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" (L) without having any effect on the object.

- Regular obj cannot be impl. with registers - need better synced primitives, so weaken.

- Background

Obstruction free objects

Weakening of wait-free obj

\[
\text{def obstruction-free} = \text{op that eventually executes solo (no contention) terminates and returns normal response}
\]

**Fact** Any obstruction-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 connect process eventually
  returns chld to caller.
- Op that encounters no conflict then returns
  "normal" response.
- An operation that encounters conflict can:
  (a) return normal response
  (b) —— I and have no effect
  (c) —— "pause" in which case it may or may
      not have taken effect.
- At this point, caller must resume until (a)
  or (b).

Fact 1: A linearizable pausable Obj shared by n
  procs can be impl. using only registers.

Fact 2: If pause is not an option, then 3 Obj.
  that can't be impl. using only registers.

Specifically, abortable consensus (without pause) for 2 procs cannot be impl. using only

Wait-free consensus for 2 procs using a "DAC" Obj. for 2
  procs. D = 2-DAC objects
  X = registers
proof \[ p_0 : \text{propose}(v) / \quad p_1 : \text{propose}(v) \]
\[
\begin{align*}
&d_0 := D.\text{prop}(v) \\
&\text{if } d_0 = 1 \text{ then return } d_0 \\
&\text{else } d_0
\end{align*}
\]
\[
\begin{align*}
&x_i := v \\
&\text{repeat } d_i := D.\text{prop} \\
&\text{until } d_i \neq 1 \\
&\text{return } d_i
\end{align*}
\]

- Now det abortable objects
  - Same as pausable but
    (a) return normal resp.
    (b) return \( \bot \) in which case it may or may
      not have taken effect

- Query abortable objects
  - Try to find all facts of aborted op-

- Back to DA objects -
  - \( \bot \) means did not take effect-

- Every def every req has a type:

\[
T = (\text{OP, RES, } q, m, \delta)
\]

- do not allow 2 proc's to access same port
- ops response states ports means, finc.

\[
\delta : Q \times \text{OP} \times \text{RES} \rightarrow Q \times \text{RES}
\]

\[
(q, op, i) \mapsto (q', res)
\]

DA counterpart of \( T \):
\[ \tau_{\text{da}} = \{ \text{op}, \text{res}, \text{us}, \text{ss}, \text{a}, \text{m}, s^d_{\text{da}} \} \]

\( \tau_{\text{da}} \) becomes

\[ \tau_{\text{op}} \]

\[ \tau_{\text{us}} \]

\[ \tau_{\text{res}} \]

\[ \tau_{\text{ss}} \]

\[ \tau_{\text{a}} \]

\[ \tau_{\text{m}} \]

\[ \tau_{s^d_{\text{da}}} \]

\[ \tau_{\text{a}} \]

\[ \tau_{\text{m}} \]

\[ \tau_{\text{ss}} \]

\[ \tau_{\text{res}} \]

\[ \tau_{\text{us}} \]

\[ \tau_{\text{op}} \]

**def** \( n \)-DAC : \( n \)-ported DA counterpart of \( n \)-consensus

**Non-triviality**

1. An op. can return \( 1 \) only if it is concurrent \( \perp \) another op.

2. An op. that is intercepted by the crash \( \perp \) of the invoker can cause only a finite number of concurrent ops to abort.

**DAC hierarchy**

\[ \text{100-DAC} \]

\[ \text{...} \]

\[ \text{3-DAC} \]

\[ \text{2-DAC} \]

\[ \text{1-DAC} \]

\[ \text{resister} \]
level of a \( T^m \) in DAC hierarchy = max \( m \) for which \( T^m \) & registers can incl. m-DAC

\[ \text{fact: } \text{DAC: } T^m \& S^m \Rightarrow \text{registers } \Rightarrow 2\text{DAC} \]
\[ T^m \& S^m \Rightarrow \text{registers } \Rightarrow 3\text{-DAC} \]

proof Suppose \( T^m \& S^m \Rightarrow 3\text{-DAC} \)

Then \( T^m \& S^m \Rightarrow 3\text{-DAC} \)
\[ \Rightarrow 2\text{-DAC} \Rightarrow 3\text{-DAC} \]
\[ \Rightarrow 2\text{-DAC} \Rightarrow 3\text{-DAC} \]

Conjecture If \( T \) is an ordinary type at level \( m \) of the Consensus hierarchy, then \( T^m \) is at level \( m \) of the DAC-hierarchy.

disproof via

Def. Sticky register: \( R \)

Initially \( \Delta \), read but write once

idea DA-stick reg.
FLP impossibility \implies FD model

- FD model step
  - receive a msg
  - query FD
  - send a msg
- NR alg. \( k < \frac{2}{3} \)
  - coordination at round \( n \) is \( \text{P}_{\text{mod} m} \) majority
  - decide iff all received msg from connected same

- Too general:
  e.g. if \( T \) has initial val. 0 then set \( T \)
  else set \( \emptyset \)

- Not general enough:
  e.g. Heartbeat FD

- Failure Pattern:
  \( T \) : processes; \( N = \text{real line} \)
  \( F : N \rightarrow \mathcal{P}(T) \)
  \( F(k) = \{ p | p \in T \text{ crashed at time } k \} \)

- Failure History (with range \( R \)) \( R \subseteq \mathcal{P}(N) \)
  \( H : T \times N \rightarrow R \)
  \( H(k) = \text{value set by } p \text{'s FD module at } k \)
FD $D$ : failure pattern $\rightarrow$ set of FD instances.

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

Redefinition:

1. $S: \forall E \in S(F) \exists p \forall q \exists t, p \cdot \text{connect}(F) \land \forall q \in \text{connect}(F) \land \forall \chi \geq t \land q \in N(p)$

2. There $p$ other properly.

Redef. step:

- Formally: $(p, m, d)$

Redef. schedule:

- seq of steps.

Redef. run of Alg $A$ using FD $D$.

$PA = (F, V, I, S, T)$ seq of lines.

$\text{FD hist, initial config.}
\text{FD hist. initial config.}$

$P \in D(F)$

$\text{props} \forall F \in T, \text{if } p \text{ ceased at } v, \text{ then } k$.

Environment def:

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

Alg $A$ solve $P$ using FD $D$ in env $E$. 
Least FD to solve P in env \( E \).

Def: \( \text{FD} \geq \text{FD}' \text{ in } E \) if \( \text{FD} \) provides at least as much info as \( \text{FD}' \text{ in } E \).

\( \exists \) an alg. that uses \( \text{FD} \) to simulate \( \text{FD}' \text{ in } E \).

\( \text{e.g. } \text{FD} \geq \text{HB} \) (change set of proc into \#)

leader FD \( \Omega \).

Quorum FD \( \Sigma \).

each public quorum: set of proc.

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

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

Least FD to solve consensus in env. \( E \).

FD \( \Sigma \) s.t. (1) alg. that uses \( \Sigma \) to solve consensus.

(2) \( \forall \text{FD} \) that uses \( \Sigma \) to solve can be used to solve consensus.

Leader quorum \( (\Omega, \Sigma) \).

\( \forall E \), the weakest FD to solve consensus is

MR \( \Rightarrow (\Omega, \Sigma) \) (Least FD that can be used to solve consensus).

For (1): modify MR

(2): \( \forall E \forall \text{FD} \) that can be used to solve consensus

(a) \( D \geq \Sigma \) \( \Omega \)

(b) \( D \geq \Sigma \) \( \Sigma \)
Exercise: implement $F$ in a system if $F < \frac{3}{4}$.

Proof of (b)

1. Any FD solving consensus in $E$ can be transformed to $D_n$ in $E$.

2. Let $\Sigma$ be any alg solving consensus in $E$.

Two interacting components:

1. DAG building
2. Extraction component (of $\Sigma$)

\[ (p, d, t) \rightarrow (p, d, z) \]

\[ (q, e, t) \]

Fair if connected

1. $G_p$ monotonically increases $\Rightarrow G_p \succeq \text{init graph}$

2. If $(q, d, k) \rightarrow (q', d', k')$ in $G_p$ then $q$ took $k$th sample and $q'$ took $k'$th sample and said $k'$th

3. If $p$ is connected to $r$ contains a "fair" path (i.e., connect $p$ to $r$ and only many samples as the path)

Simulated schedules of $A$:

Run $R = (F, H, -, -, -)$ of DAG $b$-ldy
Fix a path in $G_p$.

- $g: (p_1, d_1, k_1) \rightarrow (p_2, d_2, k_2) \rightarrow (p_3, d_3, k_3)$

Note: no need to be consecutive $k_i$, even for same process ($s \rightarrow s'$).

- Fix initial config $I$ of $A$.
- Let $S^r$: set of schedules of $A$ that are:
  - applicable to $I$
  - compatible with $g$

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

Singleton schedules of $A$

Consider $R = (F, H, \emptyset)$

$g$: path in some DAC
$I$: init config of $A$

- If $S$ is comp w/ $g$ and applicable to $I$ then $\exists T$, s.t. $(F, H, I, ST)$ is a run of $A$.

- If $g$ is fair then $\exists S$, that is compatible w/ $g$ & applicable to $I$ and $\exists T$, $(F, H, I, S, T)$ is an admissible run of $A$. 
Sch (C, I) = set of schedules compat. w/ some path \( C \) and appl. to \( I \).

- \( p \) maintains variable \( u_p \) "recent" sample taken by \( p \).
- \( p \) keeps building its DAC until all inputs = 0
  nodes \( u_p \) and taken \( \mathcal{S}_0 \) contains \( S_0 \) such that \( p \) decides in \( S_0 (I) \).
- and all inputs = 1
  \( \mathcal{S}_1 \) contains \( S_1 (I) \)
- When this happens, \( p \) computes new query

\[ \Sigma - \text{out} \_r = \text{participants} (S_0) \cup \text{part.} (S_1) \]

\[ \text{im} = \text{most recent sample of } p \]

---

proof of prop of quorum
- eventually achieve quorum, holding abort.

---

construct a 3rd run from 2 possible runs, then we don't have consensus!

No guys in common!
A solves consensus using O in E

- show \( N \subseteq E \)

Organise initial configs \( I_0 \rightarrow I_m \) (2 processes) in tree

- \( T_G \) Simulational Tree \( T_G \)
  - empty schedule
    - nodes: scheds in
      - \( Sch(G_b, I_b) \)
    - edges: \( s \rightarrow s' \)
      - \( s' = S_{\exists \langle p, m, d \rangle} \)

- \( F_G \) Simulation Forest \( F_G \)
  - \( G \rightarrow \infty \)
  - \( G \rightarrow \infty \)

- connect \( p, q \)

- \( G_p \rightarrow G_q \)

- \( G \) limit DAE

Tagging nodes in \( T_G \)

- \( \hat{v} v \) tag node \( v \) in \( T_G \) with \( 0, 1, \overline{0}, \overline{1} \)

Fact 1: \( F_G \) contains a tree w/ bounded root or \( 3 \), \( 0 \leq i \leq m \)

Such: root of \( T_{G_i-1} \) is 0-val & root of \( T_{G_i} \) is 1-val.
Fact 2: If root of $T^1_G$ is 0-val & root of $T^2_G$ is 1-val then $p_i$ is connect.

proof: by contradiction, $p_i$ changes the decision.

Fact 3: If a tree has a bival root then it has a bival root s.t. $3p, 3m, S \cdot (p, m, d)$ is a node and a node of the form $S \cdot E \cdot (p, m, -)$ is arrival.

proof: by contradiction. slow break consensus.

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

- fork: $(p, m, d) \rightarrow (p, m, d')$ (FD influencing decision)
- hook: $(p, m, d)$

proof:

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

proof: Fork $pmd \rightarrow pmd'$
hook: case 1: \( p = p' \) (same as fork)

\( (p, m, d) \)

\( (p, m, i) \rightarrow \) 0 1-val

\( (p, m, i) \rightarrow \) 0 0-val

- o Rule for picking connect \( p \) from \( F_0 \).

Let \( i \) be min \( \Theta_i \) s.t., s.t. either (a) \( T_0^i \) is bval.

or (b) \( T_0^i \) is 0-val and \( T_0^i \) is 1-val. If (b), then

pick \( p_i \) is connect (by fact 2). If (a), then choose smallest

hook on fork (in some encoding of graphs) and pick the

deciding process (by fact 3).

- o Applying rule to \( p \) cases:

Each \( p \) based on its DAG \( G_p \) constructs its own

simulation forest.

Fact 6: \( \forall p \forall S \in F_0, \exists k, A \text{ connect } p, \forall k \geq k, F_p(k)

contains \( S \) & connect bags for \( S \).


—— Have to use fact 6, be \( \exists k \), \( G_p(k) = G_q(k) \) for \( \forall p, q \)

is not have, since it is possible for 2 connect

processes one constantly exchanging messages—

So now, just wait for fact 6, then that all bags have been applied (s.t. all connect have same fork/look) —


FLP $\rightarrow$ Lema 1: 3-lived init config

A solving

- If it violates lemma 1 then consensus w/o D
- Locate connect

2 then hook

The guy who takes step (decides) is connect!

How to extract $\Sigma$ when $A$ solves non-unif consensus?

- How to bag? cannot know connect.
- Just bag iff in subtree has self deciding bag

History: $p \rightarrow p$ good hypo

How to extract $\Sigma$

- Cannot use same agents before, but can extract $\Sigma^0$ (where intersection property holds for connect)

Weakest FD to solve NU consensus: $(\Omega, \Sigma^0)$.

--- End of class ---

1. Cover Welsh
2. Present set-hindlessness ($k$-set agreement)
3. $\Omega_1$