Advanced Network Security 2017 — leader election & mutual exclusion
Homework: do these programs terminate? (assume reads and writes are atomic)

Q1: \( i = 0 \)
\( j = 0 \)

\[
\text{thread while } i = 0
\]
\[
\begin{align*}
&\text{do } j \leftarrow j + 1 \mod 2; \text{ print } i \\
&\text{print } i \\
&\text{thread while } i = 0
\end{align*}
\]
\[
\begin{align*}
&\text{do if } j = 0 \text{ then } i \leftarrow 1
\end{align*}
\]

Q2: \( a < 1 \)
\( b < 1 \)

\[
\text{thread while } a \neq 0
\]
\[
\begin{align*}
&\text{do } b \leftarrow (b + a) \mod 2 \\
&\text{do } a \leftarrow a + 1
\end{align*}
\]

\[
\text{thread while } b \neq 0
\]
\[
\begin{align*}
&\text{do } a \leftarrow a + 1
\end{align*}
\]

execute this twice

\[
\begin{align*}
&b < 0 \\
&b < 1 \\
&a < 2 \\
&a < 3
\end{align*}
\]
Distributed system: states, configurations & evolutions

State:
- every node has local state \( C[i] \): system is modeled by a graph \( G = (V, E) \)
- every edge has a state \( C[e] \)
- shared variable: contents
- message passing channel: buffer
  - not necessarily FIFO!
- global state, also called the configuration \( C \), of the system is really the cartesian product of the states of all nodes and all edges
Execution & evolution

- A node has an action enabled, depending on
  - its local state \( C[i] \)
  - the state of all incoming edges \( C[e] \)

- An execution \( \langle A, \Rightarrow \rangle \) if \( i \leq j \)
  - sequence of events \( a_0, a_1, a_2, \ldots \) (such that \( a_i \Rightarrow a_j \))
  - there is an initial configuration \( C_0 \)
  - and every action \( a_i \) changes \( C_i \) into \( C_{i+1} \)
  - \( C_0, C_1, \ldots \) the evolution of the system.
If you have a property $P$ that you want to hold

You have to prove that $P$ holds for all possible executions (when started in the initial state).

The 'logic' of the system is defined by the partial order $\rightarrow$.

In other words:

you need to prove that $P$ holds for all $\langle A, \Rightarrow \rangle$

such that $\Rightarrow$ extends $\rightarrow$.
Atomicity

write(1) \rightarrow read

Properties of distributed systems we can distinguish:
1) Communication: messages or shared memory
2) Timing: synchronous, asynchronous (partially)
3) Scheduling: fair scheduler; round robin scheduler
4) Identity: 0..N-1 or no identity at all (uniform)
5) Network topology \( G = (V, E) \)
   - do nodes know \( |V| = N \) ?
   - edges are directed?
6) "sense of direction"
synchronous

asynchronous

send a message, it will eventually be received

partially synchronous

you do have an upper bound on the time it takes a message to arrive
Reason about distributed systems

- Prove this property $P$:
  - the cost of achieving $P$:
    - time
    - messages
    - bits

- Lower bounds:
  - given a certain system $S$, you need at least $X$ to achieve property $P$

- Impossibility proofs:
  - given a certain system $S$, you cannot solve $P$
Leader election

1970

- Graph \( G = (V,E) \)
- design a protocol that
  - \( C[i].leader = \text{true} \)
  - \( \forall j \neq i \ C[j].leader = \text{false} \)
  - initially \( C[j].leader = \text{false} \) for \( \forall j \)
- \( G \) connected
- nodes have identities:
  - \( C[i].id \rightarrow \text{unique} \)
  - (not necessarily the case that \( id \in \{0, \ldots, N-1\} \))
- nodes do not know their index \( i \)

IBM token ring

- Requirements
  - Correctness: at most one leader elected
  - Progress: eventually one leader elected
Let's make node 3 with $C[3].id = 0$ the leader.

Send to your clockwise neighbour your $C[i].id$ receive from your counter-clockwise neighbour id $C[i].leader < (id > C[i].id)$
LaLamn leader election

- Ring, with FIFO message passing
- Nodes do not know the size of the ring

Protocol for node i

1. \( i \in \emptyset \)
2. \( \text{CLIT}_i . \text{leader} \leftarrow \text{false} \)
3. Send \( \text{CLIT}_i . \text{id} \) to the right
4. While \( \text{CLIT}_i . \text{id} \neq i \)
   - Do receive from the left \( \text{id} \)
   - If \( i \in \text{IU} \{ \text{id} \} \)
     - Send \( \text{id} \) to the right

\[ \text{CLIT}_i . \text{leader} \leftarrow (\text{CLIT}_i . \text{id} = \min_{i \in \text{I}}) \]
What if nodes do not have unique identifiers?

- Nodes are uniform

- Nodes can be in the same state

- Nodes that are in the same state can do the same thing

There is no deterministic solution. Can be broken by randomness.
Mutual exclusion

\[
\text{while true do enter () \leftarrow (x \text{ critical resource } x) \quad \text{only 1 in here}} \\
\text{exit () \leftarrow c.s.} \\
\text{(x remainder section x)} \\
R.S.
\]

Properties

- mutual exclusion: at most one node in critical section
- progress: if there is a node entering, and the C.S. is empty, some node will eventually get access to the C.S.
- no starvation: if it will eventually get access, it release the C.S. after they get
First try

Two nodes: 0, 1

Two shared variables C[0].flag, C[1].flag

Protocol node i:

\[ C[i].flag \leftarrow 0 \]

\[
\text{while true} \quad \text{entry:} \quad \text{while} \quad C[1-i].flag \quad \text{do} \quad \text{\# wait +} \\
? \quad \text{\#} \quad C[i].flag \leftarrow \text{true} \\
(\star \text{Critical Section } \star) \\
\text{exit:} \quad C[i].flag \leftarrow \text{false} \]
test C[i-1]. flag

node 0
atoi flag?
false

C[0]. flag ← true

node 1
atoi flag?
false

C[0]. flag ← true

both in C.S.