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.
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.
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.
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)

Object is **perturbable** if \( \gamma \) always exists.

Choose truncated \( \gamma' \) that leaves delayed write \( w_{k+1} \) to first uncovered register read by final operation.

Iterate \( n - 1 \) times to get 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-2\( n \) counters,
- fetch-and-increment,
- max registers,
- collects,
- snapshots,
- and many others.
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?
We define a new notion of \textit{\(L\)-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.
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

$$\alpha_k \lambda_k \leq k \text{ steps}$$

$$\text{final read } \rightarrow x$$

$$\alpha_k \lambda_k \leq k \text{ steps}$$

$$\text{final read } \rightarrow x' \neq x$$

Invariant: $\alpha_k \lambda_k$ includes $\leq k$ partial increments.

So $k + 1$ new increments change value.

Total over $\sqrt{m}$ stages is $\leq m \Rightarrow \Omega(\sqrt{m})$-perturbable.
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 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 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** 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.
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

<table>
<thead>
<tr>
<th></th>
<th>perturbation bound ((L))</th>
<th>step complexity, (\max(\text{steps, stalls}))</th>
<th>space complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>compare and swap</td>
<td>(\sqrt[3]{m} - 1)</td>
<td>(\Omega \left( \min \left( \log m, n \right) \right))</td>
<td>(\Omega \left( \min \left( \sqrt[3]{m}, n \right) \right))</td>
</tr>
<tr>
<td>collect</td>
<td>(m - 1)</td>
<td>(\Omega \left( \min \left( \log m, n \right) \right))</td>
<td>(\Omega \left( \min \left( m, n \right) \right))</td>
</tr>
<tr>
<td>max register</td>
<td>(m - 1)</td>
<td>(\Omega \left( \min \left( \log m, n \right) \right))</td>
<td>(\Omega \left( \min \left( m, n \right) \right))</td>
</tr>
<tr>
<td>counter</td>
<td>(\sqrt{m} - 1)</td>
<td>(\Omega \left( \min \left( \log m, n \right) \right))</td>
<td>(\Omega \left( \min \left( \sqrt{m}, n \right) \right))</td>
</tr>
<tr>
<td>counter within (\pm k)</td>
<td>(\sqrt{\frac{m}{k}} - 1)</td>
<td>(\Omega \left( \min \left( \log \frac{m}{k}, n \right) \right))</td>
<td>(\Omega \left( \min \left( \sqrt{\frac{m}{k}}, n \right) \right))</td>
</tr>
<tr>
<td>counter (randomized)</td>
<td></td>
<td>(\Omega \left( \frac{\log \log m}{\log \log \log m} \right))†</td>
<td></td>
</tr>
</tbody>
</table>

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

†Expected steps, when \(n \geq m\).