymaga nieskończonej liczby stałych procesowych.

Drogi Czytelniku. Zanim przeczytasz ciąg dalszy, spróbuj rozwiązać następujące bardzo ciekawe i pouczające ćwiczenie:

Ćwiczenie 78 *Znajdź proces symulujący działanie licznika (tzn. proces o grafie tranzycyjnym równoważnym obserwacyjnie grafowi (25)) , zdefiniowany przy pomocy skończonej liczby stałych procesowych. (Wystarczą* 3 *stałe procesowe.) Wskazówka: proces*

$$B \stackrel{def}{=} i.(B|d.0)$$

jest już prawie *poprawnym rozwiązaniem. Brakuje tylko testu na zero ...*

Oto rozwiązanie:

$$
\begin{aligned}
\mathrm{Z} &\stackrel{\mathrm{def}}{=} z.\mathrm{Z} + i.(\mathrm{A}|a.\mathrm{Z})\backslash a \\
\mathrm{A} &\stackrel{\mathrm{def}}{=} d.\overline{a}.0 + i.(\mathrm{B}|b.\mathrm{A})\backslash b \\
\mathrm{B} &\stackrel{\mathrm{def}}{=} d.\overline{b}.0 + i.(\mathrm{A}|a.\mathrm{B})\backslash a
\end{aligned}
$$

Graf tranzycyjny procesu Z wygląda następująco:

$$
\begin{tikzcd}
0 & \mathrm{Z} \arrow[d, "i"] & \sim & (0|\mathrm{Z})\backslash a & (\overline{a}.0|a.\mathrm{Z})\backslash a \arrow[l, "\tau"'] \\
1 & (\mathrm{A}|a.\mathrm{Z})\backslash a \arrow[d, "i"] \arrow[urrr, "d"] & \sim & ((0|\mathrm{A})\backslash b|a.\mathrm{Z})\backslash a & ((\overline{b}.0|b.\mathrm{A})\backslash b|a.\mathrm{Z})\backslash a \arrow[l, "\tau"'] \\
2 & ((\mathrm{B}|b.\mathrm{A})\backslash b|a.\mathrm{Z})\backslash a \arrow[d, "i"] \arrow[urrr, "d"] & & & \cdots \\
3 & (((\mathrm{A}|a.\mathrm{B})\backslash a|b.\mathrm{A})\backslash b|a.\mathrm{Z})\backslash a \arrow[d, "i"] \arrow[urrr, "d"] & & & \\
\vdots & \vdots & & &
\end{tikzcd}
$$

Wartości licznika równej 0 odpowiada proces Z. Następnie każda inkrementacja polega na zastąpieniu aktywnego procesu Z, A lub B przez większe podwyrażenie. Formalnie, zdefiniujmy wyrażenia procesowe $E_0, E_1, \ldots$ z jedną zmienna wolną X

jak następuje:

$$
\begin{array}{rcl}
E_0 & \equiv & \mathrm{Z} \\
E_1 & \equiv & (X|a.\mathrm{Z})\backslash a \\
E_{n+1} & \equiv & \begin{cases} E_n\{\ (X|b.\mathrm{A})\backslash b\ /\ X\ \}, & \text{gdy } n \text{ nieparzyste} \\ E_n\{\ (X|a.\mathrm{B})\backslash a\ /\ X\ \}, & \text{gdy } n \text{ parzyste} \end{cases}
\end{array}
$$

Ćwiczenie 79 *Pokaż, że relacja*

$$\{(A_n, E_n\{B/X\}) : n \textit{ parzyste}\} \cup \{(A_n, E_n\{A/X\}) : n \textit{ nieparzyste}\}$$

jest pre-bisymulacją z dokładnością do $\lessapprox$.

Twierdzenie 57 *Nierozstrzygalny jest następujący problem: dla danych dwóch procesów P i Q stwierdzić, czy $P \sim Q$?*

Dowód: Redukcja problemu stopu dla maszyny 2-licznikowej. Dla danej maszyny 2-licznikowej M zdefiniujemy parę procesów P_M, Q_M taką, że maszyna M zatrzymuje się dokładnie wtedy i tylko wtedy, gdy $P_M \not\approx Q_M$. Proces P_M składa się z trzech komponentów:

$$P_M \ \equiv\ (\mathrm{Z}[z_1/z, i_1/i, d_1/d] \mid \mathrm{Z}[z_2/z, i_2/i, d_2/d] \mid \mathrm{S}_1)\backslash\{z_j, i_j, d_j : j = 1, 2\}.$$

Pierwsze dwa to dwa liczniki o wartości 0. Komponent S_1 odpowiada za skończone sterowanie: stała procesowa S_i, dla $i = 1, \ldots, m$, będzie oznaczać, że maszyna znajduje się przy instrukcji $l_i : \mathtt{instr}_i$. Stałe S_i, dla $i = 1, \ldots, m-1$, definiujemy zależnie od tego, do którego typu należy instrukcja $\mathtt{instr}_i$. Dla typu I:

$$\mathrm{S}_i \ \stackrel{\text{def}}{=}\ \overline{i}_j.\mathrm{S}_k$$

a dla typu II:

$$\mathrm{S}_i \ \stackrel{\text{def}}{=}\ \overline{z}_j.\mathrm{S}_{k_1} + \overline{d}_j.\mathrm{S}_{k_2}\ .$$

Natomiast dla $i = m$ mamy

$$\mathrm{S}_m \ \stackrel{\text{def}}{=}\ 0.$$

Pozostaje tylko wskazać drugi proces Q_M. Ale ponieważ w P_M zawarliśmy już pełną informację o działaniu M, w Q_M musimy tylko uwzględnić fakt, że pytamy o to, czy M się zatrzymuje. Zatem niech

$$Q_M \ \equiv\ \mathrm{T}, \quad \text{gdzie } \mathrm{T} \ \stackrel{\text{def}}{=}\ \tau.\mathrm{T}.$$

Zauważmy, że obydwa procesy P_M i Q_M są zdeterminowane w następującym sensie: każdy potomek P' każdego z nich ma co najwyżej jedną tranzycję. Ponadto $\mathcal{L}(P_M) = \mathcal{L}(Q_M) = \emptyset$, tzn. wykonują one tylko zdarzenie τ.

Załóżmy, że maszyna M zatrzymuje się po skończonej liczbie kroków. Wtedy dla pewnego N mamy $P_M(\xrightarrow{\tau})^N P'$ i $P' \overset{\tau}{\not\to}$. Czyli P' nie ma odpowiedzi na $\mathrm{T} \xrightarrow{\tau} \mathrm{T}$.

Załóżmy przeciwnie, że maszyna M nie zatrzymuje się. Wtedy istnieje silna bisymulacja pomiędzy P_M i Q_M, której znalezienie pozostawiamy Czytelnikowi. □

Pokazaliśmy nierozstrzygalność silnej równoważności. Natychmiastowym wnioskiem jest nierozstrzygalność wszystkich innych poznanych dotychczas równoważności:

Wniosek 58 *Nierozstrzygalne są również następujące dwa problem:*

- *dla danych dwóch procesów P i Q stwierdzić, czy $P \approx Q$?*
- *dla danych dwóch procesów P i Q stwierdzić, czy $P = Q$?*

Ćwiczenie 80 *Udowodnij wniosek 58.*

Podsumujmy naszą wiedzę na temat (nie)rozstrzygalności słabej i silnej równoważności dla $\mathcal{E}_{\text{fin}}$ i pewnych jego fragmentów:

język	$\sim$	$\approx$
$\mathcal{E}_{\text{fin}}$ bez przemianowania $\_[f]$	nierozstrzygalna	nierozstrzygalna
$BPP : \mathcal{E}_{\text{fin}}$ bez przemianowania $\_[f]$ i zakazu $\_\backslash L$	rozstrzygalna PSPACE-trudna	?
procesy skończeniestanowe: 0, $\alpha.\_$, $\_ + \_$, rekurencja	P	p

7.2 Kilka ćwiczeń

Ćwiczenie 81 *Pokaż, że $P|\tau.Q = \tau.P|Q$.*

Ćwiczenie 82 *Co by się zmieniło, gdyby definicję słabej równoważności bisymulacyjnej oprzeć na nieznacznie zmodyfikowanej definicji słabej bisymulacji:*

Definicja 59 *Relację binarną R pomiędzy procesami nazywamy* słabą bisymulacją, *jeśli dla każdej pary $(P, Q) \in R$, $\forall \alpha \in$* Act,

$$(i)\quad P \xrightarrow{\alpha} P' \implies \exists Q' \ \ t.że\ \ Q \overset{\alpha}{\Longrightarrow} Q' \ \ i\ \ (P', Q') \in R$$
$$(ii)\quad Q \xrightarrow{\alpha} Q' \implies \exists P' \ \ t.że\ \ P \overset{\alpha}{\Longrightarrow} P' \ \ i\ \ (P', Q') \in R.$$

Jedyna różnica to zastąpienie relacji $\stackrel{\widehat{\alpha}}{\Longrightarrow}$ *przez* $\stackrel{\alpha}{\Longrightarrow}$. *Czyli nie pozwalamy na pustą odpowiedź w przypadku* $\alpha = \tau$.

Ćwiczenie 83 *Pokaż, że powyższa modyfikacja pociąga następującą własność:*

$$P \approx Q,\ P \stackrel{\tau}{\longrightarrow} P' \quad \Longrightarrow \exists Q' \ \ t.\dot{z}e\ \ Q \stackrel{\tau}{\longrightarrow} Q' \ \ i \ \ P' \approx Q',$$

przy założeniu, że P *i* Q *mają skończenie wiele* τ*-potomków, tzn. zbiór* $\{P' : P \stackrel{\tau}{\longrightarrow} P'\}$ *jest skończony i podobnie dla* Q.

Czy zatem konsekwencją definicji 59 jest kolaps $\approx\ =\ \sim$?

Ćwiczenie 84 *Pokaż, że wciąż* $\approx\ \neq\ \sim$.

Rozważmy teraz inną możliwą modyfikację definicji słabej bisymulacji. Jak pamiętamy z rozdziału 5 (por. ćwiczenie 65), definicję 35 można usymetrycznić następująco:

Definicja 60 *Relację binarną* R *pomiędzy procesami nazywamy* <u>*silną bisymulacją,*</u> *jeśli dla każdej pary* $(P, Q) \in R$, $\forall t \in \mathcal{L}^*$,

$$\begin{array}{ll} (i) & P \stackrel{t}{\Longrightarrow} P' \quad \Longrightarrow \exists Q' \ \ t.\dot{z}e\ \ Q \stackrel{t}{\Longrightarrow} Q' \ \ i \ \ (P', Q') \in R \\ (ii) & Q \stackrel{t}{\Longrightarrow} Q' \quad \Longrightarrow \exists P' \ \ t.\dot{z}e\ \ P \stackrel{t}{\Longrightarrow} P' \ \ i \ \ (P', Q') \in R. \end{array}$$

Przypomnijmy, ze $\mathcal{L} = \mathrm{Act} \setminus \{\tau\}$. Definicja 60 definiuje dokładnie te same relacje co definicja 35 (por. ćwiczenie 65).

Zastanówmy się nad modyfikacją, polegającą na rozpatrzeniu tylko niepustych ciągów t.

Definicja 61 *Relację binarną* R *pomiędzy procesami nazywamy* <u>*silną bisymulacją,*</u> *jeśli dla każdej pary* $(P, Q) \in R$, $\forall t \in \mathcal{L}^+$,

$$\begin{array}{ll} (i) & P \stackrel{t}{\Longrightarrow} P' \quad \Longrightarrow \exists Q' \ \ t.\dot{z}e\ \ Q \stackrel{t}{\Longrightarrow} Q' \ \ i \ \ (P', Q') \in R \\ (ii) & Q \stackrel{t}{\Longrightarrow} Q' \quad \Longrightarrow \exists P' \ \ t.\dot{z}e\ \ P \stackrel{t}{\Longrightarrow} P' \ \ i \ \ (P', Q') \in R. \end{array}$$

mamy na przykład: $a.0 + \tau.0 \approx a.0$.

Ćwiczenie 85 *Pokaż, że* $\approx$ *nie jest teraz kongruencją względem prefiksu i względem złożenia równoległego.*

Jak widać, znów konsekwencje zmian są miażdżące. Wniosek: pojęcie słabej bisymulacji nie jest ad hoc, jest starannie wybrane, a niektóre narzucające się alternatywy do niczego się nie nadają.

7.2.1 Symulacja

Definicja 62 *Relację binarną R pomiędzy procesami nazywamy silną symulacją, jeśli dla każdej pary $(P, Q) \in R$, $\forall \alpha \in \mathrm{Act}$,*

$$(i) \quad P \xrightarrow{\alpha} P' \implies \exists Q' \ t.\dot{z}e \ Q \xrightarrow{\alpha} Q' \ i \ (P', Q') \in R.$$

Niech $\precsim$ oznacza sumę wszystkich silnych symulacji:

$$\precsim \ = \bigcup_{R \ \text{silna symulacja}} R$$

Ćwiczenie 86 *Pokaż, że $\precsim$ jest pre-porządkiem, tzn. jest zwrotna i przechodnia, oraz że jest pre-kongruencją, tzn. jest zachowywana przez wszystkie operatory.*

Niech $\simeq$ oznacza $\precsim \cap \precsim^{-1}$. Relację $\simeq$ nazywamy *silną rownoważnością symulacyjną.*

Ćwiczenie 87 *Pokaż, że $\simeq$ jest kongruencją.*

Ćwiczenie 88 *Pokaż, że $\sim \subsetneq \simeq$. Tzn. wskaż parę procesów, między którymi są dwie symulacje (w dwie strony), ale nie ma bisymulacji.*

Ćwiczenie 89 *Udowodnij, że R jest silną bisymulacją wtw. gdy R oraz R^{-1} są silnymi symulacjami.*

Definicja 63 *Relację binarną R pomiędzy procesami nazywamy słabą symulacją, jeśli dla każdej pary $(P, Q) \in R$, $\forall \alpha \in \mathrm{Act}$,*

$$(i) \quad P \xrightarrow{\alpha} P' \implies \exists Q' \ t.\dot{z}e \ Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q' \ i \ (P', Q') \in R.$$

Niech $\precapprox$ oznacza sumę wszystkich słabych symulacji:

$$\precapprox \ = \bigcup_{R \ \text{słaba symulacja}} R$$

Ćwiczenie 90 *Czy własność wspomniana w ćwiczeniu 89 zachodzi też dla słabej bisymulacji i symulacji?*

Relację *słabej rownoważności symulacyjnej* $\approxeq$ definiujemy jako $\precapprox \cap \precapprox^{-1}$.

Ćwiczenie 91 *Pokaż, że $\precapprox$ jest pre-kongruencją (a zatem $\approxeq$ jest kongruencją).*

Ćwiczenie 92 *Rozważ następujący przykład:*

$$
\begin{aligned}
A &\stackrel{def}{=} a.b.A \\
S &\stackrel{def}{=} \overline{P}.\overline{V}.S \\
B &\stackrel{def}{=} a.P.b.V.B \\
A_i &\stackrel{def}{=} A[f_i] && i = 1,2 \\
B_i &\stackrel{def}{=} B[f_i] && i = 1,2,
\end{aligned}
$$

gdzie $f_i(a) = a_i, f_i(b) = b_i$, *dla* $i = 1,2$. *Pokaż, że* $A_1|A_2 \not\approx (B_1|B_2|S)\backslash\{P, V\}$, *ale* $A_1|A_2 \approxeq (B_1|B_2|S)\backslash\{P, V\}$.

Zatem $\approx\ \subsetneq\ \approxeq$.

Ćwiczenie 93 *Czy słaba i silna równoważność symulacyjna jest rozstrzygalna?*

8 Bisymulacja, logika i gry

8.1 Hierarchia aproksymantów

Definicja 64 *Definiujemy relacje* $\sim_\kappa$ *, dla* κ *przebiegającego liczby porządkowe, następująco:*

- $P \sim_0 Q$ *zawsze*
- $P \sim_{\kappa+1} Q$ *jeśli dla każdego* $t \in \mathrm{Act}^*$,

$$\begin{array}{ll} (i) & P \xrightarrow{t} P' \implies \exists Q' \ \textit{t.że}\ Q \xrightarrow{t} Q' \ \textit{i}\ P' \sim_\kappa Q' \\ (ii) & Q \xrightarrow{t} Q' \implies \exists P' \ \textit{t.że}\ P \xrightarrow{t} P' \ \textit{i}\ P' \sim_\kappa Q'. \end{array}$$

- *jeśli* κ *jest graniczną liczbą porządkową, to*

$$\sim_\kappa \ = \bigcap_{\lambda<\kappa} \sim_\lambda \ .$$

Uwaga 65 *Każda relacją* $\sim_\kappa$ *jest równoważnością.* $\sim_1$ *to dokładnie równość śladów.*

Fakt 66 $\lambda < \kappa \implies \ \sim_\lambda \ \supseteq \ \sim_\kappa$.

Fakt 67 $\sim \ = \bigcap_\kappa \sim_\kappa$.

Pierwszy fakt wynika natychmiast z monotoniczności operatora definiującego bisymulację, natomiast drugi jest instancją twierdzenia Knastera-Tarskiego o punkcie stałym w kracie zupełnej[3].

Fakt 68 $\sim_0 \ \supsetneq \ \sim_1 \ \supsetneq \dots$. *Dla* $i \in \omega$, $\sim_i \ \supsetneq \ \sim_\omega$.

Dowód: Definiujemy procesy P_i, Q_i, dla $i \in \omega$, jak następuje:

$$\begin{array}{ll} P_0 \equiv b.0 & Q_0 \equiv c.0 \\ P_{i+1} \equiv a.(P_i + Q_i) & Q_{i+1} \equiv a.P_i + a.Q_i \end{array}$$

Teraz łatwo jest pokazać indukcyjnie względem i, że

$$\begin{array}{l} P_i \ \sim_i \ Q_i \ \sim_i \ P_i + Q_i \\ P_i \ \not\sim_{i+1} \ Q_i \ \not\sim_{i+1} \ P_i + Q_i \ \not\sim_{i+1} \ P_i \end{array}$$

□

[3]Zakładamy tutaj, że mamy *zbiór* wszystkich procesów.

Ćwiczenie 94 *Dokończ dowód powyższego faktu.*

Fakt 69 $\sim_\omega \supsetneq \sim_{\omega+1}$.

Dowód: Zdefiniujemy procesy R, R' takie, że $R \sim_\omega R'$ ale $R \not\sim_{\omega+1} R'$.

$$
\begin{aligned}
R_0 &\equiv \sum\nolimits_{i\in\{0,1,2,\dots\}} P_i \\
R_1 &\equiv Q_0 + \sum\nolimits_{i\in\{1,2,\dots\}} P_i \\
R_1 &\equiv Q_0 + Q_1 + \sum\nolimits_{i\in\{2,3,\dots\}} P_i \\
&\dots \\
R_\omega &\equiv \sum\nolimits_{i\in\omega} Q_i
\end{aligned}
$$

$$
\begin{aligned}
R &\equiv \sum\nolimits_{i\in\omega} a.R_i \\
R' &\equiv R + a.R_\omega
\end{aligned}
$$

Teraz wystarczy pokazać kolejno:

- $R_\omega \sim_i R_i$ ale $R_\omega \not\sim_{i+1} R_i$, dla $i \in \omega$
- $R \sim_\omega R'$
- $R_\omega \not\sim_\omega R_i$, dla $i \in \omega$
- $R \not\sim_{\omega+1} R'$.

□

Ćwiczenie 95 *Dokończ dowód powyższego faktu.*

Uwaga 70 *Hierarchię relacji* $\sim_\kappa$ *oparliśmy na „długich krokach”* $\xrightarrow{t}$ *, dla* $t \in \mathrm{Act}^*$. *Zauważmy, że rozsądną alternatywą jest oparcie jej na „krótkich krokach”, tj. na tranzycjach* $\xrightarrow{\alpha}$ *, dla* $\alpha \in \mathrm{Act}^*$.

Ćwiczenie 96 *Pojęcie silnej bisymulacji pozostawało takie samo niezależnie od tego, czy formułowaliśmy je przy użyciu „długich” czy też „krótkich kroków”. Czy alternatywna definicja hierarchii* $\sim_\kappa$ *jest równoważna definicji 64 ?*

Ćwiczenie 97 *Wszystkie powyższe fakty pozostają prawdziwe dla alternatywnej definicji* $\sim_\kappa$ *. Udowodnij fakty 68 i 69.*

W ogólności, $\sim_\kappa$ nie jest równe $\sim$ dla żadnej liczby porządkowej κ. Natomiast przy pewnym rozsądnym założeniu mamy $\sim_\omega\ =\ \sim_{\omega+1}$, a zatem $\sim_\omega\ =\ \sim$.

Definicja 71 *Proces P ma* *<u>własność skończonego obrazu</u>, jeśli dla każdego $\alpha \in \mathrm{Act}$, zbiór jego α-potomków $\{P' : P \xrightarrow{\alpha} P'\}$ jest skończony.*

Fakt 72 *Jeśli procesy P i Q oraz ich wszyscy potomkowie mają własność skończonego obrazu, to $P \sim_\omega Q \implies P \sim_{\omega+1} Q$.*

Dowód: Wiemy, że $P \sim_\omega Q$, czyli $P \sim_i Q$ dla wszystkich $i \in \omega$.

Załóżmy, że $P \xrightarrow{\alpha} P'$. Zatem, dla każdego i, mamy $Q \xrightarrow{\alpha} Q_i$ i $P' \sim_i Q_i$. Ponieważ na mocy założenia proces Q posiada tylko skończenie wiele α-potomków, co najmniej jeden z nich, powiedzmy Q', jest równy Q_i dla nieskończenie wielu i. Teraz łatwo pokazać, że $P' \sim_\omega Q'$.

Podobnie pokazujemy, że gdy $Q \xrightarrow{\alpha} Q'$, to dla pewnego P' mamy $P \xrightarrow{\alpha} P'$ i $P' \sim_\omega Q'$. Czyli dowiedliśmy $P' \sim_{\omega+1} Q'$. □

Innymi słowy: jeśli w języku pełnym $\mathcal{E}^+$ ograniczymy się do skończonych sum, to $\sim\ =\ \sim_\omega$.

Uwaga 73 *Łatwo wyobrazić sobie hierarchię „słabych" relacji $\approx_\kappa$, aproksymujących słabą równoważność $\approx$. Niestety, dla $\approx_\kappa$ fakt 72 nie zachodzi.*

Ćwiczenie 98 *Przypomnij sobie wszystkie (równoważne) definicje słabej bisymulacji. Na ile „rozsądnych" sposobów potrafisz zdefiniować $\approx_\kappa$? Czy definicje te są równoważne?*

8.2 Logika modalna (Hennessy-Milner logic) a równoważność bisymulacyjna

Definicja 74 (Logika $\mathrm{HML}^\sim$) *Składnia formuł logiki $\mathrm{HML}^\sim$:*

$$
\begin{array}{rcll}
F & ::= & \langle\alpha\rangle F & \alpha \in \mathrm{Act} \\
 & & \neg F & \\
 & & \bigwedge_{i\in I} F_i & \textit{koniunkcja.}
\end{array}
$$

Formuła $\langle\alpha\rangle F$ dotyczy procesu (stanu) i mówi, że ma on α-potomka, w którym prawdziwa jest formuła F. Oto precyzyjna semantyka:

Definicja 75

$$
\begin{array}{ll}
P \models \langle\alpha\rangle F & \textit{jeśli dla pewnego } P',\ P \xrightarrow{\alpha} P' \textit{ i } P' \models F \\
P \models \neg F & \textit{jeśli nie zachodzi } P \models F \\
P \models \bigwedge_{i\in I} F_i & \textit{jeśli dla każdego } i,\ P \models F_i.
\end{array}
$$

Użyteczne są następujące skróty:

$$
\begin{array}{ll}
\texttt{true} \stackrel{\text{def}}{=} \bigwedge_{i\in\emptyset} F_i & \texttt{false} \stackrel{\text{def}}{=} \neg\texttt{true} \\
F_1 \wedge F_2 \stackrel{\text{def}}{=} \bigwedge_{i\in\{1,2\}} F_i & \bigvee_{i\in \mathrm{I}} F_i \stackrel{\text{def}}{=} \neg\bigwedge_{i\in \mathrm{I}} \neg F_i \\
\langle\alpha_1 \dots \alpha_k\rangle F \stackrel{\text{def}}{=} \langle\alpha_1\rangle \dots \langle\alpha_k\rangle F & [\alpha_1 \dots \alpha_k] F \stackrel{\text{def}}{=} \neg\langle\alpha_1 \dots \alpha_k\rangle\neg F
\end{array}
$$

Twierdzenie 76 $P \sim Q$ *wtw, gdy* $(\forall F \in \mathrm{HML}^{\sim}\ .\ P \models F \iff Q \models F)$.

Jest to najważniejsze twierdzenie w tym rozdziale. Mówi ono, że $\mathrm{HML}^{\sim}$ jest logiką charakterystyczną dla silnej równoważności. Istnieje wiele innych logik o tej własności, gdyż twierdzenie to jest prawdziwe dla każdej logiki bogatszej niż $\mathrm{HML}^{\sim}$, która nie potrafi odróżnić systemów silnie równoważnych. Zaletą $\mathrm{HML}^{\sim}$ jest jej prostota. Twierdzenie 76 jest bezpośrednim wnioskiem z twierdzenia 78, które sformułujemy poniżej.

Definicja 77 *Głębokość formuły modalnej definiujemy nastepująco:*

$$
\begin{array}{rcl}
\mathit{depth}(\langle t\rangle F) & = & \mathit{depth}(F) + 1, \quad t \in \mathrm{Act}^* \\
\mathit{depth}(\neg F) & = & \mathit{depth}(F) \\
\mathit{depth}(\bigwedge_{i\in I} F_i) & = & \textit{kres górny zbioru } \{\mathit{depth}(F_i) : i \in I\}
\end{array}
$$

W szczególności, $\mathit{depth}(\texttt{true}) = \mathit{depth}(\texttt{false}) = 0$. *Niech* $\mathrm{HML}^{\sim}_{\kappa}$ *oznacza fragment logiki* $\mathrm{HML}^{\sim}$ *zawierający formuły o głębokości nie większej niż* κ*:*

$$
\mathrm{HML}^{\sim}_{\kappa} \stackrel{\textit{def}}{=} \{F \in \mathrm{HML}^{\sim} : \mathit{depth}(F) \leq \kappa\}.
$$

Twierdzenie 78 *Dla dowolnej liczby porządkowej* κ, $P \sim_{\kappa} Q$ *wtw, gdy* $(\forall F \in \mathrm{HML}^{\sim}_{\kappa}\ .\ P \models F \iff Q \models F)$.

Dowód: Dowód przez indukcję pozaskończoną po κ. Załóżmy tezę dla wszystkich $\lambda < \kappa$.

„$\Rightarrow$": załóżmy, że $P \sim_\kappa Q$ i $P \models F$, $\mathrm{depth}(F) \leq \kappa$. Rozpatrzymy tylko przypadek $F \equiv \langle t \rangle F'$; pozostałe przypadki oczywiste. Niech $\lambda := \mathrm{depth}(F')$; oczywiście $\lambda < \kappa$. Ponieważ $P \models F$, wiemy, że $P \xrightarrow{t} P'$ dla pewnego P', takiego, że $P' \models F'$. Ponieważ $\lambda < \kappa$, na pewno $P \sim_{\lambda+1} Q$. Zatem $Q \xrightarrow{t} Q'$ dla pewnego Q', takiego, że $P' \sim_\lambda Q'$. Z założenia indukcyjnego, $Q' \models F'$, a więc $Q \models \langle t \rangle F' \equiv F$.

„$\Leftarrow$": załóżmy, że $P \not\sim_\kappa Q$. Skonstruujemy formułę $F \in \mathrm{HML}_\kappa^\sim$ odróżniającą P i Q, tzn. taką, że $P \models F$ a zarazem $Q \not\models F$.

a) Załóżmy najpierw, że κ jest następnikiem jakiejś liczby porządkowej, $\kappa = \lambda + 1$. Zatem bez utraty ogólności wiemy, że istnieją P' oraz $t \in \mathrm{Act}^*$ takie, że $P \xrightarrow{t} P'$ oraz dla każdego Q', jeśli $Q \xrightarrow{t} Q'$, to $P' \not\sim_\lambda Q'$.[4] Niech $\{Q_i : i \in \mathrm{I}\}$ oznacza zbiór wszystkich t-potomków procesu Q. Z założenia indukcyjnego dla każdego i istnieje formuła $F_i \in \mathrm{HML}_\lambda^\sim$ taka, że $P' \models F_i$ a zarazem $Q_i \not\models F_i$. Niech $F := \langle t \rangle \bigwedge_{i \in II} F_i$; $\mathrm{depth}(F) \leq \lambda + 1 = \kappa$. Oczywiście $P \models F$, natomiast $Q \not\models F$.

b) Teraz załóżmy, że κ jest liczbą graniczną. Zatem $P \not\sim_\kappa Q$ dla pewnej $\lambda < \kappa$. Z założenia indukcyjnego $P \models F$ a zarazem $Q \not\models F$, dla pewnej formuły $F \in \mathrm{HML}_\lambda^\sim \subseteq \mathrm{HML}_\kappa^\sim$. (zauważmy, że dla odróżnienia P i Q wystarczają już formuły należące do $\bigcup_{\lambda<\kappa} \mathrm{HML}_\lambda^\sim \subsetneq \mathrm{HML}_\kappa^\sim$). □

Ćwiczenie 99 *Czy twierdzenie 78 pozostaje prawdziwe dla alternatywnej definicji* $\sim_\kappa$ *zaproponowanej w uwadze 70? Jeśli nie, to jak trzeba zmodyfikować definicję 77?*

Ćwiczenie 100 *Uważny Czytelnik zauważył zapewne, że w punkcie b) w dowodzie twierdzenia 78, budując formułę rozróżniającą dwa procesy nie korzystamy wogóle z negacji! Czy negacja jest więc zbędna czy też może w rzeczywistości korzystamy z niej w sposób ukryty?*

Dowód twierdzenia 76: Bezpośredni wniosek z twierdzenia 78 i faktu 67. □

Ćwiczenie 101 *Znajdź najpłytszą formułę* F_i *odróżniającą procesy* P_i *i* Q_i*, dla* $i \in \omega$*, tzn. taką, że* $P_i \models F_i$ *i zarazem* $Q_i \not\models F_i$*.*

Ćwiczenie 102 *Wyobraźmy sobie, że faktycznie rezygnujemy z negacji (por. ćwiczenie 100). Czym należy zastąpić relację* $\sim$ *w twierdzeniu 76 oraz relację* $\sim_\kappa$ *w twierdzeniu 78 aby twierdzenia te były znów prawdziwe?*

[4] Albo istnieje Q' oraz t takie, że $Q \xrightarrow{t} Q'$ oraz dla każdego P', jeśli $P \xrightarrow{t} P'$, to $P' \not\sim_\lambda Q'$. Przypadek ten pomijamy a jego dowód jest idealnie symetryczny.

Uwaga 79 *Twierdzenia 76 i 78 stosują się do dowolnego systemu tranzycyjnego, gdyż* $\sim_\kappa$ *i relacja spełniania formuł są zdefiniowane tylko przy pomocy relacji tranzycji* $\stackrel{t}{\longrightarrow}$ *.*

Ćwiczenie 103 *Zaproponuj modyfikację* $\mathrm{HML}^{\approx}$ *logiki* $\mathrm{HML}^{\sim}$ *opartą na słabych tranzycjach* $\stackrel{t}{\Longrightarrow}$ *.*

Wniosek 80 *Natychmiastowy wniosek płynący z uwagi 79: obydwa twierdzenia 76 i 78 zachodzą również dla słabej równoważności i (odpowiednio zdefiniowanej) słabej logiki* $\mathrm{HML}^{\approx}$ *oraz dla odpowiednio dobranych słabych aproksymantów* $\approx_\kappa$ *(por. ćwiczenie 98).*

Logika modalna $\mathrm{HML}^{\sim}$ jest infinitarna, zatem niezbyt użyteczna w praktyce. W niektórych przypadkach wystarczy ograniczyć się do fragmentu finitarnego $\mathrm{HML}^{\sim}_{\mathrm{fin}} \subsetneq \mathrm{HML}^{\sim}$, w którym zamiast ogólnej koniunkcji mamy formułę `true` oraz koniunkcję binarną $F_1 \wedge F_2$.

Twierdzenie 81 *Jeśli procesy* P *i* Q *oraz ich wszyscy potomkowie mają własność skończonego obrazu, to* $P \sim Q$ *wtw, gdy* $(\forall F \in \mathrm{HML}^{\sim}_{fin} \,.\, P \models F \iff Q \models F)$*.*

Dowód: Wystarczy pokazać tylko jedną implikację „$\Leftarrow$”. Załóżmy, że $P \not\sim Q$, a więc na mocy faktu 72, $P \not\sim_k Q$ dla pewnego $k \in \omega$. Z własności skończonego obrazu wynika, że formuła F rozróżniająca P i Q, którą skonstruowaliśmy w punkcie b) w dowodzie twierdzenia 78, należy do $\mathrm{HML}^{\sim}_{\mathrm{fin}}$. □

8.3 Gry a równoważność bisymulacyjna

Uściślimy teraz nieprecyzyjne dotychczas pojęcie gry bisymulacyjnej pomiędzy Kowalskim a Nowakiem. Przypomnijmy: celem Kowalskiego jest wykazanie, iż dwa procesy są równoważne a celem jego oponenta jest dowiedzenie, iż się różnią.

Załóżmy, ze dany jest system tranzycyjny, składający się ze zbioru procesów (stanów) $\mathcal{P}$ oraz z relacji tranzycji $\stackrel{\alpha}{\longrightarrow}$, dla każdego $\alpha \in \mathrm{Act}$. *Konfiguracjami* gry będą elementy zbioru $V = V_N \cup V_K$, gdzie

$$\begin{aligned} V_N &= \mathcal{P} \times \mathcal{P} \\ V_K &= \mathcal{P} \times \mathcal{P} \times \mathrm{Act} \times \{\mathrm{l}, \mathrm{r}\}. \end{aligned}$$

W konfiguracji należącej do zbioru V_N kolej na ruch Nowaka; w konfiguracji ze zbioru V_K rusza się Kowalski. W konfiguracji $(P, Q) \in V_N$, Nowak może wykonać ruch prowadzący do konfiguracji $(P', Q, \alpha, \mathrm{l}) \in V_K$ pod warunkiem, że $P \stackrel{\alpha}{\longrightarrow} Q$. Symetrycznie, Nowak może wykonać ruch z konfiguracji (P, Q) do $(P, Q', \alpha, \mathrm{r}) \in V_K$ o ile

$Q \xrightarrow{\alpha} Q'$. Jak widać, konfiguracje z V_K zawierają informację o ostatnim ruchu Nowaka, potrzebną Kowalskiemu do wykonania poprawnej odpowiedzi. Kowalski może wykonać ruch z $(P', Q, \alpha, \mathrm{l}) \in V_K$ do $(P', Q') \in V_N$, o ile $Q \xrightarrow{\alpha} Q'$. Symetrycznie, może on wykonać ruch z $(P, Q', \alpha, \mathrm{r})$ do $(P', Q') \in V_N$, o ile $P \xrightarrow{\alpha} P'$. Otrzymujemy w ten sposób relację przejścia $R \subseteq V \times V$, a dokładniej $R \subseteq V_N \times V_K \cup V_K \times V_N$, opisującą wszystkie dozwolone ruchy graczy.

Gra rozpoczyna się w wybranej *konfiguracji początkowej* $v_0 = (P_0, Q_0) \in V_N$. *Rozgrywką* w grze nazywamy dowolny skończony lub nieskończony ciąg konfiguracji $r = v_0 v_1 v_2 \ldots$ taki, że

(i) $(v_i, v_{i+1}) \in R$,

(ii) jeśli ciąg r jest skończony, $r = v_0 v_1 v_2 \ldots v_m$, to nie istnieje $v' \in V$ takie, że $(v_m, v') \in R$ (rozgrywka kończy się tylko wtedy, gdy jeden z graczy nie ma żadnego ruchu).

Jeśli rozgrywka r jest skończona, $r = v_0 v_1 v_2 \ldots v_m$, to przegrywa ten gracz, który powinien wykonać ruch z v_m. W przeciwnym przypadku, tzn. gdy rozgrywka jest nieskończona, wygrywa Kowalski.

Niech $V^* V_K \subseteq V^+$ oznacza zbiór niepustych skończonych ciągów konfiguracji takich, że ostatnia konfiguracja należy do V_K. *Strategia* Kowalskiego to dowolna funkcja częściowa $s : V^* V_K \to V_N$ taka, że jeśli $s(rv) = v'$, to $(v, v') \in R$ (ruch wybrany przez strategię jest zawsze dozwolony). Mówimy, że Kowalski stosuje strategię s w rozgrywce $r = v_0 v_1 \ldots$ jeśli dla każdego v_i, jeśli $v_i \in V_K$ to albo v_i jest ostatnią konfiguracją w r, albo strategia jest określona dla $v_0 \ldots v_i$ i $s(v_0 \ldots v_i) = v_{i+1}$. Podobnie definiujemy strategię Nowaka.

Mówimy, ze strategia s Kowalskiego (Nowaka) jest *wygrywająca* jeśli Kowalski (Nowak) wygrywa każdą rozgrywkę, w której stosuje strategię s.

Twierdzenie 82

(i) $P_0 \sim Q_0$ wtw, gdy Kowalski ma strategię wygrywającą.

(ii) $P_0 \nsim Q_0$ wtw, gdy Nowak ma strategię wygrywającą.

Ćwiczenie 104 *Dowód twierdzenia pozostawiamy jako ćwiczenie Czytelnikowi.*

Bezpośredni wniosek z twierdzenia 82: gra bisymulacyjna jest zawsze *zdeterminowana*, tzn. dokładnie jeden z graczy ma strategię wygrywającą.

Ćwiczenie 105 *Czy twierdzenie 82 pozostanie prawdziwe, gdy zmienimy definicję rozgrywki usuwając punkt (ii). Umówmy się, że „niedokończona” rozgrywka oznacza poddanie się gracza, którego konfiguracja jest na końcu rozgrywki, a zatem wygraną przeciwnika.*

Ćwiczenie 106 *Zaproponuj grę dla* $\sim_\kappa$.

Ćwiczenie 107 *Zaproponuj grę dla równoważności symulacyjnej oraz dla przeporządku symulacyjnego.*

8.4 Logika modalna $\mathrm{HML}^{\sim}$ a gry

Ćwiczenie 108 *Zaproponuj grę pomiędzy dwoma graczami, taką, że jeden z nich ma strategię wygrywającą dokładnie wtedy i tylko wtedy, gdy* $P \vDash F$*, dla danych* P *i* F.