A set $A \subseteq N$ is saturated (or extensional) if whenever it contains an index for a computable function, it contains all other indices for that same function. Formally, for all $x,y \in N$:

$x \in A \wedge \varphi x = \varphi y \rightarrow y \in A$

There are several practical ways to determine if a set is saturated:

1. Direct approach: Check if A can be expressed in the form:
   $A = \{x \mid \varphi x \in A\}$ where A is some set of computable functions
2. Property-based: If A represents a property of the input/output behavior of programs (rather than syntactic or computational properties), it is typically saturated. For example:
   - "Programs computing total functions"
   - "Programs computing the constant 0 function"
   - "Programs with infinite domain"
3. To prove a set is NOT saturated (several approaches):

   a) Using Second Recursion Theorem (most common approach):

   - Show there exists e such that $\varphi e = \varphi f(e)$ for some carefully chosen f
   - Use this to find indices e, e' where $\varphi e = \varphi e'$ but $e \in A$ and $e' \notin A$

   b) Direct construction:
   - Find two indices e, e' computing the same function
   - Show that one is in A while the other is not

For example, $K = \{x \mid x \in Wx\}$ is not saturated because:

- We can construct e such that $\varphi e(x) = 0$ if $x = e$, undefined otherwise
- Then find $e' \neq e$ such that $\varphi e = \varphi e'$
- $e \in K$ but $e' \notin K$ since $\varphi e'(e') \uparrow$

Properties related to runtime, program text, or computational steps are typically not saturated since they depend on the specific program rather than just its input/output behavior.

Remember that saturation is a key property used in Rice's theorem, which states that any non-trivial saturated set is undecidable. Therefore, being able to determine if a set is saturated is crucial for applying this important undecidability result.

The key intuition is that saturated sets care only about what functions programs compute, not how they compute them.