Solution Direction to HW1

Yue Guo

September 28, 2016

Abstract

1 TA Game

Graded on "all or nothing" scale, 1 point for each part

Constrained by the if-else clause in for loop, the card A can only followed by card T, and vice versa. So the answer will be in the form of "ATAT..." or "TATA...".

Besides, strings of all different length (from 2 to 2 \cdot (1 + k)) are possible because the execution of two players can interleave in any way.

So the final answer will be all strings expressed in k = n case.

1.1 k = 1

All possible strings:

| AT | TA |
| ATA | TAT |
| ATAT | TATA |

1.2 k = 2

All possible strings:

| AT | TA |
| ATA | TAT |
| ATAT | TATA |
| ATATA | TATAT |
| ATATAT | TATATA |
1.3 \( k = n \)

Possible strings:

\((AT)^i\) for \(i \in \{1..n+1\}\), \(n + 1\) states;
\((AT)^iA\) for \(i \in \{1..n\}\), \(n\) states;
\((TA)^i\) for \(i \in \{1..n+1\}\), \(n + 1\) states;
\((TA)^iT\) for \(i \in \{1..n\}\), \(n\) states;

So total number: \(4n + 2\)

2  Aurora’s Addition

Question 2 will be graded as follows:

- Points will be split evenly 5 - 9 per part
- In part 1, -1 per missing number
- In part 2, -2 per missing number
- Minimum score for both (ie they turned something in that made it look like they attempted it) is 1
- It is possible that they may have the same 2 numbers - if they match, quickly check netIds to see they did it right, count as right
- In other cases investigate further. If they have 3 numbers, it is likely but not guaranteed they are missing one

2.1  Call Add Once

There may be other schedules leading to outputs below. Only one possible schedule listed here.

<table>
<thead>
<tr>
<th>Possible output</th>
<th>Possible schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>if (\delta &lt; 0) (MainLoop) (\rightarrow) (y = x - \delta) (MainLoop) (\rightarrow) (x = a) (Add)</td>
</tr>
<tr>
<td>Int1 + 6</td>
<td>if (\delta &lt; 0) (MainLoop) (\rightarrow) (x = a) (Add) (\rightarrow) (y = x - \delta) (MainLoop) (\rightarrow) (d) (Add) (Add)</td>
</tr>
<tr>
<td>Int1 + Int2</td>
<td>(x = a) (Add) (\rightarrow) (\delta = d) (Add) (\rightarrow) if (\delta &lt; 0) (MainLoop)</td>
</tr>
<tr>
<td>Int1 - Int2</td>
<td>if (\delta &lt; 0) (MainLoop) (\rightarrow) (x = a) (Add) (\rightarrow) (\delta = d) (Add) (\rightarrow) (y = x - \delta) (MainLoop)</td>
</tr>
</tbody>
</table>
2.2 Call Add Twice

<table>
<thead>
<tr>
<th>Possible Output</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>6, Int1 + 6, Int1 + Int2, Int1 − Int2,</td>
<td>(second notify before waiting)</td>
</tr>
<tr>
<td>(6, Int1 − Int2)</td>
<td>δ = −Int2 after second if and before (y = x + \delta)</td>
</tr>
<tr>
<td>(Int1 + 6, Int1 − Int2)</td>
<td>δ = −Int2 after second if and after (y = x + \delta)</td>
</tr>
<tr>
<td>(Int1 + Int2, Int1 − Int2)</td>
<td>or δ = −Int2 before second if</td>
</tr>
<tr>
<td>(Int1 − Int2, Int1 + Int2)</td>
<td></td>
</tr>
</tbody>
</table>

3 To Be or Not To Be There

Question 3 will be graded on a all or nothing scale:

- The answers go like this: Same, Different, Different, Same, Stuck, Stuck
- Each is worth half a point

CLARIFICATION:

- **Sequential is 1 a) and 2 a)**
- **Interleaved, v1 is 1 b)c)**
- **interleaved, v2 is 2 b)c)**

We denote:

\[ p_A \leftarrow \text{select\_party}(A) \]
\[ p_B \leftarrow \text{select\_party}(B) \]

3.1 Sequential Case

\((p_A, p_A)\)

When B gets to the if clause, A has already written \(p_A\) on the whiteboard. So B jumps to else case and set \(p[B] := \text{whiteboard} = p_A\)

3.2 Interleaved Case 1

\((p_A, p_B)\)

Because they execute lines in turn, both of them find \(\text{whiteboard} = \emptyset\) when they reach if clause. So in the next line A sets \(p[A] := p_A\) and B sets \(p[B] := p_B\).
3.3 Interleaved Case 2

$(p_A, p_B)$

$A$ have set its own choice $p[A] := p_A$ but have not written to $\text{whiteboard}$ yet. So $B$ also enters the case $\text{whiteboard} = \emptyset$ and sets its own choice $p[B] := p_B$.

3.4 For More Complex Case

(1) $(p_A, p_A)$

Same case as explained above.

(2) Getting stuck

$A$ sets $\text{alice}_\text{busy} = \text{true}$, and then $B$ sets $\text{bob}_\text{busy} = \text{true}$ immediately. Then $A$ begins to wait for $B$ and $B$ begins to wait for $A$. Both of them get stuck.

(3) Getting stuck

$A$ is still busy ($\text{alice}_\text{busy} = \text{true}$) till $\text{select}\_\text{party}(i)$ of $A$. Then $B$ begins to execute and wait for $A$ at the while loop. Now $A$ is waiting for $B$ to execute the entire function and $B$ can never finish execution before $A$ set $\text{alice}_\text{busy} = \text{false}$. Both of them get stuck.