Proof-Relevant Resolution for Constructive Automation

František Farka

April 29, 2019

Heriot-Watt University and University of St Andrews

Automation for Programming Languages

Type inference, term synthesis, and type classes

```
-- type inference, term synthesis
data maybe_A : Bool → Set where
  nothing : maybe_A false
  just : A → maybe_A true

fromJust : maybe_A true → A
fromJust (just x) = x
```

Type inference, term synthesis, and type classes

```
-- type inference, term synthesis
data maybe_A : Bool → Set where
  nothing : maybe_A false
  just : A → maybe_A true

fromJust : maybe_A true → A
fromJust (just x) = x


fromJust = λ (m : maybe_A true) →
           elim_maybe_A ?_b m
             -- nothing
             (λ (w : ?_A )
               → ?_e )
             -- just x
             (λ (w : ?_B ) (x : A)
               → x)
```

Type inference, term synthesis, and type classes

```
-- type inference, term synthesis
data maybe_A : Bool → Set where
  nothing : maybe_A false
  just : A → maybe_A true

fromJust : maybe_A true → A
fromJust (just x) = x


fromJust = λ (m : maybe_A true) →
           elim_maybe_A true m
             -- nothing
             (λ (w : true ≡ false )
               → elim≡ w )
             -- just x
             (λ (w : true ≡ true ) (x : A)
               → x)
```

Type inference, term synthesis, and type classes

```
-- type inference, term synthesis
data maybe_A : Bool → Set where
  nothing : maybe_A false
  just : A → maybe_A true

fromJust : maybe_A true → A
fromJust (just x) = x


fromJust = λ (m : maybe_A true) →
           elim_maybe_A true m
             -- nothing
             (λ (w : true ≡ false )
               → elim≡ w )
             -- just x
             (λ (w : true ≡ true ) (x : A)
               → x)
```

```
-- type class resolution
class Eq a where
  eq : a → a → Bool

instance Eq Int where
  eq  x y = ...
instance (Eq a, Eq b) ⇒ Eq (Pair a b)
  where
  eq  (x1, x2) (y1, y2) =
    eq x1 y1 ∧ eq x2 y2

test : Eq (Pair Int Int) ⇒ Bool
test = eq (1, 2) (1, 3)
```

Type inference, term synthesis, and type classes

```
-- type inference, term synthesis
data maybe_A : Bool → Set where
  nothing : maybe_A false
  just : A → maybe_A true

fromJust : maybe_A true → A
fromJust (just x) = x


fromJust = λ (m : maybe_A true) →
           elim_maybe_A true m
             -- nothing
             (λ (w : true ≡ false )
               → elim≡ w )
             -- just x
             (λ (w : true ≡ true ) (x : A)
               → x)
```

```
-- type class resolution
class Eq a where
  eq : a → a → Bool

instance Eq Int where
  eq  x y = ...
instance (Eq a, Eq b) ⇒ Eq (Pair a b)
  where
  eq  (x1, x2) (y1, y2) =
    eq x1 y1 ∧ eq x2 y2

test : Eq (Pair Int Int) ⇒ Bool
test = eq (1, 2) (1, 3)


test : Bool
test = eq { ?D : Pair Int Int } (1, 2) (1, 3)
```

Type inference, term synthesis, and type classes

$\kappa_{\mathsf{Int}} : \mathsf{Eq\ Int}$

$\kappa_{\mathsf{Pair}} : \mathsf{Eq}\ a \wedge \mathsf{Eq}\ b \Rightarrow \mathsf{Eq\ (Pair}\ a\ b)$

$\mathsf{Eq\ (Pair\ Int\ Int)} \rightsquigarrow_{\kappa_{\mathbf{Pair}}}$
$\quad \mathsf{Eq\ Int} \wedge \mathsf{Eq\ Int} \rightsquigarrow_{\kappa_{\mathbf{Int}}}$
$\quad \mathsf{Eq\ Int} \rightsquigarrow_{\kappa_{\mathbf{Int}}}$
$\quad \emptyset$

-- $\kappa_{\mathtt{Pair}}\ \kappa_{\mathtt{Int}}\ \kappa_{\mathtt{Int}}$

$\mathsf{test} : \mathsf{Bool}$

$\mathsf{test} = \mathsf{eq}\ \{\ ?_{\mathsf{D}}\ : \mathsf{Pair\ Int\ Int}\ \}\ (1, 2)\ (1, 3)$

Type inference, term synthesis, and type classes

κ_{Int} : Eq Int

κ_{Pair} : Eq $a \wedge$ Eq $b \Rightarrow$ Eq (Pair a b)

Eq (Pair Int Int) $\rightsquigarrow_{\kappa_{\mathsf{Pair}}}$
  Eq Int $\wedge$ Eq Int $\rightsquigarrow_{\kappa_{\mathsf{Int}}}$
  Eq Int $\rightsquigarrow_{\kappa_{\mathsf{Int}}}$
  $\emptyset$

-- $\kappa_{\mathtt{Pair}}$ $\kappa_{\mathtt{Int}}$ $\kappa_{\mathtt{Int}}$

```
-- type class resolution
class Eq a where
  eq : a → a → Bool

instance Eq Int where
  eq  x y = ...
instance (Eq a, Eq b) ⇒ Eq (Pair a b)
  where
  eq  (x1, x2) (y1, y2) =
    eq x1 y1 ∧ eq x2 y2

test : Eq (Pair Int Int) ⇒ Bool
test = eq (1, 2) (1, 3)
```

```
test : Bool
test = eq { κPair κInt κInt } (1, 2) (1, 3)
```

Proof-Relevant Resolution

Big-step operational semantics

$$D := A \mid G \Rightarrow D \mid \forall x : A.D$$
$$G := A \mid D \Rightarrow G \mid \forall x : A.G \mid \exists x : A.G$$
$$e := \kappa \mid e\ e \mid \lambda\kappa.e \mid \langle M, e\rangle$$

$\boxed{\mathcal{S}; \mathcal{P} \longrightarrow e : G}$

$\boxed{\mathcal{S}; \mathcal{P} \xrightarrow{e':D} e : A}$

$$\frac{}{\mathcal{S}; \mathcal{P} \xrightarrow{e:A} e : A}\ \text{init}$$

$$\frac{\mathcal{S}; \mathcal{P} \longrightarrow e_1 : A_1 \qquad \mathcal{S}; \mathcal{P} \xrightarrow{e e_1 : D} e_2 : A_2}{\mathcal{S}; \mathcal{P} \xrightarrow{e:A_1 \Rightarrow D} e_2 : A_2}\ \Rightarrow\text{L}$$

$$\frac{\mathcal{S}; \mathcal{P} \xrightarrow{e:D[M/x]} e_2 : A_2 \qquad \mathcal{S}; \cdot \vdash M : A_1}{\mathcal{S}; \mathcal{P} \xrightarrow{e:\forall x:A_1.D} e_2 : A_2}\ \forall\text{L}$$

$$\frac{\mathcal{S}; \mathcal{P} \xrightarrow{\kappa:D} e : A \qquad \kappa : D \in \mathcal{P}}{\mathcal{S}; \mathcal{P} \longrightarrow e : A}\ \text{decide}$$

$$\frac{\mathcal{S}; \mathcal{P} \longrightarrow e : G[M/x] \qquad \mathcal{S}; \cdot \vdash M : A}{\mathcal{S}; \mathcal{P} \longrightarrow \langle M, e\rangle : \exists x : A.G}\ \exists\text{R}$$

$$\frac{\mathcal{S}; \mathcal{P}, \kappa : D \longrightarrow e : G}{\mathcal{S}; \mathcal{P}, \kappa : D \longrightarrow \lambda\kappa.e : D \Rightarrow G}\ \Rightarrow\text{R}$$

$$\frac{\mathcal{S}, c : A; \mathcal{P} \longrightarrow e : G[c/x]}{\mathcal{S}; \mathcal{P} \longrightarrow e : \forall x : A.G}\ \forall\text{R}$$

Small-step operational semantics

$$\hat{e} := \kappa \mid G \mid \hat{e}\,\hat{e} \mid \langle M, \hat{e} \rangle \mid \lambda\kappa.\hat{e}$$

$$C := \bullet \mid e\ C \mid \langle M, C \rangle \mid \lambda\kappa.C$$

$$\boxed{\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \overset{\hat{e}'' : D}{\rightsquigarrow} \Gamma' \mid \hat{e}'}$$

$$\frac{\mathcal{S}; \Gamma \vdash \sigma : \Gamma' \qquad \mathcal{S}; \Gamma' \vdash \sigma A \equiv \sigma A' : \mathsf{o}}{\mathcal{S}; \mathcal{P} \vdash \Gamma \mid C\{A\} \overset{\hat{e} : A'}{\rightsquigarrow} \Gamma' \mid (\sigma C)\{\hat{e}\}}$$

$$\frac{\mathcal{S}; \mathcal{P} \vdash \Gamma \mid C\{A\} \overset{\hat{e}_{\mathbf{1}}\, A_{\mathbf{1}} : D}{\rightsquigarrow} \Gamma' \mid \hat{e}}{\mathcal{S}; \mathcal{P} \vdash \Gamma \mid C\{A\} \overset{\hat{e}_{\mathbf{1}} : A_{\mathbf{1}} \Rightarrow D}{\rightsquigarrow} \Gamma' \mid \hat{e}}$$

$$\frac{\mathcal{S}; \mathcal{P} \vdash \Gamma, Y : A_1 \mid C\{A_2\} \overset{\hat{e}_{\mathbf{1}} : D[Y/x]}{\rightsquigarrow} \Gamma' \mid \hat{e}}{\mathcal{S}; \mathcal{P} \vdash \Gamma \mid C\{A_2\} \overset{\hat{e}_{\mathbf{1}} : \forall x : A_{\mathbf{1}}.D}{\rightsquigarrow} \Gamma' \mid \hat{e}}$$

$$\boxed{\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \rightsquigarrow \Gamma' \mid \hat{e}'}$$

$$\frac{\mathcal{S}; \mathcal{P} \vdash \Gamma \mid C\{A\} \overset{\kappa : D}{\rightsquigarrow} \Gamma' \mid \hat{e} \qquad \kappa : D \in \mathcal{P}}{\mathcal{S}; \mathcal{P} \vdash \Gamma \mid C\{A\} \rightsquigarrow \Gamma' \mid \hat{e}}$$

$$\frac{\mathcal{S}; \mathcal{P} \vdash \Gamma, Y : A \mid C\{\langle Y, G[Y/x] \rangle\} \rightsquigarrow \Gamma' \mid \hat{e}}{\mathcal{S}; \mathcal{P} \vdash \Gamma \mid C\{\exists x : A.G\} \rightsquigarrow \Gamma' : A \mid \hat{e}'}$$

$$\frac{\mathcal{S}; \mathcal{P}, \kappa : D \vdash \Gamma \mid C\{\lambda\kappa.G\} \rightsquigarrow \Gamma' \mid \hat{e}}{\mathcal{S}; \mathcal{P} \vdash \Gamma \mid C\{D \Rightarrow G\} \rightsquigarrow \Gamma' \mid \hat{e}}$$

$$\frac{\mathcal{S}; \mathcal{P} \vdash \Gamma, x : A \mid C\{G\} \rightsquigarrow \Gamma' \mid \hat{e}}{\mathcal{S}; \mathcal{P} \vdash \Gamma \mid C\{\forall x : A.G\} \rightsquigarrow \Gamma' \mid \hat{e}}$$

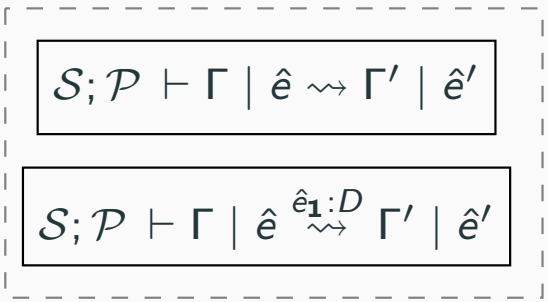
Soundness

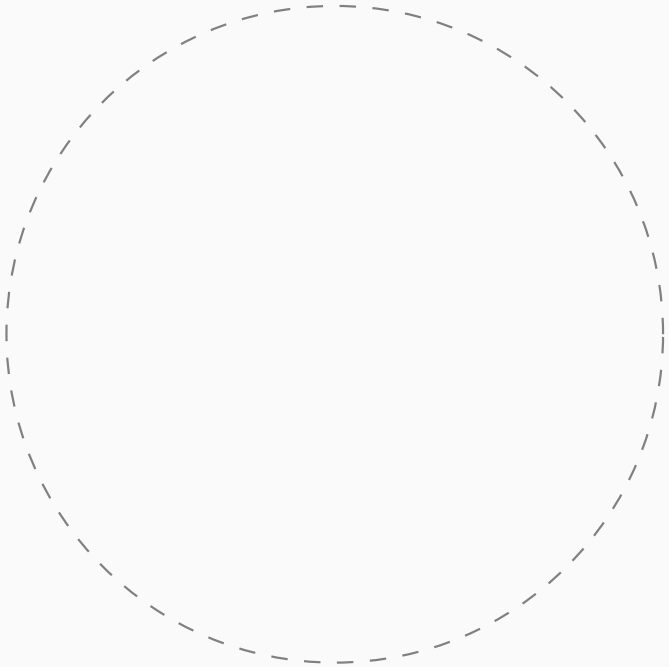
Theorem (Soundness)

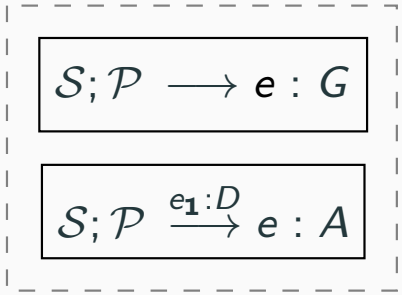
If $\mathcal{S}; \mathcal{P} \vdash \cdot \mid G \rightsquigarrow \cdot \mid e$ *then* $\mathcal{S}; \mathcal{P} \longrightarrow e : G$.

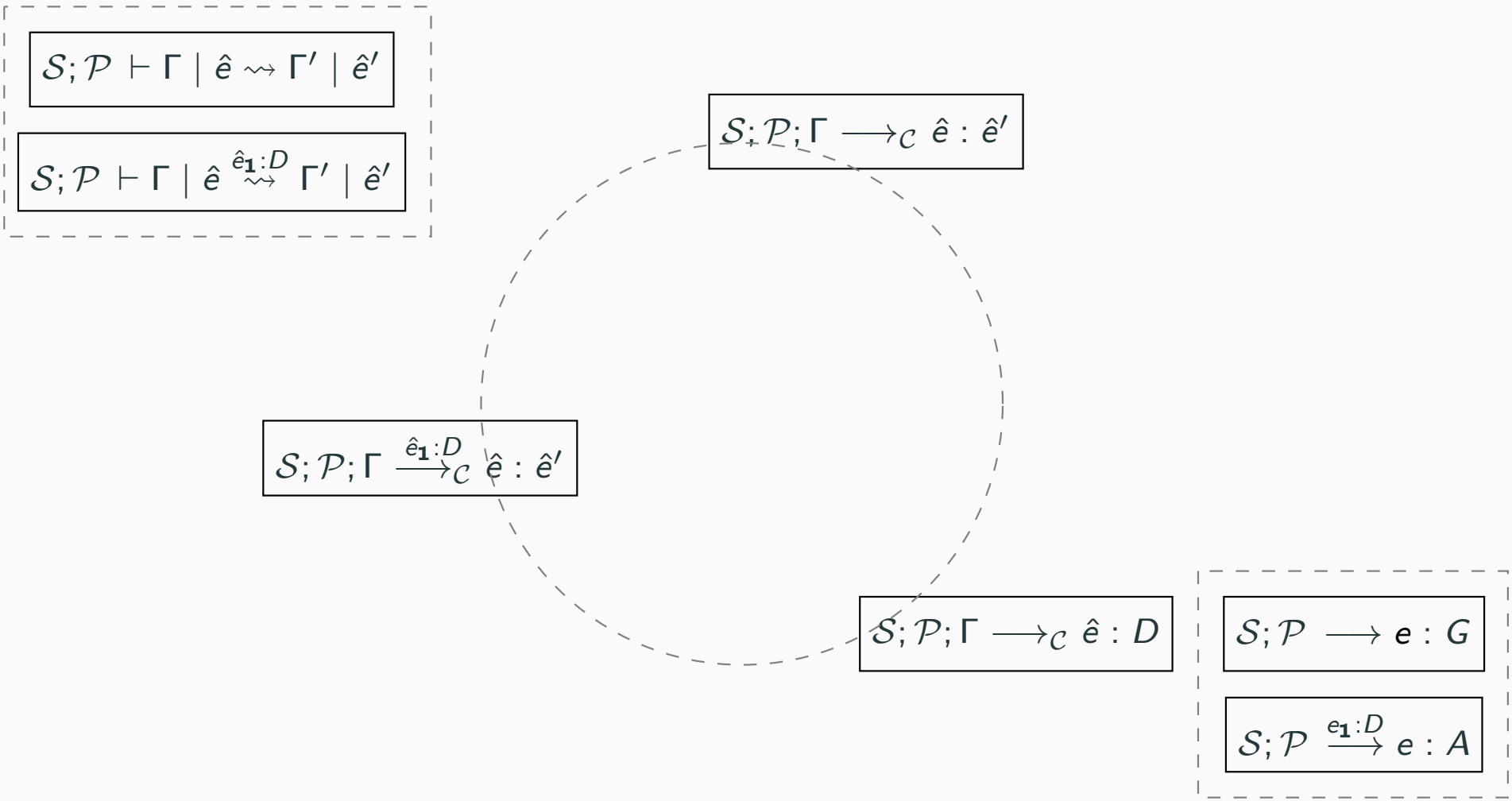
Theorem (Generalised soundness)

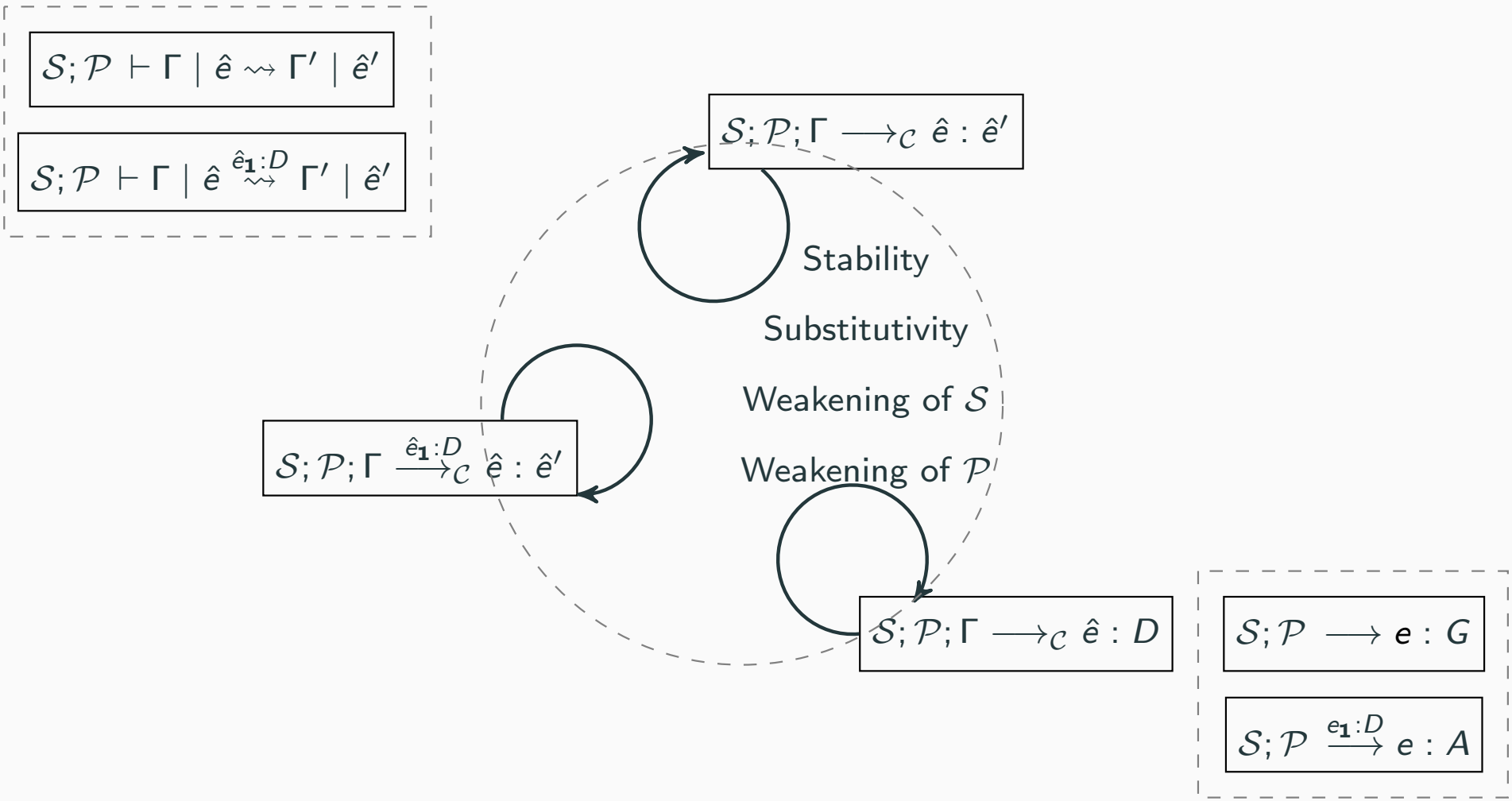
If $\mathcal{S}; \mathcal{P} \vdash \cdot \mid G \rightsquigarrow \Gamma' \mid e$ *then* $\mathcal{S}; \mathcal{P} \longrightarrow e : \forall \Gamma'.G$.

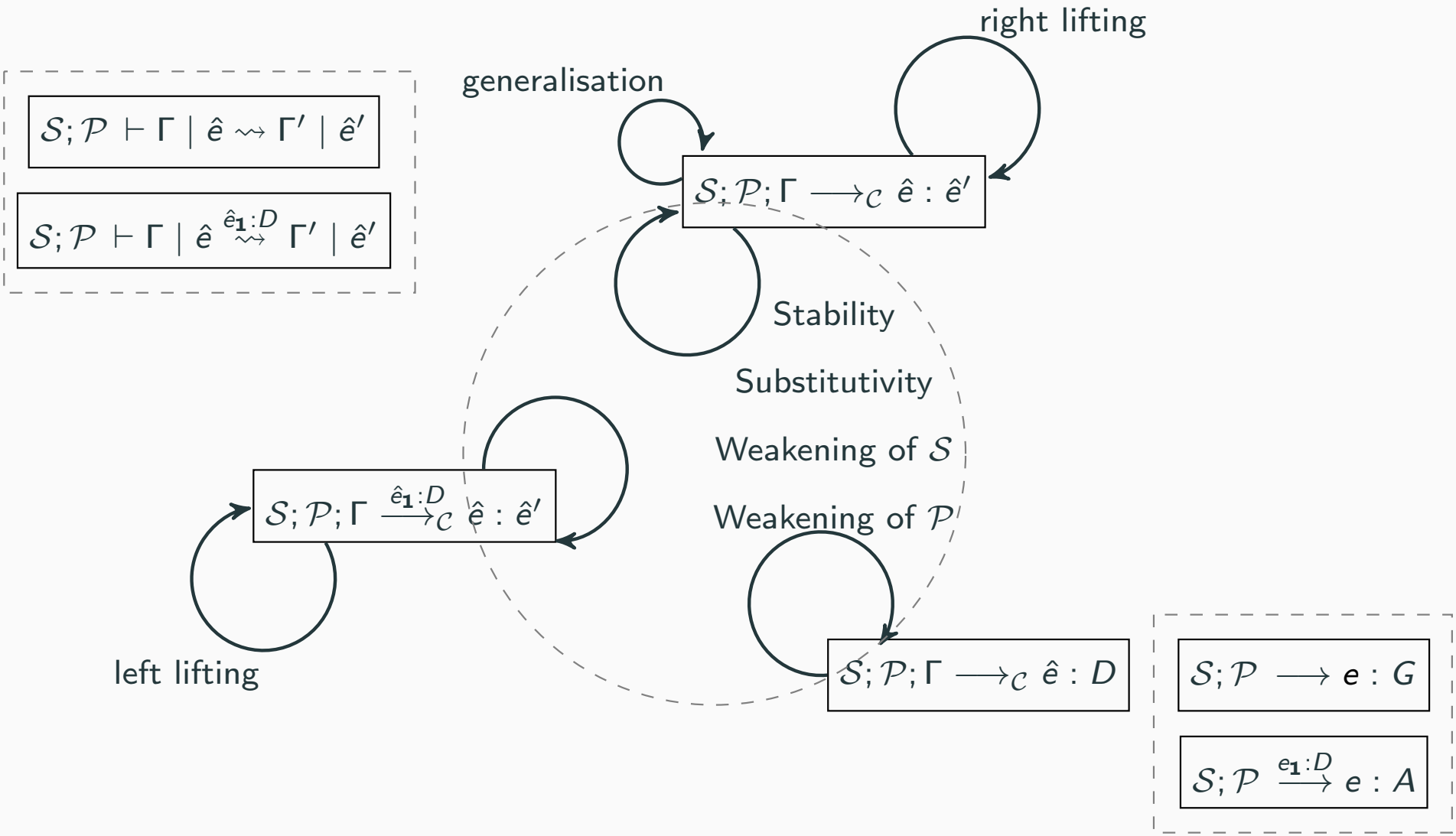

$\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \rightsquigarrow \Gamma' \mid \hat{e}'$
$\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \overset{\hat{e}_1 : D}{\rightsquigarrow} \Gamma' \mid \hat{e}'$

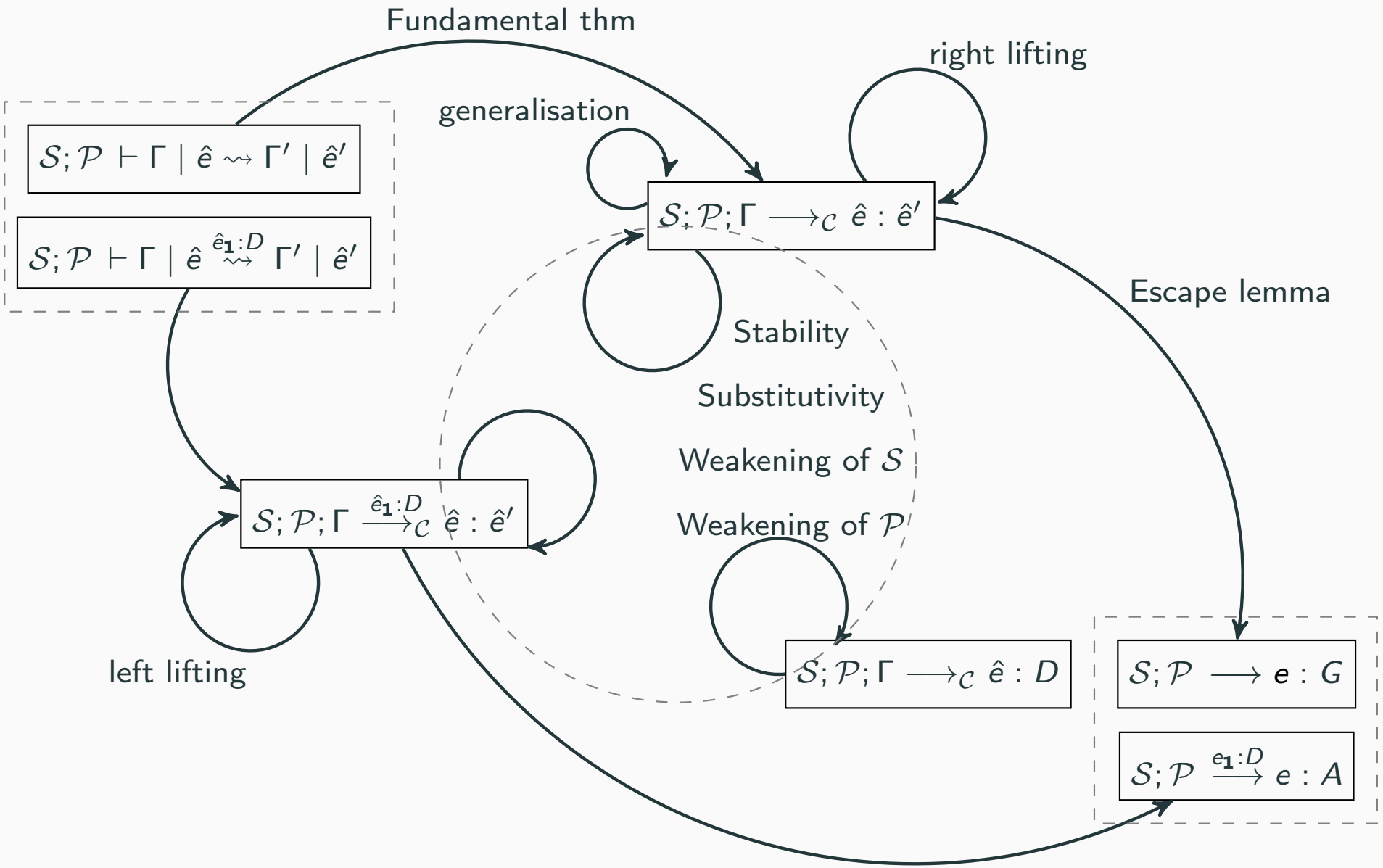


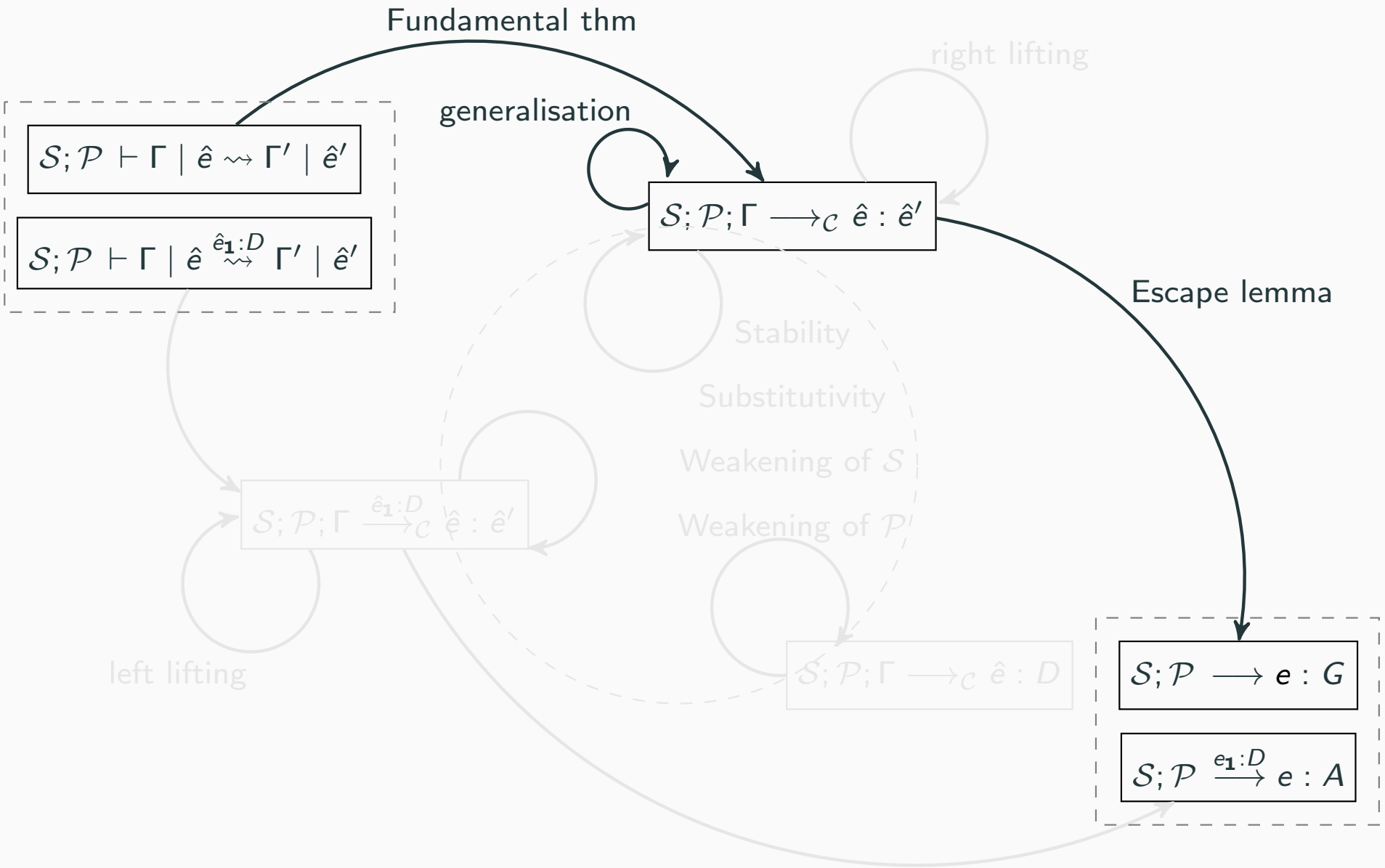

$\mathcal{S}; \mathcal{P} \longrightarrow e : G$
$\mathcal{S}; \mathcal{P} \xrightarrow{e_1 : D} e : A$


$\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \rightsquigarrow \Gamma' \mid \hat{e}'$
$\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \overset{\hat{e}_1 : D}{\rightsquigarrow} \Gamma' \mid \hat{e}'$
$\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : \hat{e}'$
$\mathcal{S}; \mathcal{P}; \Gamma \xrightarrow{\hat{e}_1 : D}_{\mathcal{C}} \hat{e} : \hat{e}'$
$\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : D$
$\mathcal{S}; \mathcal{P} \longrightarrow e : G$
$\mathcal{S}; \mathcal{P} \xrightarrow{e_1 : D} e : A$


$\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \rightsquigarrow \Gamma' \mid \hat{e}'$
$\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \overset{\hat{e}_1 : D}{\rightsquigarrow} \Gamma' \mid \hat{e}'$
$\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : \hat{e}'$
Stability
Substitutivity
Weakening of $\mathcal{S}$
Weakening of $\mathcal{P}$
$\mathcal{S}; \mathcal{P}; \Gamma \xrightarrow{\hat{e}_1 : D}_{\mathcal{C}} \hat{e} : \hat{e}'$
$\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : D$
$\mathcal{S}; \mathcal{P} \longrightarrow e : G$
$\mathcal{S}; \mathcal{P} \xrightarrow{e_1 : D} e : A$


$\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \rightsquigarrow \Gamma' \mid \hat{e}'$
$\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \overset{\hat{e}_1 : D}{\rightsquigarrow} \Gamma' \mid \hat{e}'$
generalisation
right lifting
$\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : \hat{e}'$
Stability
Substitutivity
Weakening of $\mathcal{S}$
Weakening of $\mathcal{P}$
$\mathcal{S}; \mathcal{P}; \Gamma \xrightarrow{\hat{e}_1 : D}_{\mathcal{C}} \hat{e} : \hat{e}'$
left lifting
$\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : D$
$\mathcal{S}; \mathcal{P} \longrightarrow e : G$
$\mathcal{S}; \mathcal{P} \xrightarrow{e_1 : D} e : A$

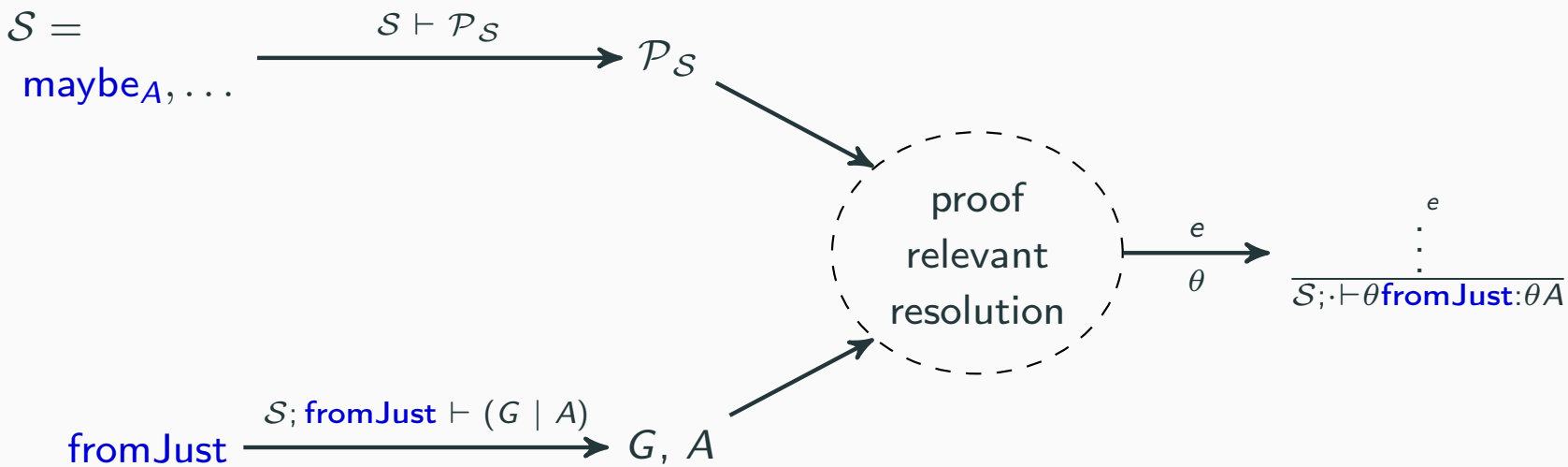

Fundamental thm
right lifting
generalisation
$\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \rightsquigarrow \Gamma' \mid \hat{e}'$
$\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \overset{\hat{e}_1 : D}{\rightsquigarrow} \Gamma' \mid \hat{e}'$
$\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : \hat{e}'$
Escape lemma
Stability
Substitutivity
Weakening of $\mathcal{S}$
Weakening of $\mathcal{P}$
$\mathcal{S}; \mathcal{P}; \Gamma \xrightarrow{\hat{e}_1 : D}_{\mathcal{C}} \hat{e} : \hat{e}'$
left lifting
$\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : D$
$\mathcal{S}; \mathcal{P} \longrightarrow e : G$
$\mathcal{S}; \mathcal{P} \xrightarrow{e_1 : D} e : A$


Fundamental thm
generalisation
right lifting
$\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \rightsquigarrow \Gamma' \mid \hat{e}'$
$\mathcal{S}; \mathcal{P} \vdash \Gamma \mid \hat{e} \overset{\hat{e}_1 : D}{\rightsquigarrow} \Gamma' \mid \hat{e}'$
$\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : \hat{e}'$
Escape lemma
Stability
Substitutivity
Weakening of $\mathcal{S}$
Weakening of $\mathcal{P}$
$\mathcal{S}; \mathcal{P}; \Gamma \xrightarrow{\hat{e}_1 : D}_{\mathcal{C}} \hat{e} : \hat{e}'$
left lifting
$\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : D$
$\mathcal{S}; \mathcal{P} \longrightarrow e : G$
$\mathcal{S}; \mathcal{P} \xrightarrow{e_1 : D} e : A$

Application to the Examples

Type inference and term synthesis[1]

$$\begin{aligned}\mathsf{fromJust} = \lambda\ (m : \mathsf{maybe}_A\ \mathsf{true}) \to\\ \mathsf{elim}_{\mathsf{maybe}_A}\ \mathsf{true}\ m\\ (\lambda\ (w : \mathsf{true} \equiv \mathsf{false}) \to \mathsf{elim}_{\equiv}\ w)\\ (\lambda\ (w : \mathsf{true} \equiv \mathsf{true})\ (x : A) \to x)\end{aligned}$$

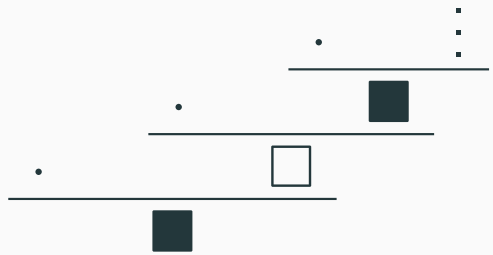


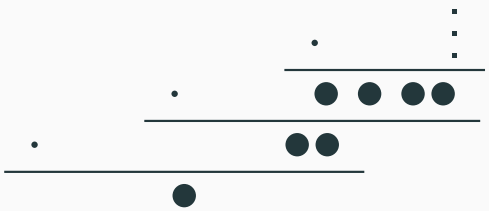
[1] https://github.com/frantisekfarka/slepice

Type class resolution[2]

```
data OddList a = OCons (EvenList a)
data EvenList a = Nil | ECons (OddList a)

instance (Eq a, Eq (OddList b)) ⇒ Eq (EvenList b)
instance (Eq a, Eq (EvenList b)) ⇒ Eq (OddList b)
```





```
data Bush a = Nil | Cons (Bush (Bush a))
instance (Eq a, Eq (Bush (Bush b))) ⇒ Eq (Bush b)
```

[2] https://github.com/frantisekfarka/cotcr

Where to Next

Future work

Applications

- coinductive proof-search for parallel and distibuted computation
- constrained Horn clauses for resource-aware computation
- automation for e.g. dependently type-based probabilistic programming

Theory of proof search

different classes of sequents for efficient search space

Type inference, term synthesis, and type classes

```
-- type inference, term synthesis
data maybe_A : Bool → Set where
  nothing : maybe_A false
  just    : A → maybe_A true

fromJust : maybe_A true → A
fromJust (just x) = x

fromJust = λ (m : maybe_A true) →
        elim_{maybe_A} ?_b m
          (λ (w : ?_A) → ?_e)
          (λ (w : ?_B) (x : A) → x)
```

```
-- type class resolution
class Eq a where
  eq : a → a → Bool

instance Eq Int where
  eq x y = ...
instance (Eq a, Eq b) ⇒ Eq (a,b)
  where
  eq (x1, x2) (y1, y2) =
    eq x1 y1 ∧ eq x2 y2

test : Eq ( Int , Int ) ⇒ Bool
test = eq (1, 2) (1, 3)

test : Bool
test = eq {?_EqD} (1, 2) (1, 3)
```


Big-step operational semantics

$$D := A \mid G \Rightarrow D \mid \forall x : A.D$$
$$G := A \mid D \Rightarrow G \mid \forall x : A.G \mid \exists x : A.G$$
$$e := \kappa \mid e\ e \mid \lambda\kappa.e \mid \langle M, e\rangle$$

$\boxed{\mathcal{S};\mathcal{P} \longrightarrow e : G}$

$\boxed{\mathcal{S};\mathcal{P} \xrightarrow{e':D} e : A}$

$$\frac{}{\mathcal{S};\mathcal{P} \xrightarrow{e:A} e : A}\ \text{init}$$

$$\frac{\mathcal{S};\mathcal{P} \longrightarrow e_1 : A_1 \qquad \mathcal{S};\mathcal{P} \xrightarrow{e_{\mathbf{1}}:D} e_2 : A_2}{\mathcal{S};\mathcal{P} \xrightarrow{e:A_1 \Rightarrow D} e_2 : A_2}\ \Rightarrow\text{L}$$

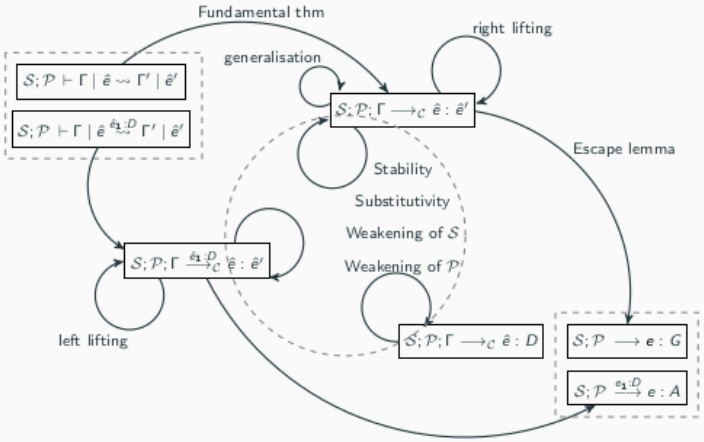
$$\frac{\mathcal{S};\mathcal{P} \xrightarrow{e:D[M/x]} e_2 : A_2 \qquad \mathcal{S};\cdot \vdash M : A_1}{\mathcal{S};\mathcal{P} \xrightarrow{e:\forall x:A_1.D} e_2 : A_2}\ \forall\text{L}$$

$$\frac{\mathcal{S};\mathcal{P} \xrightarrow{\kappa:D} e : A \qquad \kappa : D \in \mathcal{P}}{\mathcal{S};\mathcal{P} \longrightarrow e : A}\ \text{decide}$$

$$\frac{\mathcal{S};\mathcal{P} \longrightarrow e : G[M/x] \qquad \mathcal{S};\cdot \vdash M : A}{\mathcal{S};\mathcal{P} \longrightarrow \langle M, e\rangle : \exists x : A.G}\ \exists\text{R}$$

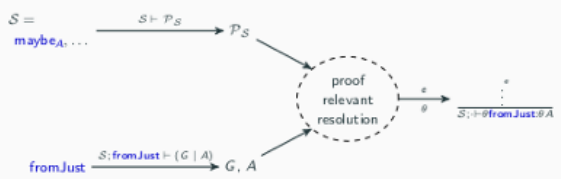
$$\frac{\mathcal{S};\mathcal{P},\kappa : D \longrightarrow e : G}{\mathcal{S};\mathcal{P} \longrightarrow \lambda\kappa.e : D \Rightarrow G}\ \Rightarrow\text{R}$$

$$\frac{\mathcal{S},c : A;\mathcal{P} \longrightarrow e : G[c/x]}{\mathcal{S};\mathcal{P} \longrightarrow e : \forall x : A.G}\ \forall\text{R}$$



Type inference and term synthesis[1]

```
fromJust = λ (m : maybe_A true) →
        elim_{maybe_A} true m
          (λ (w : true ≡ false) → elim_≡ w)
          (λ (w : true ≡ true) (x : A) → x)
```



[1] https://github.com/frantisekfarka/slepice

[2] https://github.com/frantisekfarka/cotcr

Appendix

References i

Type inference, term synthesis, and type classes

```
-- type inference, term synthesis
data maybe_A : Bool → Set where
  nothing : maybe_A false
  just : A → maybe_A true

fromJust : maybe_A true → A
fromJust (just x) = x
fromJust = λ (m : maybe_A true) →
          elim_maybe_A true m
            -- nothing
            (λ (w : true ≡ false )
              → elim≡ w )
            -- just x
            (λ (w : true ≡ true ) (x : A)
              → x)
```

```
-- type class resolution
class Eq a where
  eq : a → a → Bool

instance Eq Int where
  eq  x y = ...
instance (Eq a, Eq b) ⇒ Eq (Pair a b)
  where
  eq  (x1, x2) (y1, y2) =
    eq x1 y1 ∧ eq x2 y2

test : Eq (Pair Int Int) ⇒ Bool
test = eq (1, 2) (1, 3)



test : Bool
test = eq { κPair κInt κInt } (1, 2) (1, 3)
```

Type inference, term synthesis, and type classes

```
-- type inference, term synthesis
data maybe_A : Bool → Set where
  nothing : maybe_A false
  just : A → maybe_A true

fromJust : maybe_A true → A
fromJust (just x) = x
fromJust = λ (m : maybe_A true) →
          elim_maybe_A  ?_b  m
            -- nothing
            (λ (w :  ?_A )
              →  ?_e )
            -- just x
            (λ (w :  ?_B ) (x : A)
              → x)
```

$$\frac{\Gamma \vdash M : \Pi x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}\ \Pi\text{-t-Elim}$$

$$\mathtt{elim}_{\equiv_{\mathtt{bool}}} : (\mathtt{tt} \equiv_{\mathtt{bool}} \mathtt{ff}) \to A$$

```
-- program:

  κ_elim :  term ?_M ?_N ?_B ?_Γ ⇐
              term ?_M (Πx : ?_A.?_B') ?_Γ ∧
              term ?_N ?_A ?_Γ ∧ ?_B'[?_N/x] ≡ ?_B
κ_elim≡bool :  term elim≡bool (Πx : tt ≡bool ff . A) ?_Γ ⇐

-- goal:
-- elim_maybe_A  ?_b  m  (λ(w:?_A)→?_e)
```

$$((\mathtt{tt} \equiv_{\mathtt{bool}} \mathtt{ff}) = \ ?_A) \wedge G_{(\mathtt{elim}_{\mathtt{maybe_A}}\ \ ?_b\ \ m)} \wedge G_{\lambda(w:?_A).?_e)}$$

Type inference, term synthesis, and type classes

```
-- type inference, term synthesis
data maybe_A : Bool → Set where
  nothing : maybe_A false
  just : A → maybe_A true

fromJust : maybe_A true → A
fromJust (just x) = x
fromJust = λ (m : maybe_A true) →
           elim_maybe_A  true  m
             -- nothing
             (λ (w :  true ≡ false )
               →  elim≡ w )
             -- just x
             (λ (w :  true ≡ true ) (x : A)
               → x)
```

```
-- resolution trace:

term (?_M ?_N) A [m : maybe_A, w : tt ≡_bool ff]

  ⇝_κelim

  term ?_M (Πx : ?_A. A) [...] ∧ term ?_N ?_A
     [..., w : tt ≡_bool ff] ∧ A[?_N/x] ≡ ?_B

  ⇝_κelim≡bool

  term ?_N  tt ≡_bool ff [..., w : tt ≡_bool ff] ∧
     A[?_N/x] ≡ ?_B

  ⇝_κproj_w

  A[?_N/x] ≡ ?_B

  ⇝_κsubst_A  ⊥

-- proof-term:

  κelim κelim≡bool  κproj_w κsubst_A
```

Big-step operational semantics

Example

$$\begin{aligned}\mathcal{P} = \kappa_z &: odd(z),\\ \kappa_e &: \forall x : a.odd\ a \Rightarrow even\ (s\ x)\\ \kappa_o &: \forall x : a.even\ a \Rightarrow odd\ (s\ x)\end{aligned}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{S}, c : a; \mathcal{P}, \kappa_x : even\ c \longrightarrow \kappa_x : even\ c}{\mathcal{S}, c : a; \mathcal{P}, \kappa_x : even\ c \longrightarrow \kappa_o\ \kappa_x : odd\ (s\ c)}}{\mathcal{S}, c : a; \mathcal{P}, \kappa_x : even\ c \longrightarrow \kappa_e\ (\kappa_o\ \kappa_x) : even\ (s\ (s\ c))}}{\mathcal{S}, c : a; \mathcal{P} \longrightarrow \lambda\kappa_x.\kappa_e\ (\kappa_o\ \kappa_x) : even\ c \Rightarrow even\ (s\ (s\ c))}}{\mathcal{S}; \mathcal{P} \longrightarrow \lambda\kappa_x.\kappa_e(\kappa_o\kappa_x) : \forall x : a.even\ x \Rightarrow even\ (s\ (s\ x))}$$

Small-step operational semantics

Example

$$\cdot \mid \forall x : a.even\, x \Rightarrow even\,(s\,(s\,x)) \rightsquigarrow \cdot \mid even\, c \Rightarrow even\,(s\,(s\,c)) \rightsquigarrow$$

$$\cdot \mid \lambda\kappa_x.even\,(s\,(s\,c)) \rightsquigarrow \cdot \mid \lambda\kappa_x.even\,(s\,(s\,c))^{\kappa_e:\forall x:a.odd\, x \Rightarrow even\,(s\,x)} \rightsquigarrow$$

$$X : a \mid \lambda\kappa_x.even\,(s\,(s\,c))^{\kappa_e:odd\, X \Rightarrow even\,(s\,X)} \rightsquigarrow$$

$$X : a \mid \lambda\kappa_x.even\,(s\,(s\,c))^{\kappa_e(odd\, X):even\,(s\,X)} \rightsquigarrow$$

$$\cdot \mid \lambda\kappa_x.\kappa_e\,(odd\,(s\,c)) \rightsquigarrow \cdot \mid \lambda\kappa_x.\kappa_e\,(odd\,(s\,c))^{\kappa_o:\forall x:a.even\, x \Rightarrow odd\,(s\,x)} \rightsquigarrow$$

$$Y : a \mid \lambda\kappa_x.\kappa_e(odd\,(s\,c))^{\kappa_o:even\, Y \Rightarrow odd\,(s\,Y)} \rightsquigarrow \cdot \mid \lambda\kappa_x.\kappa_e\,(\kappa_o\,(even\, c)) \rightsquigarrow$$

$$\cdot \mid \lambda\kappa_x.\kappa_e\,(\kappa_o\,(even\, c)^{\kappa_x:even\, c}) \rightsquigarrow \cdot \mid \lambda\kappa_x.\kappa_e\,(\kappa_o\,\kappa_x)$$

Logical relation

$\boxed{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : \hat{e}'}$

$$\frac{\mathcal{S}; \mathcal{P}; \Gamma \xrightarrow{\hat{e}' : D}_{\mathcal{C}} \hat{e} : A \qquad \mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e}' : D}{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : A}$$

$$\frac{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : G[M/x] \qquad \mathcal{S}; \Gamma \vdash M : A}{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \langle M, \hat{e} \rangle : \exists x : A.G}$$

$$\frac{\mathcal{S}; \mathcal{P}, \kappa : D; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : G}{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \lambda\kappa.\hat{e} : D \Rightarrow G}$$

$$\frac{\mathcal{S}, c : A; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e}[c/x] : G[c/x]}{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : \forall x : A.G}$$

$$\frac{\mathcal{S} \vdash \mathcal{P} \qquad \mathcal{S} \vdash \Gamma}{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} A : A}$$

$$\frac{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e}_{\mathbf{1}} : \hat{e}_{\mathbf{2}}}{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} (\theta\hat{e})\, \hat{e}_{\mathbf{1}} : \hat{e}\, \hat{e}_{\mathbf{2}}}$$

$$\frac{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e}_{\mathbf{1}} : \hat{e}_{\mathbf{2}}}{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \langle \theta M, \hat{e}_{\mathbf{1}} \rangle : \langle M, \hat{e}_{\mathbf{2}} \rangle}$$

$$\frac{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e}_{\mathbf{1}} : \hat{e}_{\mathbf{2}}}{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \lambda\kappa.\hat{e}_{\mathbf{1}} : \lambda\kappa.\hat{e}_{\mathbf{2}}}$$

$\boxed{\mathcal{S}; \mathcal{P}; \Gamma \xrightarrow{\hat{e} : D}_{\mathcal{C}} \hat{e} : \hat{e}'}$

$$\frac{}{\mathcal{S}; \mathcal{P}; \Gamma \xrightarrow{\hat{e} : A}_{\mathcal{C}} \hat{e} : A}$$

$$\frac{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e}_{\mathbf{1}} : A_{\mathbf{1}} \qquad \mathcal{S}; \mathcal{P}; \Gamma \xrightarrow{\hat{e}\, \hat{e}_{\mathbf{1}} : D}_{\mathcal{C}} \hat{e}_{\mathbf{2}} : A_{\mathbf{2}}}{\mathcal{S}; \mathcal{P}; \Gamma \xrightarrow{\hat{e} : A_{\mathbf{1}} \Rightarrow D}_{\mathcal{C}} \hat{e}_{\mathbf{2}} : A_{\mathbf{2}}}$$

$$\frac{\mathcal{S}; \mathcal{P}; \Gamma \xrightarrow{\hat{e} : D[M/x]}_{\mathcal{C}} \hat{e}_{\mathbf{2}} : A_{\mathbf{2}} \qquad \mathcal{S}; \Gamma \vdash M : A_{\mathbf{1}}}{\mathcal{S}; \mathcal{P}; \Gamma \xrightarrow{\hat{e} : \forall x : A_{\mathbf{1}}.D}_{\mathcal{C}} \hat{e}_{\mathbf{2}} : A_{\mathbf{2}}}$$

$\boxed{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : D}$

$$\frac{\mathcal{S} \vdash \mathcal{P} \qquad \kappa : D \in \mathcal{P} \qquad \mathcal{S} \vdash \Gamma}{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \kappa : D}$$

$$\frac{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : A \Rightarrow D \qquad \mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e}' : A}{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e}\, \hat{e}' : D}$$

$$\frac{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : \forall x : A.D \qquad \mathcal{S}; \Gamma \vdash M : A}{\mathcal{S}; \mathcal{P}; \Gamma \longrightarrow_{\mathcal{C}} \hat{e} : D[M/x]}$$

Type class resolution - Pair

Example

$\mathcal{P}_{Pair} =$

$$\begin{array}{ll} \kappa_1 : \ \mathtt{eq}(x),\ \mathtt{eq}(y) & \Rightarrow \mathtt{eq}(\mathtt{pair}(x, y)) \\ \kappa_2 : & \Rightarrow \mathtt{eq}(\mathtt{int}) \end{array}$$

$$\dfrac{\dfrac{}{\mathcal{P}_{Pair} \longrightarrow \kappa_2 : \mathtt{eq}(\mathtt{int})}\ \text{Lp-m} \quad \dfrac{}{\mathcal{P}_{Pair} \longrightarrow \kappa_2 : \mathtt{eq}(\mathtt{int})}\ \text{Lp-m}}{\mathcal{P}_{Pair} \longrightarrow \kappa_1\kappa_2\kappa_2 : \mathtt{eq}(\mathtt{pair}(\mathtt{int}, \mathtt{int}))}\ \text{Lp-m}$$

Type class resolution - Bush

Example

$$\mathcal{P}_{Bush} =$$

$$\kappa_1 : \quad \Rightarrow \mathtt{eq}(\mathtt{int})$$

$$\kappa_2 : \ \mathtt{eq}(x),\ \mathtt{eq}(\mathtt{bush}(\mathtt{bush}(x))) \Rightarrow \mathtt{eq}(\mathtt{bush}(x))$$

$$\dfrac{\dfrac{}{\mathcal{P}_{Bush} \longrightarrow \kappa_1 : \mathtt{eq}(\mathtt{int})} \qquad \dfrac{\dfrac{\dfrac{\vdots}{\mathcal{P}_{Bush}, (\alpha : \mathtt{eq}(x) \Rightarrow \mathtt{eq}(\mathtt{bush}(x))), (\beta : \ \Rightarrow \mathtt{eq}(x)) \longrightarrow \kappa_2\beta(\alpha(\alpha\beta)) : \mathtt{eq}(\mathtt{bush}(x))}}{\mathcal{P}_{Bush}, (\alpha : \_) \longrightarrow \lambda\beta.\kappa_2\beta(\alpha(\alpha\beta)) : \mathtt{eq}(x) \Rightarrow \mathtt{eq}(\mathtt{bush}(x))}\ \text{Lam}}{\mathcal{P}_{Bush} \longrightarrow \nu\alpha.\lambda\beta.\kappa_2\beta(\alpha(\alpha\beta)) : \mathtt{eq}(x) \Rightarrow \mathtt{eq}(\mathtt{bush}(x))}\ \text{Nu}}{\mathcal{P}_{Bush} \longrightarrow (\nu\alpha.\lambda\beta.\kappa_2\beta(\alpha(\alpha\beta)))\kappa_1 : \mathtt{eq}(\mathtt{bush}(\mathtt{int}))}$$