Program constructs

In this section we construct more complex programs from existing programs. Three type of program constructs are allowed: sequence, selection and loop. Their definitions are given in the way that they suit the previously introduced definition of program. Program constructs can be described by their structogram.

1 Sequence

In case of sequence, we execute two programs after each other. If the execution of the first program from a given state is infinite or it terminates with a failure, then the second program cannot continue the execution. In this case the sequence, that is generated by the first program, is a possible execution of the sequence as well.
Let us call a state of an execution "joining state", where the first program of a sequence terminates and the second program starts its execution from the same state. We would not like, for example, the sequence of programs $x := x + 1$ and $x := x + 2$ (where x is an integer) to assign $< \{x{:}5\}, \{x{:}6\}, \{x{:}6\}, \{x{:}8\} >$ to the state $\{x{:}5\}$. Therefore, in case we have finite number of "joining states", we leave out one of them from the sequence.

Notation: *Let $\tau(\alpha)$ denote the last element of a finite sequence α. In other words, in case α is a finite sequence, $\tau(\alpha) = \alpha_{|\alpha|}$.*

Notation: *Let A be any statespace. Let $\alpha \in \bar{A}^*$ and $\beta \in (\bar{A} \cup \{fail\})^{**}$ be not empty sequences such that $\tau(\alpha) = \beta_1$. Then, let $\alpha \otimes \beta$ denote the sequence we get by eliminating the first element of β in the concatenation of sequences α and β.*
Let us generalise our notation for the case of $n \in \mathbb{N}^+$ number or even infinite number of sequences. After concatenating the sequences, we have to eliminate one out of the finite number of successive repeating "joining states".

Example 1: $A = \{1, 2, 3, 4\}$. *Then*

$$\otimes_4(< 1, 2, 3, 1 >, < 1, 2, 3, 1 >, < 1, 2, 3, 1 >, < 1, 2, 3, 1 >) =< 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1 >$$

$$\otimes_4(< 1 >, < 1 >, < 1 >, < 1, 2, 3, 1 >) =< 1, 1, 1, 2, 3, 1 >$$

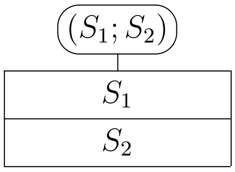
$$\otimes_\infty(< 1 >, < 1 >, < 1 >, < 1 >, \ldots) =< 1, 1, \ldots >$$

$$\otimes_\infty(< 1, 2, 3, 1 >, < 1, 2, 3, 1 >, < 1, 4 >, < 4 >, < 4 >, < 4 >, \ldots) =< 1, 2, 3, 1, 2, 3, 1, 4, 4, 4, \ldots >$$

Definition: *Let A be a common base-statespace of programs S_1 and S_2. The relation $(S_1; S_2)$ called the sequence of programs S_1 and S_2, if*

$$
\begin{aligned}
(S_1; S_2)(a) = &\{\alpha \in \bar{A}^{\infty} \mid \alpha \in S_1(a)\} \cup \\
&\{\alpha \in (\bar{A} \cup \{fail\})^{*} \mid \alpha \in S_1(a) \wedge \alpha_{|\alpha|} = fail\} \cup \\
&\{\gamma \in (\bar{A} \cup \{fail\})^{**} \mid \gamma = \alpha \otimes \beta \wedge \alpha \in S_1(a) \wedge |\alpha| < \infty \wedge \alpha_{|\alpha|} \neq fail \wedge \beta \in S_2(\alpha_{|\alpha|})\}
\end{aligned}
$$

The structogram of the sequence:



Theorem: *Let A be a common base-statespace of programs S_1 and S_2. The sequence $(S_1; S_2)$ is a program.*

Theorem: *Let A be a common base-statespace of programs S_1 and S_2. Let S denote the sequence $(S_1; S_2)$. Then*

$$p(S) = p(S_2) \odot p(S_1)$$

2 Selection

Definition: *Let A be a common base-statespace of $S_1, \dots S_n$ programs. Let $\pi_1, \dots \pi_n \in A \to \mathbb{L}$ be logical functions. Then $(\pi_1{:}S_1, \dots \pi_n{:}S_n) \subseteq A \times (\bar{A} \cup \{fail\})^{**}$ relation is called selection constituted from programs S_i and restricted by conditions π_i. The selection is denoted by IF and*

$$\forall a \in A : IF(a) = \omega_0(a) \cup \bigcup_{i=1}^{n} \omega_i(a)$$

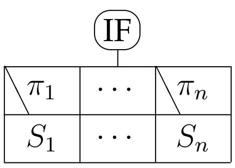
where $\forall i \in [1..n]$:

$$\omega_i(a) = \begin{cases} S_i(a), & \text{if } a \in \mathcal{D}_{\pi_i} \wedge \pi_i(a) \\ \emptyset, & \text{if } a \in \mathcal{D}_{\pi_i} \wedge \neg\pi_i(a) \\ \{< a, fail >\}, & \text{if } a \notin \mathcal{D}_{\pi_i} \end{cases}$$

and

$$\omega_0(a) = \begin{cases} \{< a, fail >\}, & \text{if } \forall i \in [1..n] : (a \in \mathcal{D}_{\pi_i} \wedge \neg\pi_i(a)) \\ \emptyset, & \text{otherwise} \end{cases}$$

The structogram of the selection:



In the literature, the following notation is also used to describe the selection $(\pi_1{:}S_1, \ldots \pi_n{:}S_n)$:

$$
\begin{array}{l}
\textbf{if} \\
\quad \pi_1 \to S_1 \,\Box \\
\quad \ldots \\
\quad \pi_{n-1} \to S_{n-1} \,\Box \\
\quad \pi_n \to S_n \\
\textbf{fi}
\end{array}
$$

Theorem: *Let A a common base-statespace of programs $S_1, \ldots S_n$. Let $\pi_1, \ldots \pi_n \in A \to \mathbb{L}$ be logical functions. Then the selection $IF = (\pi_1{:}S_1, \ldots \pi_n{:}S_n)$ is a program.*

Theorem: *Let IF denote the $(\pi_1{:}S_1, \ldots \pi_n{:}S_n)$ selection constituted of programs $S_1, \ldots S_n$ and restricted by logical functions $\pi_1, \ldots \pi_n \in A \to \mathbb{L}$. Then*

$$\mathcal{D}_{p(IF)} = \{a \in A \mid a \in \bigcap_{i=1}^{n} \mathcal{D}_{\pi_i} \wedge a \in \bigcup_{i=1}^{n} \lceil \pi_i \rceil \wedge \forall i \in [1..n] : a \in \lceil \pi_i \rceil \implies a \in \mathcal{D}_{p(S_i)}\}$$

and

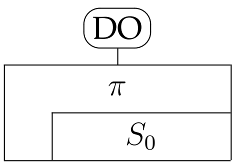
$$\forall a \in \mathcal{D}_{p(IF)}\colon p(IF)(a) = \bigcup_{i=1}^{n} p(S_i)|_{\lceil \pi_i \rceil}$$

3 Loop

Definition: *Let $\pi \in A \to \mathbb{L}$ be a logical function and $S_0 \subseteq A \times (\bar{A} \cup \{fail\})^{**}$ be a program. $DO \subseteq A \times (\bar{A} \cup \{fail\})^{**}$ relation is called loop constituted of program S_0 and logical function π and denoted by (π, S_0), if $\forall a \in A$:*

$$DO(a) = \begin{cases} (S_0; DO)(a) & \textit{if} \quad a \in \mathcal{D}_\pi \wedge \pi(a) \\ \{< a >\} & \textit{if} \quad a \in \mathcal{D}_\pi \wedge \neg\pi(a) \\ \{< a, fail >\} & \textit{if} \quad a \notin \mathcal{D}_\pi \end{cases}$$

The structogram of the loop:



In the literature, the following notation is also used to describe the loop (π, S_0):

$$
\begin{array}{l}
\textbf{while} \;\; \pi \;\; \textbf{do} \\
\quad S_0 \\
\textbf{od}
\end{array}
$$

The loop also can be defined in the way that is similar to the thoughts in the definition of sequence and selection.

Definition: $\forall a \in A$:

$$DO(a) = \begin{cases} \{<a, fail>\}, & \textit{if } a \notin \mathcal{D}_\pi \\ \{<a>\}, & \textit{if } a \in \mathcal{D}_\pi \wedge \neg\pi(a) \\ \{\alpha \in (\bar{A} \cup \{fail\})^{**} \mid \exists \alpha^1, \ldots, \alpha^n \in (\bar{A} \cup \{fail\})^{**} \colon \alpha = \otimes_n(\alpha^1, \ldots, \alpha^n) \wedge & \\ \alpha^1 \in S_0(a) \wedge \forall i \in [1..n-1] \colon (\alpha^i \in \bar{A}^* \wedge \tau(\alpha^i) \in \lceil\pi\rceil \wedge \alpha^{i+1} \in S_0(\tau(\alpha^i))) \wedge & \\ ((\alpha^n \in \bar{A}^\infty \vee (\alpha^n \in (\bar{A} \cup \{fail\})^* \wedge \tau(\alpha^n) = fail) \vee & \\ (\alpha^n \in \bar{A}^* \wedge \tau(\alpha^n) \in \mathcal{D}_\pi \wedge \tau(\alpha^n) \notin \lceil\pi\rceil)\} & \\ \cup & \\ \{\alpha \in \bar{A}^\infty \mid \exists \alpha^1, \alpha^2, \ldots \in \bar{A}^* \colon \alpha = \otimes_\infty(\alpha^1, \alpha^2, \ldots) \wedge \alpha^1 \in S_0(a) \wedge \forall i \in \mathbb{N}^+ \colon & \\ (\alpha_i \in \bar{A}^* \wedge \tau(\alpha^i) \in \lceil\pi\rceil \wedge \alpha^{i+1} \in S_0(\tau(\alpha^i)))\} & \\ \cup & \\ \{\alpha \in (\bar{A} \cup \{fail\})^* \mid \exists \alpha^1, \ldots, \alpha^n \in (\bar{A} \cup \{fail\})^{**} \colon \alpha = \otimes_n(\alpha^1, \ldots, \alpha^n) \wedge & \\ \alpha^1 \in S_0(a) \wedge \forall i \in [1..n-2] \colon (\alpha^i \in \bar{A}^* \wedge \tau(\alpha^i) \in \lceil\pi\rceil \wedge \alpha^{i+1} \in S_0(\tau(\alpha^i))) & \\ \wedge(\alpha^{n-1} \in \bar{A}^* \wedge \tau(\alpha^{n-1}) \notin \mathcal{D}_\pi \wedge \alpha^n = <\tau(\alpha^{n-1}), fail>)\}, & \textit{if } a \in \mathcal{D}_\pi \wedge \pi(a) \end{cases}$$

At first sight, the latter definition might seem to be difficult. It is not difficult, if we take into account all the possible executions of the loop starting from a given state a:

- The loop terminates with failure, in case the loop condition is not defined in the state a.
- The loop does nothing but terminates faultlessly, in case the loop condition is defined but does not hold for the state a.
- We execute the loop body finite number of times, and the last execution of the loop body
  - produces an endless execution, or
  - produces an execution that terminates in the $fail$ state, or
  - terminates in a state where the π loop condition does not hold.
- We execute the loop body infinite number of times.
- We execute the loop body finite number of times, and the last execution of the loop body leads to a state where π cannot be evaluated.

Notice that the last case differs from the previously mentioned case, when the loop resides in a state that satisfies the loop condition, and the last execution of the loop body generates a sequence that ends with the $fail$ state.