FLP and RSMs
The Consensus Trilogy - Part 1
FLP and RSMs

The Consensus Trilogy - Part 1
Announcements
Announcements

• No Lab 1. We will just skip ahead to Lab 2 in 2 weeks.
  • Laziness.
  • More time for you to spend on Lab 2 which looks more complex.
  • More time for final project.
• Some people still have not filled out the form for associating Github accounts.
  • Do it now!
Consensus and FLP
What is Consensus?
Consensus: Setting

- Some set of nodes.
- Each receives some input: Considering binary consensus here so just 0/1.
- Each produces some output: Again just 0/1.
Consensus Protocol: Requirements

- **Termination**: All correct nodes *eventually* decide on a value to output.

- **Agreement**: All decided nodes decide on the *same* value.

- **Non-Triviality**: There must exist *some* input leading to all possible decisions.
  
  - Some input must result in algorithm deciding 0.
  
  - Some input must result in algorithm deciding 1.

- **Validity**: The decision must be one of the inputs.
  
  - Notice that validity implies non-triviality.
Consensus: Agreement
Consensus: Validity
FLP Impossibility Theorem

- No **deterministic** 1-crash-robust consensus algorithm exists for **async model**.
- Highlighted bits important since things break if you do not consider them.
Walk Through FLP Proof
System Model/Configuration

Configuration $c_0$

Network

$(p_1, m_0)$

$(p_0, m_1)$

$(p_0, m_0)$

$(p_1, m_1)$

$p_0$

$s_0$

Process State

$p_1$

Process
Events

Process $p_0$ and $p_1$ connected by the network.

Event $(p_0, m_0)$

Event $(p_0, m_1)$

Event $(p_1, m_2)$
System Model/Configuration

$c_0 \rightarrow c_1$  Transition from $c_0$ to $c_1$

$c_0 \Rightarrow c_1$  $c_1$ reachable from $c_0$
Definitions

• 0-decided: A configuration where some process has decided on 0.
• 1-decided: A configuration where some process has decided on 1.
• 0-valent: All reachable decided configuration are 0-decided.
• 1-valent: All reachable decided configuration are 1-decided.
• Bivalent: Both 0 and 1 decided reachable configuration.
Definitions

0-decided

1-decided
Definitions

0-valent
Definitions

Bivalent
Proof Sketch

- **Lemma 1**: Any 1-crash tolerant consensus protocol has an initial bivalent config.

- **Lemma 2**: Given a bivalent configuration ($\gamma$) and event $e$ can find configuration $\gamma'$
  
  - Event $e$ is enabled in $\gamma'$
  
  - When $e$ is applied to $\gamma'$ the resulting configuration is bivalent.
Proof Sketch

- No **deterministic** 1-crash-robust consensus algorithm exists for **async model**.

1. Use Lemma 1 to pick an initial bivalent configuration.

2. Given a bivalent configuration \((c)\) and event \(e\) which has been enabled longest.
   - Take the path from \(c\) to \(c'\) where \(e\) is still enabled in \(c'\).
   - Apply \(e\) to \(c'\) to get a new bivalent configuration \(c''\).

3. Repeat step 2.
Lemma 1

• Why must an initial bivalent configuration exist?

• Consider a system with 4 processes.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Lemma 1

• Why must an initial bivalent configuration exist?

• Consider a system with 4 processes.

<table>
<thead>
<tr>
<th></th>
<th>p₀</th>
<th>p₁</th>
<th>p₂</th>
<th>p₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Decision must flip from 0 to 1 somewhere
Lemma 1

- Why must an initial bivalent configuration exist?
- Consider a system with 4 processes.

<table>
<thead>
<tr>
<th>$p_0$</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Decision must flip from 0 to 1 somewhere
Lemma 1

• Why must an initial bivalent configuration exist?

• Consider a system with 4 processes.

\[ \begin{array}{cccc}
  p_0 & p_1 & p_2 & p_3 \\
  0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 1 & ? \\
  0 & 0 & 1 & 1 & ? \\
  \text{...} & & & & \\
  1 & 1 & 0 & 0 & 0 \\
  1 & 1 & 1 & 0 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
\end{array} \]

Decision must flip from 0 to 1 somewhere
Identical except at \( p_2 \)
Lemma 1

0 0 0 0 0 0
0 0 0 1 ?
0 0 1 1 ?

... Decision must flip from 0 to 1 somewhere
1 1 0 0 0
1 1 1 0 1
1 1 1 1 1

Identical except at $p_2$

Consensus protocol is 1-crash tolerant

1 1 X 0 ?
Lemma 1

Consensus protocol is 1-crash tolerant

If result is 0 then (1, 1, 1, 0) is bivalent (depending on whether $p_2$ crashes)

If result is 1 then (1, 1, 0, 0) is bivalent (depending on whether $p_2$ crashes)
Lemma 2 is a bit more involved
What is Lemma 2

- **Lemma 2:** Given a bivalent configuration ($\gamma$) and event $e$ can find configuration $\gamma'$
  - Event $e$ is enabled in $\gamma'$
  - When $e$ is applied to $\gamma'$ the resulting configuration is bivalent.
Configurations

- Any configuration of 1-crash resistant robust consensus protocol is:
  - Bivalent
  - 0-valent
  - 1-valent

- Why?
Bivalent Configurations

\[ \gamma \xrightarrow{\text{Bivalent}} \gamma' \]

\[ \gamma \xrightarrow{\text{Bivalent}} \gamma' \]

Challenge with the lemma: showing that left side is what happens and e is applicable
Diamond Theorem: Events

• Consider some configuration \( C \) and two events \( e_0 \) and \( e_1 \).

• Assume \( e_0 \) involves deliver message to process \( p \) and \( e_1 \) to process \( q \).

Why?
Diamond Theorem: Schedules

- A schedule $\sigma$ is a sequence of events.
- Define two schedules $\sigma_0$ and $\sigma_1$ as non-interfering iff no process appears in both.

Why?
Walking through Proof for Lemma 2

• Assume starting configuration $\gamma$ and event $e$.

• Assume $e$ involves some process $p$. 
Proof Setup

- Assume starting configuration $\gamma$ and event $e$.

- Assume $e$ involves some process $p$. 
Proof

• Does D contain any bivalent configurations?

• Prove this by contradiction.
Proof Sketch

• Assume no bivalent configuration in D.

• All configurations must be 0-valent or 1-valent.

• First show that there exist both 0-valent and 1-valent configuration in D.
Proof

- Assume $D$ contains no bivalent configurations.
- We can reach a 0-valent and 1-valent configuration from $\gamma$.
  - Call these $\gamma_0$ and $\gamma_1$ respectively.
- If $\gamma_0$ is in $R$, then $e(\gamma_0)$ is in $D$ and is 0-valent.
Proof
Proof

• We can reach a 0-valent and 1-valent configuration from $\gamma$.
  
  • Call these $\gamma_0$ and $\gamma_1$ respectively.

• If $\gamma_0$ is in R, then $e(\gamma_0)$ is in D and is 0-valent.

• If $\gamma_0$ is not in R then there must exist configuration C in R such that
  
  • C is on the path between $\gamma$ and $\gamma_0$.
  
  • $e(C)$ is in D and is 0-valent.
Proof
Proof
Proof Sketch

• Assume no bivalent configuration in D.

• All configurations must be 0-valent or 1-valent.

• First, show that there exist both 0-valent and 1-valent configuration in D.

• Show that there exist two neighboring configurations \( c \) and \( c' \) in R s.t.:
  • \( d_0 = e(c) \) and \( d_1 = e(c') \); \( d_0 \) is 0-valent and \( d_1 \) is 1-valent
  • Show this is a contradiction to original assumption.
Proof

• D contains a 0 and 1 valent configuration -- $d_0$ and $d_1$.

• Claim: There exist $c$ and $c'$ in $C$ such that

  • $c' = f(c)$, $d_0 = e(c)$, $d_1 = e(c')$
Proof
Proof
Proof
Proof

- Two options
  - Event $e$ and $f$ happen on the same process
  - Event $e$ and $f$ happen on different processes
Proof

- If e and f happen on different processes.
- Apply Diamond theorem to move event
- Contradiction: Left event is not 0-valent and is bivalent.
Proof

- If $e$ and $f$ happen on the same process $p$.
- Consensus algorithm must work even if $p$ is silent.

Contradiction: Nodes in $A$ had decided, $A$ cannot be bivalent.
Proof Sketch

• No deterministic 1-crash-robust consensus algorithm exists for async model.

1. Use Lemma 1 to pick an initial bivalent configuration.

2. Given a bivalent configuration \((c)\) and event \(e\) which has been enabled longest.
   • Take the path from \(c\) to \(c'\) where \(e\) is still enabled in \(c'\).
   • Apply \(e\) to \(c'\) to get a new bivalent configuration \(c''\).

3. Repeat step 2.
Quiz!!