In Section 7.1, we know that in an asynchronous shared memory model where processes only communicate via registers, wait-free consensus is unsolvable. However, consensus is solvable with weaker termination conditions. There are randomized algorithms which terminate in finite expected time. There are deterministic obstruction-free algorithms, in which a process terminates if it is given sufficiently many consecutive steps. The termination condition we use in this section, nondeterministic solo termination, is weaker than both of these. It requires that, from every configuration and for all processes, $p$, there is a finite solo execution by $p$ in which $p$ terminates.

Even though we are considering a weaker termination condition, the outputs of all executions must still satisfy the agreement and validity properties. To show that an algorithm is faulty, we construct an execution that decides both 0 and 1. For example, suppose there is a reachable configuration $C$ in which all the registers are covered by a set of processes $P$ and there is a solo history $\alpha$ by a process $q \notin P$ that decides 1 and can be performed starting from $C$. Consider the execution illustrated in Figure 7.1, in which the processes in $P$ perform a block write $\beta$ and, from the resulting configuration, a process $p \in P$ performs a solo history $\gamma$ that decides 0. Then the execution of $\alpha\beta\gamma$ starting from $C$ decides both 0 and 1.

The proof does not talk about probabilities, nor does it put any restrictions on the adversary, since the lower bound is proved for the worst-case space complexity of the algorithm, namely, the space used in some execution, regardless of its probability.
stands in contrast to the lower bound on the total step complexity (proved in the Section 7.5), which shows that, with high probability, an execution takes many steps.

We use a combination of valency and covering arguments to prove an $\Omega(\sqrt{n})$ lower bound on the number of registers used by any consensus algorithm that satisfies nondeterministic solo termination. Specifically, we show that, if the number of processes is sufficiently large relative to the number of registers, then it is possible to construct an execution that decides both 0 and 1. Starting from any bivalent configuration, we either construct an execution that decides both 0 and 1, or prove that it is possible to reach a bivalent configuration in which more registers are covered. This gives an upper bound on the number of processes (as a function of the number of registers) in any correct algorithm satisfying nondeterministic solo termination, which implies a lower bound on the number of registers (as a function of the number of processes). In particular, this lower bound applies to randomized algorithms and deterministic obstruction-free algorithms.

### 7.4.1 Anonymous Processes

We begin by proving the lower bound in a system of anonymous processes. This means that all processes are identical and they run the same code. If two such processes are in the same state, they apply the same primitive to the same object when they are next allocated a step and, if the results are the same (for example, they read the same value or they get the same outcome from a coin flip), then they go to the same state. Initially, all processes with the same input value will be in the same state. Although this model is quite restrictive, it provides important insight for the lower bound in the general case.

A clone of a process $p$ in an execution is a process with the same input as $p$, which proceeds in lockstep with $p$, reading and writing the same values as $p$, until immediately before some write to a register. We only require that there exists an execution in which this process takes the same steps as process $p$, not that it takes the same steps as $p$ in every execution. An adversary can have the clone apply that write in some extension of the execution to ensure that the value $p$ reads from that register is the same as the value that $p$ last wrote there. After applying its delayed write, a clone takes no further
steps. Note that, until a clone does its delayed write, other processes, including \( p \), are unaware of its existence. In other words, the execution with this clone and the execution without it are indistinguishable to the other processes. Although these executions are not technically the same and they result in different configurations, we will ignore these distinctions.

Let \( r \) denote the number of registers. The following lemma shows that, if there is a bivalent configuration in which there are sufficiently many processes available to be used as clones, then it is possible to construct an execution that decides both 0 and 1. The situation is illustrated in Figure 7.2.

**Lemma 7.4.** Consider a reachable configuration \( C \) in which there is a set of processes \( P \) covering a set of registers \( V \), a disjoint set of processes \( Q \) covering a (not necessarily disjoint) set of registers \( W \), and at least \( r^2 - r + (|V| + |W| - |V|^2 - |W|^2)/2 \) processes that have taken no steps and are not in \( P \cup Q \). Suppose that after the block write by \( P \) there is a solo execution with history \( \alpha \) by a process \( p \in P \) in which \( p \) decides 0 and after the block write by \( Q \) there is a solo execution with history \( \beta \) by a process \( q \in Q \) in which \( q \) decides 1. Then there is an execution that decides both 0 and 1.

**Proof.** The proof is by induction on \((V, W)\), where pairs of sets are partially ordered by component-wise inclusion. Let \( \delta \) be the history of an execution starting from an initial configuration and ending in configuration \( C \), which exists since \( C \) is reachable.

![Figure 7.2. The situation in Lemma 7.4.](image-url)
7.4. Space Complexity of Consensus

The base case is when at least one of these two sets consists of all the registers. Suppose that \(|W| = r\). The argument when \(|V| = r\) is symmetric. Consider the execution, illustrated in Figure 7.3, consisting of \(\delta\), the block write to \(V\) by \(P\), \(\alpha\), the block write to \(W\) by \(Q\), and \(\beta\). To the processes in \(P\), this execution is indistinguishable from \(\delta\) followed by the block write to \(V\) by \(P\) and \(\alpha\), so \(p\) decides 0. To the processes in \(Q\), this execution is indistinguishable from \(\delta\) followed by the block write to \(W\) by \(Q\) and \(\beta\), so \(q\) decides 1.

Now consider any pair \((V, W)\) with \(|V|, |W| \neq r\) and suppose that the claim is true for all \((V', W') \neq (V, W)\) such that \(V \subseteq V'\) and \(W \subseteq W'\). There are two cases.

**Case 1.** \(V \subseteq W\) or \(W \subseteq V\).

Suppose that \(V \subseteq W\). The argument when \(W \subseteq V\) is symmetric. If all writes that occur during \(\alpha\) are to registers in \(W\), then, as in the base case, the execution in Figure 7.3 decides both 0 and 1.

Otherwise, let \(\alpha'\) be the longest prefix of \(\alpha\) that only contains writes to registers in \(W\), let \(R \not\in W\) be the register covered by \(p\) immediately after \(\alpha'\), and let \(\alpha''\) be the remainder of \(\alpha\), following the write by \(p\) to \(R\). Let \(C'\) be the configuration immediately after \(\alpha'\), except that there is a clone covering each register in \(V\), which was left behind when that register was last written to during \(\alpha'\) or during the block write to \(V\) by \(P\), if \(p\) doesn’t write to that register during \(\alpha'\). The block write to \(V\) by the set, \(P'\), of these clones starting from configuration \(C'\) does not change the values of any registers, so starting from the resulting configuration, \(p\) can perform a write to \(R\) followed by the solo history \(\alpha''\). When the processes in \(Q\) perform a block write starting from \(C'\) instead of from \(C\), the resulting configurations are indistinguishable to process \(q\), so the solo execution by \(q\) with history \(\beta\) still decides 1. Therefore, we have the situation depicted in Figure 7.4.

Let \(V' = V \cup \{R\}\), so \(|V'| = |V| + 1\). In \(C'\), the number of processes that have taken no steps and are not in \(P' \cup Q\) is at least

\[
r^2 - r + (|V| + |W| - |V|^2 - |W|^2)/2 - |V| = r^2 - r + (|V'| + |W| - |V'|^2 - |W|^2)/2.
\]
By the induction hypothesis for \((V', W)\), with \(C'\) instead of \(C\) and \(P' \cup \{p\}\) instead of \(P\), there is an execution that decides both 0 and 1.

**Case 2.** \(V \not\subseteq W\) and \(W \not\subseteq V\). Let \(U = V \cup W\). Then \(V, W \not\subseteq U\). Consider any terminating execution starting from \(C\) that begins with a block write to \(U\) and continues with a terminating solo execution of a history \(\gamma\) by any one of these processes that covered a register in \(U - (V \cup W)\). The existence of \(\gamma\) is guaranteed by nondeterministic solo termination. Without loss of generality, suppose that \(\gamma\) decides 0. Let \(C'\) be the configuration that is the same as \(C\), except there is a clone of a process in \(Q\) covering each register in \(W - V\) = \(U - V\), which was left behind when that register was last written to. Let \(P'\) consist of these clones plus the processes in \(P\). Then \(P'\) is disjoint from \(Q\). If the process performing \(\gamma\) is in \(P\), let \(p'\) be that process. Otherwise, let \(p'\) be its clone in \(P'\). This is illustrated in Figure 7.5.

In \(C'\), the number of processes that have taken no steps and are not in \(P' \cup Q\) is at least

\[
r^2 - r + (|V| + |W| - |V|^2 - |W|^2)/2 - |W - V|
\]

\[
= r^2 - r + (|U| - |W - V| + |W| - (|U| - |W - V|)^2 - |W|^2 - 2|W - V|)/2
\]

\[
= r^2 - r + (|U| + |W| - |U|^2 - |W|^2)/2 + |W - V| \cdot (|V| - |W - V|)/2 - 3/2
\]

\[
\geq r^2 - r + (|U| + |W| - |U|^2 - |W|^2)/2,
\]

since \(|V| \geq 1\) and \(|W - V| \geq 1\). By the induction hypothesis for \((U, W)\), with \(C'\) instead of \(C\) and \(P'\) instead of \(P\), there is an execution that decides both 0 and 1.

Using this lemma, we show that no consensus algorithm exists, if the number of anonymous processes is sufficiently large compared to the number of registers.

**Theorem 7.5.** There is no consensus algorithm using \(r\) registers for \(r^2 - r + 2\) or more processes that satisfies nondeterministic solo termination.
7.4. Space Complexity of Consensus

Proof. Suppose there is such an algorithm. Let \( C_0 \) be an initial configuration in which \( p \) has input 0 and \( q \) has input 1. Nondeterministic solo termination implies that there is a solo execution from \( C_0 \) by \( p \) that decides 0. Let \( \alpha \) be the history of that execution. Similarly, there is a solo history \( \beta \) by \( q \) that decides 1 and can be performed starting from \( C_0 \). If \( p \) doesn’t write during \( \alpha \), then the execution starting from \( C_0 \) with history \( \alpha \beta \) decides both 0 and 1. Similarly, if \( q \) doesn’t write during \( \beta \), then both 0 and 1 are decided in the execution starting from \( C_0 \) with history \( \beta \alpha \). So, suppose that \( p \) first writes to \( R \) and \( q \) first writes to \( R' \). Say that \( \alpha = \alpha' \alpha'' \), where \( \alpha' \) is the longest prefix of \( \alpha \) that contains no writes and \( \beta = \beta' \beta'' \), where \( \beta' \) is the longest prefix of \( \beta \) that contains no writes. Let \( P = \{ p \} \), \( Q = \{ q \} \), \( V = \{ R \} \), \( W = \{ R' \} \), and \( C = C_0 \alpha' \beta' \). In configuration \( C \), the number of processes that have taken no steps and are not in \( P \cup Q \) is at least \( r - \frac{|V| + |W| - r^2 + |W|^2}{2} \). Therefore, by Lemma 7.4, there is an execution that decides both 0 and 1. This contradicts the correctness of the algorithm. \( \square \)

7.4.2 The General Case

When processes are not anonymous, it is more difficult for the adversary to get multiple processes to cover the same register. Although the structure of the proof is the same as in the case of anonymous processes, there is more bookkeeping and combinatorics involved. The following notation is helpful. For any set of registers \( V \), let \( V' \) denote the set of registers not in \( V \). Then \( |V'| = r - |V| \), where \( r \) is the number of registers in the system.

The key to the lower bound is the following definition. It is the analogue of a block write followed by a terminating solo execution with added clones.

Definition 1. Let \( P \) be a set of processes and let \( V \) be a set of registers. An execution with history \( \alpha = \alpha_1 \alpha' \) starting from configuration \( C \) is interruptible for \( P \) and \( V \) if

(a) in \( C \), there are at least \( |V'| + 1 \) processes in \( P \) covering every register in \( V' \),

(b) \( \alpha_1 \) begins with a block write to \( V' \),

(c) all writes in \( \alpha_1 \) are to registers in \( V' \),

(d) all steps of \( \alpha \) are by processes in \( P \),

(e) some process in \( P \) decides during \( \alpha \), and

(f) either \( \alpha = \alpha_1 \) or there exist a set of processes \( P' \subseteq P \) and a set of registers \( V' \supseteq V \) such that the execution of \( \alpha' \) starting from configuration \( C \alpha_1 \) is interruptible for \( P' \) and \( V' \).

Note that, if an execution is interruptible for \( P \) and \( V \), it is also interruptible for \( P'' \) and \( V' \), for all \( P'' \supseteq P \). It may be helpful to consider the following equivalent, noninductive definition: An execution with history \( \alpha \) starting from configuration \( C \) is interruptible for a set of processes \( P \) and a set of registers \( V \) if \( \alpha \) can be divided into one or more pieces \( \alpha = \alpha_1 \cdots \alpha_k \) and there exist \( V = V_1 \subset V_2 \subset \cdots \subset V_k \) and \( P = P_1 \supseteq P_2 \supseteq \cdots \supseteq P_k \) such that, for \( i = 1, \ldots, k \),
(a) in configuration \( C\alpha_1 \cdots \alpha_{i-1} \) (or configuration \( C \), if \( i = 1 \)) there are at least \( |V_i| + 1 \) processes in \( P_i \) covering every register in \( V_i \),

(b) \( \alpha_i \) begins with a block write to a set of registers \( V_i \),

(c) all writes in \( \alpha_i \) are to registers in \( V_i \),

(d) all steps of \( \alpha_i \) are by processes in \( P_i \), and

(e) some process in \( P_k \) decides during \( \alpha = \alpha_1 \cdots \alpha_k \).

After each piece of an interruptible execution, there are more registers covered, but there may be fewer processes covering a particular register. This execution is interruptible in the sense that it is possible to insert certain steps by processes not in \( P \) between the pieces of \( \alpha \) so that the resulting execution is indistinguishable from the original execution to the processes in \( P \). Specifically, let \( \beta_1, \ldots, \beta_k, \beta_{k+1} \) be histories of executions by processes not in \( P \) such that for \( i = 1, \ldots, k \), all writes in \( \beta_i \) are to registers in \( V_i \). Then \( \alpha \sim P \beta_1 \alpha_1 \cdots \beta_k \alpha_k \beta_{k+1} \). These two executions are illustrated in Figure 7.6.

Figure 7.6. An interruptible execution starting from \( C \) (top) with steps by other processes inserted (bottom).

While it is straightforward to add clones to an execution when processes are anonymous, the existence of an interruptible execution for \( P \) and \( V \) starting from configuration \( C \) requires a careful proof, which depends on \( P \) being sufficiently large and, in configuration \( C \), each register in \( V \) being covered by sufficiently many processes in \( P \). We prove this in Lemma 7.6. Then, using this result, we prove Lemma 7.7, which is analogous to Lemma 7.4, and Theorem 7.8, which is analogous to Theorem 7.5.

A simplified version of Lemma 7.6 (with \( Y = \phi \)) can be used to show that, if \(|P| > (r^2 - r + |V| - |V|^2)/2 \) and, in configuration \( C \), there are at least \( |V| + 1 \) processes in \( P \) covering every register in \( V \), then there is an interruptible execution for \( P \) and \( V \) starting from \( C \). However, in the proof of the second case of Lemma 7.4, clones of processes in \( P \) that are covering registers in \( V \) are combined with clones of processes in \( Q \) covering
register in processes not in Proof. The proof is by induction on covering every register in Note that, if an execution reserves processes for Then \( R \) covering \( \alpha \) in configuration \( C_\alpha \). Let \( \hat{\alpha} \) be a sorted list of the numbers \( x_1 \leq x_2 \leq \cdots \leq x_{|V|} \) be a sorted list of the numbers \( x(R) \), for \( R \in V \). Then

\[
\sum_{i=1}^{|V|} x_i = \sum_{R \in V} x'(R) - |Y| \cdot |Y \cap V|
\]

\[
= |P - \hat{P}| - |Y| \cdot |Y \cap V|
\]

\[
= |P| - |V| \cdot |V| - |Y| \cdot |V \cap Y|
\]

\[
> (r^2 - r + |V| - |V|^2)/2 - |V| \cdot |V|
\]

\[
= |V|(|V| - 1)/2.
\]

If \( x_i \leq i - 1 \) for \( i = 1, \ldots, |V| \), then

\[
\sum_{i=1}^{|V|} x_i \leq \sum_{i=1}^{|V|} (i - 1) = |V|(|V| - 1)/2.
\]
Hence, there exists \( i \in \{1, \ldots, |V|\} \) such that \( x_i \geq i \). Since \( x_i |V| \geq \cdots \geq x_i \), there is a set \( S \subseteq V \) of \( |V| - i + 1 \) registers such that \( x(R) \geq i \) for all \( R \in S \). Let \( V' = V \cup S \), so \( |V'| = |V| + |S| = |V| + |V| - i + 1 = r - i + 1 \). Let \( P' \) be obtained from \( P \) by removing \( |Y| \) processes covering each register in \( Y \cap S \). Since only processes in \( P \) take steps in \( \alpha_1 \), there are at least \( |Y| \) processes not in \( P \) and, hence not in \( P' \), covering each register in \( Y \cap V \). Thus, there are at least \( |Y| \) processes not in \( P' \) covering each register in \( Y \cap V' \).

There are \( x'(R) - |Y| = x(R) \) processes in \( P' \) covering each register in \( Y \cap S \) and \( x'(R) = x(R) \) processes in \( P' \) covering each register in \( Y \cap S \). Thus, there are \( x(R) \geq i = |V'| + 1 \) processes in \( P' \) covering each register in \( S \). There are also \( |V| \geq |V'| + 1 \) processes in \( P \subseteq P' \) covering each register in \( V \). Hence, there are at least \( |V'| + 1 \) processes in \( P' \) covering each register in \( V' \).

Since \( |V'| > |V| \geq 1 \) and \( f(v) = v - v^2 \) is a non-increasing function of the non-negative integers, it follows that

\[
|P'| = |P| - |Y| \cdot |Y \cap S| \\
> |Y| \cdot |Y \cap V| + (r^2 - r - |V|^2 + |V|)/2 - |Y| \cdot |Y \cap S| \\
= |Y| \cdot |Y \cap V| + (r^2 - r - |V|^2 + |V|)/2 \\
\geq |Y| \cdot |Y \cap V| + (r^2 - r - |V'|^2 + |V'|)/2
\]

So, by the induction hypothesis, there is an interruptible execution for \( P' \) and \( V \) starting from \( C' \) that reserves processes for \( Y \). Let \( \alpha' \) be the history of that execution. Then \( \alpha_1 \alpha' \) is the history of an interruptible execution for \( P \) and \( V \) starting from \( C \) that reserves processes for \( Y \).

The situation in the following lemma is illustrated in Figure 7.7, which is very similar to Figure 7.2. The proof of this result is also very similar to the proof of Lemma 7.4.

**Lemma 7.7.** Let \( P \) and \( Q \) be disjoint sets of processes and let \( V \) and \( W \) be (not necessarily disjoint) sets of registers. Consider a configuration \( C \) from which there is an interruptible execution with history \( \alpha = \alpha_1 \alpha' \) for \( P \) and \( V \) that decides 0 and reserves processes for \( \bar{W} \) and an interruptible execution with history \( \beta \) for \( Q \) and \( W \) that decides 1 and reserves processes for \( \bar{V} \). If \( |P| \geq |W| \cdot |W \cap V| + (r^2 - r + |V| - |V'|^2)/2 \) and \( |Q| \geq |V| \cdot |V \cap W| + (r^2 - r + |W| - |W'|^2)/2 \), then there is an execution starting from \( C \) that decides both 0 and 1.

**Proof.** The proof is by induction on \((V, W)\), as in the proof of Lemma 7.4.

The base case is when at least one of these two sets consists of all the registers. If \(|W| = r\), then the execution with history \( \alpha \beta \) starting from \( C \) decides both 0 and 1. Likewise, if \(|V| = r\), the execution with history \( \beta \alpha \) starting from \( C \) decides both 0 and 1.

Now consider any pair \((V, W)\) with \(|V|, |W| \neq r\) and suppose that the claim is true for all \((V', W') \neq (V, W)\) such that \( V \subseteq V' \) and \( W \subseteq W' \). There are two cases.

**Case 1.** \( V \subseteq W \) or \( W \subseteq V \).

Suppose that \( V \subseteq W \). The argument when \( W \subseteq V \) is symmetric. If all \( \text{writes} \) that occur during \( \alpha \) are to registers in \( W \), then \( \alpha \beta \) starting from \( C \) decides both 0 and 1.
Otherwise, the execution with history $\alpha'$ starting from $C\alpha_1$ is interruptible for some $P' \subseteq P$ and $V' \supseteq V$ and reserves processes for $\overline{W}$. Since all writes in $\alpha_1$ are to registers in $V$ and $\beta$ begins with a block write to $W \supseteq V$, the executions with history $\beta$ starting from $C$ and $C\alpha_1$ are indistinguishable to processes in $Q$. Furthermore, $V' \subseteq V$. Therefore, the execution with history $\beta$ starting from $C\alpha_1$ is interruptible for $Q$ and $W$ and leaves behind processes for $\overline{V}$.

Since the execution with history $\alpha'$ starting from $C\alpha_1$ reserves processes for $\overline{W}$, in configuration $C\alpha_1$, there are at least $|\overline{W}|$ processes not in $P'$ covering each register in $\overline{W} \cap V'$. Let $P''$ be a maximum size set such that $P' \subseteq P'' \subseteq P$ and, in configuration $C\alpha_1$, there are at least $|\overline{W}|$ processes not in $P''$ covering each register in $\overline{W} \cap V'$. Then the execution with history $\alpha'$ starting from $C\alpha_1$ is also interruptible for $P''$ and $V'$ and reserves processes for $\overline{W}$. Furthermore, $P''$ and $Q$ are disjoint. Since only processes in $P$ take steps in $\alpha_1$, there are at least $|\overline{W}|$ processes not in $P$ covering each register in $\overline{W} \cap V$ in configuration $C\alpha_1$. Hence

$$|P''| \geq |P| - |\overline{W}| \cdot |\overline{W} \cap (V' - V)|$$
$$\geq |\overline{W}| \cdot |\overline{W} \cap V| + (r^2 - r + |V| - |V'|^2)/2 - |\overline{W}| \cdot |\overline{W} \cap (V' - V)|$$
$$= |\overline{W}| \cdot |\overline{W} \cap V| + (r^2 - r + |V| - |V'|^2)/2$$
$$\geq |\overline{W}| \cdot |\overline{W} \cap V| + (r^2 - r + |V'| - |V'|^2)/2.$$  

By the induction hypothesis applied to $(V', W)$, there is an execution starting from $C\alpha_1$ that decides both 0 and 1. Let $\gamma$ be the history of this execution. Then the execution with history $\alpha_1\gamma$ starting from $C$ decides both 0 and 1.

*Case 2.* $V \not\subseteq W$ and $W \not\subseteq V$. Let $U = V \cup W$. Then $V, W \not\subseteq U$, so $|V|, |W| \geq |U| + 1$. 

Figure 7.7. The situation in Lemma 7.7.
7. Valency Arguments

Consider configuration \( C \). Since the execution with history \( \alpha \) starting from \( C \) is interruptible for \( P \) and \( V \) and reserves processes for \( \overline{W} \), there are at least \( |\overline{V}|+1 \) processes in \( P \) covering each register in \( V \) and at least \( |\overline{W}| \) processes not in \( P \) covering each register in \( \overline{W} \cap V \). Let \( Q'' \) be a set consisting of \( |\overline{W}| \) processes not in \( P \) covering each register in \( \overline{W} \cap V = U - W \). Similarly, since \( \beta \) is interruptible for \( Q \) and \( W \) and reserves processes for \( \overline{V} \), there are at least \( |\overline{V}|+1 \) processes in \( Q \) covering each register in \( W \) and a set \( P'' \) consisting of \( |\overline{V}| \) processes not in \( Q \) covering each register in \( \overline{V} \cap W = U - V \). The processes in \( P'' \) and \( Q'' \) cover disjoint sets of registers, so \( P'' \cap Q'' = \phi \). Let \( P' = P \cup P'' \) and \( Q' = Q \cup Q'' \). Since \( P \cap Q = \phi \), \( P \cap Q'' = \phi \), and \( P'' \cap Q = \phi \), it follows that \( P' \) and \( Q' \) are disjoint.

There are at least \( |\overline{V}| \geq |\overline{U}|+1 \) processes in \( P' \) covering each register in \( V \cup (U-V) = U \) and there are at least \( |\overline{W}| \) processes not in \( P' \) covering each register in \( \overline{W} \cap V = \overline{W} \cap U \). Since \( |P'| \geq |P| \geq |\overline{W}| \cdot |\overline{W} \cap V| + (r^2 - r + |V| - |V|^2)/2 > |\overline{W}| \cdot |\overline{W} \cap U| + (r^2 - r + |U| - |U|^2)/2 \), Lemma 7.6 implies that there is an interruptible execution for \( P' \) and \( U \) starting from \( C \) that reserves processes for \( \overline{W} \). Let \( \alpha' \) be the history of this execution. Similarly, \( |Q'| > |\overline{V}| \cdot |\overline{V} \cap U| + (r^2 - r + |U| - |U|^2)/2 \) and there is an execution with history \( \beta' \) starting from \( C \) that is interruptible for \( Q' \) and \( U \) and reserves processes for \( \overline{V} \).

First suppose that \( \alpha' \) decides 0. Since \( U \subseteq \overline{V} \), the execution with history \( \beta' \) starting from \( C \) is interruptible for \( Q \) and \( W \) and reserves processes for \( \overline{U} \). Then, by the induction hypothesis, there is an execution starting from \( C \) that decides both 0 and 1.

Similarly, if \( \beta' \) decides 1, there is an execution starting from \( C \) that decides both 0 and 1.

Otherwise, \( \alpha' \) decides 1 and \( \beta' \) decides 0. Since \( \overline{U} \subseteq \overline{V} \), \( \overline{W} \), it follows that the execution with history \( \alpha' \) starting from \( C \) is interruptible for \( P' \) and \( U \) and reserves processes for \( \overline{U} \), the execution with history \( \beta' \) starting from \( C \) is interruptible for \( Q' \) and \( U \) and reserves processes for \( \overline{U} \), and \( |P'|, |Q'| > |\overline{U}| \cdot |\overline{U} \cap U| + (r^2 - r + |U| - |U|^2)/2 \). Hence, by the induction hypothesis, there is an execution starting from \( C \) that decides both 0 and 1. \( \square \)

Now we prove the main result, by showing that, if there are too many processes compared to the number of registers, then there is a execution in which both 0 and 1 are decided. Like the proof of Theorem 7.5, which is a simple application of Lemma 7.4, this result follows using a simple application of Lemma 7.7. However, it also needs Lemma 7.6 to set up the conditions for Lemma 7.7.

**Theorem 7.8.** There is no consensus algorithm using \( r \) registers for \( 3r^2 - r + 2 \) or more processes that satisfies nondeterministic solo termination.

**Proof.** Suppose there is such an algorithm. Divide the processes into two sets, \( P \) and \( Q \), with more than \( (3r^2 - r)/2 \) processes each. Let \( C_0 \) be an initial configuration in which each process in \( P \) has input 0 and each process in \( Q \) has input 1. Let \( V = W = \phi \). Then \( W \cap V = \overline{V} \cap W = \phi \) and \( |\overline{W}| \cdot |\overline{W} \cap V| + (r^2 - r + |V| - |V|^2)/2 = |\overline{V}| \cdot |\overline{V} \cap W| + (r^2 - r + |W| - |W|^2)/2 = (3r^2 - r)/2 \). Lemma 7.6 implies that, starting from \( C_0 \), there is an interruptible execution for \( P \) and \( V \) that reserves processes for \( \overline{W} \) and an interruptible
7.5. Expected Work for Randomized Consensus

execution for $Q$ and $W$ that reserves processes for $V$. Since all processes in $P$ have input 0, the former execution must decide 0. Similarly, the latter execution must decide 1. Then, by Lemma 7.7, there is an execution starting from $C_0$ that decides both 0 and 1. This contradicts the correctness of the algorithm.

These lower bounds, the use of clones, and the concept of nondeterministic solo termination first appeared in the paper *On the Space Complexity of Randomized Synchronization*, by Faith Ellen Fich, Maurice Herlihy, and Nir Shavit, Journal of the ACM, 1998 [37]. They also extended these lower bounds to solving consensus using historyless objects.

7.5 A Lower Bound on Expected Work for Randomized Consensus

In a randomized consensus algorithm, each process $p_i$ has an input value $x_i$ and should decide on an output value $y_i$ within a finite expected number of rounds. The outputs are required to satisfy both agreement and validity. In this section, we prove an $\Omega(n^2)$ lower bound on the expected work performed by any randomized consensus algorithm, that is, on the expected total number of steps taken by all processes.

Our model is the asynchronous message passing system, in which up to $f$ processes might fail by crashing. As in Section 7.3, we restrict attention to a class of almost synchronous executions. These executions proceed in rounds. A round starts with processes performing all the local computation (including the coin flips) they wish to perform. If no process has decided, at least $n - f$ processes take a step, each sending at most one message to every other process and then each receiving all the messages that have been delivered to it. If exactly $n - f$ processes take a step in a given round, then all of their messages are delivered. If more processes take a step in that round, then there is at most one process such that one or more of the messages it sent are not delivered. If a message sent by process $p$ to process $q$ in round $i$ is delivered to process $q$ in round $j \geq i$, then all messages sent by process $p$ to process $q$ in previous rounds are delivered, in order, at or before round $i$. If a process does not take a step in a given round, it does no local computation in the next round. An adversary determines which processes take steps each round and which of their messages are delivered. After one or more processes have decided, the adversary is fault-free: each process that has not yet decided takes a step each round and all messages are delivered. Once all processes have decided, nothing happens during a round.

The $\Omega(n^2)$ lower bound on expected work is achieved by fixing an arbitrary randomized consensus algorithm and showing that there is a high probability of getting an $n$-round execution in which no process has decided. We do not exploit the length of executions with more than $n$ rounds, nor steps that are taken by a process after some other process has decided. Therefore, the rest of this section considers only $k$-round executions, where $k \leq n$. A 0-round execution consists of an initial configuration.