```
proc reconstruir (in s: señal, in prof: ℤ, in freq: ℤ, out result: señal)  {
        Pre {esSeñalAux(s, prof, freq) ∧ tieneAlMenos2MuestrasDistintasDeCero(s)}
        Post {
          |s| = |result|  ∧_L (
            enDondeNoSeaCeroDebenCoincidir(s, result) ∧
            enDondeEsCeroDebeSerElPromedioDeSusVecinosNoNulos(s, result))
        }
}

    pred tieneAlMenos2MuestrasDistintasDeCero (s: señal) {(|s| − #apariciones(s, 0)) ≥ 2}

    pred enDondeNoSeaCeroDebenCoincidir (original: señal, reconstruida: señal) {
      (∀i : ℤ) 0 ≤ i < |original|  ⟶_L ((original[i] ≠ 0) ⟶ (original[i] = reconstruida[i]))
    }

    pred enDondeEsCeroDebeSerElPromedioDeSusVecinosNoNulos (original: señal, reconstruida: señal) {
      (∀i : ℤ) 0 ≤ i < |original|  ⟶_L ((original[i] = 0) ⟶
        esUnPromedioDeSusVecinosNoNulosMasCercanos(original, reconstruida, i))
    }

    pred esUnPromedioDeSusVecinosNoNulosMasCercanos (original: señal, reconstruida: señal, i: ℤ) {
      (∃j, k : ℤ) 0 ≤ j, k < |original|  ∧ j ≠ k ∧_L
        (noSonNulas(original, j, k) ∧
          sonLasMuestrasMasCercanas(original, i, j, k) ∧
          reconstruida[i] = promedio(original[j], original[k]) )
    }

    pred noSonNulas (s: señal, j: ℤ, k: ℤ) {s[j] ≠ 0 ∧ s[k] ≠ 0}

    pred sonLasMuestrasMasCercanas (s: señal, i: ℤ, j: ℤ, k: ℤ) {
        (∀m : ℤ) 0 ≤ m < |s|  ∧ m ∉ [j, k] ⟶_L
          s[m] ≠ 0  ⟶ ( dist(i, m) > dist(i, j) ∧ dist(i, m) ≥ dist(i, k) )
    }

    fun dist (x: ℤ, y: ℤ) : ℤ = abs(x − y);

    fun promedio (a: ℤ, b: ℤ) : ℤ = (a + b) div 2;
```