CSC2415 Advanced Distributed Computing

- Deterministic abortable (DA) obj.
  - → wait free
  - → linearizable

Like ordinary obj except ops that experience contention (i.e. are concurrent with other operations) may return "abort" ($\perp$) without having any effect on the object.

- → Regular obj cannot be impl. with registers. Need better sync primitives, so weaken.
- → background
  - Obstruction free objects
    Weakening of wait-free obj.

    def obstruction-freedom = op that eventually executes solo (no contention) terminates and returns normal response.

    fact $\forall n$, $\forall$ linearizable obs-free obj shared by n procs can be impl. using only registers.

exercise give an obs-free impl. of consensus for 2 procs using only registers.

- "Pausable" objects
  - → Every op invoked by correct process eventually returns ctrl to caller
  - → Op. that encounters no contention returns "normal" response.
  - → An operation that encounters conten° can:
    - (a) return normal response
    - (b) ——— $\perp$ and have no effect.
    - (c) ——— "pause" in which case it may or may not have taken effect. ~~Ab~~
      - → at this point, caller must resume until (a) or (b).

fact $\forall n$, $\forall$ linearizable pausable obj shared by n procs can be impl. using only registers.

fact if pause is not an option, then $\exists$ obj. that can't be impl. using only registers.
- → specifically abortable consensus (without pause) for 2 procs cannot be impl. using only registers.

→ Wait-free consensus for 2 procs using a "DAC" obj. for 2 procs. $\mathcal{D}$ = 2-DAC objects
$\mathcal{R}$ = registers

proof

P_0 : propose (v)	P_1 : propose (v)
$d_0 := D.\mathrm{prop}(v)$	$x := v$
if $d_0 = \perp$ then return x_1	repeat $d_1 := D.\mathrm{prop}$
else d_0	until $d_1 \neq \perp$
	return d_1

can be impl using registers

- → Non det abortable objects
  Same as pausable but (a) return normal resp.
  (b) return $\perp$ in which case it may or may not have taken effect.
- → Query abortable objects
  Try to find out fate of aborted op.

• Back to DA objects.

$\perp$: means did not take effect.

~~Reminder~~ Def every obj has a type:

$$T = (OP, RES, Q, n, \delta)$$

ops, response, states, ports, trans. func.

⚠ not allow 2 procs to access same port ⚠

$$\delta : Q \times OP \times \{1..n\} \to Q \times RES$$

$$(q, op, i) \mapsto (q', res)$$

DA counterpart of T:

$$T^{da} = (OP, RES \cup \{\perp\}, Q, n, \delta^{da})$$

$$\delta: \quad q \xrightarrow{op\uparrow R} q' \qquad \text{becomes} \qquad \delta^{da}: \quad q \xrightarrow{op\uparrow R} q'$$

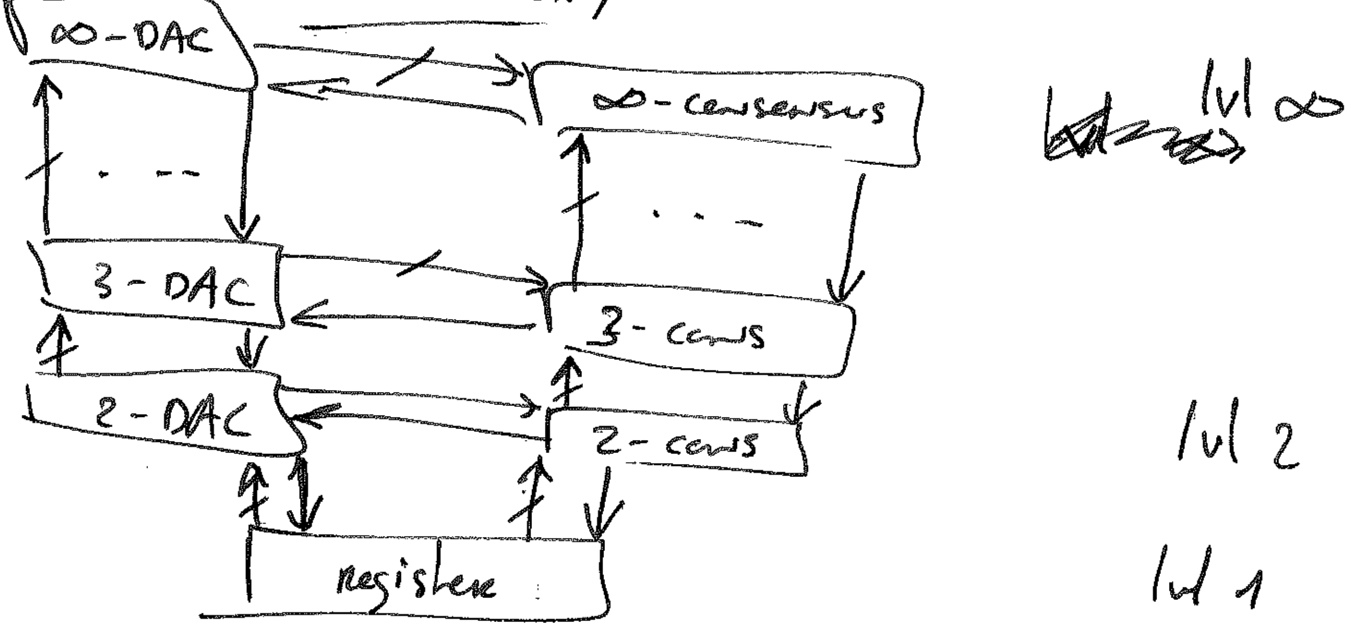
$$q \xrightarrow{op\uparrow \perp} q$$

<u>def</u> n-DAC : n-ported DA counterpart of n-consensus

<u>Non-triviality</u> (1) An op. can return $\perp$ only if it is concurrent w/ another op.
[safety]

(2) An op. that is interrupted by the crash of the invoker can cause only a finite number of concurrent ops to abort.
[liveness]

<u>DAC hierarchy</u>



Level$^{n}_{\vee}$ of a T^{da} in DAC hierarchy = max n for which T^{da} & registers can impl. n-DAC.

<u>fact</u> ~~DA~~: • T & S^{da} ~~is~~ $\wedge$ registers $\to$ 2DAC

• T & S^{da} ~~⇒~~ $\wedge$ registers $\not\to$ 3-DAC

<u>proof</u> Sup. T & $S^{da} \to$ 3-DAC

then $T\&S \to$ 3-DAC

$\Rightarrow$ 2-$T\&S \to$ 3-DAC

$\Rightarrow$ 2-~~T&S~~ $\to$ 3-DAC

$\Rightarrow$ 2-DAC $\to$ 3-DAC

<u>Conjecture</u> If T is an ordinary type at level n of the Consensus hierarchy, then T^{da} is at level n of the DAC-hierarchy.

<u>disproof</u> ~~sta~~

<u>def</u> sticky register

~~ini~~tially $\triangle$, read but write once.

<u>idea</u> DA-stick. reg.

FLP impossibility -> FD model

- FD model step
  - receive a msg
  - query FD
  - send a msg
- MR alg ($t < \frac{n}{2}$)
  - coordinator at round r is $p_{r \bmod n}$ ~~majority~~
  - decide iff all ~~received~~ msg from correct ~~are~~ same.

- Too general:
  - e.g. if $\exists p$ has initial val. 0 then ret Π
    else ret $\emptyset$

- Not general enough:
  - e.g. Heartbeat FD –

- Failure Pattern:
  - Π : processes ; $\mathbb{N}$ = real time
  - $F : \mathbb{N} \to \mathcal{P}(\Pi)$
    $F(t) = \{ p \in \Pi \mid \text{perished at time } t \}$

- Failure History (with range R) — $R \subseteq \mathcal{P}(\Pi)$
  - $H : \Pi \times \mathbb{N} \to R$
    $H(p,t) =$ value ret by p's FD module at t.

. FD $\mathcal{D}$: failure pattern $\mapsto$ set of FD histories.

$\mathcal{D}(F)$ = set of possible beh of $\mathcal{D}$ when Fail.Pat. is F.

. Redefinition:

$\Diamond S$: $H \in \Diamond S(F) \Leftrightarrow$ ① $\forall p \forall q \exists t, p \in correct(F) \wedge q \notin correct(F) \wedge t' \geq t \wedge q \in H(p,t')$

② ~~...~~ other property.

. Redef step:

. formally: (p, m, d) — process, msg received, FD value

. Redef schedule:

. seq of steps.

. Redef run of alg $\mathcal{A}$ using FD $\mathcal{D}$.

. $R_{\mathcal{A}} = (F, H, I, S, T)$ — seq of times. — seq of steps in run (sched) — initial config — FD hist. $\to H \in \mathcal{D}(F)$ — failure pat.

props $\to$ $\forall t \in T$, if p crashed at t' ~~then $t \geq t'$~~

. Environment def:

$\mathcal{E}$ = set of failure pat.

Alg A solve P using FD D in env $\mathcal{E}$

. Weakest FD to solve P in env $\mathcal{E}$.

. Def: $D \geq_{\mathcal{E}} D'$ $\Leftrightarrow$ D provides at least as much info as D' (in $\mathcal{E}$)

$\Leftrightarrow$ $\exists$ A alg that uses D to simulate D' (in $\mathcal{E}$).

e.g. $P \geq_{\mathcal{E}} HB$ (change set of procs into #)

. Leader FD Ω

. Quorum FD Σ

. output: quorum: set of procs

. reqs: (1) any 2 quorums (at any times and any processes) must intersect.

(2) eventually quorum of correct process must contain only correct.

. Weakest FD to solve consensus in env. $\mathcal{E}$.

. FD $\chi_{\mathcal{E}}$ s.t. (1) $\exists$ alg that uses $\chi_{\mathcal{E}}$ to solve consensus.

(2) $\forall$ FD D that ~~uses $\chi_{\mathcal{E}}$ to solve~~ can be used to solve consensus,

$D \geq_{\mathcal{E}} \chi_{\mathcal{E}}$

anecdote

leader quorum

(Ω, Σ)

. Thm $\forall \mathcal{E}$, the weakest FD to solve consensus is

$$MR \longleftrightarrow (\Omega, \Sigma)$$

intersec° prop gives maj. Leon aacron!

Ben-Or also can be

For (1): modify MR

(2): $\forall \mathcal{E}, \forall D$ FD that can be used to solve consensus in $\mathcal{E}$

(a) $D \geq_{\mathcal{E}} \Omega$ [CHT92]

(b) $D \geq_{\mathcal{E}} \Sigma$ [DFG03]

Exercise: implement Σ in asynch if $\mathcal{E} \to \exists\, f < \frac{n}{2}$
→ wait for maj of

. Proof of (b)

. Any FD $\mathcal{D}$ solving consensus $\overset{\text{wlog}}{\text{in}}$ $\mathcal{E}$ can be transformed to $\underbrace{\mathcal{D}^*}_{\Sigma}$ in $\mathcal{E}$.

. Let $\mathcal{A}$ be any alg solving consensus in $\mathcal{E}$.

Two interacting components: directed acyclic graph

(1) DAG building

(2) extraction component (of Σ)

e.g. $(p, d, 1) \to (p, d', 2)$

$(q, e, 1) \to (p, d', 2)$ — q sent its DAG to p.

fair if $\forall$ correct, has infinitely many samples.

① G_p monotonically increases $\Rightarrow G_p^\infty$ "limit graph" of p

② If $(q, d, k) \to (q', d', k')$ in G_p then q took kth sample and saw d before q' took its k'th sample and saw d'

$p: G_p(t) \xrightarrow{\infty}$

$q: G_q(t') \xrightarrow{\infty}$

$\exists\, t' > t$

③ if p is correct G_p^∞ contain a "fair" path (all correct have ∞ly many samples on that path).

. Simulated schedules of A.
run $R = (F, H, -, -, -)$ of DAG-bldg alg

Fix g path in G_p.

e.g.: $g: (p_1, d_1, k_1) \to (p_2, d_2, k_2) \to (p_3, d_3, k_3)$

note: no need to be consecutive ks, even for same process ($\circ \to \circ$?)

. Fix initial config I of $\mathcal{A}$

. Let S : set of schedules of $\mathcal{A}$ that are:

. applicable to I

. compatible with g

$$I = C_0 \longrightarrow C_1 \longrightarrow C_2$$

$$S = (p_1, m_1, d_1) \quad (p_2, m_2, d_2)$$

$m_1 = \lambda(\text{null})$

. Simulated schedules of A

Consider $R = (F, H$ $\cancel{*}$

g: path in some DAG

I: init config of $\mathcal{A}$

G_p | (S1) If S is comp w/ g and applicable to I then $\exists T$, s.t. (F, H, I, S, T) is a run of $\mathcal{A}$.

G_p^{∞} | (S2) If g is fair then $\exists S^{\infty}$ that is compatible w/ g & appl. to I and $\exists T^{\infty}$, $(F, H, I, S^{\infty}, T^{\infty})$ is an admissible run of $\mathcal{A}$.

- $Sch(G, I)$ = set of schedules compat w/ some path $\in G$ and appl. to I.
- p maintains variable u_p = "recent" sample taken by p.
- p keeps building its DAG while

G_p restricted to nodes u_p and ~~taken~~

$Sch(G_p \mid u_p, I_0)$ contains S_0 such that p decides in $S_0(I)$.

all inputs = 0

- and

all inputs = 1

$Sch(G_p \mid u_p, I_1)$ contains $S_1(I_1)$

- When this happens, p computes new quorum

$\Sigma\text{-out}_p :=$ participants $(S_0) \cup$ part. (S_1)

ok.

$u_p :=$ most recent sample of p.

careful about things happening at the same time.

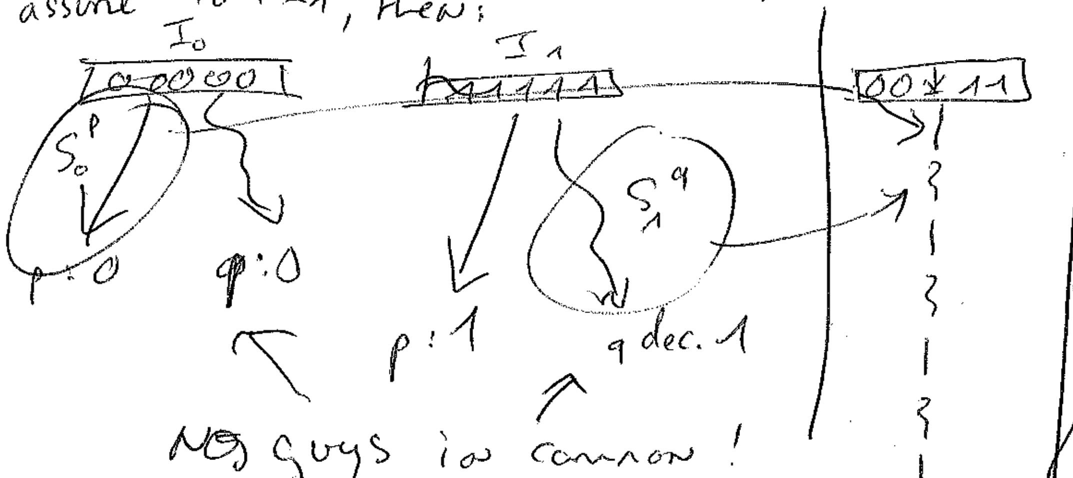
~~$Sch(G, I)$~~ =

proof of prop(?) of quorum

~ eventually continue pointing to things ahead?

(1)

assume $I_0 \neq I_1$, ~~then~~:



construct a 3rd RUN from 2 possible RUNS, then we don't have CONSENSUS!

$\mathcal{A}$ solves consensus using $\mathcal{D}$ in $\mathcal{E}$

→ show $\mathcal{D} \succeq_{\mathcal{E}} \Omega$

Organise initial configs $I_0 \dots I_n$ (n processes) in tree

→ Simulation Tree T_G^I

nodes: scheds in $Sch(G, I)$

edges: $S \to S'$ iff $S' = S \cdot \langle p, m, d \rangle$

→ Simulation Forest $\tilde{F}_G$

$\tilde{F}_G = \{T_G^{I_0}, T_G^{I_1} \dots T_G^{I_n}\}$

$\forall$ connect p, q

$G_p^\infty = G_q^\infty$

$= G$ ← limit DAG

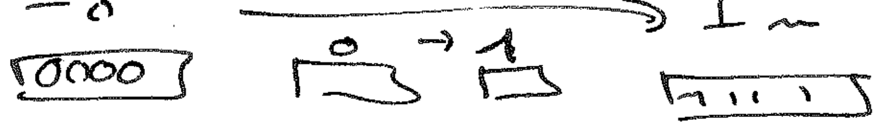
~~→~~ $\tilde{F}_G$ has infinite length path (inf sched.)

\+ fan out (for FD infinite domain)

→ Tagging nodes in T_G^I

tag node v of T_G^I with $\overbrace{0, 1}^{\text{unival.}}$, $\overbrace{\{0, 1\}}^{\text{bivalent}}$

Fact 1: $\tilde{F}_G$ contains a tree w/ bivalent root or $\exists i, 0 \le i \le n$ s.t. root of $T_G^{I_{i-1}}$ is 0-val & root of $T_G^{I_i}$ is 1-val.



Fact 2: If root of $T_G^{I_{i-1}}$ is 0-val & root of $T_G^{I_i}$ is 1-val then p_i is correct.

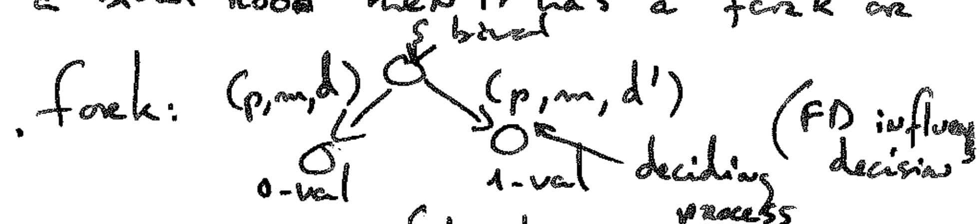
proof: by contradiction, p_i changes the decision.

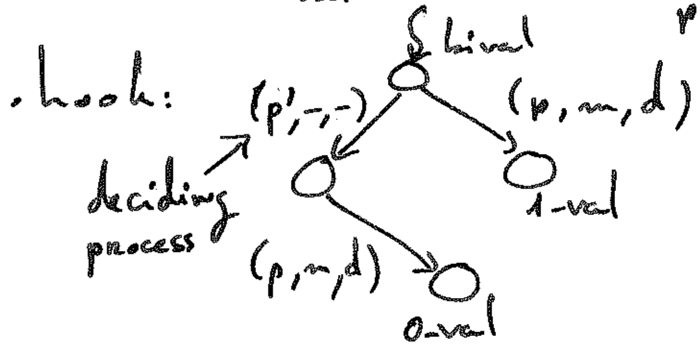
Fact 3: If a tree has a bival root then it has a bival node, s.t. $\exists p, \exists m$, $S \cdot (p,m,d)$ is a node and

$\forall$ node of the form $S \cdot E \cdot (p,m,-)$ is unival.

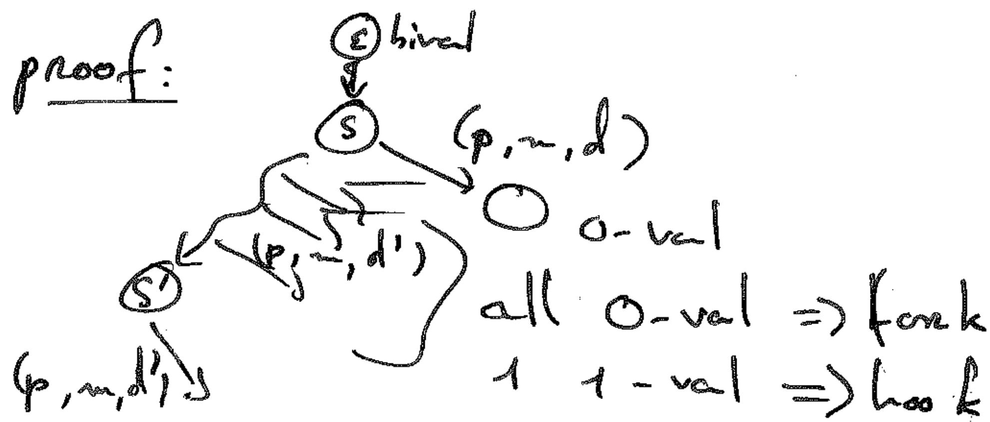
proof: by contradiction. show break consensus

Fact 4: If a tree has a bival root then it has a fork or a hook:



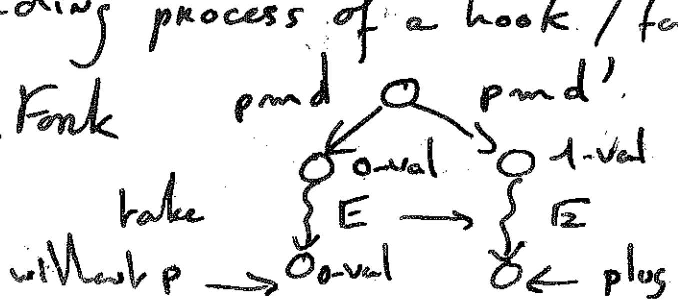


proof:



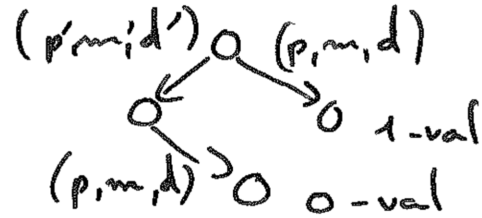
Fact 5: The deciding process of a hook / fork is correct.

proof: . Fork



Assume p is $\neg$correct then $\exists Z$ such that compatible w/ right side.

. Hook: case 1: $p = p'$
(same as fork)



Comment: FD is not monotone!

$(p, S, t_1) \to$

$(p, 4, t_2) \to$

case 2, $p \neq p'$

E w/out p' ~~...~~ taking steps after d'.

→ Rule for picking correct p from $\mathcal{F}_G$.

Let i be min ~~s.t.~~ $0 \le i \le n$, s.t. either (a) $T_G^{I_i}$ is bival. or (b) $T_G^{I_{i-1}}$ is 0-val and $T_G^{I_i}$ is 1-val. If (b), then pick p_i is correct (by fact 2). If (a), then choose smallest hook or fork (in some encoding of graphs) and pick the deciding process (by fact 5).

→ Applying rule to procs:
Each p based on its DAG G_p constructs its own simulation forest.

Fact 6: ~~For~~ $\forall S \in \mathcal{F}_G, \exists t, \forall$ correct p, $\forall t' \geq t, \mathcal{F}_p(t')$ contains S & correct tags for S.

Have to use fact 6, bc $\exists t, \; G_p(t) = G_q(t)$ for correct p, q is not true, since it is possible for 2 correct processes are constantly exchanging messages.

So now, just wait for fact 6, then that all tags ~~done~~ have been applied (s.t. all correct have same min. fork/hook).

FLP → Lemma 1 : ∃ bival init config _
(origin of proof)
Lemma 2 : ∃ hook _

A solving consensus w/ D

→ if it violates lemma 1 then

0 → 1

locate connect!

→ ________ 2 then hook

the guy who takes step (decides) is connect!

→ How to extract Ω when A solves Non-unif consensus

→ how to tag? cannot know connect. ← NU agreem^t. not 2 connect ~~pass~~ decide $\neq$

just tag iff in subtree has self deciding tag

<u>history</u>: ~~x~~ → p good hypo ^_^

→ How to extract Σ ________

→ cannot use same arg as before, but can extract Σ^{ν} (where intersec° property holds for connect)

→ Weakest FD to solve NU consensus:

(Ω, Σ^{ν}).

— End of class —

1. Cover Welsh
2. Present set-timeliness (k-set agreement)
3. Ω^{i}_{j} ~ ~~task~~