The consensus hierarchy (continued)

Reference: [Herlihy 1991].

Consensus numbers A type T has consensus number n if T objects and registers can solve wait-free (i.e. the model is asynchronous, tolerating any number of halting failures) consensus among n processes but not among $n+1$ processes (if no such n, then consensus number $= \infty$).

∞	compare&swap
.	
.	
.	
4	
3	
2	queue, stack, test&set
1	registers, snapshots

Atomic m-register Collection of registers
- can perform writes to m registers in a single atomic action.

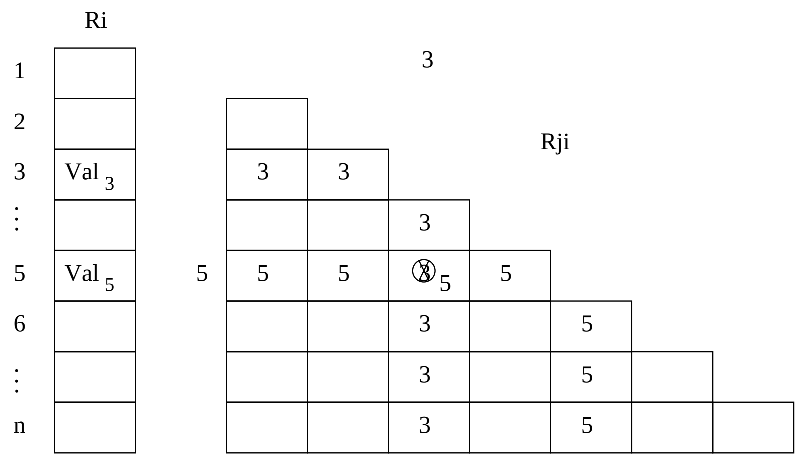
Formally:

State Vector of natural numbers.

Operations have form

read(i) $\longrightarrow$ read i^{th} component of vector

m-write(i_1, v_1, i_2, v_2, ..., i_m, v_m) $\longrightarrow$ updates component i_j to v_j ($\forall$ j)



We give a consensus algorithm that uses an atomic n-register that has $n(n+1)/2$ components, $R_1, \ldots, R_n$ and R_{ij}, for $1 \le j < i \le n$. In a single step, process j writes its input value to R_j and its id to every register in row j and column j of the R_{ij} array.

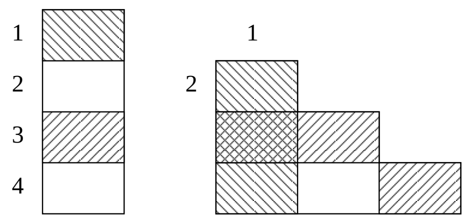
Let $i > j$ be distinct processes. How to tell which of processes i, j wrote first?

Read R_{ij}
Read R_i
Read R_j
If R_{ij} is non-empty
    If R_i is empty then R_j went first
    Else if R_j is empty then R_i went first
    Else re-read R_{ij}, which will contain later of the 2 processes
Else $\longrightarrow$ neither process has gone

consensus algorithm (for process i):

write input value into R_i %%% single
write i into row i & column i % atomic action
look for anyone who went before you
    if so, look for process who preceded him
        etc.
        until you find someone who precedes everyone
output value stored in his register

Every process returns the input value of the first process to access the atomic n-register. Thus we have cons(atomic n-register) $\geq n$



Fact: can improve algorithm to solve consensus among 2m-2 processes.
can't do any better.
so cons(m-register) = 2m-2, if m $\neq$ 1

Consensus hierachy

∞ compare&swap
.
.
.
2m-2 atomic m-registers
.
.
.
2 stacks, queues
1 registers, counters

Theorem If cons(T_1) = n, and cons(T_2) = m ($m < n$), then
there does not exist wait-free implementation of T_1
from object of type T_2 and registers that works in a system of n processes.

Proof Suppose $\exists$ implementation.
By assumption, $\exists$ consensus algorithm A for n processes using T_1 and registers.
In A, replace each object of type T_1 by implementation from T_2 and registers
$\longrightarrow$ yields algorithm A' that solves consensus for n processes using T_2 objects and registers
contradicts the assumption that cons(T_2) $<$ n

e.g. There doesn't exist wait-free implementation of a stack from registers in a system of 2 processes.

Universality of consensus

Herlihy's Universality

Theorem If cons(T) = n, then
$\exists$wait-free implementation of *every* object type from T-objects and registers in a system of $\leq$ n processes.

e.g. Stacks and registers can implement anything in system of 2 processes,
but cannot implement C&S in 3-process system.

Proof Idea: try to come up with a linearization of all the simulated operations on object of type T.
Use linked list that will store operations in the order they are to be linearized.