```
-- Simultaneous substitution
exts : ∀{Γ Δ} → (∀{A} → A ∈ Γ → Δ ⊢ A)
               → (∀{A B} → B ∈ Γ , A → Δ , A ⊢ B)
exts σ Zero = Var Zero
exts σ (Suc x) = rename Suc (σ x)

subst : ∀{Γ Δ} → (∀{A} → A ∈ Γ → Δ ⊢ A)
               → (∀{A} → Γ ⊢ A → Δ ⊢ A)
subst σ (Var a) = σ a
subst σ (Sub a A≤:B) = Sub (subst σ a) A≤:B
subst σ (Lambda f) = Lambda (subst (exts σ) f)
subst σ (App f e) = App (subst σ f) (subst σ e)
subst σ Skip = Skip
subst σ (Seq c₁ c₂) = Seq (subst σ c₁) (subst σ c₂)
subst σ (NewVar c) = NewVar (subst (exts σ) c)
subst σ (Assign a e) = Assign (subst σ a) (subst σ e)
subst σ (Lit i) = Lit i
subst σ (Neg e) = Neg (subst σ e)
subst σ (Plus e₁ e₂) = Plus (subst σ e₁) (subst σ e₂)

-- Single substitution
_[_] : ∀{Γ A B} → Γ , B ⊢ A → Γ ⊢ B → Γ ⊢ A
_[_] {Γ} {A} {B} N M = subst {Γ , B} {Γ} σ {A} N
  where
  σ : ∀ {A} → A ∈ Γ , B → Γ ⊢ A
  σ Zero = M
  σ (Suc x) = Var x

-- Reduction
data _⟶_ : ∀{Γ A} → (Γ ⊢ A) → (Γ ⊢ A) → Set where
  App-cong₁ : ∀{Γ A B} {F F′ : Γ ⊢ A ⇒ B} {E : Γ ⊢ A}
                → F ⟶ F′ → App F E ⟶ App F′ E
  App-cong₂ : ∀{Γ A B} {V : Γ ⊢ A ⇒ B} {E E′ : Γ ⊢ A}
                → Value V → E ⟶ E′ → App V E ⟶ App V E′
  Lambda-β : ∀{Γ A B} {F : Γ , A ⊢ B} {V : Γ ⊢ A}
                → Value V → App (Lambda F) V ⟶ F [ V ]
```