```
assign : (sd : SD) → (sd' : SD) → (S ⇒ₛ Compl) sd
              → sd ≤ₛ sd' → R sd' → I sd'
assign ⟨ f , d ⟩ ⟨ f' , d' ⟩ β sd≤ₛsd' r with (≤-compare {suc d} {d'})
... | leq δ₁≤δ₂
        = assign-dec
            ((d' - (suc d)) δ₁≤δ₂) (-→≤ δ₁≤δ₂)
              (I-var ⟨ f , d ⟩
                (sd≤ₛsd'→sd≤ₛsd'-ₛ[d'-[suc-d]] sd≤ₛsd' δ₁≤δ₂))
              r
              (β ((sd≤ₛsd'→sd≤ₛsd'-ₛ[d'-[suc-d]] sd≤ₛsd' δ₁≤δ₂))
                (s-I (I-var ⟨ f , d ⟩
                  ((sd≤ₛsd'→sd≤ₛsd'-ₛ[d'-[suc-d]] sd≤ₛsd' δ₁≤δ₂)))))


... | geq δ₂≤δ₁ = assign-inc (((suc d) - d') δ₂≤δ₁)
                    (I-var ⟨ f , d ⟩ (≤ₛ-trans sd≤ₛsd' +ₛ→≤ₛ)) r
                    (β ((≤ₛ-trans sd≤ₛsd' +ₛ→≤ₛ))
                      (s-I (I-var ⟨ f , d ⟩ ((≤ₛ-trans sd≤ₛsd' +ₛ→≤ₛ)))))

use-temp : ∀ {sd sd'} → (S ⇒ₛ Compl) sd → sd ≤ₛ sd' → R sd' → I sd'
use-temp β sd≤ₛsd' (r-s s) = β sd≤ₛsd' s
use-temp {sd} {sd'} β sd≤ₛsd' (r-unary uop s) =
  assign sd sd' β sd≤ₛsd' (r-unary uop s)
use-temp {sd} {sd'} β sd≤ₛsd' (r-binary s₁ bop s₂) =
  assign sd sd' β sd≤ₛsd' (r-binary s₁ bop s₂)
```