FDS 2012:
Consensus and Set Agreement

So far…

Shared-memory communication:

- safe bits => multi-valued atomic registers
- safe bit => atomic bit (with constant overhead)
- Atomic registers => atomic/immediate snapshot
- Atomic registers ⇔ Iterated Immediate Snapshot (non-blocking, for tasks)

Today

Reaching agreement in shared memory:

- Consensus
  - ✓Impossibility of wait-free consensus
- Set agreement
  - ✓ Impossibility of wait-free consensus (Sperner's lemma)
- 1-resilient consensus impossibility

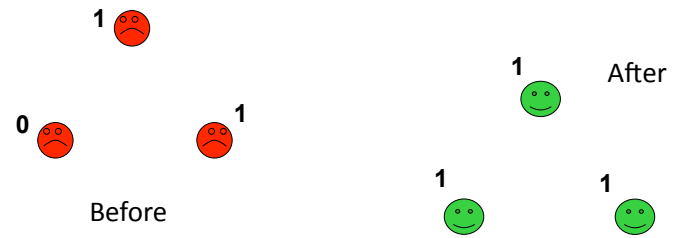
System model

- N *asynchronous* (no bounds on relative speeds) processes $p_0,\ldots,p_{N-1}$ ($N\geq 2$) communicate via atomic read-write registers

- Processes can fail by crashing
  - ✓A crashed process takes only finitely many steps (reads and writes)
  - ✓Up to t processes can crash: t-resilient system
  - ✓t=N-1: wait-free

Consensus

Processes *propose* values and must *agree* on a common decision value so that the decided value is a proposed value of some process



Consensus: definition

A process *proposes* an *input* value in V (|V|≥2) and tries to *decide* on an *output* value in V

- *Agreement:* No two process decide on different values
- *Validity:* Every decided value is a proposed value
- *Termination:* No process takes infinitely many steps without deciding
  (Every correct process decides)

Optimistic (0-resilient) consensus

Consider the case t=0, no process fails

Shared: 1WNR register D, initially T (default value not in V)

Upon propose(v) by process p_i:
  if i = 0 then D.write(v) // if p_0 decide on v
  wait until D.read() ≠ T // wait until p_0 decides
  return D

(every process decides on p_0's input)

Impossibility of wait-free consensus [FLP85,LA87]

Theorem 1 No wait-free algorithm solves consensus

We give the proof for N=2, assuming that
p_0 proposes 0 and p_1 proposes 1

Implies the claim for all N≥2

Proof of Theorem 1

- We show that no 2-process wait-free solution exists for IIS (iterated immediate snapshot memory)

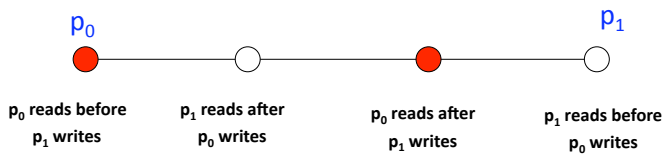
```
r := 0
repeat
        r := r+1
        v_i := IS_r.WriteRead_i(v_i)
until not decided (v_i)
```

- The IIS=>AS non-blocking simulation (last class) implies that no solution exists for (non-iterated) read-write registers

Proof of Theorem 1

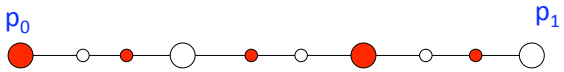
Initially each p_i only knows its input

One round of IIS:



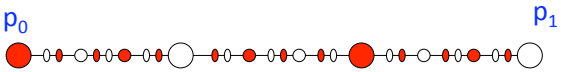
Proof of Theorem 1

Two rounds of IIS:



Proof of Theorem 1

And so on…



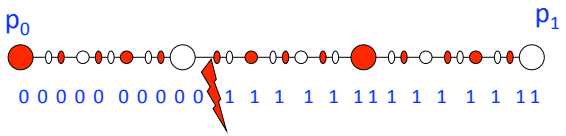
Solo runs remain connected - no way to decide!

Proof of Theorem 1

Suppose p_i (i=0,1) proposes i

- p_i must decide i in a solo run!

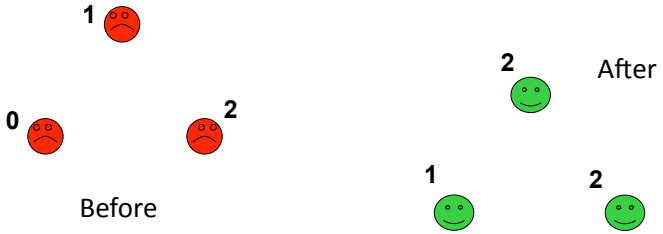
Suppose by round r every process decides



There exists a run with conflicting decisions!

Thus

- Wait-free consensus is impossible (for all N)
- What about weaker forms of agreement?
  - ✓E.g., decide on at most k (k<N) different values



k-Set Agreement: definition

A process *proposes* an *input* value in V (|V|≥k+1) and tries to *decide* on an *output* value in V

- *k-Agreement:* At most k distinct values are decided
- *Validity:* Every decided value is a proposed value
- *Termination:* No process takes infinitely many steps without deciding
  (Every correct process decides)

1-set agreement = consensus
(N-1)-set agreement = set agreement

Impossibility of Set Agreement

Theorem 2 No wait-free algorithm solves set agreement in read-write

- Starting with N values, there is no way to drop one (decide on at most N-1)
- Implies the impossibility of wait-free k-set agreement for all k≤N-1

Solving Set Agreement in IIS

- Each p_i proposes its id i
- There is an IIS round r, such that each process (that reaches round r) outputs some id
- In each run, the set of output ids is a subset of participants of size ≤N-1

Implies Sperner coloring of the subdivided simplex IIS^r

Sperner's coloring of a subdivided simplex

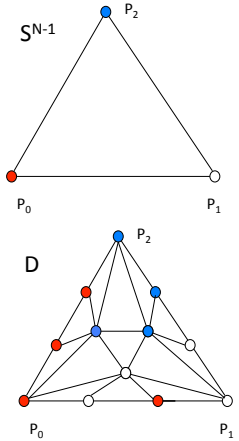
Each vertex of the original simplex S^{N-1} (initial state) is colored with distinct a process id

Each simplex in subdivision D is contained in a face (subset) of S and the union of simplexes of D is S (|S|=|D|)

Sperner's coloring:

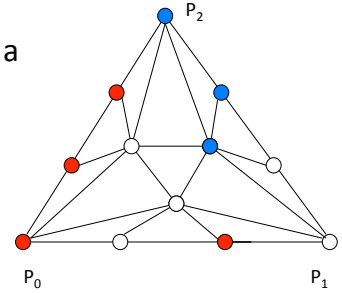
Each vertex v in the subdivision D is colored with colored d:

✓If the vertex belongs to a face S of S^{N-1}, then d is in colors(S)



Sperner's Lemma

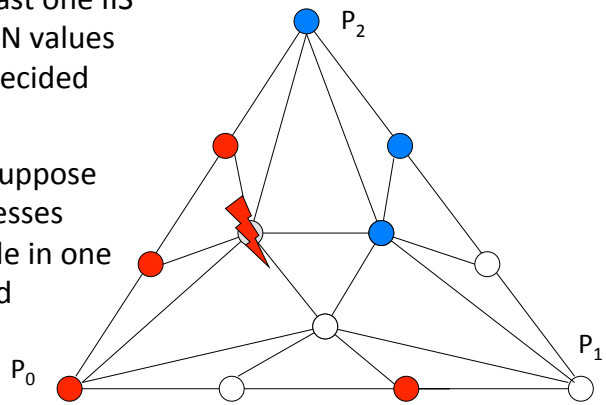
Sperner's coloring of any subdivision D of S^{N-1} has a simplex with all N colors



Corollary: wait-free set agreement is impossible in IIS (and, thus, in read-write)

In at least one IIS run, N values are decided

Here: suppose processes decide in one round

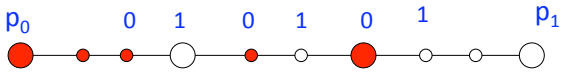


Sperner's lemma: inductive step

Claim: for each k=0,…,N-1, face $p_0,\dots,p_{k-1}$ of S^{N-1} has an odd number of k-dimensional simplexes colored 0,…,k-1

By induction: k=0 - trivial (exactly one)

K=1, simple counting



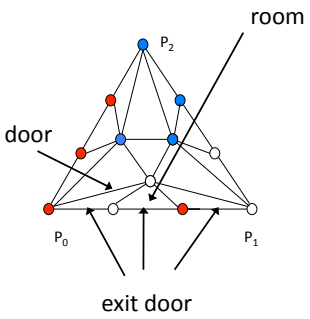
Suppose the claim holds for k<N and consider the face 1,…,k+1

Sperner's: rooms and doors

Each k-simplex contained in face $p_0,\dots,p_k$ is a room

A (k-1)-dimensional face (a subset of k-1 vertices) of a room colored in 0,…,k-1 is a door

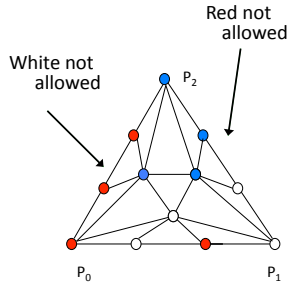
A door is an exit if it is contained in the boundary of $p_0,\dots,p_k$



Sperner's: rooms and doors

There is an odd number of exits!

- By Sperner's coloring no faces other than in $p_0,\dots,p_{k-1}$ can contain simplexes colored 0,…,k-1
- Exits may only be contained in $p_0,\dots,p_{k-1}$
- By induction, $p_0,\dots,p_{k-1}$ contains an odd number of doors (colored with 0,…,k-1)



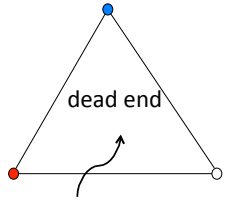
Sperner's: corridors and dead ends

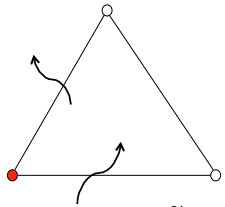
Consider a room with a door (k vertices colored 0,…,k-1)

Two cases possible (depending on the color of the remaining vertex):

- the remaining vertex is colored k - the room has exactly one door (dead end)
  - ✓the room is fully colored (0,…,k)
- The remaining vertex is colored in one of {0,…,k-1} - the room has two doors

We show that there is an odd number of dead ends contained in face $p_0,\dots,p_k$



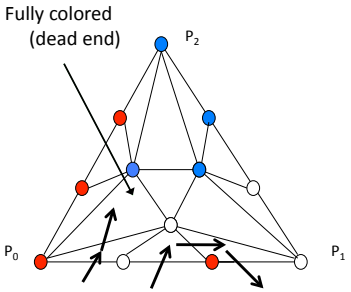


Sperner's: counting fully colored rooms

Start with an exit and walk through the doors

There are only finitely many rooms, thus, two only cases possible:

- Stop in a dead end (fully colored simplex)
- Reach another exit



Sperner's: crossing the rooms

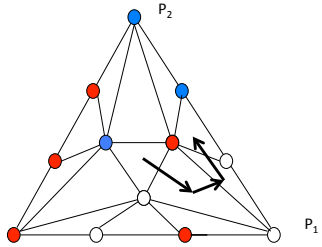
The number of exits is odd

=> The number of dead ends (fully colored simplexes) reachable from exit is odd!

Now start in a dead end not reachable from an exit: can only stop in another dead end not reachable from an exit

⇒ The number of dead ends not reachable from exit is even!

⇒ The total number of fully colored rooms is odd



Thus

- No algorithm can wait-free solve (N-1)-set agreement in IIS
  - ✓otherwise Sperner's coloring of some IIS^r would have no fully colored simplexes (but there is at least one!)
- No algorithm can wait-free solve set agreement in AS (or any read-write model):
  - ✓otherwise we can (non-blocking) simulate it in IIS and thus find a wait-free algorithm in IIS
- We cannot tolerate N-1 failures: can we tolerate less?
  - ✓E.g., can we solve consensus (1-set agreement) 1-resiliently?

1-resilient consensus?

What if we have 1000000 processes and one of them can crash?

NO

We present a direct proof now

(an indirect proof by reduction to the wait-free impossibility also exists)

Impossibility of 1-resilient consensus [FLP85,LA87]

Theorem 2 No 1-resilient (assuming that one process might fail) algorithm solves consensus in read-write

Proof

By contradiction, suppose that an algorithm A solves 1-resilient binary consensus among $p_0,\ldots p_{N-1}$

Proof

By contradiction, suppose that an algorithm A solves wait-free binary consensus among $p_0,\ldots p_{N-1}$

A run of A is a sequence of atomic *steps* (reads or writes) applied to the initial state

A run of A can be seen as and initial input configuration (one input per process) and a sequence of process ids $i_1,i_2,\ldots i_k,\ldots$ (all registers are atomic)

Every correct (taking sufficiently many steps) process decides!

Proof: valence

Let R be a finite run

- We say that R is *v-valent* (for v in {0,1}) if v is decided in every infinite extension of R
- We say that R is *bivalent* if R is neither 0-valent nor 1-valent
  (there exists a 0-valent extension of R and a 1-valent extension of R)

Proof: valence claims

Claim 1 Every finite run is 0-valent, or 1-valent, or bivalent.
(by Termination)

Claim 2 Any run in which some process decides v is v-valent
(by Agreement)

Corollary 1: No process can decide in a bivalent run (by Agreement).

Bivalent input

Claim 3 There exists a bivalent input configuration (empty run)

Proof

Suppose not

Consider sequence of input configurations $C_0,\ldots,C_N$:

C_i: $p_0,..,p_{i-1}$ propose 1, and $p_i,..,p_{N-1}$ propose 0

- All C_i's are univalent
- C_0 is 0-valent (by Validity)
- C_N is 1-valent (by Validity)

Bivalent input

There exists i in {0,…N-2} such that C_i is 0-valent and C_{i+1} is 1-valent!

C_i and C_{i+1} differ only in the input value of p_i (it proposes 1 in C_i and 0 in C_{i+1})

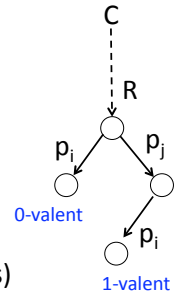
Consider a run R starting from C_i in which p_i takes no steps (crashes initially): eventually all other processes decide 0

Consider R' that is like R except that it starts from C_{i+1}

- R and R' are indistinguishable!
- Thus, every process decides 0 in R' --- contradiction (C_{i+1} is 1-valent)

Critical run

Claim 4 There exists a critical (bivalent) finite run R and two processes p_i and p_j such that R.i is 0-valent and R.j.i is 1-valent (or vice versa)



Proof of Claim 4: By construction, take the bivalent empty run C (by Claim 3 it exists)

We construct an ever-extending fair (giving each process enough steps) run which results in R

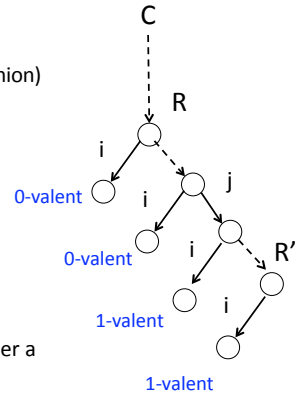
Proof of Claim 4: critical run

repeat forever

take the next process p_i (in round-robin fashion)

if for some R', an extension of R, R.i is bivalent **then** R:=R'.i

else stop



- If never stops – ever extending (infinite) bivalent runs in which every process is correct (takes infinitely many steps – contradiction with termination
- If stops – (suppose R.i is 0-valent) – consider a 1-valent extension
  - ✓ There is a critical configuration between R and R'

Proof (contd.)

Take a critical run R (exists by Claim 4) such that:

- R.0 is 0-valent
- R.1.0 is 1-valent

(without loss of generality, we can always rename processes or inputs appropriately ☺)

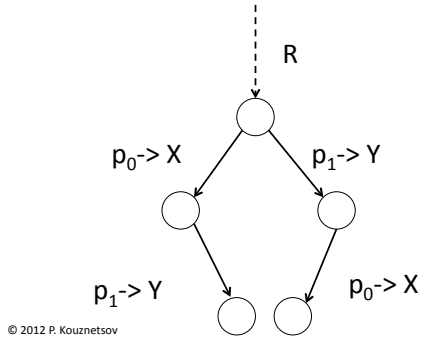
Proof (contd.): the next steps in R

Four cases, depending on the next steps of p_0 and p_1 in R

- p_0 and p_1 are about to access different objects in R
- p_1 reads X and p_0 reads X
- p_0 writes in X
- p_1 reads X

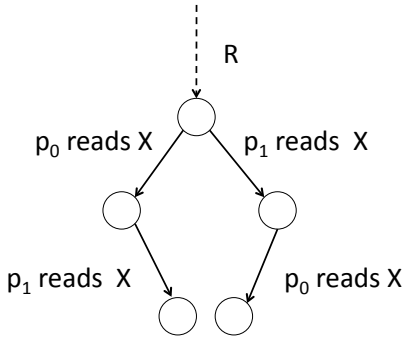
Proof (contd.): cases and contradiction

- p_0 and p_1 are about to access different objects in R
  - ✓R.0.1 and R.1.0 are indistinguishable



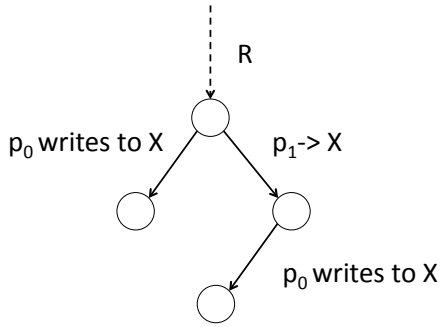
Proof (contd.): cases and contradiction

- p_0 and p_1 are about to read the same object X
  R.0.1 and R.1.0 are indistinguishable



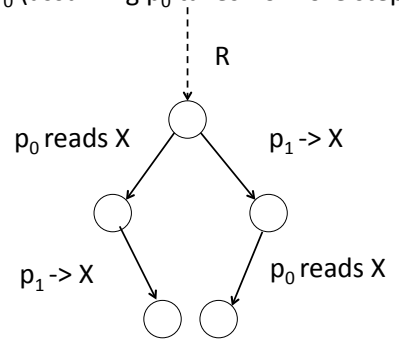
Proof (contd.): cases and contradiction

- p_0 is about to write to X
  - ✓Extensions of R.0 and R.1.0 are indistinguishable for all except p_1 (assuming p_1 takes no more steps)



Proof (contd.): cases and contradiction

- p_0 is about to read to X
  - ✓Extensions of R.0.1 and R.1.0 are indistinguishable for all but p_0 (assuming p_0 takes no more steps)



So…

- No critical run exists
- A contradiction with **Claim 4**

⇒1-resilient consensus is impossible in read-write

- 1-resilient consensus is impossible in read-write
- Can be generalized to:
  - ✓k-resilient k-set agreement is impossible in read-write
  - ✓(by a reduction to the wait-free case)

Next class:

- showing that consensus is universal (wait-free implements any sequential object)
- Solving consensus with stronger (than read-write) objects

Next tutorial: Q&A (homework: go through the previous homework ☺)