Program constructs - examples

Exercise 1 *Let $A = [1..6]$ be the common base-statespace of the following programs $S_1, S_2 \subseteq A \times (\bar{A} \cup \{fail\})^{**}$:*

$$S_1 = \begin{Bmatrix} 1 \to <1,4,3> & 1 \to <1,2,4> & 2 \to <2,2,\dots> \\ 2 \to <2,1,4,6> & 3 \to <3,5,1> & 4 \to <4,5,3> \\ 5 \to <5,1,fail> & 6 \to <6,3,1,5> & \end{Bmatrix}$$

$$S_2 = \begin{Bmatrix} 1 \to <1,3,2> & 1 \to <1,2,4> & 2 \to <2,6> \\ 3 \to <3,4> & 4 \to <4,fail> & 4 \to <4,5,1> \\ 5 \to <5> & 6 \to <6,4,3,2> & \end{Bmatrix}$$

– *Determine the sequence $(S_1; S_2)$.*
Informally: Briefly saying we get the sequence of two programs by executing the two porgrams after each other: first we execute S_1, then we execute S_2. Program S_1 can assign three different kind of sequences to an arbitrary state a: an infinite sequence, a finite sequence ending in the $fail$ state, or a finite sequence ending in a state of A. In the first two cases program S_2 cannot do anything: in case program S_1 assigned an infinite sequence to state a or a finite sequence with the last element $fail$, then the sequence $(S_1; S_2)$ assigns the same sequence to the given state a.
In the third case, we concatenate two sequences but not repeat the joint element: the first sequence (α) is finite and assigned to state a by S_1, the second sequence (β) is assigned to the last element of α by program S_2.

$$(S_1; S_2) = \begin{Bmatrix} 1 \to <1,4,3,4> & 1 \to <1,2,4,fail> & 1 \to <1,2,4,5,1> \\ 2 \to <2,2,\dots> & 2 \to <2,1,4,6,4,3,2> & \\ 3 \to <3,5,1,3,2> & 3 \to <3,5,1,2,4> & 4 \to <4,5,3,4> \\ 5 \to <5,1,fail> & 6 \to <6,3,1,5> & \end{Bmatrix}$$

– *Let $\pi_1, \pi_2 \in A \to \mathbb{L}$ be logical functions such that*
$\pi_1 = \{(1, true), (2, true), (4, true), (5, false), (6, false)\}$ *and*
$\pi_2 = \{(1, true), (2, false), (3, true), (4, true), (5, false)\}$.
Determine the selection $(\pi_1{:}S_1, \pi_2{:}S_2)$.
The selection assigns the sequence $<a, fail>$ to state a, in case both the π_1 and π_2 conditions are false for a or none of them is defined for a. In this case, the sequence $<a, fail>$ is the only sequence assigned to state a.

In case π_i holds in state a, then the selection assigns all the sequences to a that are assigned to a by the program S_i. If π_i is not defined for the state a, then the selection assigns the sequence $<a, fail>$ to a (in order to indicate that there is an error when evaluating condition π_i for the state a).

Be careful: in the exercise π_1 intentionally do not assign anything to 3, that means logical function π_1 is not defined in the state 3. This is the reason why the logical functions are not defined by their truth set, the fact *that π_1 is not true for a state a would not imply that $\neg\pi_1(a)$; the function can be neither true or flase in state a.*

As in our case for state 1 both π_1 and π_2 are satisfied, $IF(1) = S_1(1) \cup S_2(1)$ holds. Let us investigate for every state a, which one of the two conditions holds for a. To the state 5, the selection assigns the sequence $< 5, fail >$, as both conditions are false for 5. In case of stete 6, there is only one possible execution generated by the selection, that is the sequence $< 6, fail >$; since π_1 is defined in 6 but evaluates to false, and π_2 is not defined for the state 6.

$$IF = \begin{Bmatrix} 1 \to< 1,4,3 > & 1 \to< 1,2,4 > & 1 \to< 1,3,2 > \\ 2 \to< 2,2,\dots > & 2 \to< 2,1,4,6 > & \\ 3 \to< 3, fail > & 3 \to< 3,4 > & \\ 4 \to< 4,5,3 > & 4 \to< 4, fail > & 4 \to< 4,5,1 > \\ 5 \to< 5, fail > & & \\ 6 \to< 6, fail > & & \end{Bmatrix}$$

Exercise 2 *Let $A = [1..5]$, $S_0 \subseteq A \times (\bar{A} \cup \{fail\})^{**}$ be programs, and $\pi\colon A \to \mathbb{L}$ such that $\lceil \pi \rceil = \{1, 2, 3, 4\}$.*

$$S_0 = \begin{Bmatrix} 1 \to< 1,2,4 > & 2 \to< 2 > & 3 \to< 3,4,2 > \\ 3 \to< 3,5 > & 3 \to< 3,3,3,\dots > & 4 \to< 4,5,3,4 > \\ 4 \to< 4,1,3 > & 5 \to< 5,5,\dots > & \end{Bmatrix}$$

Determine the loop (π, S_0). We denote this loop by DO, whereas S_0 denotes the loop body and π denotes the loop condition, respectively.

Informally:

Let us start the execution of the loop from state 1. As the loop condition is true for 1, it is guaranteed that we reach the state 4 by executing the loop body once (as only the sequence $< 1, 2, 4 >$ is assigned to the state 1 by the loop body S_0). From the state 4, there are two possible ways to continue the execution of the loop (recall: the loop can stop only in the state 5, as 5 is the only state where π is defined but false): we end up un the state 3 (due to the sequence $< 4, 1, 3 >$ assigned to 4 by the loop body; or we end up in 4 again due to the sequence $< 4, 5, 3, 4 >$). Notice, the fact that the loop body can take us from the state 4 to the state 4, we are enabled to execute the loop body in the same way infinite number of times (by traversing the states 5 and 3 and reaching the state 4 again); or after executing the loop body finite number of times, we choose the sequence $< 4, 1, 3 >$ and continue the execution in a different way.

Let $< (4, 5, 3)\infty >$ denote the execution, where starting from the state 4, after executing the loop body once, we reach the state 4 again; that means we repeat the traverse of states 4, 5 and 3 (in this order) infinite number of times.

$< (4, 5, 3)k\, 4, 1, 3, 5 > \; k \in \mathbb{N}^+$ denotes the execution, where starting from the state 4, the states 4, 5 and 3 traversed recurrently infinite number of times (k), then from the state 4 the execution continues according to the sequence $< 4, 1, 3 >$ (that is assigned to the state 4 by the bofy of the loop). Moreover, as the sequence $< 3, 5 >$ is assigned to 3 by the loop body, the loop stops in the state 5.

Notice that by allowing the case when $k = 0$, the case when the sequence $< 4, 1, 3, 5 >$ is assigned to 4 by the loop is covered as well.

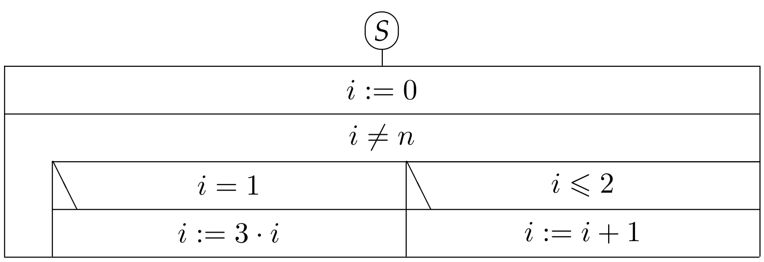
$$DO = \begin{cases} 1 \rightarrow < 1,2,4,(5,3,4)\infty > \\ 1 \rightarrow < 1,2,4,(5,3,4)k,1,3,5 > k \in \mathbb{N} \\ 1 \rightarrow < 1,2,4,(5,3,4)l,1,3,4,2,... > l \in \mathbb{N} \\ 1 \rightarrow < 1,2,4,(5,3,4)m,1,3,3,3,3,... > m \in \mathbb{N} \\ 2 \rightarrow < 2,2,..... > \\ 3 \rightarrow < 3,4,2,... > \\ 3 \rightarrow < 3,5 > \\ 3 \rightarrow < 3,3,3,... > \\ 4 \rightarrow < (4,5,3)\infty > \\ 4 \rightarrow < (4,5,3)g,4,1,3,4,2,... > g \in \mathbb{N} \\ 4 \rightarrow < (4,5,3)i,4,1,3,5 > i \in \mathbb{N} \\ 4 \rightarrow < (4,5,3)j,4,1,3,3,... > j \in \mathbb{N} \\ 5 \rightarrow < 5 > \end{cases}$$

Exercise 3 $A = (i : \mathbb{N}_0, n : \mathbb{N}_0)$

Write down the sequences that are assigned to the states

- $(1,3)$ *and*
- $(4,6)$

by the following S *program.*



Our S *program is a sequence constructed from an assignment* $i := 0$ *and a loop. The loop will stop if its loopcondition becomes* $false$*, that is when* i *and* n *are equal.*

The semantics of sequence and loop is well known, however we defined the meaning of selection in a different way than it is common in programming languages like JAVA or C++. In our mathematical model, non-determinism is allowed. Particularly, if there are more branches of a selection where the conditions are all true for a given state in the statespace, then we do not select one branch from them according a special rule, instead, it is allowed to take any branch where the corresponding condition is $true$

For example, when i *equals to* 1*, then both the conditions* $i = 1$ *and* $i \leqslant 2$ *are* $true$*, we do not prefer any of the corresponding programs, both* $i := i \cdot 3$ *and* $i := i + 1$ *can be executed starting from the state where the value of variable* i *is* 1*.*

Another difference in our model compared to programming languages, that if none of the conditions of a selection is $true$ for a given state, then the selection aborts at the given state.
For example, when i equals to 3 then neither $i = 1$ nor $i \leqslant 2$ is $true$, the selection aborts starting from a state where the value of variable i is 3.

A program is a relation that assigns sequences to every state in the statespace. All the sequences assigned to any state start with the same state they are assigned to. Our S program maps sequences to every state of the statespace such that the elements of the sequences are states of A (i.e. the elements of the sequences are pairs).

The assignment $i := 0$ takes us from a state (i, n) to a state where i is set to 0 and n is unchanged, that is $(0, n)$.
There are two sequences assigned to the state $(1, 3)$:

$< (1, 3), (0, 3), (1, 3), (3, 3) >$ and $< (1, 3), (0, 3), (1, 3), (2, 3), (3, 3) >$
Notice that the reason why two sequences are assigned to $(1, 3)$ is, that when we are in the state $(1, 3)$, then both the conditions $i = 1$ and $i \leqslant 2$ are $true$, there are two possible ways how we can continue the execution of the program. In the state $(3, 3)$ the loop stops, as its loopcondition becomes $false$.

There are two sequences assigned to the state $(4, 6)$:

$< (4, 6), (0, 6), (1, 6), (3, 6), fail >$ and $< (4, 6), (0, 6), (1, 6), (2, 6), (3, 6), fail >$
Notice that the reason why these sequences terminates in the special $fail$ state is, that in the state $(3, 6)$ neither the conditions $i = 1$ and nor $i \leqslant 2$ is $true$, the selection (the body of the loop) aborts.