2 Język CCS

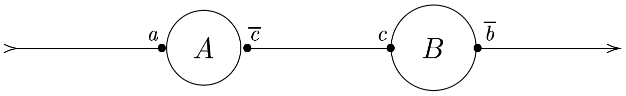
Czas na precyzyjną definicję składni i semantyki naszego języka.

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:



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

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

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

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

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

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

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

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

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):

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

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:

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

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

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

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

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

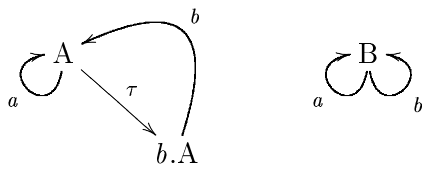
gdzie

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

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 X *oznacza dany zbiór zmiennych;* $X, Y, \ldots \in \mathrm{X}$. *Niech* K *oznacza zbiór stałych;* $A.B, \ldots \in \mathrm{K}$. *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:*

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

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:

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

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 $0 \stackrel{\text{def}}{=} \sum_{i\in\emptyset} E_i$.

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:

$$
\begin{array}{lll}
\dfrac{}{\alpha.E \xrightarrow{\alpha} E} & \dfrac{E_j \xrightarrow{\alpha} E'}{\sum_{i\in \mathrm{I}} E_i \xrightarrow{\alpha} E'}\ (j \in \mathrm{I}) & \dfrac{(\mathrm{A} \stackrel{\mathrm{def}}{=} E) \qquad E \xrightarrow{\alpha} E'}{\mathrm{A} \xrightarrow{\alpha} E'} \\[2ex]
\dfrac{E \xrightarrow{\alpha} E'}{E|F \xrightarrow{\alpha} E'|F} & \dfrac{F \xrightarrow{\alpha} F'}{E|F \xrightarrow{\alpha} E|F'} & \dfrac{E \xrightarrow{a} E' \qquad F \xrightarrow{\overline{a}} F'}{E|F \xrightarrow{\tau} E'|F'} \\[2ex]
\dfrac{E \xrightarrow{\alpha} E'}{E\backslash L \xrightarrow{\alpha} E'\backslash L}\ (\alpha, \overline{\alpha} \notin L) & \dfrac{E \xrightarrow{\alpha} E'}{E[f] \xrightarrow{f(\alpha)} E'[f]} &
\end{array}
$$

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 $(a.E + b.0|\overline{a}.F)\backslash a \xrightarrow{\tau} (E|F)\backslash a$:

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

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:

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

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:

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

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:*

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

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 $\mathcal{E}$:

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

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*

if b **then** E_1 **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	$\longmapsto$	$\widehat{F}$
$\mathrm{A}(e_1, \ldots, e_k)$		$\mathrm{A}_{e_1,\ldots,e_k}$
$a(x).E$		$\sum_{v\in\mathcal{V}} a_v.\widehat{E\{v/x\}}$
$\overline{a}(e).E$		$\overline{a}_e.\widehat{E}$
$\tau.E$		$\tau.\widehat{E}$
$E_1 \mid E_2$		$\widehat{E_1} \mid \widehat{E_2}$
$\sum_{i\in\mathrm{I}} E_i$		$\sum_{i\in\mathrm{I}} \widehat{E_i}$
$E \backslash L$		$\widehat{E}\backslash\{l_v : l \in L, v \in \mathcal{V}\}$
$E[f]$		$\widehat{E}[\widehat{f}]$, 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:*

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

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

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

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.*