A Model Equivalence Proof

Consider the two distributed message-passing models below:

*Model A*: asynchronous processors and asynchronous communication
*Model B*: synchronous processors and asynchronous communication

**Definition 1** Model X is at least as strong as Model Y if for every execution $e$ in Model X there exists an execution $e'$ in Model Y such that every processor goes through the same sequence of states in both executions.

We show that Model A and B are both equally strong. It is trivial to argue that Model B is at least as strong as Model A since any execution in Model B is also an execution in Model A. We prove that Model A is at least as strong as Model B.

**Theorem 1** Model A is at least as strong as Model B.

**Proof**: Consider any execution $e$ of Model A. We construct execution $e'$ of Model B as follows. Consider the computation events in $e$ in order and place them in the same order in $e'$. For each $\text{comp}(p)$ event $\phi$ in $e$, for all messages received (processed) by $p$ in event $\phi$ in $e$, insert the corresponding delivery events to right before $\phi$ in $e'$. At this point, there is a bijection between the events in $e$ and $e'$.

Now, we add computation events to $e'$ resulting in execution $e''$ to satisfy processor synchrony. To make sure these are dummy events, we make an assumption on processor algorithms that a processor does not change state or send messages in a computation event in which it receives no messages.

We argue that the sequence of states of each processor is the same in both executions. Also, we prove that each event in $e''$ is applicable at the point it is applied, proving that $e''$ is valid. We also show that $e''$ is admissible, completing the proof of correctness.

**Claim 2** Let $\phi_1, \phi_2, \ldots$, be the computation events in order in $e$. For all $i$, the state of each processor right after event $\phi_i$ is the same in execution $e$ and $e'$. Also, the prefix of execution $e'$ upto event $\phi_i$ is valid.
Proof: (by induction on the number of computation events)

(Base case) Initially, all processors are in initial state in both $e$ and $e'$. Also, empty executions are trivially valid.

(Induction step) Assume $e'$ is valid upto $\phi_k$ and $p$ has the same state after $\phi_k$ in executions $e$ and $e'$, for all processors $p$. Let $q$ be the processor whose computation event is $\phi_{k+1}$. By the construction, for each message $m$ processed by $q$ in $\phi_{k+1}$, we add the corresponding delivery event $\alpha_m$ right before $\phi_{k+1}$ in $e'$.

We argue that these events are applicable in $e'$. Message $m$ was delivered in $e$ before $\phi_{k+1}$ and was sent in event $\phi_\ell$, where $\ell \leq k$. By the inductive hypothesis, $m$ was sent in event $\phi_\ell$ in $e'$ as well, implying that delivery event $\alpha_m$ is applicable in $e'$. Now, $\phi_{k+1}$ is applicable in $e'$ since processor $q$ has the same state at that point in both executions $e$ and $e'$, and all messages processed by $\phi_{k+1}$ have been delivered in $q$’s inbuffer. So, $e'$ is valid upto $\phi_{k+1}$, as required. Also, $q$ is in the same state and receives the same messages in both $e$ and $e'$, resulting in the same state after $\phi_{k+1}$ in both $e$ and $e'$. All other processors remain in the same state before and after $\phi_{k+1}$ in both executions. This satisfies the inductive hypothesis.

End of Proof of Claim

We now construct $e''$ from $e'$ by inserting extra processor computation events periodically to satisfy processor synchrony. We assumed that if a computation event does not process any message, there is no processor state change. Since each message was delivered in $e'$ immediately before the computation event $\phi_k$ where it was processed, it follows that the inbuffers are empty when these extra computation events take place in $e''$. This, the resulting execution $e''$ also satisfies validity and equivalence with $e$.

Execution $e''$ contains all computation and delivery events in $e$, so since $e$ is admissible, this implies that $e''$ satisfies admissibility for Model A. The processor synchrony required for Model B is satisfied by the additional computation events, making $e''$ admissible for Model B as well.

\[\square\]