1 Consensus Reduction

Proposition 1 \( \forall n, \) Consensus is unsolvable for \( n \) processors given Byzantine failures \( f \) if \( n \leq 3f \).

Proposition 2 Consensus is unsolvable given a system of 3 processors with one Byzantine fault.

We want to perform a reduction from proposition 2 to proposition 1. Remember from complexity, for a reduction of an algorithm \( B \) to \( A \), \( B \leq_c A \), if there’s an algorithm for \( A \) then there’s an algorithm for \( B \). Then there’s the contrapositive: if there isn’t an algorithm for \( B \) then there isn’t an algorithm for \( A \).

Lemma 1 If proposition 2 holds, then proposition 1 holds.

Proof: For contradiction, assume \( \forall n, \) if for \( n \leq 3f \) consensus is solvable for \( n \) processors given a number of Byzantine failures \( f \) then Consensus is solvable given a system of 3 processors with one Byzantine fault. Assume proposition 2 holds. For contradiction, assume that proposition 1 does not hold such that there is an algorithm \( A \) that solves consensus in a system with \( 3f \) processors and up to \( f \) Byzantine failures. (Since we know there’s not an algorithm for \( B \). The reduction will show there’s not an algorithm for \( A \).)

Let three processors \( P_1, P_2 \) and \( P_3 \) be processors. Copy \( P_1 \)'s output to virtual processors \( 0 \ldots f - 1 \), \( P_2 \)'s output to virtual processors \( f \ldots 2f - 1 \) and \( P_3 \)'s output to virtual processors \( 2f \ldots 3f - 1 \). One of the processors \( P_1, P_2 \) or \( P_3 \) will experience a Byzantine failure; therefore, \( f \) virtual processors will also have a Byzantine fault. Since we assumed proposition 1 does not hold, algorithm \( A \) solves consensus, and its output can be fed back to processors \( P_1, P_2 \) and \( P_3 \). Thus there exists an algorithm solving consensus for 3 processors with one Byzantine fault. We have a contradiction since we assumed that 2 holds. Thus Lemma 1 holds.
2 Atomic Snapshot Objects

Atomic Snapshot Objects have two operations:

- \( \text{scan}_i \): where the response is \( \text{return}_i(v) \) where \( v \) is an \( n \)-element vector.
- \( \text{update}_i(d) \): where \( d \) is the data written to \( p_i \)'s segment, whose response is \( \text{ack}_i \).

We will be using atomic snapshot objects for our Borowsky Gafni simulation. It is important to note, that atomic snapshot objects are no more powerful than single-writer, multi-reader, atomic registers.

2.1 BG (Borowsky Gafni) Simulation

The Borowsky Gafni Simulation is a shared memory algorithm that allows a set of \( f + 1 \) processors to wait-free simulate a larger system of \( n \) processors of which at most \( f \) may crash.

The simulation uses a \textit{safe agreement module} that runs on the atomic snapshot model. Lines 2-7 are wait-free, while lines 8-10 are non-blocking, but require busy-waiting. If any processor fails in lines 2-7, it is possible that \( \text{level}_i = 1 \). Then for all other processors, line 8 will result in an infinite loop once it reaches \( \text{level}_i \). For any thread where a failure occurs, all other processors will also not reach agreement. There are \( f + 1 \) such threads, \( f \) of which can contain a failure. There exists at least one thread on which all \( n \) processors will successfully execute the safe agreement module since at most \( f \) processors will fail. Thus our safe agreement module lets us simulate a system of \( n \) processors with at most \( f \) failures using a shared memory algorithm for \( f + 1 \) processors with \( f \) failures.
Algorithm 1 Protocol for $P_i$

1: procedure $\text{propose}(v)_i$
2: \hspace{1em} $\text{val}_i := \text{value}$ \hspace{2em} $\triangleright$ Initially $\text{level}_i = 0, \text{val}_i = 1$
3: \hspace{1em} $\text{level}_i := 1$
4: \hspace{1em} $\text{scan}([\text{val}_j, \text{level}_j])_j$
5: \hspace{1em} if $\exists j, \text{level}_j = 2$
6: \hspace{2.1em} then $\text{level}_i := 0$
7: \hspace{2.1em} else $\text{level}_i := 2$
8: \hspace{1em} repeat $\text{scan}(\text{val}_i, \text{level}_i)$ of all $P_j$
9: \hspace{2.1em} until $\forall j, \text{level}_j \neq 1$
10: \hspace{1em} return $\text{val}_i$
11: end procedure