8.4 Lower Bounds using Any Objects with Limited Contention via Turan's Theorem

Some objects limit the number of processes that can access them. For example, a single-writer register can be accessed by only one process. More generally, in this section, we are interested in objects that can be simultaneously accessed by a only a limited number of processes. We will prove a lower bound on implementations from *any* objects with this restriction.

The *contention* of an object is the maximum, over all reachable configurations, C, of the number of processes that are poised to perform non-trivial operations on the object in configuration C.

The model we consider is synchronous shared memory containing any types of objects with contention at most w. We allow processes to crash, but only at the beginning of an execution.

We prove a lower bound on the number of steps taken in a solo execution, i.e. when all but one of the processes crash before taking any steps. To do so, we consider executions which may involve multiple processes, but which are indistinguishable from solo executions to every nonfaulty process. For any set of processes Q, we define a *Q-independent execution* to be an execution in which only processes in Q take steps and each process in Q only accesses objects that have not been modified by any other processes (in Q). In particular, a solo execution by process p is a $\{p\}$-independent execution. Also, an empty execution is Q-solo for any set of processes Q.

If all the steps of some process $p \in Q$ are removed from a Q-independent execution, then the resulting execution is $(Q - \{p\})$-independent and both executions are indistinguishable to all of the processes in $Q - \{p\}$.

We use the following result from graph theory to find a set of processes that do not acquire any information about one another during a round.

Theorem 8.7 (Turan)**.** *Any graph $G = (V, E)$ has an independent set of size at least $\frac{|V|^2}{|V|+2|E|}$.*

The next lemma shows how to construct Q-independent executions.

Lemma 8.8. *Consider any algorithm for n processes that uses only objects with contention at most w. Suppose that, in every Q-independent execution with $|Q| > 1$, at most one process terminates. Then, for every integer $0 \leq t < \log_{w+2} n$, there is a t-round Q_t-independent execution α_t, where $|Q_t| \geq \frac{n}{(w+2)^t}$.*

Proof. The proof is by induction on t. The case $t = 0$ holds with Q_0 being the set of all processes and α_0 being the empty execution. So, suppose that $t > 0$.

By the induction hypothesis, there is a $(t-1)$-round Q_{t-1}-independent execution α_{t-1}, where $|Q_{t-1}| \geq \frac{n}{(w+2)^{t-1}}$. Let C be the configuration at the end of α_{t-1}. Consider the operation (if any) each process in Q_{t-1} is poised to perform in C.

Let $G = (Q_{t-1}, E)$ be an undirected graph where $\{p_i, p_j\} \in E$ if and only if either p_i and p_j are both poised to perform a non-trivial operation to the same object in C or one of them is poised to access an object in C that was modified by the other process during α_{t-1}.

For each object B, let $w(B)$ denote the number of processes poised to perform non-trivial operations to B in configuration C. Since the contention of every object is at most w, $w(B) \leq w$. The number of pairs of processes that are poised to apply non-trivial operations to the same object in C is $\sum\{\binom{w(B)}{2} \mid B \text{ is an object}\} \leq \frac{|Q_{t-1}|}{w}\binom{w}{2} = |Q_{t-1}|\frac{w-1}{2}$. This is because, by convexity, the sum is maximized when there are as many groups with size w as possible.

Since α_{t-1} is a Q_{t-1}-independent execution, each object is modified by at most one process during α_{t-1}. In C, there are most $|Q_{t-1}|$ processes that are poised to access one of these objects. Thus, there are most $|Q_{t-1}|$ pairs of processes where one of them is poised to access an object that was modified by the other process during α_{t-1}.

Hence, $|E| \leq |Q_{t-1}|\frac{w-1}{2} + |Q_{t-1}| = |Q_{t-1}|\frac{w+1}{2}$. By Turan's Theorem, there is an independent set Q_t in G such that

$$|Q_t| \geq \frac{|Q_{t-1}|^2}{|Q_{t-1}| + 2|E|} \geq \frac{|Q_{t-1}|^2}{|Q_{t-1}| + |Q_{t-1}|(w+1)} = \frac{|Q_{t-1}|}{w+2} \geq \frac{n}{(w+2)^t} \ .$$

Let α'_t be obtained from α_{t-1} by removing all steps of all processes in $Q_{t-1} - Q_t$. Let α_t be an execution obtained by extending α'_t with the next step of each process in Q_t, where, in the last round, all trivial operations precede all non-trivial operations. Since $|Q_t| > 1$, at least one of these processes takes a step in the last round.

By the induction hypothesis, each access in α'_t is to an object that has not been modified by any other process. By construction, each access in the last round of α_t has this property. Thus α_t is a t-round Q_t-independent execution. □

8.5 The Solo Step Complexity of Implementing Weak Test&Set

A *weak test&set* object supports a single operation, T&S, which can either *succeed* or *fail*. In every execution, at most one T&S operation on each object succeeds. A T&S operation must succeed if no other T&S operations on the same object begin until after it has completed.

Theorem 8.9. *Any implementation of an n-process weak test&set object from objects with contention at most w has a solo execution of a T&S operation with $\Omega(\frac{\log n}{\log(w+2)})$ steps.*

Proof. Consider any implementation of a weak test&set object that uses only objects with contention at most w. Let α be any Q-independent execution α of this implementation with $|Q| > 1$. For each process in Q, α is indistinguishable from a solo execution. So, if it terminates, its T&S operation must succeed. In every execution of a correct implementation, at most one T&S operation succeeds. Thus, at most one process terminates in α.

Let $t = \lceil \log_{w+2} n \rceil - 1$. By Lemma 8.8, there is a t-round Q_t-independent execution α_t, where $|Q_t| \geq n/(w+2)^t \geq 2$. Hence, there is a solo execution of a T&S operation that takes at least $t \in \Omega(\frac{\log n}{\log(w+2)})$ steps. $\square$

A somewhat weaker version of this lower bound appears in Hagit Attiya, Faith Ellen Fich, and Yaniv Kaplan, *Lower Bounds for Adaptive Collect and Related Objects*, PODC, 2004, pages 60–69. For $w > 1$, they also give a weak test&set implementation with $O(\log n/\log w)$ step complexity, using $O(n/w)$ registers with contention at most w. Thus, the step complexity and solo-step complexity of implementing weak test&set are the same to within a constant factor and does not improve when more powerful base objects are available.

8.6 Yao's Principle

Given any deterministic algorithm D, an adversary can produce an execution by specifying the inputs each process receives and the order in which the processes take steps.

Consider any complexity measure that assigns a cost *c(D,σ)* to an execution produced by an adversary σ applied to a deterministic algorithm D. Then the *worst case complexity* of D for a set of adversaries S is

$$\max_{\sigma \in S} c(D, \sigma).$$

For any probability distribution over a set of adversaries S, the *average case complexity* of a deterministic algorithm D is

$$\underset{\sigma \in S}{\mathrm{E}} \; [c(D, \sigma)].$$

This is also called the *distributional complexity* of the algorithm D. The *distributional complexity of a problem* is the distributional complexity of the best algorithm that solves the problem, i.e.

$$\min_{D} \; \underset{\sigma \in S}{\mathrm{E}} \; [c(D, \sigma)],$$

where the minimum is taken over all deterministic algorithms D that solve the problem.

For a randomized algorithm, an adversary defines a set of executions, rather than a single execution. The execution that results depends on the outcomes of the coin tosses that each process performs. Thus, a randomized algorithm A can be viewed as a probability distribution over deterministic algorithms $A[\rho]$, where ρ denotes the sequences of coin toss outcomes, one for each process. For example, if each process performs at most r coin tosses, then ρ is distributed uniformly over $(\{0,1\}^r)^n$.

Given an adversary σ, the *expected cost* of a randomized algorithm A is

$$\underset{\rho}{\mathrm{E}} \; [c(A[\rho], \sigma)]$$

and its *worst case expected complexity* for a set of adversaries S is

$$\max_{\sigma \in S} \mathop{\mathrm{E}}_{\rho} [c(A[\rho], \sigma)].$$

The *worst case expected complexity of a problem* is the worst case expected complexity of the best algorithm that solves it, i.e.

$$\min_{A} \max_{\sigma \in S} \mathop{\mathrm{E}}_{\rho} [c(A[\rho], \sigma)],$$

where the minimum is taken over all randomized algorithms A that solve the problem.

We restrict attention to *oblivious adversaries*, which choose the inputs each process receives and the order in which processes take steps before the execution begins. Therefore, they do not depend on the outcomes of the coin tosses performed by the processes during the execution of the algorithm. Note that lower bounds proved for oblivious adversries also apply for more powerful adversaries.

Yao's principle says that the worst case expected cost of a problem against a set of oblivious adversaries S is bounded below by its distributional complexity, for any probability distribution over S.

Lemma 8.10 (Yao's Principle)**.** *For any probability distribution over a set of oblivious adversaries S, $\min_{A} \max_{\sigma \in S} \mathop{E}_{\rho} [c(A[\rho], \sigma)] \geq \min_{D} \mathop{E}_{\sigma \in S} [c(D, \sigma)]$, where A is chosen from a set of randomized algorithms, D is chosen from the set of all deterministic algorithms obtained from randomized algorithms in this set by fixing the outcomes of their its coin tosses, and ρ is chosen from the possible sequences of coin toss outcomes.*

Proof. Consider any randomized algorithm A from the set. Then

$$\max_{\sigma \in S} \mathop{\mathrm{E}}_{\rho} [c(A[\rho], \sigma)] \geq \mathop{\mathrm{E}}_{\sigma \in S} \mathop{\mathrm{E}}_{\rho} [c(A[\rho], \sigma)] = \mathop{\mathrm{E}}_{\rho} \mathop{\mathrm{E}}_{\sigma \in S} [c(A[\rho], \sigma)].$$

The order of the expectations can be interchanged, because all the adversaries are oblivious. Since $A[\rho]$ is a deterministic algorithm for each choice of ρ,

$$\max_{\sigma \in S} \mathop{\mathrm{E}}_{\rho} [c(A[\rho], \sigma)] \geq \mathop{\mathrm{E}}_{\rho} \mathop{\mathrm{E}}_{\sigma \in S} [c(A[\rho], \sigma)] \geq \min_{D} \mathop{\mathrm{E}}_{\sigma \in S} [c(D, \sigma)].$$

Finally, since A is an arbitrary randomized algorithm from the set,

$$\min_{A} \max_{\sigma \in S} \mathop{\mathrm{E}}_{\rho} [c(A[\rho], \sigma)] \geq \min_{D} \mathop{\mathrm{E}}_{\sigma \in S} [c(D, \sigma)].$$

□

Thus, a lower bound on the worst case expected complexity of a problem against a set of oblivious adversaries, S, can be obtained by (carefully) choosing a probability distribution over S and deriving a lower bound on the distributional complexity of the problem under that distribution.

Yao's principle was first presented for the decision tree model in Andrew Yao's paper *Probabilistic Computations: Toward a Unified Measure of Complexity*, 18th FOCS, 1977, pages 222-227. It is possible to obtain versions of Yao's principle for distributed algorithms against more general adversaries, but care is needed. See, for example, the papers *New Lower Bound Techniques for Distributed Leader Finding and Other Problems on Rings of Processors*, by Hans Bodlaender, which appeared in TCS, volume 81, 1991, pages 237-256, *Average and Randomized Complexity of Distributed Problems*, by Nechama Allenberg-Navony, Alon Itai, and Shlomo Moran, which appeared in SIAM J. Computing, volume 25, 1996, pages 1254-1267, and *A Tight RMR Lower Bound for Randomized Mutual Exclusion*, by George Giakkoupis and Philipp Woelfel, which appeared in STOC, 2012, pages 983-1001.

8.7 A Lower Bound for Randomized Implementations of a Max Register

A *max register* is an object that supports two operations,

- $ReadMax$, which returns the current value of the object, and
- $WriteMax(v)$, which sets the value of the object to be the maximum of its current value and v.

We use Yao's principle (Lemma 8.10) to derive a lower bound for the expected step complexity of randomized implementations of a max register from registers against oblivious adversaries. The input to an implementation is the sequence of operations each process performs.

We begin by considering a set of adversaries in which process p_0 is allocated no steps and process p_i performs a single WriteMax(i) operation and is allocated at most w steps, for $i = 1, \ldots, n-1$. For $k_1, \ldots, k_{n-2} \in \{0, \ldots, w-1\}$ and $i \in \{1, \ldots, n-1\}$, let $\sigma'(k_1, \ldots, k_{n-2}, 0)$ denote the adversary that allocates no steps to any process and let $\sigma'(k_1, \ldots, k_{n-2}, i)$ denote the adversary that allocates k_j steps to process p_j, for $j = 1, \ldots, i-1$, followed by w steps to process i, followed by one more step to process p_j for $j = i-1, \ldots, 1$. A process does nothing when it is allocated a step after it has finished its operation.

Let D be an arbitrary deterministic implementation of a max register such that WriteMax has worst case step complexity w. Fix $k_1, \ldots, k_{n-2} \in \{0, \ldots, w-1\}$ and consider the sequence of steps α'_i that occur when D is performed starting from its initial configuration C_0 using the adversary $\sigma'(k_1, \ldots, k_{n-2}, i)$, for $i = 0, \ldots, n-1$.

Then

$$\begin{aligned}
\alpha'_0 &= \\
\alpha'_1 &= \beta_1 \\
\alpha'_2 &= \beta'_1\beta_2\delta_1 \\
\alpha'_3 &= \beta'_1\beta'_2\beta_3\delta_2\delta_1, \\
&\vdots \\
\alpha'_i &= \beta'_1\beta'_2\cdots\beta'_{i-1}\beta_i\delta_{i-1}\cdots\delta_2\delta_1, \\
&\vdots \\
\text{and } \alpha'_{n-1} &= \beta'_1\beta'_2\cdots\beta'_{n-2}\beta_{n-1}\delta_{n-1}\cdots\delta_2\delta_1.
\end{aligned}$$

Here β_i is a sequence of at most w consecutive steps by process p_i in which it completes WriteMax(i). Either β'_i is the prefix of β_i consisting of k_i steps and δ_i consists of the next step by process p_i or $\beta'_i = \beta_i$ and δ_i is empty. In general, β_i depends on the values of $k_1, \ldots, k_{i-1}$ and β'_i and δ_i also depend on the value of k_i, but none of them depend on the values of $k_{j+1}, \ldots, k_{n-2}$. Note that, if δ_i is a read step, it might return different values when $\alpha'_{i+1}, \ldots, \alpha'_{n-1}$ are performed starting from C_0.

Let $v_R(i)$ denote the value of register R in configuration $C_0\alpha'_i$. In particular, $v_R(0)$ is the initial value of register R and, if δ_i is a write to R, then $v_R(i+1) = \cdots = v_R(n-1)$. Let $V_R = \#\{v_R(i) \mid i = 0, \ldots, n-1\}$ denote the number of different values in register R at the end of these n executions. If none of $\beta_1, \ldots, \beta_{n-1}$ write to R, then $V_R = 1$.

Suppose that $v_R(i) \notin \{v_R(0), \ldots, v_R(i-1)\}$. Then $\delta_1, \ldots, \delta_{i-2}$ are not writes to R. Since $v_R(i) \neq v_R(i-1)$, either δ_{i-1} is a write to R, β_i contains a write to R, or there exists $1 \leq f(i) \leq i-1$ such that $\beta'_{f(i)}$ contains a write to R, but $\beta'_{f(i)+1}\cdots\beta'_{i-1}\beta_i$ does not. Note that f is an injective partial function.

For example, suppose R is initially 0, suppose β_1 writes the value 10 to R during β'_1 and later writes the value 20, suppose β_2 writes the value 30 to R during β'_2 and has no further writes to R, and suppose β_3 contain no writes to R. Then $v_R(0) = 0$, $v_R(1) = 20$, $v_R(2) = 30$, $v_R(3) = 10$, and $f(3) = 1$.

If $V_R > d$, then there exist $0 < i_1 < \cdots < i_d \leq n-1$ such that, for $1 \leq u \leq d$, $v_R(i_u) \notin \{v_R(0), \ldots, v_R(i_u - 1)\}$. Since $v_R(i_d) \neq v_R(i_d - 1)$, it follows that $\delta_1, \delta_2, \ldots, \delta_{i_d-2}$ are not writes to R. Furthermore, for $1 \leq u \leq d-1$, if β_{i_u} does not contain a write to R, then $\beta_{f(i_u)}$ contains a write to R. Thus, at least $(d-2)/2 = d/2 - 1$ of the sequences of steps $\beta_1, \beta_2, \ldots, \beta_{i_d-2}$ contain a write to R.

The following lemma shows it is unlikely that a register can contain many different values at the ends of the different sequences of steps $\alpha'_0, \ldots, \alpha'_{n-1}$. The idea is that, if a process p_i writes to the register when it is allocated w steps, then, with probability at least $1/w$, process p_i will cover the register after the adversary first allocates steps to p_i and the register will contain the same value at the ends of $\alpha'_{i+1}, \ldots, \alpha'_{n-1}$.

Lemma 8.11. *For every register R, if $k_1, \ldots, k_{n-2}$ are chosen independently and uniformly from $\{0, \ldots, w-1\}$, then, for any positive integer $d \leq n$,* $\text{Prob}\,[V_R > d] \leq (1 - 1/w)^{d/2-1}$.

Proof. We prove by backwards induction on m that, for all $0 \leq m \leq n-2$, for all $0 \leq c \leq n-m-2$, and for all choices of $k_1, \ldots, k_m \in \{0, \ldots, w-1\}$, if $k_{m+1}, \ldots, k_{n-2}$ are chosen independently and uniformly from $\{0, \ldots, w-1\}$, then

$$\text{Prob}\left[\begin{array}{l}\text{for some } i \in \{0, \ldots, n-1\},\ \text{none of } \delta_1, \ldots, \delta_i \\ \text{are writes to } R \text{ and at least } c \text{ of the sequences} \\ \beta_{m+1}, \ldots, \beta_i \text{ contain a write to } R\end{array}\right] \leq (1 - 1/w)^c.$$

If $c = 0$, then choosing $i = 0$ vacuously satisfies the condition, so the probability is $1 = (1-1/w)^c$. In particular, if $m = n-2$, then $c = 0$, so the claim is true.

Now suppose that $m < n-2$, $c > 0$, and the claim holds for $m+1$. Fix $k_1, \ldots, k_m \in \{0, \ldots, w-1\}$, a deterministic algorithm, and a non-negative integer $c \leq n-m-2$. This defines $\beta_1, \ldots, \beta_{m+1}$ and $\delta_1, \ldots, \delta_m$.

First, suppose that β_{m+1} does not contain a write to R. Then for each choice of $k_{m+1} \in \{0, \ldots, w-1\}$, δ_{m+1} is not a write to R and, by the induction hypothesis, if $k_{m+2}, \ldots, k_{n-2}$ are chosen independently and uniformly from $\{0, \ldots, w-1\}$, then

$$\text{Prob}\left[\begin{array}{l}\text{for some } i \in \{0, \ldots, n-1\},\ \text{none of } \delta_1, \ldots, \delta_i \\ \text{are writes to } R \text{ and at least } c \text{ of the sequences} \\ \beta_{m+2}, \ldots, \beta_i \text{ contain a write to } R\end{array}\right] \leq (1 - 1/w)^c.$$

Hence, if $k_{m+1}, \ldots, k_{n-2}$ are chosen independently and uniformly from $\{0, \ldots, w-1\}$, then the claim also holds for m.

Now suppose β_{m+1} does contain a write to R. Let

$$K = \{k \in \{0, \ldots, w-1\} \mid \delta_{m+1} \text{ is a write to } R \text{ when } k_{m+1} = k\}.$$

In general, K depends on the choices of $k_1, \ldots, k_m$, like β_{m+1} does. If k_{m+1} is chosen uniformly from $\{0, \ldots, w-1\}$, then $\text{Prob}[k_{m+1} \in K] \geq 1/w$. Since $c > 0$, if at least c of the sequences $\beta_{m+1}, \ldots, \beta_i$ contain a write to R, then $i \geq m+1$. Hence, for each $k \in K$,

$$\text{Prob}\left[\begin{array}{l}\text{for some } i \in \{0, \ldots, n-1\},\ \text{none of } \delta_1, \ldots, \delta_i \\ \text{are writes to } R \text{ and at least } c \text{ of the sequences} \\ \beta_{m+1}, \ldots, \beta_i \text{ contain a write to } R\end{array}\ \middle|\ k_{m+1} = k\right] = 0.$$

By the induction hypothesis, for each $k \notin K$, if $k_{m+1}, k_{m+2}, \ldots, k_{n-2}$ are chosen independently and uniformly from $\{0,\ldots,w-1\}$,

$$\begin{aligned}
&\text{Prob}\left[\begin{array}{l}\text{for some } i\in\{0,\ldots,n-1\},\ \text{none of } \delta_1,\ldots,\delta_i\\ \text{are writes to } R \text{ and at least } c \text{ of the sequences}\\ \beta_{m+1},\ldots,\beta_i \text{ contain a write to } R\end{array}\ \middle|\ k_{m+1}=k\right]\\
&=\text{Prob}\left[\begin{array}{l}\text{for some } i\in\{0,\ldots,n-1\},\ \text{none of } \delta_1,\ldots,\delta_i\\ \text{are writes to } R \text{ and at least } c-1 \text{ of the sequences}\\ \beta_{m+2},\ldots,\beta_i \text{ contain a write to } R\end{array}\right]\\
&\le (1-1/w)^{c-1}.
\end{aligned}$$

It follows that, when $k_{m+1},k_{m+2},\ldots,k_{n-2}$ are chosen independently and uniformly from $\{0,\ldots,w-1\}$,

$$\begin{aligned}
&\text{Prob}\left[\begin{array}{l}\text{for some } i\in\{0,\ldots,n-1\},\ \text{none of } \delta_1,\ldots,\delta_i\\ \text{are writes to } R \text{ and at least } c \text{ of the sequences}\\ \beta_{m+1},\ldots,\beta_i \text{ contain a write to } R\end{array}\right]\\
&=\sum_{k=0}^{w-1}\text{Prob}\left[\begin{array}{l}\text{for some } i\in\{0,\ldots,n-1\},\ \text{none of}\\ \delta_1,\ldots,\delta_i \text{are writes to } R \text{ and at least}\\ c\text{of the sequences}\beta_{m+1},\ldots,\beta_i\\ \text{contain a write to } R\end{array}\ \middle|\ k_{m+1}=k\right]\cdot\text{Prob}[k_{m+1}=k]\\
&=\sum_{k\notin K}\text{Prob}\left[\begin{array}{l}\text{for some } i\in\{0,\ldots,n-1\},\ \text{none of}\\ \delta_1,\ldots,\delta_i \text{are writes to } R \text{ and at least}\\ c\text{of the sequences}\beta_{m+1},\ldots,\beta_i\\ \text{contain a write to } R\end{array}\ \middle|\ k_{m+1}=k\right]\cdot\text{Prob}[k_{m+1}=k]\\
&\le\sum_{k\notin K}(1-1/w)^{c-1}\,\text{Prob}[k_{m+1}=k]\\
&=(1-1/w)^{c-1}\,\text{Prob}[k_{m+1}\notin K]\\
&\le(1-1/w)^{c}.
\end{aligned}$$

This proves the claim for m.

By induction, when $m=0$ and $c=d/2-1$, if $k_1,\ldots,k_{n-2}$ are chosen independently and uniformly from $\{0,\ldots,w-1\}$, then

$$\text{Prob}[V_R>d]\le\left[\begin{array}{l}\text{for some } i\in\{0,\ldots,n-1\},\ \text{none of } \delta_1,\ldots,\delta_i\\ \text{are writes to } R \text{ and at least } d/2-1 \text{ of the}\\ \text{sequences } \beta_1,\ldots,\beta_i \text{ contain a write to } R\end{array}\right]\le(1-1/w)^{d/2-1}.$$

□

For each oblivious adversary $\sigma'(k_1,\ldots,k_{n-2},i)$, with $k_1,\ldots,k_{n-2}\in\{0,\ldots,w-1\}$ and $i\in\{0,\ldots,n-1\}$, let $\sigma(k_1,\ldots,k_{n-2},i)=\sigma'(k_1,\ldots,k_{n-2},i)\rho$ be the oblivious adversary that allocates steps to a single ReadMax operation by process p_0 fillowing the steps it allocates to other processes performing WriteMax operations. When the deterministic implementation D is performed starting from its initial configuration C_0 using the adversary $\sigma(k_1, \ldots, k_{n-2}, i)$, an instance of WriteMax(i) is completed (by the steps of β_i) and no instances of WriteMax(i') are invoked, for $0 < i < i' \leq n-1$, so p_0 must return i. If $i = 0$, then no instance of WriteMax are invoked, so p_0 must return 0.

For each choice of $k_1, \ldots, k_{n-2}$, the executions of ReadMax starting from C_0 using the adversaries $\sigma(k_1, \ldots, k_{n-2}, 0), \ldots, \sigma(k_1, \ldots, k_{n-2}, n-1)$ can be described by a decision tree with n leaves. If $V_R \leq d$, then the branching factor at each node of this tree is at most d. In this case, the decision tree has fewer than d^{2r} leaves at depth at most $2r$, so the average depth of the leaves in this tree is greater than $2r(n - d^{2r})/n$. Let $r = \log(n/2)/2\log d$. Then the average step complexity of ReadMax for an adversary chosen uniformly from $\{\sigma(k_1, \ldots, k_{n-2}, 0), \ldots, \sigma(k_1, \ldots, k_{n-2}, n-1)\})$ is greater than $2r(n - d^{2r})/n = r$. From Lemma 8.11, $\text{Prob}[V_R > d] \leq (1 - 1/w)^{d/2-1} \leq e^{-(d/2-1)/w}$. It follows that the average step complexity of ReadMax for an adversary chosen uniformly from $\{\sigma(k_1, \ldots, k_{n-2}, i) \mid k_1, \ldots, k_{n-2} \in \{0, \ldots, w-1\}, i \in \{0, \ldots, n-1\}\}$ is greater than

$$\text{Prob}[V_R > d] \cdot 0 + \text{Prob}[V_R \leq d] \cdot r \geq \left(1 - e^{-(d/2-1)/w}\right) \log(n/2)/2\log d.$$

Setting $d = 2 + 2w \ln n$ gives a lower bound of $(1 - 1/n)\log(n/2)/2log(2 + 2w\ln n) \in \Omega((\log n)/\log(w \log n))$.

Among all deterministic implementations such that WriteMax has worst case step complexity at most w, the implementation D was chosen arbitrarily. Thus, this is a lower bound on the distributional step complexity of ReadMax and, hence, by Yao's principle, on the worst case expected step complexity of ReadMax for all randomized implementations such that WriteMax has worst case step complextiy at most w. This holds against the set of adversaries

$$\{\sigma(k_1, \ldots, k_{n-2}, i) \mid k_1, \ldots, k_{n-2} \in \{0, \ldots, w-1\}, i \in \{0, \ldots, n-1\}\}$$

and, more generally, against the set of all oblivious adversaries.

Theorem 8.12. *For any randomized implementation of a max register that can be accessed by n processes and for which the worst case step complexity of WriteMax is at most w, the worst case expected step complexity of ReadMax against an oblivious adversary is in $\Omega((\log n)/\log(w \log n))$.*

When w is polylogarithmic in n, the worst case expected step complexity of ReadMax is in $\Omega(\log n/\log\log n)$. When w is polynomial in n, the worst case expected step complexity of ReadMax is in $\Omega(1)$.

The $\Omega((\log n)/\log(w \log n))$ lower bound on the worst case expected step complexity of ReadMax can be extended to randomized implementations of a max register for which the worst case expected step complexity of WriteMax is at most w, even when a ReadMax operation can return an incorrect result with low probability. This result is from the paper *Polylogarithmic concurrent data structures from monotone circuits*, by James Aspnes, Hagit Attiya and Keren Censor-Hillel, J.ACM, volume 59, number 1, 2012, pages 2:1-2:24.

This paper also contains a deterministic implementation of a max register in which the worst case step complexities of ReadMax and WriteMax are both in $O(\log n)$ when at most n operations are performed and a randomized implementation of a max register in which the worst case step complexity of WriteMax is in $O(n^3)$, the worst case step complexity of ReadMax is in $O(1)$, and ReadMax returns an incorrect result with probability $O(1/n)$.

8.8 Anonymous Conflict Detectors

In the m-valued conflict detector problem, each process p_i has an input value $v_i \in \{1, \ldots, m\}$ and each nonfaulty process must output a Boolean value b_i that satisfy the following properties:

- If all the inputs are the same, all the outputs are false.
- If $v_i \neq v_j$, then at least one of b_i and b_j is true.

Algorithms that solve this problem are components of many randomized consensus algorithms. (See, for example, the paper *A Modular Approach to Shared-Memory Consensus, with Applications to the Probabilistic-Write Model*, by James Aspnes, which appeared in PODC 2010, pages 460–467.)

The model we consider is asynchronous shared memory, where processes are anonymous and only communicate using registers.

Theorem 8.13. *Any deterministic algorithm that solves the m-valued conflict detector problem for n anonymous processes has $\Omega(\min(n, \log m/\log\log m))$ step complexity.*

Consider any deterministic algorithm for the problem. We will prove that it has a solo execution that contains $\Omega(\min(n, \log m/\log\log m))$ steps.

For each $v \in \{1, \ldots, m\}$ and each positive integer k, let $E_k(v)$ denote the solo execution by a process with input v, if it has length at most k, or the first k steps of that execution, if it does not terminate within k steps.

Suppose that there is a subset $V_k \subseteq \{1, \ldots, m\}$ of size at least 2 such that, for all $v, w \in V_k$, $v \neq w$, and all $1 \leq i \leq k$, if p reads register R in step i of $E_k(v)$ and does not write to R before step i of $E_k(v)$, then q does not write to R before step i of $E_k(w)$. We show that an adversary can construct an execution E' that is indistinguishable from $E_k(v)$ to a process p performing $E_k(v)$ and indistinguishable from $E_k(w)$ to another process q, performing $E_k(w)$ for any $v, w \in V_k$.

Starting with $E_k(v)$ performed by process p, we construct an execution $E^*_k(v)$ that also includes steps by at most $k-1$ clones of p. Specifically, $E^*_k(v)$ consists of a sequence of rounds in which p and some of its clones each takes one step. In round i, process p performs its i'th step. For each step of $E_k(v)$ in which p reads from a register, R, to which it has previously written, there is a clone of p which is scheduled to perform the same step as p in every round up to, but not including p's last write to R prior to the read. This clone delays its next step until immediately before the read and then takes no further steps. Note that the last step of this clone rewrites the value that is already in R, so $E_k(v)$ and $E^*_k(v)$ are indistinguishable to p.

Then it is possible to interleave $E^*_k(v)$ and $E^*_k(w)$ to create the desired execution E'. Specifically, E' consists of a sequence of rounds in which p, q and their clones each take at most one step. If p writes to register R in step i of $E_k(v)$, then p and all of its clones that take steps in round i of $E^*_k(v)$ write to R at the end of round i of E'. If p reads register R in step i of $E_k(v)$, but has written to R earlier, then there is a clone of p whose last step in $E^*_k(v)$ is a write to R immediately before step i by process p. In this case, this clone writes to R in the middle of round i of E', immediately followed by the reads of R by p and those of its clones that read R in round i of $E^*_k(v)$. Thus they all read the same value that they read in round i of $E^*_k(v)$. Finally, if p reads from register R in step i of $E_k(v)$ and does not write to R before step i of $E_k(v)$, then p and all of its clones that take steps in round i of $E^*_k(v)$ read R at the beginning of round i of E'. Note that, since $v, w \in V_k$, it follows that q and, hence, none of its clones, write to R before step i of $E_k(w)$. Therefore, p and its clones read the initial value of R in round i of E', just as they do in $E^*_k(v)$. Process q and its clones behave analogously to p and its clones.

If $k \leq n/2$, then there are at most n processes taking steps in E'. Hence execution E' can be constructed.

Note that $E_k(v)$ and E' are indistinguishable to p. Thus, if p returns during $E_k(v)$, it also returns false in E'. Similarly, if q returns during $E_k(w)$, it also returns false in E'. Since p and q cannot both return false in E', it follows that either p or q must perform more than k steps.

To prove our lower bound, it remains to prove the existence of the set V_k, for some $k \in \Omega(\log m / \log\log m)$. This is an easy consequence of the following result.

Lemma 8.14. *For all $k \geq 1$, there is a subset $V_k \subseteq \{1, \ldots, m\}$ of size at least $m/(e+2)^{k-1}(k-1)!$ such that, for all $v, w \in V_k$, $v \neq w$, and all $1 \leq i \leq k$, if p reads register R in step i of $E_k(v)$ and does not write to R before step i of $E_k(v)$, then q does not write to R before step i of $E_k(w)$.*

Proof. by induction on k. The claim is vacuously true for $V_1 = \{1, \ldots, m\}$. So, suppose the claim is true for some $k \geq 1$.

Construct a directed bipartite graph with vertex set $V_k \times U$, where $U = \cup\{U_k(v) \mid v \in V_k\}$ and $U_k(v)$ is the set of all registers written to in $E_k(v)$. For each $v \in V_k$, there is an edge from v to each register in $U_k(v)$ and, if step $k+1$ of $E_{k+1}(v)$ is the read of a register $R \in U - U_k(v)$, there is also an edge from R to v. Note that, in this graph, each $v \in V_k$ has outdegree at most k and indegree at most 1. Also, there are no cycles of length 2.

Randomly partition U into two parts, U' and U'', where each register $R \in U$ is in U' independently with probability $1/(k+1)$. Let $V' = \{v \in V_k \mid$ there is no edge from v to U' and no edge from U'' to $v\}$.

For each $v \in V_k$,

$$\Pr[v \in V'] \geq \frac{1}{k+1}\left(1 - \frac{1}{k+1}\right)^k = \frac{1}{k}\left(1 - \frac{1}{k+1}\right)^{k+1} > 1/(e+2)k,$$

so the expected size of V' is at least $|V_k|/(e+2)k \geq m/(e+2)^k k!$. Hence there exists a subset $V_{k+1} \subseteq V_k$ of size at least $m/(e+2)^k k!$ such that each vertex $v \in V_{k+1}$ has no edge to any vertex in U' and no edge from any vertex in U''.

Now suppose that $v, w \in V_{k+1} \subseteq V_k$ and $v \neq w$. Note that the first k steps of $E_k(v)$ and $E_{k+1}(v)$ are the same and the first k steps of $E_k(w)$ and $E_{k+1}(w)$ are the same. Hence, if p reads register R in step i of $E_{k+1}(v)$ and does not write to R before step i of $E_{k+1}(v)$, where $1 \leq i \leq k$, then q does not write to R before step i of $E_{k+1}(w)$. If p reads register R in step $k+1$ of $E_{k+1}(v)$ and does not write to R before step $k+1$ of $E_{k+1}(v)$, there is an edge from R to v in the graph. Hence, by definition of V_{k+1}, $R \notin U''$, so $R \in U'$ and there is no edge from w to R. This means that q does not write to R during $E_k(w)$ and, hence, before step $k+1$ of $E_{k+1}(w)$. Thus the claim is true for $k+1$. □

These proofs are due to Jim Aspnes, Faith Ellen, and Nati Linial. The m-valued conflict detector problem has asymptotically matching upper bounds. In their paper, *Tight Bounds for Anonymous Adopt-Commit Objects*, SPAA 2011, Jim Aspnes and Faith Ellen give one algorithm that solves this problem in $O(n)$ steps and another that solves it in $O(\log m / \log \log m)$ steps.