Equational Reasoning

"When are two expressions equal? Whenever we can't tell them apart!" -Harper, PFPL

Syntactic Equality $\bar{2} = \bar{2}$

$\lambda x. x+\bar{1} = \lambda x. x+\bar{1}$

$\lambda y. y+\bar{1} \neq \lambda x. x+\bar{1}$

$\lambda x. x+\bar{1} \neq \lambda x. \bar{1}+x \ldots$

α Equivalence $\lambda x. x \equiv \lambda y. y$

Kleene Equality $\simeq$
(for "observable types", e.g. int, bool)
$e \simeq e'$ if $\exists v.\ e \mapsto^* v$ and $e' \mapsto v$

e.g. $\bar{1}+\bar{2} \simeq \bar{3} \simeq \bar{2}+\bar{1}$

What about $\lambda x. \lambda y. x+y$ and $\lambda x. \lambda y. y+x$?

Observational Equivalence $\cong$
Can't do an "experiment" that can tell them apart
Only looking at results! (" program)
e.g. $\bar{1}+\bar{2}+\bar{3}+\bar{4}+\bar{5} \cong \bar{15}$
Quicksort $\cong$ Bubble sort

Expression context $C ::= \circ \mid \lambda x. C \mid C\, e \mid e\, C \mid (C, e) \mid (e, C)$
$\mid$ fst $C \mid$ snd $C \mid$ inl $C \mid$ inr C
$\mid$ case C of $\{x. e; y. e\}$
$\mid$ case e of $\{x. C; y. e\}$
$\mid$ case e of $\{x. e; y. e\}$
$\mid \Lambda\alpha. e \mid e[C]$

Like evaluation contexts but the hole can be anywhere!

Program Context: Exp. context that has type int and no free vars at outer level.
i.e., closed exp of $\underbrace{\text{type int}}$ w/ one hole
need to be able to tell the result of the experiment!

Types for contexts: $(\Delta;\Gamma \triangleright \tau) \rightsquigarrow (\Delta';\Gamma' \triangleright \tau')$

exp to fill hole has type τ under Γ, Δ — outer exp has type τ' under Γ', Δ'

If C is a pgm context, $C : (\Delta;\Gamma \triangleright \tau) \rightsquigarrow (\cdot;\cdot \triangleright \text{int})$
If $\Delta;\Gamma \vdash e : \tau$, then $\cdot;\cdot \vdash C[e] : \text{int}$.

<u>Observational equivalence</u>
If $\Delta;\Gamma \vdash e : \tau$ and $\Delta;\Gamma \vdash e' : \tau$, then $e \cong e'$ if for <u>all program contexts</u> $C : (\Delta;\Gamma \triangleright \tau) \rightsquigarrow (\cdot;\cdot \triangleright \text{int})$, $C[e] \simeq C[e']$.

So is $\lambda x. \lambda y. x+y \cong \lambda x. \lambda y. y+x$?
Reasoning about all program contexts is hard!

<u>Logical Equivalence</u> $\sim_\tau$ - defined inductively on the type!

$e \sim_{\text{unit}} e'$ if $e \simeq e'$
$e \sim_{\text{int}} e'$ if $e \simeq e'$
$e \sim_{\tau_1 \to \tau_2} e'$ if $\forall e_1, e_1'$ s.t. $e_1 \sim_{\tau_1} e_1'$, we have $e\, e_1 \sim_{\tau_2} e'\, e_1'$
...
$e \sim_{\forall \alpha.\tau} e'$ if $\forall \rho, \rho'$ and <u>admissible</u> $R : \rho \times \rho' \to \text{Bool}$,
$e[\rho] \sim_\tau^{\alpha \mapsto R} e'[\rho']$
use R to compare at type α
$e \sim_\alpha^{\alpha \mapsto R} e'$ if $R(e, e')$

Definition 1. A relation $R : \rho \times \rho' \to Bool$ is <u>admissible</u> if

1. It respects observational eq.: If $R(e, e')$ and $d \cong e$ and $d' \cong e'$ then $R(d, d')$.
2. "Closure under converse evaluation": If $R(e, e')$, then:
   a. If $d \mapsto e$, then $R(d, e')$
   b. If $d' \mapsto e'$, then $R(e, d')$

R "can't tell apart things it shouldn't be able to."

Theorem 1. If $\cdot ; \cdot \vdash e : \tau$ and $\cdot ; \cdot \vdash e' : \tau$ then $e \sim_\tau e' \Leftrightarrow e \cong e'$.
(Can also generalize log. eq. to non-empty contexts)

Theorem 2. (Parametricity) If $\cdot ; \cdot \vdash e : \tau$, then $e \sim_\tau e$.

Theorem 3. Let $\cdot ; \cdot \vdash e : \forall \alpha . \alpha \to \alpha$ and let $id \triangleq \Lambda \alpha . \lambda x : \alpha . x$.
Then $e \sim_{\forall \alpha . \alpha \to \alpha} id$ (and by Thm 1, $e \cong id$)

Proof. Let ρ, ρ' be types and $R : \rho \times \rho' \to Bool$ admissible. Suppose $R(e_0, e_0')$.
WTS (by def of $\sim$) $R(e[\rho]\, e_0, id[\rho']\, (e_0'))$
Because $id[\rho']\, e_0' \mapsto^* e_0'$, suffices to show $R(e[\rho]\, e_0, e_0')$ by Def 1.2

Because $R(e_0, e_0')$ and $e_0' \cong e_0'$, by Def 1.1, suffices to show $e[\rho]\, e_0 \cong e_0$.

By Thm 2, $e \sim_{\forall \alpha . \alpha \to \alpha} e$. So, for any admissible $S : \rho \times \rho \to Bool$, if $S(e_0, e_0)$, then $S(e[\rho]\, e_0, e[\rho]\, e_0)$.
Let $S(d, d')$ iff $d \cong e_0 \cong d'$.
Clearly $S(e_0, e_0)$, so $S(e[\rho]\, e_0, e[\rho]\, e_0) \Rightarrow e[\rho]\, e_0 \cong e_0$. □