Lecture: Backtranslation

Marco Patrignani
thanks to: Akram El-Korashy, Dominique Devriese, Daniel Patterson

1 Languages

1.1 Source

$$
\begin{aligned}
P &::= f(x) \mapsto e \\
e &::= \mathsf{true} \mid \mathsf{false} \mid e \odot e \mid n \mid x \mid e \oplus e \mid \mathsf{let}\ x = e\ \mathsf{in}\ e \mid \mathsf{if}\ e\ \mathsf{then}\ e\ \mathsf{else}\ e \mid \mathsf{fail} \\
v &::= \mathsf{true} \mid \mathsf{false} \mid n \\
O &::= \mathsf{let}\ y = \mathsf{call}\ f\ e\ \mathsf{in}\ e \\
f &::= n \mid \mathsf{fail} \\
E &::= [\cdot] \mid e \oplus E \mid E \oplus n \mid \mathsf{let}\ x = E\ \mathsf{in}\ e \mid \mathsf{if}\ E\ \mathsf{then}\ e\ \mathsf{else}\ e \mid e \odot E \mid E \odot n \\
\gamma &::= \varnothing \mid \gamma; [v \,/\, x]
\end{aligned}
$$

The program state is just an expression e.

1.1.1 Static Semantics (Typing)

$$
\begin{aligned}
\tau &::= \mathsf{Bool} \mid \mathsf{Nat} \mid \mathsf{Nat} \to \mathsf{Nat} \\
\Gamma &::= \varnothing \mid \Gamma; x : \tau
\end{aligned}
$$

$\Gamma \vdash e : \tau$ — Well-typed expression e of type τ

$\vdash f(x) \mapsto e : \mathsf{Nat} \to \mathsf{Nat}$ — Well-typed program e of type Nat to Nat

$\vdash \mathsf{let}\ y = \mathsf{call}\ f\ e_1\ \mathsf{in}\ e_2 : \mathsf{Bool}$ — Well-typed context of type Bool

$$\frac{}{\Gamma \vdash \mathsf{true} : \mathsf{Bool}}\ (\text{T-true}) \qquad \frac{}{\Gamma \vdash \mathsf{false} : \mathsf{Bool}}\ (\text{T-false}) \qquad \frac{}{\Gamma \vdash n : \mathsf{Nat}}\ (\text{T-n}) \qquad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ (\text{T-x})$$

$$\frac{\Gamma \vdash e_1 : \mathsf{Nat} \quad \Gamma \vdash e_2 : \mathsf{Nat}}{\Gamma \vdash e_1 \odot e_2 : \mathsf{Bool}}\ (\text{T-bop}) \qquad \frac{\Gamma \vdash e_1 : \mathsf{Nat} \quad \Gamma \vdash e_2 : \mathsf{Nat}}{\Gamma \vdash e_1 \oplus e_2 : \mathsf{Nat}}\ (\text{T-op}) \qquad \frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma; x : \tau_1 \vdash e_2 : \tau}{\Gamma \vdash \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : \tau}\ (\text{T-let})$$

$$\frac{\Gamma \vdash e : \mathsf{Bool} \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \mathsf{if}\ e\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 : \tau}\ (\text{T-if}) \qquad \frac{}{\Gamma \vdash \mathsf{fail} : \tau}\ (\text{T-fail})$$

$$\frac{x : \mathsf{Nat} \vdash e : \mathsf{Nat} \quad \mathsf{fail} \notin e}{\vdash f(x) \mapsto e : \mathsf{Nat} \to \mathsf{Nat}}\ (\text{T-prog}) \qquad \frac{\varnothing \vdash e_1 : \mathsf{Nat} \quad y : \mathsf{Nat} \vdash e_2 : \mathsf{Bool}}{\vdash \mathsf{let}\ y = \mathsf{call}\ f\ e_1\ \mathsf{in}\ e_2 : \mathsf{Bool}}\ (\text{T-ctx})$$

1.1.2 Dynamic Semantics

$$\text{Judgement: } \mathsf{e} \hookrightarrow \mathsf{e}'$$

$$\mathsf{let\ x = n\ in\ e} \hookrightarrow \mathsf{e[n\ /\ x]}$$

$$\mathsf{n} \oplus \mathsf{n}' \hookrightarrow \mathsf{n}'' \text{ where } n \oplus n' = n''$$

$$\mathsf{n} \odot \mathsf{n}' \hookrightarrow \mathsf{v} \text{ where } n \odot n' = v$$

$$\mathsf{if\ true\ then\ e\ else\ e}' \hookrightarrow \mathsf{e}$$

$$\mathsf{if\ false\ then\ e\ else\ e}' \hookrightarrow \mathsf{e}'$$

$$\mathsf{E}\,[\mathsf{e}] \hookrightarrow \mathsf{E}\,[\mathsf{e}'] \text{ if } \mathsf{e} \hookrightarrow \mathsf{e}'$$

$$\mathsf{E}\,[\mathsf{fail}] \hookrightarrow \mathsf{fail}$$

We can use evaluation contexts (which are not observers) to determine where the reductions happen.

1.2 Target

$$\begin{aligned}
\mathbf{P} &::= \mathbf{f(x)} \mapsto \mathbf{e} \\
\mathbf{v} &::= \mathbf{n} \mid \mathbf{true} \mid \mathbf{false} \qquad \mathbf{b} ::= \mathbf{true} \mid \mathbf{false} \\
\mathbf{e} &::= \mathbf{n} \mid \mathbf{x} \mid \mathbf{e} \oplus \mathbf{e} \mid \mathbf{let\ x = e\ in\ e} \mid \mathbf{if\ e\ then\ e\ else\ e} \mid \mathbf{fail} \\
&\quad \mid \mathbf{true} \mid \mathbf{false} \mid \mathbf{e} \odot \mathbf{e} \mid \mathbf{e\ has\ T} \\
\mathbf{T} &::= \mathbf{NAT} \mid \mathbf{BOOL} \\
\mathbf{O} &::= \mathbf{let\ y = call\ f\ e\ in\ e} \\
\mathbf{f} &::= \mathbf{n} \mid \mathbf{fail} \\
\mathbf{E} &::= [\cdot] \mid \mathbf{e} \oplus \mathbf{E} \mid \mathbf{E} \oplus \mathbf{v} \mid \mathbf{let\ x = E\ in\ e} \mid \mathbf{if\ E\ then\ e\ else\ e} \mid \mathbf{e} \odot \mathbf{E} \mid \mathbf{E} \odot \mathbf{v} \\
\gamma &::= \varnothing \mid \gamma; [\mathbf{v} \,/\, \mathbf{x}]
\end{aligned}$$

Semantics.

$$\text{Judgement: } \mathbf{e} \hookrightarrow \mathbf{e}'$$

$$\mathbf{let\ x = n\ in\ e} \hookrightarrow \mathbf{e[n\ /\ x]}$$

$$\mathbf{n} \oplus \mathbf{n}' \hookrightarrow \mathbf{n}'' \text{ where } n \oplus n' = n''$$

$$\mathbf{n} \odot \mathbf{n}' \hookrightarrow \mathbf{b} \text{ where } n \odot n' = b$$

$$\mathbf{if\ true\ then\ e\ else\ e}' \hookrightarrow \mathbf{e}$$

$$\mathbf{if\ false\ then\ e\ else\ e}' \hookrightarrow \mathbf{e}'$$

$$\mathbf{E}\,[\mathbf{e}] \hookrightarrow \mathbf{E}\,[\mathbf{e}'] \text{ if } \mathbf{e} \hookrightarrow \mathbf{e}'$$

$$\mathbf{E}\,[\mathbf{fail}] \hookrightarrow \mathbf{fail}$$

$$\mathbf{v} \oplus \mathbf{v}' \hookrightarrow \mathbf{fail} \text{ if } \mathbf{v} = \mathbf{b} \text{ or } \mathbf{v}' = \mathbf{b}$$

$$\mathbf{v} \odot \mathbf{v}' \hookrightarrow \mathbf{fail} \text{ if } \mathbf{v} = \mathbf{b} \text{ or } \mathbf{v}' = \mathbf{b}$$

$$\mathbf{if\ n\ then\ e\ else\ e}' \hookrightarrow \mathbf{fail}$$

1.3 Common

We assume some standard properties of substitutions γ, such as capture avoidance, distributivity over terms, and weakening.

In the following, we assume: $\oplus ::= +, -, \cdots$, $\odot ::= <, >, ==$.

Definition 1.1 (Contextual equivalence).

$$P_1 \simeq_{ctx} P_2 \stackrel{\text{def}}{=} \forall O.\ O\,[P_1] \hookrightarrow^* f \wedge O\,[P_2] \hookrightarrow^* f$$

Definition 1.2 (Plugging). Given that $O = \mathit{let}\ y = \mathit{call}\ f\ e_{arg}\ \mathit{in}\ e_{cont}$ and $P = f(x) \mapsto e_{fun}$.

$$O[P] \stackrel{\text{def}}{=} \mathit{let}\ y = \mathit{let}\ x = e_{arg}\ \mathit{in}\ e_{fun}\ \mathit{in}\ e_{cont}$$

Example 1.3 (Equivalent and inequivalent programs).

$\mathsf{x} + 2$	$\mathsf{x} + 1 + 1$
$\mathsf{if\ x} > 0\ \mathsf{then\ 0\ else\ 1}$	x
$\mathsf{let\ z} = \mathsf{x\ in\ z} + \mathsf{z}$	$\mathsf{x} + \mathsf{x}$

⊡

2 Compiler

$$\begin{aligned}
[\![\mathsf{f(x)} \mapsto \mathsf{e}]\!]^{\mathsf{S}}_{\mathbf{T}} &= \mathbf{f(x) \mapsto if\ x\ has\ NAT\ then}\ [\![\mathsf{e}]\!]^{\mathsf{S}}_{\mathbf{T}}\ \mathbf{else\ fail} \\
[\![\mathsf{true}]\!]^{\mathsf{S}}_{\mathbf{T}} &= \mathbf{true} \\
[\![\mathsf{false}]\!]^{\mathsf{S}}_{\mathbf{T}} &= \mathbf{false} \\
[\![\mathsf{n}]\!]^{\mathsf{S}}_{\mathbf{T}} &= \mathbf{n} \\
[\![\mathsf{x}]\!]^{\mathsf{S}}_{\mathbf{T}} &= \mathbf{x} \\
[\![\mathsf{e} \oplus \mathsf{e'}]\!]^{\mathsf{S}}_{\mathbf{T}} &= [\![\mathsf{e}]\!]^{\mathsf{S}}_{\mathbf{T}} \oplus [\![\mathsf{e'}]\!]^{\mathsf{S}}_{\mathbf{T}} \\
[\![\mathsf{let\ x} = \mathsf{e\ in\ e'}]\!]^{\mathsf{S}}_{\mathbf{T}} &= \mathbf{let\ x} = [\![\mathsf{e}]\!]^{\mathsf{S}}_{\mathbf{T}}\ \mathbf{in}\ [\![\mathsf{e'}]\!]^{\mathsf{S}}_{\mathbf{T}} \\
[\![\mathsf{if\ b\ then\ e\ else\ e'}]\!]^{\mathsf{S}}_{\mathbf{T}} &= \mathbf{if}\ [\![\mathsf{b}]\!]^{\mathsf{S}}_{\mathbf{T}}\ \mathbf{then}\ [\![\mathsf{e}]\!]^{\mathsf{S}}_{\mathbf{T}}\ \mathbf{else}\ [\![\mathsf{e'}]\!]^{\mathsf{S}}_{\mathbf{T}} \\
[\![\mathsf{fail}]\!]^{\mathsf{S}}_{\mathbf{T}} &= \mathbf{fail} \\
[\![\mathsf{e} \odot \mathsf{e'}]\!]^{\mathsf{S}}_{\mathbf{T}} &= [\![\mathsf{e}]\!]^{\mathsf{S}}_{\mathbf{T}} \odot [\![\mathsf{e'}]\!]^{\mathsf{S}}_{\mathbf{T}} \\
[\![[\mathsf{v}\ /\ \mathsf{x}]]\!]^{\mathsf{S}}_{\mathbf{T}} &= \left[[\![\mathsf{v}]\!]^{\mathsf{S}}_{\mathbf{T}} \Big/ [\![\mathsf{x}]\!]^{\mathsf{S}}_{\mathbf{T}} \right]
\end{aligned}$$

2.1 Compiler Correctness (Lemma 2.2)

We rely on a helper lemma for this result, as explained by Leroy.

Lemma 2.1 (Forward simulation).

$$\text{if } \mathsf{e}\gamma \hookrightarrow^* \mathsf{v} \text{ then } [\![\mathsf{e}]\!]^{\mathsf{S}}_{\mathbf{T}}[\![\gamma]\!]^{\mathsf{S}}_{\mathbf{T}} \hookrightarrow^* [\![\mathsf{v}]\!]^{\mathsf{S}}_{\mathbf{T}}$$

Lemma 2.2 (Expression correctness).

$$\text{if } [\![\mathsf{e}]\!]^{\mathsf{S}}_{\mathbf{T}}[\![\gamma]\!]^{\mathsf{S}}_{\mathbf{T}} \hookrightarrow^* [\![\mathsf{f}]\!]^{\mathsf{S}}_{\mathbf{T}} \text{ then } \mathsf{e}\gamma \hookrightarrow^* \mathsf{f}$$

Proof. By contradiction we assume: $\mathsf{e}\gamma \hookrightarrow \mathsf{v}' \neq \mathsf{v}$ (the case for b is analogous).

By Lemma 2.1 (Forward simulation) we get that $[\![\mathsf{e}]\!]^{\mathsf{S}}_{\mathbf{T}}[\![\gamma]\!]^{\mathsf{S}}_{\mathbf{T}} \hookrightarrow^* [\![\mathsf{v}']\!]^{\mathsf{S}}_{\mathbf{T}}$.

By determinism of the compiler we have $[\![\mathsf{v}]\!]^{\mathsf{S}}_{\mathbf{T}} \neq [\![\mathsf{v}']\!]^{\mathsf{S}}_{\mathbf{T}}$.

So we have that the same term $[\![\mathsf{e}]\!]^{\mathsf{S}}_{\mathbf{T}}$ reduces to two different terms, which contradicts the determinism of the semantics.

3 Compiler Full Abstraction

Theorem 3.1 (Full abstraction of $[\![\cdot]\!]^{\mathsf{S}}_{\mathbf{T}}$).

$$\forall \mathsf{P}_1, \mathsf{P}_2.\ \ \mathsf{P}_1 \simeq_{ctx} \mathsf{P}_2 \iff [\![\mathsf{P}_1]\!]^{\mathsf{S}}_{\mathbf{T}} \simeq_{ctx} [\![\mathsf{P}_2]\!]^{\mathsf{S}}_{\mathbf{T}}$$

Proof. The $\Leftarrow$ case:

$$[\![\mathsf{P}_1]\!]^{\mathsf{S}}_{\mathbf{T}} \simeq_{ctx} [\![\mathsf{P}_2]\!]^{\mathsf{S}}_{\mathbf{T}} \Leftarrow \mathsf{P}_1 \simeq_{ctx} \mathsf{P}_2$$

in contrapositive form:

$$\mathsf{P}_1 \not\simeq_{ctx} \mathsf{P}_2 \Rightarrow [\![\mathsf{P}_1]\!]^{\mathsf{S}}_{\mathbf{T}} \not\simeq_{ctx} [\![\mathsf{P}_2]\!]^{\mathsf{S}}_{\mathbf{T}}$$

and expand the definitions of $\simeq_{ctx}$:

$$\exists \mathsf{O}.\ \mathsf{O}[\mathsf{P}_1] \hookrightarrow^* \mathsf{f} \wedge \mathsf{O}[\mathsf{P}_2] \hookrightarrow^* \mathsf{f}' \Rightarrow \exists \mathbf{O}.\ \mathbf{O}\left[[\![\mathsf{P}_1]\!]^{\mathsf{S}}_{\mathbf{T}}\right] \hookrightarrow^* \mathbf{f} \wedge \mathbf{O}\left[[\![\mathsf{P}_2]\!]^{\mathsf{S}}_{\mathbf{T}}\right] \hookrightarrow^* \mathbf{f}'$$

Picking $\mathbf{O}$ is simple, assuming $[\![\cdot]\!]^{\mathsf{S}}_{\mathbf{T}}$ can be applied to context (as is generally the case, like here), $\mathbf{O}{=}[\![\mathsf{O}]\!]^{\mathsf{S}}_{\mathbf{T}}$. At this point, a clever usage of Lemma 2.1 (Forward simulation) gives this result.

The $\Rightarrow$ case:

$$\forall \mathsf{P}_1, \mathsf{P}_2.\ \ \mathsf{P}_1 \simeq_{ctx} \mathsf{P}_2 \Rightarrow [\![\mathsf{P}_1]\!]^{\mathsf{S}}_{\mathbf{T}} \simeq_{ctx} [\![\mathsf{P}_2]\!]^{\mathsf{S}}_{\mathbf{T}}$$

in contrapositive form:

$$\forall \mathsf{P}_1, \mathsf{P}_2.\ \ [\![\mathsf{P}_1]\!]^{\mathsf{S}}_{\mathbf{T}} \not\simeq_{ctx} [\![\mathsf{P}_2]\!]^{\mathsf{S}}_{\mathbf{T}} \Rightarrow \mathsf{P}_1 \not\simeq_{ctx} \mathsf{P}_2$$

and expand the definitions of $\simeq_{ctx}$:

$$\forall \mathsf{P}_1, \mathsf{P}_2.\ \ \exists \mathbf{O}.\ \mathbf{O}\left[[\![\mathsf{P}_1]\!]^{\mathsf{S}}_{\mathbf{T}}\right] \hookrightarrow^* \mathbf{f} \wedge \mathbf{O}\left[[\![\mathsf{P}_2]\!]^{\mathsf{S}}_{\mathbf{T}}\right] \hookrightarrow^* \mathbf{f}' \Rightarrow \exists \mathsf{O}.\ \mathsf{O}[\mathsf{P}_1] \hookrightarrow^* \mathsf{f} \wedge \mathsf{O}[\mathsf{P}_2] \hookrightarrow^* \mathsf{f}'$$

We can try to build O starting from $\mathbf{O}$ as we cannot rely on any correctness result.

4 Backtranslation of Contexts

4.1 The Universal Type

We need a universal type, something to backtranslate target expression to in order for them to be valid.

Example 4.1 (Backtranslation type). We cannot backtranslate $\mathbf{true}$ to true because when backtranslating $\mathbf{3 + true}$ we would get $3 + \mathsf{true}$ that is not a valid source expression according to the grammar of a. Also, we need a mechanism that scales for all operations quantified over by $\oplus$, e.g., $\mathbf{3 * 2}$ etc. ⊡

Anything that the target is backtranslated to, must be of this universal type.

This universal type is *natural numbers*.

4.2 Helper Functions

Then we need to convert to and from normal types and the universal type in order to ensure proper communication occurs. In fact, if we backtranslate $\mathbf{call\ f\ true}$ to $\mathsf{call\ f\ 0}$, the former will fail (by the typecheck inserted by the compiler) and the second will not.

Inject takes something of a type and injects it into the universal type, extract takes from the universal type and extracts to a type.

$$\begin{aligned}
\mathsf{inject_{NAT}(e)} &= \mathsf{e + 2} \\
\mathsf{inject_{BOOL}(e)} &= \mathsf{if\ e\ then\ 1\ else\ 0} \\
\mathsf{extract_{NAT}(e)} &= \mathsf{let\ x = e\ in\ if\ x \geq 2\ then\ x - 2\ else\ fail} \\
\mathsf{extract_{BOOL}(e)} &= \mathsf{let\ x = e\ in\ if\ x \geq 2\ then\ fail\ else\ if\ x - 1 \geq 1\ then\ false\ else\ true}
\end{aligned}$$

4.2.1 Properties of these Helpers

Lemma 4.2 (The Helpers are well-typed). The following holds:

- If $\Gamma \vdash \mathsf{e : Nat}$ then $\Gamma \vdash \mathsf{inject_{NAT}(e) : Nat}$
- If $\Gamma \vdash \mathsf{e : Bool}$ then $\Gamma \vdash \mathsf{inject_{BOOL}(e) : Nat}$
- If $\Gamma \vdash \mathsf{e : Nat}$ then $\Gamma \vdash \mathsf{extract_{NAT}(e) : Nat}$
- If $\Gamma \vdash \mathsf{e : Nat}$ then $\Gamma \vdash \mathsf{extract_{BOOL}(e) : Bool}$

4.3 The Backtranslation

The backtranslation is based on the observer structure.

$$\begin{aligned}
\langle\!\langle \mathbf{let\ y = call\ f\ e'\ in\ e''} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= \mathsf{let\ y = inject_{NAT}(call\ f\ (extract_{NAT}} \langle\!\langle \mathbf{e'} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} \mathsf{))\ in\ } \langle\!\langle \mathbf{e''} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} \\
\langle\!\langle \mathbf{n} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= \mathsf{n + 2}
\end{aligned}$$

$$
\begin{aligned}
\langle\!\langle \mathbf{x} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= \mathsf{x} \\
\langle\!\langle \mathbf{true} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= 0 \\
\langle\!\langle \mathbf{false} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= 1 \\
\langle\!\langle \mathbf{e} \oplus \mathbf{e'} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= \mathsf{let\ x1} = \mathsf{extract}_{\mathsf{NAT}} \langle\!\langle \mathbf{e} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{in} \\
&\qquad \mathsf{let\ x2} = \mathsf{extract}_{\mathsf{NAT}} \langle\!\langle \mathbf{e'} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{in} \\
&\qquad \mathsf{inject}_{\mathsf{NAT}} \mathsf{x1} \oplus \mathsf{x2} \\
\langle\!\langle \mathbf{e} \odot \mathbf{e'} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= \mathsf{let\ x1} = \mathsf{extract}_{\mathsf{NAT}} \langle\!\langle \mathbf{e} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{in} \\
&\qquad \mathsf{let\ x2} = \mathsf{extract}_{\mathsf{NAT}} \langle\!\langle \mathbf{e'} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{in} \\
&\qquad \mathsf{inject}_{\mathsf{BOOL}} \mathsf{x1} \oplus \mathsf{x2} \\
\langle\!\langle \mathbf{let\ x = e\ in\ e'} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= \mathsf{let\ x} = \langle\!\langle \mathbf{e} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{in}\ \langle\!\langle \mathbf{e'} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} \\
\langle\!\langle \mathbf{if\ e\ then\ e'\ else\ e''} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= \mathsf{let\ x} = \mathsf{extract}_{\mathsf{BOOL}} \langle\!\langle \mathbf{e} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{in\ if\ x\ then}\ \langle\!\langle \mathbf{e'} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{else}\ \langle\!\langle \mathbf{e''} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} \\
\langle\!\langle \mathbf{e\ has\ T} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= \begin{cases} \mathsf{let\ x} = \langle\!\langle \mathbf{e} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{in\ if\ x} \geq 2\ \mathsf{then\ 1\ else\ 0} \\ \quad \mathsf{if}\ \mathbf{T} = \mathbf{NAT} \\ \mathsf{let\ x} = \langle\!\langle \mathbf{e} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{in\ if\ x} \geq 2\ \mathsf{then\ 0\ else\ 1} \\ \quad \mathsf{if}\ \mathbf{T} = \mathbf{BOOL} \end{cases} \\
\langle\!\langle \mathbf{fail} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= \mathsf{fail} \\
\langle\!\langle [\mathbf{v}\ /\ \mathbf{x}] \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= \left[\langle\!\langle \mathbf{v} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} \ \middle/\ \langle\!\langle \mathbf{x} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} \right]
\end{aligned}
$$

The case for the $\mathbf{e\ has\ T}$ from should be $\mathsf{inject}_{\mathsf{BOOL}}(\mathsf{if\ x} \geq 2\ \mathsf{then\ true\ else\ false})$ (in the case for BOOL, swap true and false for the NAT case) but we shorten it to the definition above because we already know how the "if" expression and the subsequent inject will execute.

Remark 4.3 (Letin). The backtranslation of $\mathbf{let\ x = e\ in\ e'}$ may seem confusing, as it does not insert $\mathsf{inject}_{\mathsf{NAT}}$ for its subexpressions.

We argue why it is right using this example, for which we indicate the reductions:

- $\mathbf{t_1 = let\ x = 2\ in\ x + 1}$ and $\mathbf{t_1 \hookrightarrow^* 3}$;

Currently, what we get is:

- $\mathsf{t_1 = let\ x = 4\ in\ let\ x1 = extract_{NAT}\ x\ in\ let\ x2 = extract_{NAT}\ 3\ in\ inject_{NAT}\ x1 + x2}$

If we unfold the reductions, we see that

$$
\begin{array}{rl}
 & \mathsf{let\ x = 4\ in\ let\ x1 = extract_{NAT}\ x\ in\ let\ x2 = extract_{NAT}\ 3\ in\ inject_{NAT}\ x1 + x2} \\
\hookrightarrow & \mathsf{let\ x1 = extract_{NAT}\ 4\ in\ let\ x2 = extract_{NAT}\ 3\ in\ inject_{NAT}\ x1 + x2} \\
\mathsf{t_1} \hookrightarrow^* & \mathsf{let\ x2 = extract_{NAT}\ 3\ in\ inject_{NAT}\ 2 + x2} \\
\hookrightarrow^* & \mathsf{inject_{NAT}\ 2 + 1} \\
\hookrightarrow^* & \mathsf{5}
\end{array}
$$

and these reductions proceed as expected.

However if we insert an additional $\mathsf{extract}_{\mathsf{NAT}}$ for the value bound to the x, these reductions will not go right, as we get an additional $+2$. We could eliminate it by adding an $\mathsf{inject}_{\mathsf{NAT}}$ when variables are backtranslated, but this is hard to do correctly as we do not know if a variable will be used as a Nat or as a Bool in the target, as in this other valid expression:

- $\mathbf{t_2 = let\ x = true\ in\ if\ x\ then\ 3\ else\ 0}$

This gets backtranslated to

- $\mathsf{t_2 = let\ x = 0\ in\ let\ z = extract_{BOOL} x\ in\ if\ z == 1\ then\ 5\ else\ 2}$

These expressions reduce correctly, but we wouldn't know how to carry the information that $\mathbf{x}$ is technically a Boolean (the $\mathbf{true}$ expression may be a lot more complex that that and provide no help).

However, we know that *when a variable is going to be used*, e.g., inside a $\oplus$ expression, the $\mathsf{extract.}$ will be there.

4.3.1 Properties of the Backtranslation

In order to use the context backtranslation, we need to prove that it is correct:

Lemma 4.4 (Backtranslation correctness).

$$\text{if } \mathbf{e}\gamma \hookrightarrow^* \mathbf{f}$$
$$\text{then } \langle\!\langle \mathbf{e} \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}} \langle\!\langle \gamma \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}} \hookrightarrow^* \langle\!\langle \mathbf{f} \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}}$$

5 Using the Context Backtranslation

We resume our proof for the $\Rightarrow$ direction of fully abstract compilation.

What we have is:

- $\exists \mathbf{O}.\ \mathbf{O}\left[\![\mathsf{P}_1]\!]_{\mathbf{T}}^{\mathsf{S}}\right] \hookrightarrow^* \mathbf{f} \wedge \mathbf{O}\left[\![\mathsf{P}_2]\!]_{\mathbf{T}}^{\mathsf{S}}\right] \hookrightarrow^* \mathbf{f}' \Rightarrow \exists \mathsf{O}.\ \mathsf{O}\,[\mathsf{P}_1] \hookrightarrow^* \mathsf{f} \wedge \mathsf{O}\,[\mathsf{P}_2] \hookrightarrow^* \mathsf{f}'$

We can instantiate O with $\langle\!\langle \mathbf{O} \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}}$.

So we can assume:

1. $\mathbf{O}\left[\![\mathsf{P}_1]\!]_{\mathbf{T}}^{\mathsf{S}}\right] \hookrightarrow^* \mathbf{f}$
2. $\mathbf{O}\left[\![\mathsf{P}_2]\!]_{\mathbf{T}}^{\mathsf{S}}\right] \hookrightarrow^* \mathbf{f}'$

and prove this

- $\langle\!\langle \mathbf{O} \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}}\,[\mathsf{P}_1] \hookrightarrow^* \mathsf{f}$
- $\langle\!\langle \mathbf{O} \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}}\,[\mathsf{P}_2] \hookrightarrow^* \mathsf{f}'$

If we unfold the definition of $O[P]$ in the hypotheses, assuming that $\mathsf{P}_1 = \mathsf{f}(\mathsf{x}) \mapsto \mathsf{e}_1$ and $\mathsf{P}_2 = \mathsf{f}(\mathsf{x}) \mapsto \mathsf{e}_2$ we obtain

1. $\mathbf{let\ y = let\ x = e'\ in\ if\ x\ has\ NAT\ then}\ [\![\mathsf{e}_1]\!]^{\mathsf{S}}_{\mathbf{T}}\ \mathbf{else\ fail\ in\ e''} \hookrightarrow^* \mathbf{f}$
2. $\mathbf{let\ y = let\ x = e'\ in\ if\ x\ has\ NAT\ then}\ [\![\mathsf{e}_2]\!]^{\mathsf{S}}_{\mathbf{T}}\ \mathbf{else\ fail\ in\ e''} \hookrightarrow^* \mathbf{f'}$

We can unfold the reductions to see that:

$$
[\![\mathsf{P}_1]\!]^{\mathsf{S}}_{\mathbf{T}} \left|
\begin{array}{l}
\quad \mathbf{let\ y = let\ x = e'\ in\ if\ x\ has\ NAT\ then}\ [\![\mathsf{e}_1]\!]^{\mathsf{S}}_{\mathbf{T}}\ \mathbf{else\ fail\ in\ e''} \\
\hookrightarrow^* \mathbf{let\ y = let\ x = v'\ in\ if\ x\ has\ NAT\ then}\ [\![\mathsf{e}_1]\!]^{\mathsf{S}}_{\mathbf{T}}\ \mathbf{else\ fail\ in\ e''} \\
\hookrightarrow \mathbf{let\ y = if\ v'\ has\ NAT\ then}\ ([\![\mathsf{e}_1]\!]^{\mathsf{S}}_{\mathbf{T}}[\mathbf{v'}\ /\ \mathbf{x}])\ \mathbf{else\ fail\ in\ e''}
\end{array}
\right.
$$

Also, by determinism and given that the observer is the same in both cases, we know that until here, the reductions are the same for the second program too, so:

$$
[\![\mathsf{P}_2]\!]^{\mathsf{S}}_{\mathbf{T}} \left|
\begin{array}{l}
\quad \mathbf{let\ y = let\ x = e'\ in\ if\ x\ has\ NAT\ then}\ [\![\mathsf{e}_2]\!]^{\mathsf{S}}_{\mathbf{T}}\ \mathbf{else\ fail\ in\ e''} \\
\hookrightarrow^* \mathbf{let\ y = if\ v'\ has\ NAT\ then}\ ([\![\mathsf{e}_1]\!]^{\mathsf{S}}_{\mathbf{T}}[\mathbf{v'}\ /\ \mathbf{x}])\ \mathbf{else\ fail\ in\ e''}
\end{array}
\right.
$$

Given that the two executions must differ, we know that $\mathbf{v'}$ really is a natural number $\mathbf{n'}$, so:

$$
[\![\mathsf{P}_1]\!]^{\mathsf{S}}_{\mathbf{T}} \left|
\begin{array}{l}
\quad \mathbf{let\ y = if\ n'\ has\ NAT\ then}\ ([\![\mathsf{e}_1]\!]^{\mathsf{S}}_{\mathbf{T}}[\mathbf{n'}\ /\ \mathbf{x}])\ \mathbf{else\ fail\ in\ e''} \\
\hookrightarrow \mathbf{let\ y} = ([\![\mathsf{e}_1]\!]^{\mathsf{S}}_{\mathbf{T}}[\mathbf{n'}\ /\ \mathbf{x}])\ \mathbf{in\ e''} \\
\hookrightarrow^* \mathbf{let\ y} = ([\![\mathsf{n}_1]\!]^{\mathsf{S}}_{\mathbf{T}})\ \mathbf{in\ e''} \\
\hookrightarrow \mathbf{e''}\left[[\![\mathsf{n}_1]\!]^{\mathsf{S}}_{\mathbf{T}}\ /\ \mathbf{y}\right] \\
\hookrightarrow^* \mathbf{f}
\end{array}
\right.
$$

The execution of P_2, instead, must differ, so for a $\mathsf{n}_2 \neq \mathsf{n}_1$:

$$
[\![\mathsf{P}_2]\!]^{\mathsf{S}}_{\mathbf{T}} \left|
\begin{array}{l}
\quad \mathbf{let\ y = if\ n'\ has\ NAT\ then}\ ([\![\mathsf{e}_2]\!]^{\mathsf{S}}_{\mathbf{T}}[\mathbf{n'}\ /\ \mathbf{x}])\ \mathbf{else\ fail\ in\ e''} \\
\hookrightarrow^* \mathbf{e''}\left[[\![\mathsf{n}_2]\!]^{\mathsf{S}}_{\mathbf{T}}\ /\ \mathbf{y}\right] \\
\hookrightarrow^* \mathbf{f'}
\end{array}
\right.
$$

So we can assume that $\mathbf{e''}$ is $\mathbf{if\ y == [\![n1]\!]^{S}_{T}\ then\ 0\ else\ 1}$ and thus $\mathbf{f} = \mathbf{0}$ and $\mathbf{f'} = \mathbf{1}$.

Let's take a look at the source reductions. By Lemma 4.4 (Backtranslation correctness), we know the following:

$$
\mathsf{P}_1 \left|
\begin{array}{l}
\quad \mathsf{let\ y = inject_{NAT}(let\ x = extract_{NAT}}\langle\!\langle \mathbf{e'} \rangle\!\rangle^{\mathbf{T}}_{\mathbf{S}}\ \mathsf{in\ e_1)\ in}\ \langle\!\langle \mathbf{e''} \rangle\!\rangle^{\mathbf{T}}_{\mathbf{S}} \\
\hookrightarrow^* \mathsf{let\ y = inject_{NAT}(let\ x = extract_{NAT}}\langle\!\langle \mathbf{v'} \rangle\!\rangle^{\mathbf{T}}_{\mathbf{S}}\ \mathsf{in\ e_1)\ in}\ \langle\!\langle \mathbf{e''} \rangle\!\rangle^{\mathbf{T}}_{\mathbf{S}}
\end{array}
\right.
$$

By inspecting the target reductions we know, $\mathsf{extract_{NAT}}\langle\!\langle \mathbf{v'} \rangle\!\rangle^{\mathbf{T}}_{\mathbf{S}}$ cannot fail, as $\mathbf{v'}$ is a natural number. Additionally, we know that $\mathbf{v'} = \langle\!\langle \mathbf{n'} \rangle\!\rangle^{\mathbf{T}}_{\mathbf{S}} - 2$. By Lemma 2.2 (Expression correctness) we know:

$$
\mathsf{P}_1 \left|
\begin{array}{l}
\quad \mathsf{let\ y = inject_{NAT}e_1[v'\ /\ x]\ in}\ \langle\!\langle \mathbf{e''} \rangle\!\rangle^{\mathbf{T}}_{\mathbf{S}}\left[\langle\!\langle \mathbf{v} \rangle\!\rangle^{\mathbf{T}}_{\mathbf{S}}\ /\ \mathsf{z}\right] \\
\hookrightarrow^* \mathsf{let\ y = inject_{NAT}n_1\ in}\ \langle\!\langle \mathbf{e''} \rangle\!\rangle^{\mathbf{T}}_{\mathbf{S}}
\end{array}
\right.
$$

So by definition of $\mathsf{inject_{NAT}}$ we have that $\mathsf{n'_1} = \mathsf{n_1} + 2$:

$$
\begin{array}{l|l}
P_1 & \quad \mathsf{let\ y = inject_{NAT} n_1\ in}\ \langle\!\langle \mathbf{e''} \rangle\!\rangle_S^{\mathbf{T}} \\
 & \hookrightarrow \mathsf{let\ y = n_1'\ in}\ \langle\!\langle \mathbf{e''} \rangle\!\rangle_S^{\mathbf{T}} \\
 & \hookrightarrow^* \langle\!\langle \mathbf{e''} \rangle\!\rangle_S^{\mathbf{T}}[n_1' \ / \ y]
\end{array}
$$

So we need to show the reductions for P_2, again by Lemma 4.4 (Backtranslation correctness):

$$
\begin{array}{l|l}
P_2 & \quad \mathsf{let\ y = inject_{NAT}(let\ x = extract_{NAT}}\langle\!\langle \mathbf{e'} \rangle\!\rangle_S^{\mathbf{T}}\ \mathsf{in\ e_2)\ in}\ \langle\!\langle \mathbf{e''} \rangle\!\rangle_S^{\mathbf{T}} \\
 & \hookrightarrow^* \mathsf{let\ y = inject_{NAT} a_2[v' \ / \ x]\ in}\ \langle\!\langle \mathbf{e''} \rangle\!\rangle_S^{\mathbf{T}}
\end{array}
$$

And by how the reductions proceeded in the target and Lemma 2.2 (Expression correctness), we know that:

$$
\begin{array}{l|l}
P_2 & \hookrightarrow \mathsf{let\ y = inject_{NAT} n_2\ in}\ \langle\!\langle \mathbf{e''} \rangle\!\rangle_S^{\mathbf{T}}
\end{array}
$$

We can define $n_2'{=}n_2 + 2$ and we get:

$$
\begin{array}{l|l}
P_2 & \hookrightarrow \mathsf{let\ y = n_2'\ in}\ \langle\!\langle \mathbf{e''} \rangle\!\rangle_S^{\mathbf{T}} \\
 & \hookrightarrow \langle\!\langle \mathbf{e''} \rangle\!\rangle_S^{\mathbf{T}}[n_2' \ / \ y]
\end{array}
$$

Given that we know $\mathbf{e''}$, let's work out its backtranslation (modulo some optimisation and elimination of bits that we know how will reduce):

$$
\begin{array}{l|l}
P_1 \ \& \ P_2 & \quad \mathsf{let\ z = extract_{BOOL}}\langle\!\langle \mathbf{y == n_1} \rangle\!\rangle_S^{\mathbf{T}}\ \mathsf{in\ if\ z == 1\ then}\ \langle\!\langle \mathbf{0} \rangle\!\rangle_S^{\mathbf{T}}\ \mathsf{else}\ \langle\!\langle \mathbf{1} \rangle\!\rangle_S^{\mathbf{T}} \\
 & = \mathsf{let\ z = extract_{BOOL}\ let\ x1 = extract_{NAT}}\langle\!\langle \mathbf{y} \rangle\!\rangle_S^{\mathbf{T}}\ \mathsf{in} \quad \mathsf{in\ if\ z == 1\ then\ 2\ else\ 3} \\
 & \qquad\qquad\qquad\qquad \mathsf{let\ x2 = extract_{NAT}}\langle\!\langle \mathbf{n_1} \rangle\!\rangle_S^{\mathbf{T}}\ \mathsf{in} \\
 & \qquad\qquad\qquad\qquad \mathsf{inject_{BOOL} x1 == x2} \\
 & = \mathsf{let\ z = extract_{BOOL}\ let\ x1 = extract_{NAT} y\ in} \quad \mathsf{in\ if\ z == 1\ then\ 2\ else\ 3} \\
 & \qquad\qquad\qquad\qquad \mathsf{let\ x2 = extract_{NAT} n_1 + 2\ in} \\
 & \qquad\qquad\qquad\qquad \mathsf{inject_{BOOL} x1 == x2} \\
 & = \mathsf{let\ z = extract_{BOOL}\ let\ x1 = y - 2\ in} \quad \mathsf{in\ if\ z == 1\ then\ 2\ else\ 3} \\
 & \qquad\qquad\qquad\qquad \mathsf{let\ x2 = n_1\ in} \\
 & \qquad\qquad\qquad\qquad \mathsf{inject_{BOOL} x1 == x2} \\
 & = \mathsf{if\ y - 2 == n_1\ then\ 2\ else\ 3}
\end{array}
$$

So we have

$$
\begin{array}{l|l}
P_1 & \quad \mathsf{if\ y - 2 == n_1\ then\ 2\ else\ 3}[n_1' \ / \ y] \\
 & = \mathsf{if\ n_1' - 2 == n_1\ then\ 2\ else\ 3} \\
 & = \mathsf{if\ n_1 + 2 - 2 == n_1\ then\ 2\ else\ 3} \\
 & \hookrightarrow 2
\end{array}
$$

Now we know $\langle\!\langle \mathbf{e''} \rangle\!\rangle_S^{\mathbf{T}}$ and we know that $n_2 \neq n_1$, so we have that

$$
\begin{array}{l|l}
P_2 & \quad = \mathsf{if\ y - 2 == n_1\ then\ 2\ else\ 3}[n_2' \ / \ y] \\
 & = \mathsf{if\ n_2 + 2 - 2 == n_1\ then\ 2\ else\ 3} \\
 & = \mathsf{if\ n_2 == n_1\ then\ 2\ else\ 3} \\
 & \hookrightarrow 3
\end{array}
$$

So $f{=}2$ and $f'{=}3$ and this proof holds.

5.1 Proper differentiation

Note that the assumption on what $\mathbf{e}'$ looks like is a simplifying assumption just for the sake of explanation. To conclude the proof properly we need to apply again Lemma 4.4 (but in the more general case with substitutions) and we need the following trivial lemma too:

Lemma 5.1 (Differentiation).

$$\text{if } \mathbf{f} \neq \mathbf{f}' \text{ then } \langle\!\langle \mathbf{f} \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}} \neq \langle\!\langle \mathbf{f}' \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}}$$

6 Trace-Based Backtranslation

The source language, the compiler and the definitions of contextual equivalence and full abstraction do not change.

6.1 Target

$$\mathbf{e} ::= \cdots \mid \mathbf{refl}\ \mathbf{e}$$

$$\mathbf{refl}\ \mathbf{e} \hookrightarrow \mathbf{n} \qquad \text{where } \mathbf{n} = \|\mathbf{e}\|$$

Function $\|\cdot\|$ returns a hash of its argument so each term has its numerical representation.

6.2 Target Traces

$$\mathbf{t} ::= \mathbf{call}\ \mathbf{n}? \cdot \mathbf{ret}\ \mathbf{n}!$$

Definition 6.1 (Traces of a program).

$$\mathsf{TR}(\mathbf{P}) = \{\mathbf{call}\ \mathbf{n}? \cdot \mathbf{ret}\ \mathbf{n}'! \mid \mathbf{P} = \mathbf{f}(\mathbf{x}) \mapsto \mathbf{e} \text{ and } \mathbf{let}\ \mathbf{x} = \mathbf{n}\ \mathbf{in}\ \mathbf{e} \hookrightarrow^* \mathbf{n}'\}$$

Definition 6.2 (Trace equivalence).

$$\mathbf{P_1} \overset{\mathrm{T}}{=} \mathbf{P_2} \overset{\text{def}}{=} \mathsf{TR}(\mathbf{P_1}) = \mathsf{TR}(\mathbf{P_2})$$

6.2.1 Properties of the Target Traces

Lemma 6.3 (Soundness of traces).

$$\mathbf{P_1} \overset{\mathrm{T}}{=} \mathbf{P_2} \Rightarrow \mathbf{P_1} \simeq_{ctx} \mathbf{P_2}$$

6.3 Backtranslation of Traces

At this point we are given two different traces which, by definition, agree on the call.. parameters and we must build both a source observer O that leads to the differentiation.

$$\mathbf{t_1} = \mathbf{call}\ \mathbf{n}?\cdot\mathbf{ret}\ \mathbf{n^1}! \qquad\qquad \mathbf{t_2} = \mathbf{call}\ \mathbf{n}?\cdot\mathbf{ret}\ \mathbf{n^2}!$$

$$\begin{aligned}\langle\!\langle \mathbf{n} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= \mathsf{n}\\ \langle\!\langle \mathbf{t_1}, \mathbf{t_2} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} &= \mathsf{let\ y = call\ f}\ \langle\!\langle \mathbf{n} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{in\ if\ y ==}\ \langle\!\langle \mathbf{n^1} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{then\ 1\ else\ 2}\end{aligned}$$

6.3.1 Properties of the Backtranslation of Traces

Lemma 6.4 (Correctness of the backtranslation of traces).

$$\begin{aligned}&\text{if } \mathbf{call}\ \mathbf{n}?\cdot\mathbf{ret}\ \mathbf{n'}! \in \mathsf{TR}(\llbracket \mathsf{P} \rrbracket^{\mathsf{S}}_{\mathbf{T}})\\ &\quad \mathsf{P} = \mathsf{f(x)} \mapsto \mathsf{e}\\ &\text{then } \mathsf{let\ y = let\ x =}\ \langle\!\langle \mathbf{n} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{in\ e\ in\ e'} \hookrightarrow^* \mathsf{e'}\left[\langle\!\langle \mathbf{n'} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} \,/\, \mathsf{y}\right]\end{aligned}$$

Proof. This follows by unfolding the definitions of the trace semantics and by Lemma 2.2 (Expression correctness).

6.4 Using the Backtranslation of Traces

Back to proving compiler full abstraction:

The ⇒ case:

$$\forall \mathsf{P_1}, \mathsf{P_2}.\ \ \mathsf{P_1} \simeq_{ctx} \mathsf{P_2} \Rightarrow \llbracket \mathsf{P_1} \rrbracket^{\mathsf{S}}_{\mathbf{T}} \simeq_{ctx} \llbracket \mathsf{P_2} \rrbracket^{\mathsf{S}}_{\mathbf{T}}$$

We can apply Lemma 6.3 (Soundness of traces) and we get:

$$\forall \mathsf{P_1}, \mathsf{P_2}.\ \ \mathsf{P_1} \simeq_{ctx} \mathsf{P_2} \Rightarrow \llbracket \mathsf{P_1} \rrbracket^{\mathsf{S}}_{\mathbf{T}} \overset{\mathbb{T}}{=} \llbracket \mathsf{P_2} \rrbracket^{\mathsf{S}}_{\mathbf{T}}$$

in contrapositive form:

$$\forall \mathsf{P_1}, \mathsf{P_2}.\ \ \llbracket \mathsf{P_1} \rrbracket^{\mathsf{S}}_{\mathbf{T}} \overset{\mathbb{T}}{\neq} \llbracket \mathsf{P_2} \rrbracket^{\mathsf{S}}_{\mathbf{T}} \Rightarrow \mathsf{P_1} \not\simeq_{ctx} \mathsf{P_2}$$

expanding the definitions:

$$\begin{aligned}\forall \mathsf{P_1}, \mathsf{P_2}.\ \ &\exists \mathbf{t} \in \mathsf{TR}(\llbracket \mathsf{P_1} \rrbracket^{\mathsf{S}}_{\mathbf{T}}) \wedge \mathbf{t} \notin \mathsf{TR}(\llbracket \mathsf{P_2} \rrbracket^{\mathsf{S}}_{\mathbf{T}})\\ &\Rightarrow \exists \mathsf{O}.\ \mathsf{O}\,[\mathsf{P_1}] \hookrightarrow^* \mathsf{n'} \wedge \mathsf{O}\,[\mathsf{P_2}] \hookrightarrow^* \mathsf{n''}\end{aligned}$$

We pick another trace $\mathbf{t'}$ from $\mathsf{TR}(\llbracket \mathsf{P_2} \rrbracket^{\mathsf{S}}_{\mathbf{T}})$ such that the first part is the same as in $\mathbf{t}$ for the backtranslation.

We can now use the backtranslation of traces with $\mathbf{t}$ and $\mathbf{t'}$ to instantiate O.

The reductions proceed as follows. By Lemma 6.4 (Correctness of the back-translation of traces) we know:

$$
\begin{array}{l|l}
[\![\mathsf{P}_1]\!]^{\mathsf{S}}_{\mathbf{T}} & \quad \mathsf{let\ y} = \mathsf{call\ f}\ \langle\!\langle \mathbf{n} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{in\ if\ y} == \langle\!\langle \mathbf{n^1} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{then\ 1\ else\ 2} \\
& \hookrightarrow^* \mathsf{if\ y} == \langle\!\langle \mathbf{n^1} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{then\ 1\ else\ 2}\left[\langle\!\langle \mathbf{n_1} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}} \,/\, \mathsf{y}\right]) \\
& \quad \equiv \mathsf{if}\ \mathsf{n}_1 == \langle\!\langle \mathbf{n^1} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{then\ 1\ else\ 2}
\end{array}
$$

and

$$
\begin{array}{l|l}
[\![\mathsf{P}_2]\!]^{\mathsf{S}}_{\mathbf{T}} & \quad (\mathsf{let\ y} = \mathsf{call\ f}\ \langle\!\langle \mathbf{n} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{in\ if\ y} == \langle\!\langle \mathbf{n^1} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{then\ 1\ else\ 2} \\
& \hookrightarrow^* \mathsf{if}\ \mathsf{n}_2 == \langle\!\langle \mathbf{n^1} \rangle\!\rangle^{\mathbf{T}}_{\mathsf{S}}\ \mathsf{then\ 1\ else\ 2}
\end{array}
$$

The different reductions now are straightforward, so $\mathsf{f}= 1$ and $\mathsf{f}'=2$ so this case holds.

Appendix

The appendix provides general indications on how the proofs proceed and omits some cases for students to practice proofs.

A Proof of Lemma 2.1

Proof. By structural induction over e.

Base case $\mathsf{e}{=}\mathsf{n}$

$\mathsf{e}{=}\mathsf{true}$

$\mathsf{e}{=}\mathsf{false}$

$\mathsf{e}{=}\mathsf{x}$

Inductive case In this case we identify these inductive hypotheses:

1. if $\mathsf{e}{\hookrightarrow^*}\mathsf{n}$ then $[\![\mathsf{e}]\!]^{\mathsf{S}}_{\mathbf{T}}{\hookrightarrow^*}[\![\mathsf{n}]\!]^{\mathsf{S}}_{\mathbf{T}}$;
2. if $\mathsf{e}'{\hookrightarrow^*}\mathsf{n}'$ then $[\![\mathsf{e}']\!]^{\mathsf{S}}_{\mathbf{T}}{\hookrightarrow^*}[\![\mathsf{n}']\!]^{\mathsf{S}}_{\mathbf{T}}$;
3. if $\mathsf{e}_\mathsf{b}{\hookrightarrow^*}\mathsf{v}$ then $[\![\mathsf{e}_\mathsf{b}]\!]^{\mathsf{S}}_{\mathbf{T}}{\hookrightarrow^*}[\![\mathsf{v}]\!]^{\mathsf{S}}_{\mathbf{T}}$ (only in the "if" case).

The following cases arise:

$\mathsf{e}{=}\mathsf{let\ x = e\ in\ e}$

$\mathsf{e}{=}\mathsf{if\ e_b\ then\ e\ else\ e'}$

$\mathsf{e}{=}\mathsf{e}\oplus\mathsf{e}'$

$\mathsf{e}{=}\mathsf{e}\odot\mathsf{e}'$

B Proof of Lemma 4.2

Proof. Simple case analysis.

C Proof of Lemma 4.4

Proof. By structural induction over $\mathbf{e}$.

Base case $\mathbf{e}{=}\mathbf{n}$

$\mathbf{e}{=}\mathbf{true}$

$\mathbf{e}{=}\mathbf{false}$

$\mathbf{e}{=}\mathbf{fail}$

$\mathbf{e}{=}\mathbf{x}$

Inductive case In this case we identify these inductive hypotheses:

1. if $\mathbf{e} \hookrightarrow^* \mathbf{f}$ then $\langle\!\langle \mathbf{e} \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}} \hookrightarrow^* \langle\!\langle \mathbf{f} \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}}$;
2. if $\mathbf{e}' \hookrightarrow^* \mathbf{f}'$ then $\langle\!\langle \mathbf{e}' \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}} \hookrightarrow^* \langle\!\langle \mathbf{f}' \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}}$;
3. if $\mathbf{e}'' \hookrightarrow^* \mathbf{f}''$ then $\langle\!\langle \mathbf{e}'' \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}} \hookrightarrow^* \langle\!\langle \mathbf{f}'' \rangle\!\rangle_{\mathsf{S}}^{\mathbf{T}}$ (only needed in the "if" case).

The following cases arise:

$\mathbf{e} = \mathbf{let\ x = e\ in\ e}'$

$\mathbf{e} = \mathbf{if\ e\ then\ e}'\ \mathbf{else\ e}''$

$\mathbf{e} = \mathbf{e} \oplus \mathbf{e}'$

$\mathbf{e} = \mathbf{e} \odot \mathbf{e}'$

D Proof of Lemma 6.3

Proof. The proof proceeds by contradiction.

Assume the thesis is false: wlog we have: $\mathbf{P_1} \not\simeq_{ctx} \mathbf{P_2}$, so $\exists \mathfrak{C}.\mathfrak{C}[\mathbf{P_1}]\Downarrow \wedge \mathfrak{C}[\mathbf{P_2}]\Uparrow$

Let us take a look at the traces of $\mathfrak{C}[\mathbf{P_1}]$ and $\mathfrak{C}[\mathbf{P_2}]$ respectively.

They are of the form $\mathbf{call\ n?} \cdot \mathbf{ret\ n_1!}$ and $\mathbf{call\ n?} \cdot \mathbf{ret\ n_2!}$.

By determinism of the semantics, $\mathbf{n}$ must coincide since it comes from the same $\mathfrak{C}$.

By analysing the semantics, the only way for $\mathfrak{C}$ to behave differently is to receive two different numbers $\mathbf{n_1}$ and $\mathbf{n_2}$.

This contradicts the assumption that traces are equal.