MITRO207
Homework 1: Solutions

Problem 1: muddy children

How would the state evolution depicted in slides 88-91 (Fig. 4.1 in the textbook) change if the father announces at noon:

- *Child* 1 *is dirty.*
- *There are an odd number of dirty children.*
- *There are an even number of dirty children.*

- By declaring child 1 to be dirty, the father excludes all configurations in which the first position (blue node) is set to 0. As a result, at 1:01pm, only three possible configurations remain: precisely those in which at least one of the remaining processes is dirty. The complex is, however, connected, and the children 2 and 3 may still have to wait until 3pm to announce themselves (when everyone is dirty). As before, at 2:01pm, the only possible configuration is the one in which everybody is dirty. See Figure 1.

  Remark: Note that here we assume that child 1 cooperates by running the original protocol ("only announce your self when you know exactly who is dirty"). If the child leaves the yard immediately after the father's announcement, the remaning kids won't be able to decide on their own (the complex is connected and no new information can be gained). In the distributed computing literature, there is a similar distinction between "departing after obtaining an output" and "helping the others to output."

- For an odd number of dirty children, we have only four disconnected 2-dimensional simplices at 12:01 (just after the news has arrived). Thus, all dirty children can announce themselves at 1pm (Figure 2).

- For an even number of dirty children, we have only four disconnected 2-dimensional simplices at 12:01, and the children will announce themselves at 1pm (Figure 3).

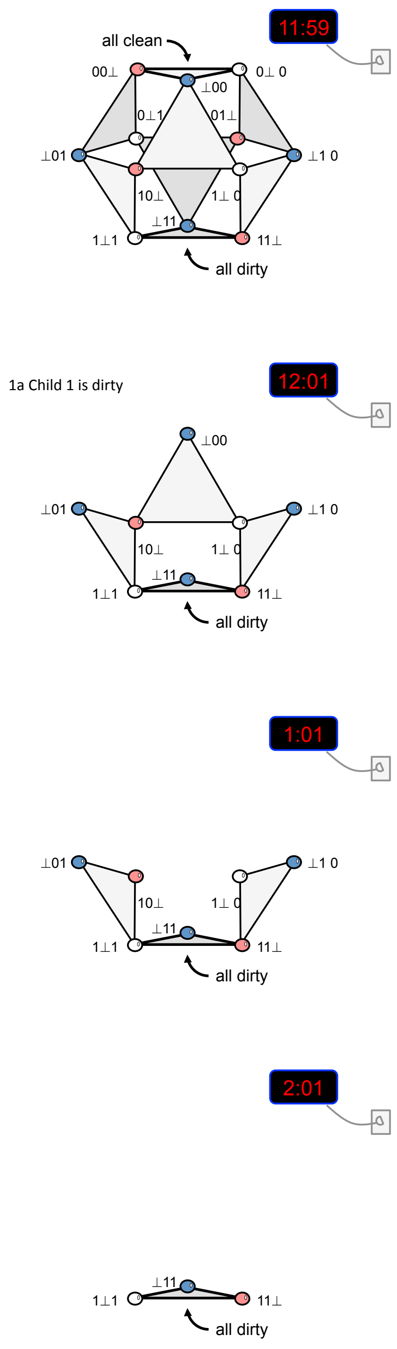

Figure 1: Possible simplices assuming that child 1 is dirty.

Problem 2: distinct input complex

Three processes, P, Q, and R, are assigned distinct *inputs from the set* $\{0, 1, 2\}$.

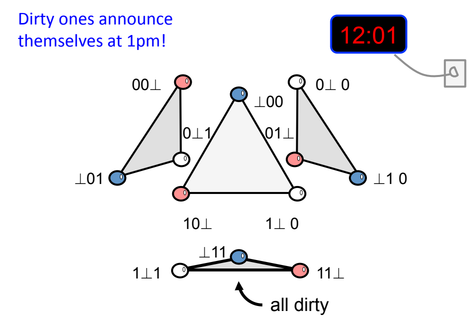

Figure 2: Possible simplices assuming that an odd number of children are dirty.

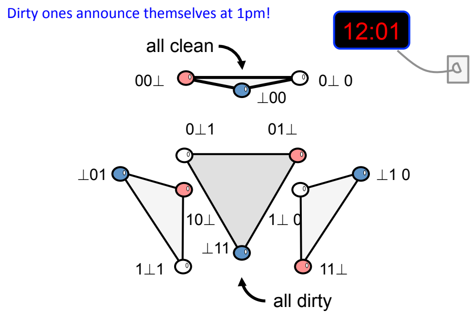

Figure 3: Possible simplices assuming that an even number of children are dirty.

Draw the complex of all possible assignments. Draw the complex with the values set $\{0,1,2,3\}$.

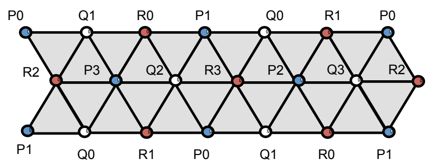

Figure 4: States of P, Q, and R with inputs $\{0,1,2,3\}$. Note that vertices in the top and bottom rows are repeated.

We begin with the value set $\{0,1,2,3\}$. Each of the three processes can take each of the four values. Thus there are $3*4 = 12$ vertices in the simplex. In a simplex of our complex, the three processes are assigned distinct values in one of the 4 subsets of three values of $\{0,1,2,3\}$, and there are $3! = 6$ assignments

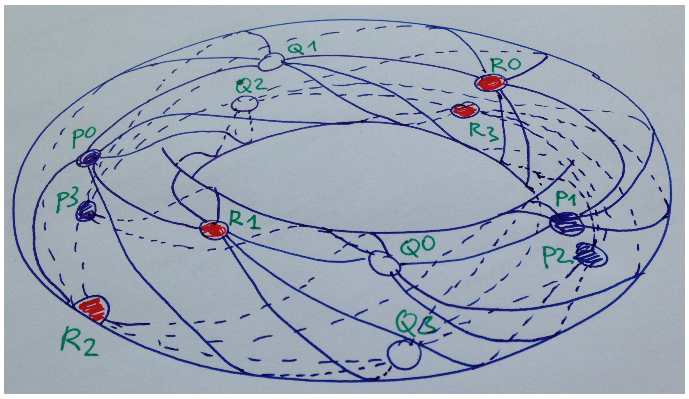

Figure 5: Complex of P, Q, and R with inputs $\{0,1,2,3\}$.

for each such subset. Thus, there are $4 * 6 = 24$ 2-dimensional simplices in the complex.

The 24 simplicies (with some vertices repeated) are drawn in Figure 4. Notice that the three vertices on the left are identified with the tree vertices on the right, and the seven vertices at the top are identified with the seven vertices at the bottom ("distorted" by three simplices to the right). By "glueing" the identified faces of the boundaries, we obtain a torus (Figure 5, apologies for the quality of the drawing).

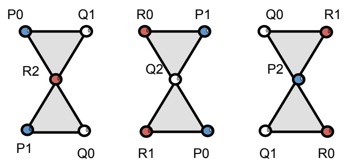

Figure 6: States of P, Q, and R with inputs $\{0,1,2\}$.

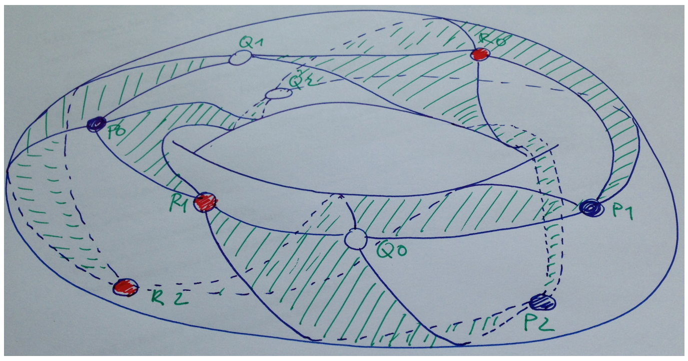

Figure 7: Complex of P, Q, and R with inputs $\{0,1,2\}$.

To obtain the complex corresponding to the value set $\{0,1,2\}$, we simply remove from the complex in Figure 4 all simplices containing value 3. The resulting complex of 6 2-dimensional simplices (with repeated vertices) is drawn in Figure 6. By "glueing" the identified vertices, we obtain the "spiral" complex that winds around the torus surface (Figure 7).

Problem 3: exact message losses

Draw the complex of states corresponding to one round of the 2-process system in slide 52, assuming that exactly *one message is lost in each round.*

The complex is like the one in slide 63 (the right side of Figure 1.3 in the textbook), except that the middle interval of the subdivision of each edge is removed: this interval corresponds to the execution in which both messages are delivered, which is excluded by our model.

Problem 4: Peterson's lock

Consider the following simplified variant of Peterson's 2-process mutual exclusion algorithm:

```
Shared atomic variables:
  flag[0], flag[1]: boolean atomic registers
  turn: atomic register

Code for process p_i (i=0,1):
  flag[i] := true
  turn := 1-i
  if (flag[1-i]) and (turn=1-i) then
     return false // failure
  else
     return true // critical section
```

Draw the complex of states reachable by the two processes after they complete the algorithm.

Possible vertices of the resulting *protocol complex* are (p_0, CS), (p_1, CS), (p_0, F), (p_1, F). Here CS means that the process returns `true`, and F means that the process returns `false`.

We are going to show that only two configurations presented in Figure 8 are possible.

Suppose, without loss of generality, that p_i $(i = 0, 1)$ was the first to write $1 - i$ to variable `turn`. By the algorithm, the other process p_{1-i} will overwrite `turn` with i, and then read `true` in `flag[i]` and i in `turn`. Thus, p_{1-i} will have to return `false`. Thus, $(p_0, CS), (p_1, CS)$ cannot be an edge in the resulting complex.

Now if p_i reads `turn` before the write of p_{1-i}, then it also returns `false`, which will result in the middle interval in Figure 8. Otherwise it returns `true`, which will result in one of the side intervals in Figure 8.

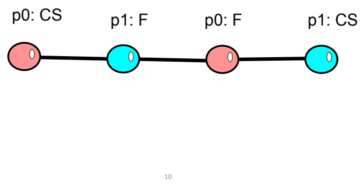

Figure 8: The reachable states of (simplified) Peterson's lock.