2020 Model of Computation Examination Solution

1. a. i) $\langle x := 1, s\rangle \rightarrow_c \langle \texttt{skip}, s[x \rightarrow 1]\rangle$

   ii) The evaluation path is as follows

$$\begin{aligned} &\langle \texttt{while true do skip}, s\rangle \\ \rightarrow_c&\langle \texttt{if true then (skip; while true do skip) else skip}, s\rangle \\ \rightarrow_c&\langle \texttt{skip; while true do skip}, s\rangle \\ \rightarrow_c&\langle \texttt{while true do skip}, s\rangle \\ \rightarrow_c&(\text{back to line 2})\end{aligned}$$

   iii) There are two possible evaluation paths (supposed to write both in the exam for full credit)

   The first way is

$$\begin{aligned} &\langle (x := 1) \text{ or } \texttt{while true do skip}, s\rangle \\ \rightarrow_c&\langle x := 1, s\rangle \\ \rightarrow_c&\langle \texttt{skip}, s[x \rightarrow 1]\rangle\end{aligned}$$

   The second way is

$$\begin{aligned} &\langle (x := 1) \text{ or } \texttt{while true do skip}, s\rangle \\ \rightarrow_c&\langle \texttt{while true do skip}, s\rangle \\ \rightarrow_c&(\text{back to line 2 in ii})\end{aligned}$$

   b. To prove `while true do skip` always diverges, it is enough to show that the evaluation path of it is infiinte and theree is no normal form corresponding to it.

$$\begin{aligned} &\langle \texttt{while true do skip}, s\rangle \\ \rightarrow_c&\langle \texttt{if true then (skip; while true do skip) else skip}, s\rangle \\ \rightarrow_c&\langle \texttt{skip; while true do skip}, s\rangle \\ \rightarrow_c&\langle \texttt{while true do skip}, s\rangle \\ \rightarrow_c&(\text{back to line 2})\end{aligned}$$

   We can see that the evaluation path will never stop and will not reach to a normal state (e.g. $\langle \texttt{skip}, s'[x \rightarrow 1]\rangle$). In other words, $\langle \texttt{while true do skip}, s\rangle \rightarrow^* \langle \texttt{while true do skip}, s\rangle$.

   c. Command i) and iii) are the ONLY commands that are equivalent to each other

   Let's denote $C_1 = (x := 1)$ and $C_2 = ((x := 1) \text{ or } (\texttt{while true do skip}))$. From a i) we have known that $\exists n \in \mathbb{R}.\ \langle C_1, s\rangle \rightarrow^n \langle \texttt{skip}, s'\rangle$ where $n = 1$ and $s' = s[x \rightarrow 1]$. From iii) we know that $\exists n \in \mathbb{R}.\ \langle C_2, s\rangle \rightarrow^n \langle \texttt{skip}, s'\rangle$ where $n = 2$ and $s' = s[x \rightarrow 1]$. Hence both sides of the if-and-only-if statement are true and $C_1 \sim C_2$.

   d. i) ***Notice: the statement in this question might be incorrect. The 'skip' in the first statement should be 'C'***.

   We first prove the base case when $n = 1$.

   Trivially, we can prove this by showing the evaluation path of $\langle C, s[x \rightarrow 0, z \rightarrow 0]\rangle$

$$\begin{aligned}
&\langle C, s[x \to 0, z \to 0]\rangle \\
\to_c &\langle \texttt{if (z=0) then (x:=x+1; \{z:=0\} or \{z:=1\}; while ...) else skip}, s[x \to 0, z \to 0]\rangle \\
\to_c &\langle \texttt{x:=x+1; \{z:=0\} or \{z:=1\}; while ...)}, s[x \to 0, z \to 0]\rangle \\
\to_c &\langle x := 1; \texttt{\{z:=0\} or \{z:=1\}; while ...}, s[x \to 0, z \to 0]\rangle \\
\to_c &\langle \texttt{skip}; \texttt{\{z:=0\} or \{z:=1\}; while ...}, s[x \to 1, z \to 0]\rangle \\
\to_c &\langle \texttt{\{z:=0\} or \{z:=1\}; while ...}, s[x \to 1, z \to 0]\rangle
\end{aligned}$$

Now, if we choose to use the left-hand-side rule of or-evaluation, the rest of the evaluation will look like this

$$\begin{aligned}
\to_c &\langle \texttt{z:=0; while ...}, s[x \to 1, z \to 0]\rangle \\
\to_c &\langle \texttt{skip; while ...}, s[x \to 1, z \to 0]\rangle \\
\to_c &\langle \texttt{C}, s[x \to 1, z \to 0]\rangle
\end{aligned}$$

which has proved that $\langle C, s[x \to 0, z \to 0]\rangle \to^* \langle C, s[s \to 1, z \to 0]\rangle$.

If we instead choose to use the right-hand-side rule of or-evaluation, the rest of the evaluation will look like this

$$\begin{aligned}
\to_c &\langle \texttt{z:=1; while ...}, s[x \to 1, z \to 0]\rangle \\
\to_c &\langle \texttt{skip; while ...}, s[x \to 1, z \to 1]\rangle \\
\to_c &\langle \texttt{while (z=0) do ...}, s[x \to 1, z \to 1]\rangle \\
\to_c &\langle \texttt{if (z=0) then (...) else skip}, s[x \to 1, z \to 1]\rangle \\
\to_c &\langle \texttt{skip}, s[x \to 1, z \to 1]\rangle
\end{aligned}$$

which has proved that $\langle C, s[x \to 0, z \to 0]\rangle \to^* \langle \texttt{skip}, s[s \to 1, z \to 1]\rangle$.

We now show the inductive case

The inductive hypothesis is that for a given number $k \in \mathbb{N}^+$, we have both

$$\langle C, s[x \to 0, z \to 0]\rangle \to^* \langle C, s[s \to k, z \to 0]\rangle$$

and

$$\langle C, s[x \to 0, z \to 0]\rangle \to^* \langle \texttt{skip}, s[s \to k, z \to 1]\rangle$$

We need to show that for $k + 1 \in \mathbb{N}^+$

$$\langle C, s[x \to 0, z \to 0]\rangle \to^* \langle C, s[s \to k+1, z \to 0]\rangle$$

and

$$\langle C, s[x \to 0, z \to 0]\rangle \to^* \langle \texttt{skip}, s[s \to k+1, z \to 1]\rangle$$

Using the first part of the inductive hypothesis we can prove that

$$\begin{aligned}
&\langle C, s[x \to 0, z \to 0]\rangle \\
\to^* &\langle C, s[x \to k, z \to 0]\rangle \\
\to^* &\langle \texttt{if (z=0) then (...) else skip}, s[x \to k, z \to 0]\rangle \\
\to^* &\langle x := x + 1; \texttt{\{z:=0\} or \{z:=1\}; while ...}, s[x \to k, z \to 0]\rangle \\
\to^* &\langle x := k + 1; \texttt{\{z:=0\} or \{z:=1\}; while ...}, s[x \to k, z \to 0]\rangle \\
\to^* &\langle \texttt{skip}; \texttt{\{z:=0\} or \{z:=1\}; while ...}, s[x \to k+1, z \to 0]\rangle \\
\to^* &\langle \texttt{\{z:=0\} or \{z:=1\}; while ...}, s[x \to k+1, z \to 0]\rangle
\end{aligned}$$

Again, if we choose the left-hand-side rule of or-evaluation, we have

$$\begin{aligned}
&\rightarrow^* \langle \texttt{z:=0; while ...}, s[x \rightarrow k+1, z \rightarrow 0]\rangle \\
&\rightarrow^* \langle \texttt{skip; while ...}, s[x \rightarrow k+1, z \rightarrow 0]\rangle \\
&\rightarrow^* \langle C, s[x \rightarrow k+1, z \rightarrow 0]\rangle
\end{aligned}$$

which proved the first part of our inductive case.

If we choose the right-hand-side rule of or-evaluation, we have

$$\begin{aligned}
&\rightarrow^* \langle \texttt{z:=1; while ...}, s[x \rightarrow k+1, z \rightarrow 0]\rangle \\
&\rightarrow^* \langle \texttt{skip; while ...}, s[x \rightarrow k+1, z \rightarrow 1]\rangle \\
&\rightarrow^* \langle \texttt{while (z=0) do ...}, s[x \rightarrow k+1, z \rightarrow 1]\rangle \\
&\rightarrow^* \langle \texttt{skip}, s[x \rightarrow k+1, z \rightarrow 1]\rangle
\end{aligned}$$

which proved the second part of our inductive case.

ii) They are equivalent to each other. Below is the proof

Given a state s and $n \in \mathbb{N}^+$, the evaluation path of $x := \texttt{somenumber}; z := 1$ is

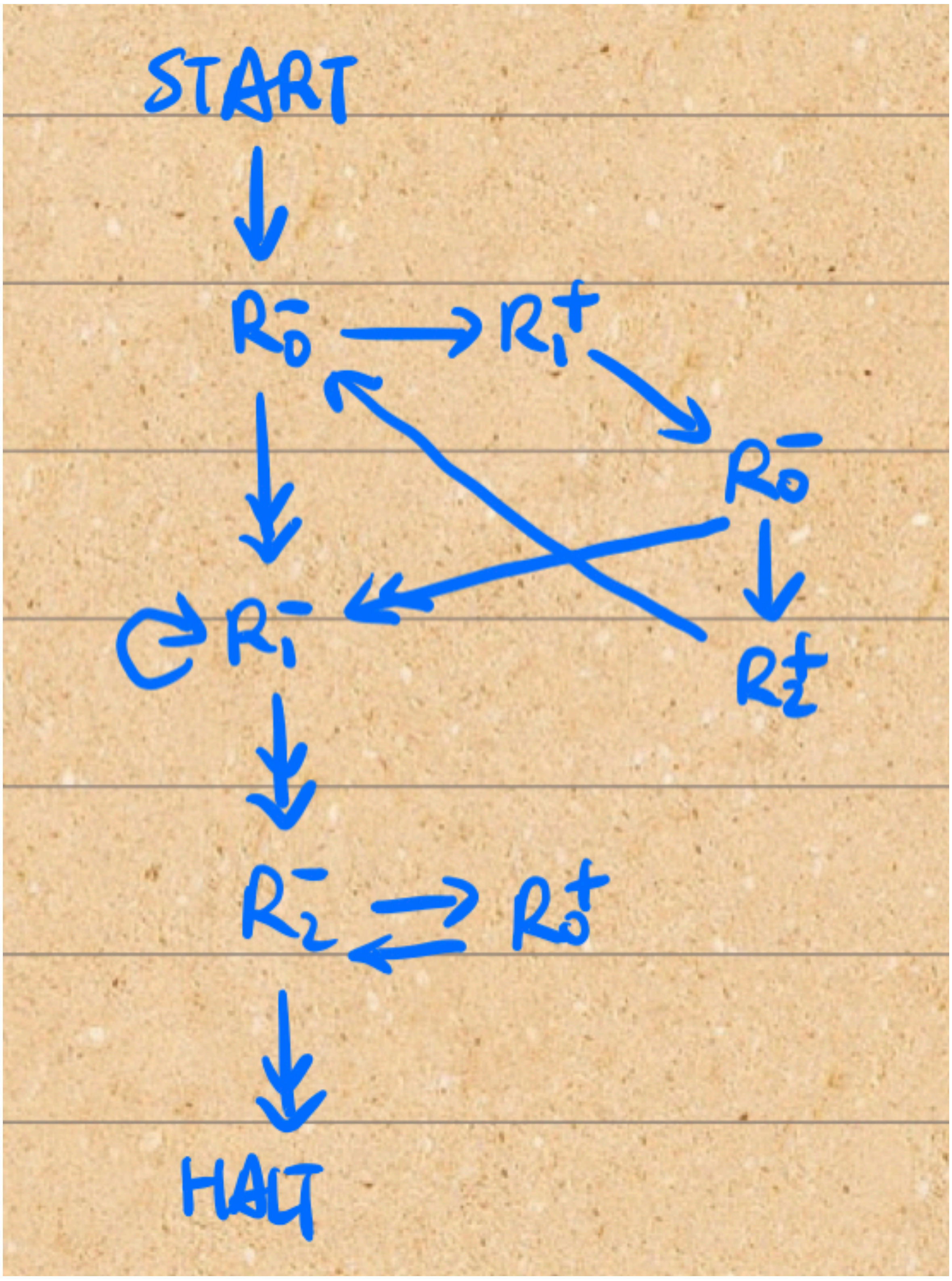
$$\begin{aligned}
&\langle x := \texttt{somenumber}; z := 1, s\rangle \\
\rightarrow_c &\langle \texttt{skip}; z := 1, s[x \rightarrow n]\rangle \\
\rightarrow_c &\langle z := 1, s[x \rightarrow n]\rangle \\
\rightarrow_c &\langle \texttt{skip}, s[x \rightarrow n, z \rightarrow 1]\rangle
\end{aligned}$$

The evaluation path of $x := 0; z := 0; C$ is

$$\begin{aligned}
&\langle x := 0; z := 0; C, s\rangle \\
\rightarrow_c &\langle \texttt{skip}; z := 0, s[x \rightarrow 0]\rangle \\
\rightarrow_c &\langle z := 0; C, s[x \rightarrow 0]\rangle \\
\rightarrow_c &\langle \texttt{skip}; C, s[x \rightarrow 0, z \rightarrow 0]\rangle \\
\rightarrow_c &\langle C, s[x \rightarrow 0, z \rightarrow 0]\rangle \\
\rightarrow^* &\langle \texttt{skip}, s[x \rightarrow n, z \rightarrow 1]\rangle
\end{aligned}$$

which results in the same end state as $x := \texttt{somenumber}; z := 1$. Hence, both statements exist a finite number of evaluation steps that result in the same state and hence they are equivalent to each other.

2. a. i) The graphical representaion of M will be something looks like this



$$\begin{tikzcd}
\text{START} \arrow[d] & & \\
R_0^- \arrow[r] \arrow[dd] & R_1^+ \arrow[dr] & \\
& & R_0^- \arrow[dll] \arrow[d] \\
R_1^- \arrow[loop left] \arrow[d] & & R_2^+ \arrow[uull] \\
R_2^- \arrow[r, shift left] \arrow[d] & R_0^+ \arrow[l, shift left] & \\
\text{HALT} & &
\end{tikzcd}$$

The function M computes $R_0 = \lfloor \frac{m}{2} \rfloor$

ii) The encodings of the first three instructions are

$$\begin{aligned}
\ulcorner R_0^- \rightarrow L_1, L_4 \urcorner &= \langle\!\langle 2 \times 0 + 1, \langle 1, 4 \rangle \rangle\!\rangle \\
&= \langle\!\langle 1, 2^1 \times (2 \times 4 + 1) - 1 \rangle\!\rangle \\
&= \langle\!\langle 1, 17 \rangle\!\rangle \\
&= 2^1 \times (2 \times 17 + 1) = 70
\end{aligned}$$

$$\begin{aligned}\ulcorner R_1^+ \to L_2 \urcorner &= \langle\!\langle 2\times 1, 2\rangle\!\rangle \\ &= 2^2 \times (2\times 2+1) \\ &= 20\end{aligned}$$

$$\begin{aligned}\ulcorner R_0^- \to L_3, L_4 \urcorner &= \langle\!\langle 2\times 0+1, \langle 3,4\rangle\rangle\!\rangle \\ &= \langle\!\langle 1, 2^3\times(2\times 4+1)-1\rangle\!\rangle \\ &= \langle\!\langle 1, 71\rangle\!\rangle \\ &= 2^1\times(2\times 71+1) = 186\end{aligned}$$

iii) The calculation is shown below

$$1144 = 8\times 143 = 2^3\times 143 = \langle\!\langle 2\times 1+1, \langle j,k\rangle\rangle\!\rangle$$
$$\text{where } 143 = 2\cdot\langle j,k\rangle + 1 \Rightarrow \langle j,k\rangle = 71$$
$$71 = 2^3\times(2\times 4+1)-1 = \langle 3,4\rangle$$

Therefore, the instruction encoded by 1144 is $R_1^- \to L_3, L_4$

$$448 = 2^6\times(2\times 3+1) = \langle\!\langle 2\times 3, 3\rangle\!\rangle$$

Therefore, the insturction encoded by 448 is $R_3^+ \to L_3$.

b. i) The λ-terms are

$$\begin{aligned}\underline{T_l} &= \lambda sbl.\, s\ \underline{0}\ (b\ (l\ \underline{2})\ \underline{1}\ (l\ \underline{3})) \\ \underline{T_r} &= \lambda sbl.\, b\ (l\ \underline{1})\ \underline{0}\ (s\ \underline{2}\ (l\ \underline{3}))\end{aligned}$$

ii) The λ-term of B is

$$B = \lambda t_1 e t_2 sbl.\, b\ t_1\ e\ t_2$$

The beta reduction of $B\ (l\ \underline{1})\ \underline{0}\ (s\ \underline{2}\ (l\ \underline{3}))$ is

$$\begin{aligned}&B\ (l\ \underline{1})\ \underline{0}\ (s\ \underline{2}\ (l\ \underline{3})) \\ \to_\beta &(\lambda t_1 e t_2 sbl.\, b\ t_1\ e\ t_2)(l\ \underline{1})\ \underline{0}\ (s\ \underline{2}\ (l\ \underline{3})) \\ \to_\beta &(\lambda e t_2 sbl.\, b\ (l\ \underline{1})\ e\ t_2)\ \underline{0}\ (s\ \underline{2}\ (l\ \underline{3})) \\ \to_\beta &(\lambda t_2 sbl.\, b\ (l\ \underline{1})\ \underline{0}\ t_2)\ (s\ \underline{2}\ (l\ \underline{3})) \\ \to_\beta &\lambda sbl.\, b\ (l\ \underline{1})\ \underline{0}\ (s\ \underline{2}\ (l\ \underline{3})) \\ =_\beta &\underline{T_r}\end{aligned}$$

iii) The λ-term of C is

$$C = \lambda d.\, d\ (\lambda el.\, \underline{1})\ (\lambda ler.\, \underline{2})\ (\lambda e.\, \underline{0})$$

The construction is based on the constructors, particularly the number of arguments of the constructors, of the three data structures, as well as the return value for each of the constructor. (e.g. leaf return 0, branch return 2 and single child return 1). Notice that the λ-term is also based on the order of letters in the λ-term we defined in i). (e.g. λsbl and λbsl will result in different C with order changes as well).

We now show that $C\ \underline{T_l} =_\beta \underline{1}$

$$\begin{aligned}
& C\ \underline{T_l} \\
\rightarrow_\beta & (\lambda d.\, d\ (\lambda el.\, \underline{1})\ (\lambda ler.\, \underline{2})\ (\lambda e.\, \underline{0}))(\lambda sbl.\, s\ \underline{0}\ (b\ (l\ \underline{2})\ \underline{1}\ (l\ \underline{3}))) \\
\rightarrow_\beta & (\lambda sbl.\, s\ \underline{0}\ (b\ (l\ \underline{2})\ \underline{1}\ (l\ \underline{3})))\ (\lambda el.\, \underline{1})\ (\lambda ler.\, \underline{2})\ (\lambda e.\, \underline{0}) \\
\rightarrow_\beta & (\lambda bl.\, (\lambda el.\, \underline{1})\ \underline{0}\ (b\ (l\ \underline{2})\ \underline{1}\ (l\ \underline{3})))\ (\lambda ler.\, \underline{2})\ (\lambda e.\, \underline{0}) \\
\rightarrow_\beta & (\lambda bl.\, (\lambda l.\, \underline{1})\ (b\ (l\ \underline{2})\ \underline{1}\ (l\ \underline{3})))\ (\lambda ler.\, \underline{2})\ (\lambda e.\, \underline{0}) \\
\rightarrow_\beta & (\lambda bl.\, \underline{1})\ (\lambda ler.\, \underline{2})\ (\lambda e.\, \underline{0}) \\
\rightarrow_\beta & \underline{1}
\end{aligned}$$

We now show that $C\ \underline{T_r} =_\beta \underline{2}$

$$\begin{aligned}
& C\ \underline{T_r} \\
\rightarrow_\beta & (\lambda d.\, d\ (\lambda el.\, \underline{1})\ (\lambda ler.\, \underline{2})\ (\lambda e.\, \underline{0}))(\lambda sbl.\, b\ (l\ \underline{1})\ \underline{0}\ (s\ \underline{2}\ (l\ \underline{3}))) \\
\rightarrow_\beta & \lambda sbl.\, b\ (l\ \underline{1})\ \underline{0}\ (s\ \underline{2}\ (l\ \underline{3}))\ (\lambda el.\, \underline{1})\ (\lambda ler.\, \underline{2})\ (\lambda e.\, \underline{0}) \\
\rightarrow_\beta & \lambda bl.\, b\ (l\ \underline{1})\ \underline{0}\ ((\lambda el.\, \underline{1})\ \underline{2}\ (l\ \underline{3}))\ (\lambda ler.\, \underline{2})\ (\lambda e.\, \underline{0}) \\
\rightarrow_\beta & \lambda l.\, (\lambda ler.\, \underline{2})\ (l\ \underline{1})\ \underline{0}\ ((\lambda el.\, \underline{1})\ \underline{2}\ (l\ \underline{3}))\ (\lambda e.\, \underline{0}) \\
\rightarrow_\beta & \lambda l.\, \underline{2}\ (\lambda e.\, \underline{0}) \\
\rightarrow_\beta & \underline{2}
\end{aligned}$$

The bracketing here is bit messy and might not be completely right, so remember that :).