Consensus

Justin and Li

Real life case to think about

-You ask your girlfriend what she wants for dinner and she says “anything is fine”, but you end up choosing wrong almost all of the time anyways. How do you know what dinner to choose for both of you when one isn’t making a choice or making a “negative” choice?

-9 generals must decide whether to retreat or fight. 1 of the generals is a traitor and can easily swing the vote if 4 choose retreat and 4 choose fight. The traitor general can also tell the 4 retreat voting generals that he chose to retreat and the 4 fighting generals that he chooses to fight which would be very bad for the 8 generals. On top of this the couriers can be killed/corrupted so some messages may not be received/trusted. How can the generals know how to make a decision without all of this uncertainty in communications and error.

Consensus:Example

- Proposal: move midterm date to another day
- Consensus needed
    - All students must be OK with new date (input)
    - Everyone must know the final decision (agreement)

Definition of Consensus Problem

Agreement: All non-faulty processes must agree on the same (single) value.

Validity: If all the non-faulty processes have the same initial value, then the agreed upon value by all of the non-faulty processes must be the same. (Similar to a black box)

Termination: Each non-faulty process must eventually decide on a value.

-Models for consensus include graphs, rings, and trees.

Fault Tolerance

-Say for a vote to choose a leader, more than half of the votes must go towards a choice. In the case of a fault free system, we need not worry about the validity of the votes for the choice. However, given the possibility that faults can happen in our system, a choice may be chosen incorrectly because a vote had an error and went towards the wrong choice. So the incorrect global choice was chosen.

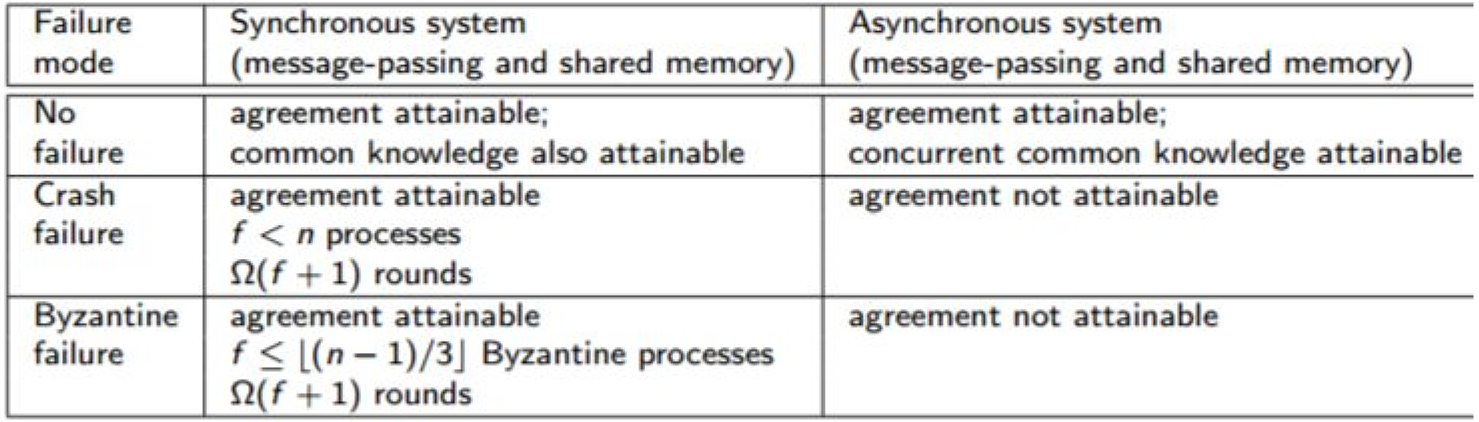
-A process is correct iff it does not experience a failure.

-A protocol that can guarantee consensus of n process in which t fail is t-resilient.

Failure

-See Justin's grades for more details.

- There are 2 types of failures that a process can undergo, crash (Dinner story) or byzantine (Army Story).



Failure mode	Synchronous system (message-passing and shared memory)	Asynchronous system (message-passing and shared memory)
No failure	agreement attainable; common knowledge also attainable	agreement attainable; concurrent common knowledge attainable
Crash failure	agreement attainable $f < n$ processes $\Omega(f+1)$ rounds	agreement not attainable
Byzantine failure	agreement attainable $f \leq \lfloor (n-1)/3 \rfloor$ Byzantine processes $\Omega(f+1)$ rounds	agreement not attainable

Asynchronous Vs Synchronous

- Synchronous system: bounds on
    - Message delays
    - Max time for each process step
    - e.g., multiprocessor (common clock across processors)

- Asynchronous system:
    - No bound on message delays
    - Unbounded delay is indistinguishable from failure of a process\
    - E.g., Internet

- There is no asynchronous deterministic consensus algorithm that tolerates even a single fault.(FLP impossibility proof)

Consensus on synchronous system

- Assumption: Processes fail only by crash-stopping
- For a system with at most f processes crashing, the algorithm proceeds in f+1 rounds (with timeout), using basic multicast (B-multicast).

Consensus on synchronous system-continued
The algorithm:

```
Initially $Values^{0}_{i} = \{\}$ ; $Values^{1}_{i} = \{v_i = x_p\}$
    for round $r$ = 1 to f+1 do
        multicast ($Values^{r}_{i}$)
         $Values^{r+1}_{i} \leftarrow Values^{r}_{i}$
        for each $V_j$ received
         $Values^{r+1}_{i} = Values^{r+1}_{i} \cup V_j$
        end
    end
    $y_p = d_i$ = minimum($Values^{f+1}_{i}$)
```

Why does it work?

- Proof by contradiction

Application to Our Group

- Consider each robot that we connect with as an agent. Each agent is a node in a graph and each connection is an edge.

- We are applying graph grammar rules to a local set of agents to try to apply a general rule to the global set of multi-agents.

- Consider example in paper “Programmable Self-Assembly” by Eric Klavins where some graph grammar rules are:

$$\Phi = \begin{cases} a \quad a \rightharpoonup b - b & (r_1), \\ a \quad b \rightharpoonup b - c & (r_2), \\ b \quad b \rightharpoonup c - c & (r_3). \end{cases} \qquad (1)$$

Using the rules in the previous slide we can obtain the following can obtain state b,c, or d depending on probability and programming. For our application some hooking mechanisms will likely be move favoured, and of course mean that only some parts of the robots can connect to other parts of the robots which means we will have different rules than what was presented in the paper, but some general ideas still overlap.

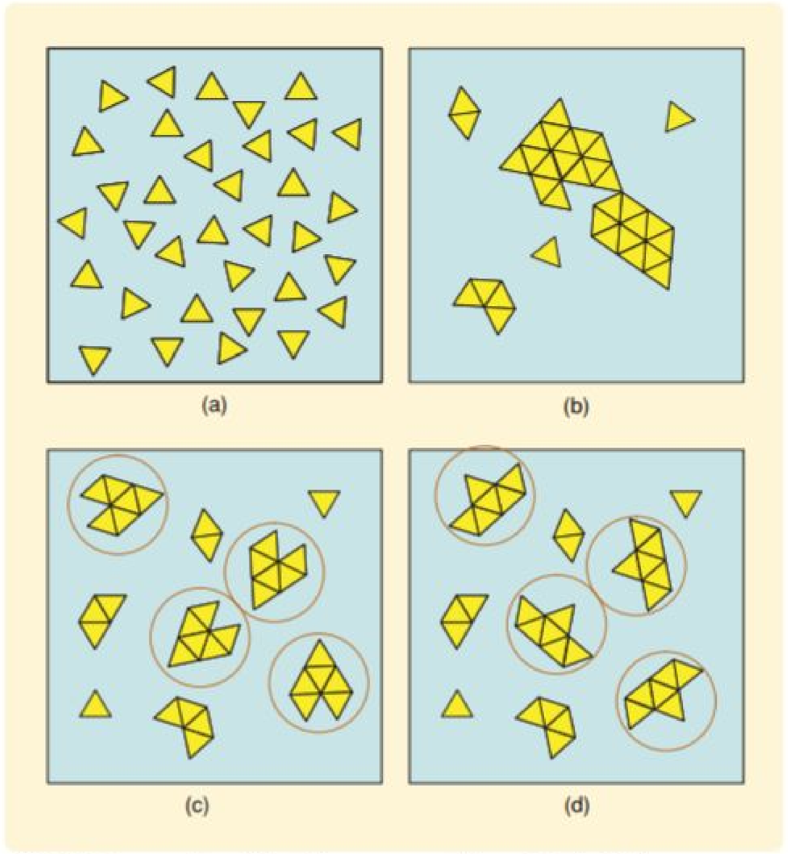


FIGURE 3 The reconfigurability of the programmable parts testbed. The programmable parts can be programmed to self-assemble into a specified structure by means of a graph grammar. (a) Initially, the programmable parts are not assembled and have identical internal states. (b) With no programming, the parts aggregate into a crystalline structure whose overall shape cannot be precisely predicted. (c), (d) Different graph grammars result in the predictable emergence of structures.

Research Application Continued…

- The purpose of using consensus algorithms is for the robots to decide on a state, and for us to be able to consider any errors that can happen. Our communication is asynchronous; however, in applications it is typical to model things as a synchronous system. We don't want to stop if one robot faults.

- An example of a graph grammar application for robotics is to move a skid and can be achieved through the following example:

$$\Phi_3 = \left\{\begin{array}{lll}
a \;-\; c \;\rightharpoonup\; d \quad e, & & (r_4)\\[6pt]
\begin{tikzcd}[sep=small] & b \arrow[dl, dash] \arrow[dr, dash] & \\ e & & g \end{tikzcd} \;\rightharpoonup\; \begin{tikzcd}[sep=small] & b \arrow[dl, dash] \arrow[dr, dash] & \\ c \arrow[rr, equal] & & h \end{tikzcd}, & & (r_5)\\[6pt]
b \;-\; h \;\rightharpoonup\; f \;-\; b, & & (r_6)\\[6pt]
d \;-\; f \;\rightharpoonup\; g \;-\; a, & & (r_7)
\end{array}\right.$$

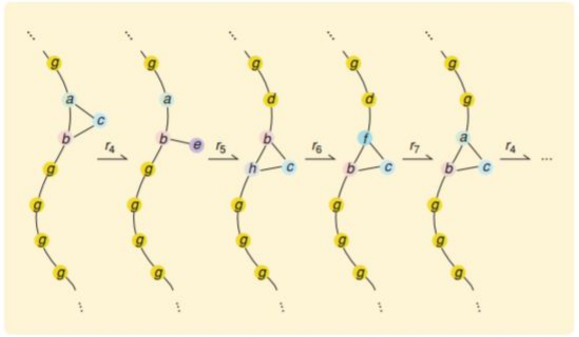


FIGURE 12 Illustrative trajectory of the ratchet system. A grammar need not describe a self-assembling system. In this example, the grammar instead determines how a particle moves along a substrate in a repeating sequence of four basic steps.