Patterns

```
(define-macro (checked-app e1 e2)
 #:with (→ τ1 τ2) (typeof e1)
 #:with τ1-arg (typeof e2)
 #:when (τ=? τ1-arg τ1)
 #:with e1’ (erase e1)
 #:with e2’ (erase e2)
 (⊢ (#%app e1’ e2’) : τ2))
```

$$\frac{\Gamma \vdash e_1 \ : \ \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 \ : \ \tau_1}{\Gamma \vdash e_1\, e_2 \ : \ \tau_2} \ (\text{T-App})$$

Patterns

```
(define-macro (checked-app e1 e2)
 [⊢ [e1 ≫ e1’ ⇒ (~→ τ1 τ2)]]
 [⊢ [e2 ≫ e2’ ⇒ τ1-arg]]
 #:when (τ=? τ1-arg τ1)
 --------
 [⊢ [_ ≫ (#%app e1’ e2’) ⇒ τ2]])
```

$$\frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1\, e_2 : \tau_2}\ (\text{T-App})$$

Bidirectional-style type rules

```
(define-macro (checked-app e1 e2)
 [⊢ [e1 ≫ e1' ⇒ (~→ τ1 τ2)]]
 [⊢ [e2 ≫ e2' ⇐ τ1]]
 --------
 [⊢ [_ ≫ (#%app e1' e2') ⇒ τ2]])
```

$$\frac{\Gamma \vdash e_1 \ : \ \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 \ : \ \tau_1}{\Gamma \vdash e_1 \, e_2 \ : \ \tau_2} \ (\text{T-App})$$

#lang turnstile

```
(define-typed-syntax checked-app
 [(checked-app e1 e2) ≫
  [⊢ [e1 ≫ e1’ ⇒ (~→ τ1 τ2)]]
  [⊢ [e2 ≫ e2’ ⇐ τ1]]
  --------
  [⊢ [_ ≫ (#%app e1’ e2’) ⇒ τ2]]])
```

$$\frac{\Gamma \vdash e_1 \ : \ \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 \ : \ \tau_1}{\Gamma \vdash e_1 \, e_2 \ : \ \tau_2} \ (\text{T-App})$$

#lang turnstile

```
(define-typed-syntax checked-app
 [(checked-app e1 e2 ...) ≫
  [⊢ [e1 ≫ e1’ ⇒ (~→ τ1 ... τ2)]]
  [⊢ [e2 ≫ e2’ ⇐ τ1] ...]
  --------
  [⊢ [_ ≫ (#%app e1’ e2’ ...) ⇒ τ2]])
```

#lang turnstile

```
(define-typed-syntax checked-let
 [(checked-let ([x1 e1]) e2) ≫
  [⊢ [e1 ≫ e1’ ⇒ τ1]]
  [([x1 ≫ x1’ : τ1]) ⊢ [e2 ≫ e2’ ⇒ τ2]]
  --------
  [⊢ [_ ≫ (let ([x1’ e1’]) e2’) ⇒ τ2]])
```

Try it out!

```
raco pkg install turnstile
```