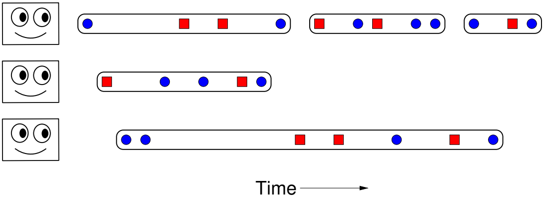
Lower bounds for restricted-use objects

James Aspnes (Yale)
Hagit Attiya (Technion)
Keren Censor-Hillel (MIT)
Danny Hendler (BGU)

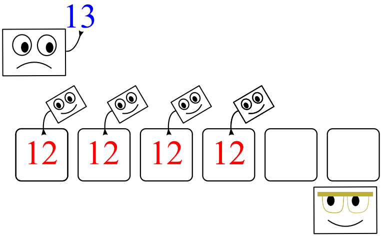
June 26th, 2012

Model



- High-level **operations** implemented by low-level **steps**.
- **Asynchronous**: interleaving of steps controlled by **adversary**.
- **Obstruction-free**: any operation finishes if it runs alone.
- **Historyless** base objects, where a step either doesn't change the state or wipes out previous history.
  - Examples: read/write registers, test-and-set, swap.

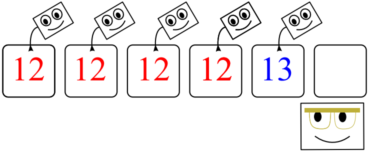
Covering arguments



Historyless objects permit **covering arguments**:

- Suppose first k registers read by reader are **covered** by pending update steps.
- Any new operation must update some other register to be visible.
- This new update can be delayed to cover another register.

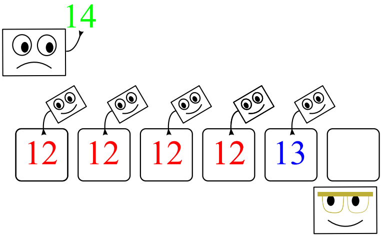
Covering arguments



Historyless objects permit **covering arguments**:

- Suppose first k registers read by reader are **covered** by pending update steps.
- Any new operation must update some other register to be visible.
- This new update can be delayed to cover another register.

Covering arguments



Historyless objects permit **covering arguments**:

- Suppose first k registers read by reader are **covered** by pending update steps.
- Any new operation must update some other register to be visible.
- This new update can be delayed to cover another register.

Perturbable objects

(Jayanti, Tan, and Toeug, SICOMP 2000)

$$\underbrace{\alpha_k}_{\text{prefix}} \quad \underbrace{w_1 w_2 \dots w_k}_{\text{delayed writes } \lambda_k} \quad \underbrace{r_1 r_2 \dots r_k \dots}_{\text{final read} \to x}$$

$$\underbrace{\alpha_k}_{\text{prefix}} \quad \underbrace{\gamma}_{\text{perturbation}} \quad \underbrace{w_1 w_2 \dots w_k}_{\text{delayed writes } \lambda_k} \quad \underbrace{r_1 r_2 \dots r_k \dots}_{\text{final read} \to x' \neq x}$$

$$\underbrace{\alpha_k \quad \underbrace{\gamma'}_{\text{truncated perturbation}}}_{\text{new prefix } \alpha_{k+1}} \quad \underbrace{w_1 w_2 \dots w_k w_{k+1}}_{\text{delayed writes } \lambda_{k+1}} \quad \underbrace{r_1 r_2 \dots r_k r_{k+1} \dots}_{\text{final read}}$$

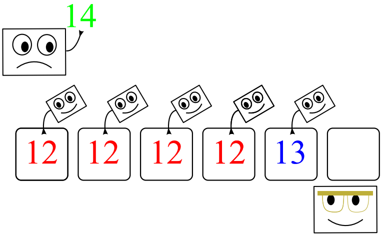
- Object is **perturbable** if γ always exists.
- Choose truncated γ' that leaves delayed write w_{k+1} to first uncovered register read by final operation.
- Iterate $n-1$ times to get lower bound.

JTT lower bound

Theorem (JTT): Any obstruction-free implementation of a perturbable object from historyless base objects requires $n - 1$ steps and $n - 1$ space in the worst case.

Gives lower bounds on:

- counters,
- mod-$2n$ counters,
- fetch-and-increment,
- max registers,
- collects,
- snapshots,
- and many others.



Restricted-use objects

$$\underbrace{\alpha_k}_{\text{prefix}} \quad \underbrace{w_1 w_2 \dots w_k}_{\text{delayed writes } \lambda_k} \quad \underbrace{r_1 r_2 \dots r_k \dots}_{\text{final read } \to m}$$

$$\underbrace{\alpha_k}_{\text{prefix}} \quad \underbrace{\gamma}_{\text{perturbation}} \quad \underbrace{w_1 w_2 \dots w_k}_{\text{delayed writes } \lambda_k} \quad \underbrace{r_1 r_2 \dots r_k \dots}_{\text{final read } \to m}$$

- Consider an m-bounded counter that returns m after any number of increments $\geq m$.
- This is not perturbable: after m increments, further increments have no effect.
- So JTT bound doesn't apply.
- In general, can make *any* object m-limited-use by ignoring all but first m updates.

Examples of restricted-use objects

- m-valued max registers cost $O(\log m)$ (Aspnes, Attiya, Censor-Hillel, JACM 2012).
- m-valued counters cost $O(\log^2 m)$ (ibid).
- m-limited-use snapshots cost $O(\log^2 m \log n)$ (Aspnes, Attiya, Censor-Hillel, Ellen, PODC 2012, to appear).

Unrestricted versions are all perturbable $\Rightarrow \Omega(n)$ cost.
Can we adapt perturbability to apply to restricted-use objects?

L-perturbable objects

perturbable

$\downarrow$

L-perturbable

$\Omega(\log L)$ steps $\quad$ $\Omega(L)$ space $\quad$ $\Omega(\log L)$ stalls

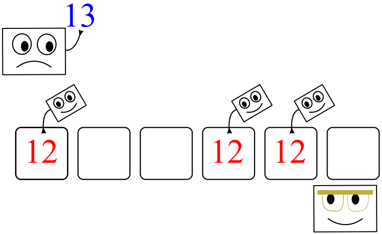
We define a new notion of L-perturbable objects to extend JTT to restricted-use objects.

- Intuition: object is L-perturbable if we can perturb it L times.
- But also have fewer restrictions on structure of executions.

Backtracking covering

(Fich, Hendler, Shavit, FOCS 2005)

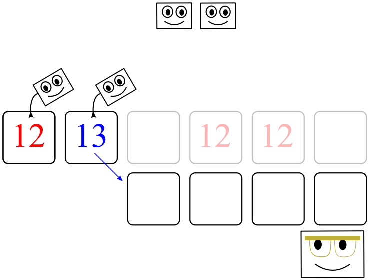
- Can't necessarily cover first k registers read by reader.
- Write to early register might divert reader away from later covered registers.
- This frees up covering processes for re-use.



Backtracking covering

(Fich, Hendler, Shavit, FOCS 2005)

- Can't necessarily cover first k registers read by reader.
- Write to early register might divert reader away from later covered registers.
- This frees up covering processes for re-use.



L-perturbable objects: definition

$$\underbrace{\alpha_k}_{\text{prefix}} \quad \underbrace{\lambda_k}_{\leq 1 \text{ step per process}} \quad \underbrace{op_n}_{\text{final read} \to x}$$

$$\underbrace{\alpha_k}_{\text{prefix}} \quad \underbrace{\gamma}_{\text{ops by } p_\ell \notin \lambda_k} \quad \underbrace{\lambda_k}_{\leq 1 \text{ step per process}} \quad \underbrace{op_n}_{\text{final read} \to x' \neq x}$$

$$\underbrace{\alpha_k}_{\text{prefix}} \quad \underbrace{\gamma' e}_{\text{prefix of } \gamma} \quad \underbrace{\lambda_k}_{\leq 1 \text{ step per process}} \quad \underbrace{op_n}_{\text{final read} \to x''}$$

$$\underbrace{\underbrace{\alpha_k}_{\text{prefix}} \quad \gamma' \quad \underbrace{\lambda'}_{\text{subpermutation of } \lambda_k e}}_{\alpha_{k+1}} \quad \underbrace{\underbrace{\lambda''}_{\text{rest of } \lambda_k e}}_{\lambda_{k+1}} \quad \underbrace{op_n}_{\text{final read}}$$

- Object is L-perturbable if this works until $k = L$ *or* we reach a **saturated** execution where $|\lambda_k| = n - 1$, no matter how we do the $\gamma'/\lambda'/\lambda''$ split.
- Perturbable objects are L-perturbable.

Example: m-bounded counters

$$\underbrace{\alpha_k}_{\text{prefix}} \quad \underbrace{\lambda_k}_{\le k \text{ steps}} \quad \underbrace{op_n}_{\text{final read} \to x}$$

$$\underbrace{\alpha_k}_{\text{prefix}} \quad \underbrace{\underbrace{I'_0}_{\text{finish partial inc}} \quad \underbrace{I_1 \dots I_{k+1}}_{\text{new incs}}}_{\gamma} \quad \underbrace{\lambda_k}_{\le k \text{ steps}} \quad \underbrace{op_n}_{\text{final read} \to x' \neq x}$$

$$\underbrace{\underbrace{\alpha_k}_{\text{prefix}} \quad \gamma' \quad \underbrace{\lambda'}_{\text{subpermutation of } \lambda_k e}}_{\alpha_{k+1}} \quad \underbrace{\underbrace{\lambda''}_{\le k+1 \text{ steps}}}_{\lambda_{k+1}} \quad \underbrace{op_n}_{\text{final read}}$$

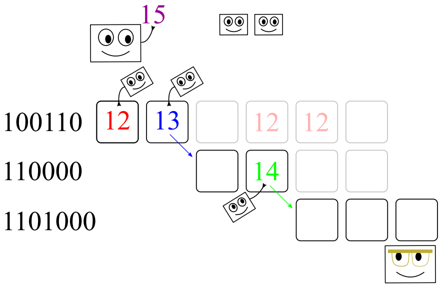
- Invariant: $\alpha_k \lambda_k$ includes $\le k$ partial increments.
- So $k+1$ new increments change value.
- Total over $\sqrt{m}$ stages is $\le m \Rightarrow \Omega(\sqrt{m})$-perturbable.

Lower bounds

We'll use different sequences of perturbations to get different lower bounds:

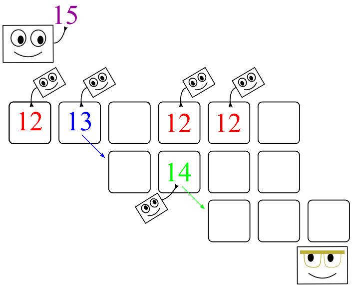
- **Access-perturbation sequence**: gives lower bound on steps.
- **Cover-perturbation sequence**: gives lower bound on space.
- **Access-stall-perturbation sequence**: gives lower bound on stalls (contention) or steps, even for non-historyless base objects.

Access-perturbation sequence



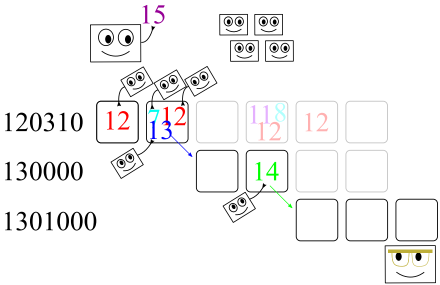
- **Access-perturbation sequence** is a sequence of L perturbations that shows many accesses by reader.
- Associate a bit-vector with each sequence of reader operations: $1 =$ covered register, $0 =$ uncovered register.
- Bit vectors are **lexicographically increasing** ($\Rightarrow$ no repetitions) and **prefix-free**.
- L distinct vectors $\Rightarrow$ some vector has length $\geq \log_2 L$ (or $n-1$ if saturated) $\Rightarrow$ $\Omega(\min(\log L, n))$ steps.

Cover-perturbation sequence



- **Cover-perturbation sequence** shows many registers are covered.
- Like access-perturbation sequence, but never release covering processes.
- L stages $\Rightarrow$ L covered registers (or $n-1$ if saturated) $\Rightarrow$ $\Omega(\min(L, n))$ space.

Access-stall-perturbation sequence



- **Access-stall-perturbation sequence** shows high contention or high steps with arbitrary base objects.
- Vector of bits becomes vector of counts: still lexicographically increasing.
- Gives $\Omega(\min(\log L, n))$ stalls *or* steps.

Randomized implementations

- For randomized implementations, we do not have a general lower bound.
- But we use similar techniques to show an $\Omega\left(\frac{\log\log m}{\log\log\log m}\right)$ lower bound on expected steps for approximate counters, with an **oblivious adversary**, for $m \leq n$.
- This is close to $O(\log\log n)$ upper bound for single-use approximate counters (Bender and Gilbert, FOCS 2011).
- Still open: adapt L-perturbability for general randomized implementations.

Summary of lower bounds

	perturbation bound (L)	step complexity, max(steps, stalls)	space complexity
compare and swap	$\sqrt[3]{m} - 1$	$\Omega(\min(\log m, n))$	$\Omega\left(\min\left(\sqrt[3]{m}, n\right)\right)$
collect	$m - 1$	$\Omega(\min(\log m, n))$	$\Omega(\min(m, n))$
max register	$m - 1$	$\Omega(\min(\log m, n))^*$	$\Omega(\min(m, n))$
counter	$\sqrt{m} - 1$	$\Omega(\min(\log m, n))^*$	$\Omega\left(\min\left(\sqrt{m}, n\right)\right)$
counter within $\pm k$	$\sqrt{\frac{m}{k}} - 1$	$\Omega\left(\min\left(\log \frac{m}{k}, n\right)\right)^*$	$\Omega\left(\min\left(\sqrt{\frac{m}{k}}, n\right)\right)$
counter (randomized)		$\Omega\left(\frac{\log \log m}{\log \log \log m}\right)^\dagger$	

*Step complexity bounds also in (Aspnes, Attiya, Censor-Hillel, JACM 2012)

†Expected steps, when $n \geq m$.