Relational Reasoning
(Relationel ræsonnement)

Mathias Pedersen, 201808137

Bachelor Report (15 ECTS) in Computer Science
Advisor: Amin Timany
Department of Computer Science, Aarhus University
October 2021

AARHUS
UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Abstract

►in English. . . ◄

Mathias Pedersen,
Aarhus, October 2021.

Contents

Chapter 1

Introduction

►motivate and explain the problem to be addressed◄
►example of a citation: [1]◄ ►get your bibtex entries from `https://dblp.org/`◄

Chapter 2

Definition of Language

▶create draft◀

Syntax

$$
\begin{array}{lll}
e ::= () \mid & & \text{(unit value)} \\
\quad x \mid & & \text{(variables)} \\
\quad \overline{n} \mid e + e \mid e - e \mid e \leq e \mid e < e \mid e = e \mid & & \text{(integers)} \\
\quad \mathsf{true} \mid \mathsf{false} \mid \mathsf{if}\ e\ \mathsf{then}\ e\ \mathsf{else}\ e \mid & & \text{(booleans)} \\
\quad (e, e) \mid \mathsf{fst}\ e \mid \mathsf{snd}\ e \mid & & \text{(products)} \\
\quad \mathsf{inj}_1\ e \mid \mathsf{inj}_2\ e \mid \mathsf{match}\ e\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e \mid \mathsf{inj}_2\ x \Rightarrow e\ \mathsf{end} \mid & & \text{(sums)} \\
\quad \mathsf{rec}\ f(x) := e \mid e\ e & & \text{(recursive functions)} \\
\quad \Lambda\ e \mid e\ \_ & & \text{(polymorphism)}
\end{array}
$$

$$
v ::= () \mid \overline{n} \mid \mathsf{true} \mid \mathsf{false} \mid (v, v) \mid \mathsf{inj}_1\ v \mid \mathsf{inj}_2\ v \mid \mathsf{rec}\ f(x) := e \mid \Lambda\ e \qquad \text{(values)}
$$

$$
\tau ::= \mathsf{Unit} \mid \mathbb{Z} \mid \mathbb{B} \mid \tau \times \tau \mid \tau + \tau \mid \tau \to \tau \mid \forall X.\ \tau \qquad \text{(types)}
$$

$$
\begin{array}{lll}
K ::= [] \mid K + e \mid v + K \mid K - e \mid v - K \mid K \leq e \mid v \leq K \mid K < e \mid v < K \mid & & \text{(evaluation context)} \\
\quad K = e \mid v = K \mid \mathsf{if}\ K\ \mathsf{then}\ e\ \mathsf{else}\ e \mid (K, e) \mid (v, K) \mid \mathsf{fst}\ K \mid \mathsf{snd}\ K \mid & & \\
\quad \mathsf{inj}_1\ K \mid \mathsf{inj}_2\ K \mid \mathsf{match}\ K\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e \mid \mathsf{inj}_2\ x \Rightarrow e\ \mathsf{end} \mid K\ e \mid v\ K \mid K\ \_ & &
\end{array}
$$

Typing rules

$$\text{T-VAR}\ \frac{(x:\tau)\in\Gamma}{\Xi \mid \Gamma \vdash x:\tau} \qquad \text{T-UNIT}\ \frac{}{\Xi \mid \Gamma \vdash () : \mathsf{Unit}} \qquad \text{T-INT}\ \frac{}{\Xi \mid \Gamma \vdash \overline{n} : \mathbb{Z}}$$

$$\text{T-ADD}\ \frac{\Xi \mid \Gamma \vdash e_1 : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 + e_2 : \mathbb{Z}} \qquad \text{T-SUB}\ \frac{\Xi \mid \Gamma \vdash e_1 : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 - e_2 : \mathbb{Z}}$$

$$\text{T-LE}\ \frac{\Xi \mid \Gamma \vdash e_1 : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 \le e_2 : \mathbb{B}} \qquad \text{T-LT}\ \frac{\Xi \mid \Gamma \vdash e_1 : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 < e_2 : \mathbb{B}}$$

$$\text{T-EQ}\ \frac{\Xi \mid \Gamma \vdash e_1 : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 = e_2 : \mathbb{B}} \qquad \text{T-TRUE}\ \frac{}{\Xi \mid \Gamma \vdash \mathsf{true} : \mathbb{B}} \qquad \text{T-FALSE}\ \frac{}{\Xi \mid \Gamma \vdash \mathsf{false} : \mathbb{B}}$$

$$\text{T-IF}\ \frac{\Xi \mid \Gamma \vdash e_1 : \mathbb{B} \qquad \Xi \mid \Gamma \vdash e_2 : \tau \qquad \Xi \mid \Gamma \vdash e_3 : \tau}{\Xi \mid \Gamma \vdash \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 : \tau}$$

$$\text{T-PAIR}\ \frac{\Xi \mid \Gamma \vdash e_1 : \tau_1 \qquad \Xi \mid \Gamma \vdash e_2 : \tau_2}{\Xi \mid \Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2} \qquad \text{T-FST}\ \frac{\Xi \mid \Gamma \vdash e : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \mathsf{fst}\ e : \tau_1}$$

$$\text{T-SND}\ \frac{\Xi \mid \Gamma \vdash e : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \mathsf{snd}\ e : \tau_2} \qquad \text{T-INJ1}\ \frac{\Xi \mid \Gamma \vdash e : \tau_1}{\Xi \mid \Gamma \vdash \mathsf{inj}_1\ e : \tau_1 + \tau_2} \qquad \text{T-INJ2}\ \frac{\Xi \mid \Gamma \vdash e : \tau_2}{\Xi \mid \Gamma \vdash \mathsf{inj}_2\ e : \tau_1 + \tau_2}$$

$$\text{T-MATCH}\ \frac{\Xi \mid \Gamma \vdash e_1 : \tau_1 + \tau_2 \qquad \Xi \mid \Gamma, x:\tau_1 \vdash e_2 : \tau \qquad \Xi \mid \Gamma, x:\tau_2 \vdash e_3 : \tau}{\Xi \mid \Gamma \vdash \mathsf{match}\ e_1\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e_2 \mid \mathsf{inj}_2\ x \Rightarrow e_3\ \mathsf{end} : \tau}$$

$$\text{T-REC}\ \frac{\Xi \mid \Gamma, f:\tau_1 \to \tau_2, x:\tau_1 \vdash e : \tau_2}{\Xi \mid \Gamma \vdash \mathsf{rec}\ f(x) := e : \tau_1 \to \tau_2} \qquad \text{T-APP}\ \frac{\Xi \mid \Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Xi \mid \Gamma \vdash e_2 : \tau_1}{\Xi \mid \Gamma \vdash e_1\ e_2 : \tau_2}$$

$$\text{T-TLAM}\ \frac{\Xi, X \mid \Gamma \vdash e : \tau}{\Xi \mid \Gamma \vdash \Lambda\, e : \forall X.\, \tau} \qquad \text{T-TAPP}\ \frac{\Xi \mid \Gamma \vdash e : \forall X.\, \tau}{\Xi \mid \Gamma \vdash e\ \_ : \tau\left[\tau'/X\right]}$$

Dynamics

$$\textsc{head-step-step}\ \frac{e \to_h e'}{K[e] \to K[e']}$$

$$\textsc{E-add}\ \overline{n_1} + \overline{n_2} \to_h \overline{n_1 + n_2} \qquad \textsc{E-sub}\ \overline{n_1} - \overline{n_2} \to_h \overline{n_1 - n_2} \qquad \textsc{E-eq}\ \frac{n_1 = n_2}{\overline{n_1} = \overline{n_2} \to_h \mathsf{true}}$$

$$\textsc{E-not-eq}\ \frac{n_1 \neq n_2}{\overline{n_1} = \overline{n_2} \to_h \mathsf{false}} \qquad \textsc{E-le}\ \frac{n_1 \leq n_2}{\overline{n_1} \leq \overline{n_2} \to_h \mathsf{true}} \qquad \textsc{E-not-le}\ \frac{n_1 \not\leq n_2}{\overline{n_1} \leq \overline{n_2} \to_h \mathsf{false}}$$

$$\textsc{E-lt}\ \frac{n_1 < n_2}{\overline{n_1} < \overline{n_2} \to_h \mathsf{true}} \qquad \textsc{E-not-lt}\ \frac{n_1 \not< n_2}{\overline{n_1} < \overline{n_2} \to_h \mathsf{false}} \qquad \textsc{E-if-true}\ \mathsf{if\ true\ then}\ e_2\ \mathsf{else}\ e_3 \to_h e_2$$

$$\textsc{E-if-false}\ \mathsf{if\ false\ then}\ e_2\ \mathsf{else}\ e_3 \to_h e_3 \qquad \textsc{E-fst}\ \mathsf{fst}\ (v_1, v_2) \to_h v_1 \qquad \textsc{E-snd}\ \mathsf{snd}\ (v_1, v_2) \to_h v_2$$

$$\textsc{E-match-inj1}\ \mathsf{match}\ (\mathsf{inj}_1\ v)\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e_2 \mid \mathsf{inj}_2\ x \Rightarrow e_3\ \mathsf{end} \to_h e_2[v/x]$$

$$\textsc{E-match-inj2}\ \mathsf{match}\ (\mathsf{inj}_2\ v)\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e_2 \mid \mathsf{inj}_2\ x \Rightarrow e_3\ \mathsf{end} \to_h e_3[v/x]$$

$$\textsc{E-rec-app}\ (\mathsf{rec}\ f(x) := e)v \to_h e[\mathsf{rec}\ f(x) := e/f][v/x] \qquad \textsc{E-tapp-tlam}\ (\Lambda\ e)\ \_ \to_h e$$

Lemma 1. $K[e] \to_h e' \wedge \neg(K = []) \implies \mathrm{Val}(e)$

Proof. This can be shown by doing case distinction on the head-step $K[e] \to_h e'$. We show it here only for case E-ADD, as all the other cases are similar.

So assume $K[e] = \overline{n_1} + \overline{n_2}$, and $e' = \overline{n_1 + n_2}$. Then there are three cases: $K = []$ and $e = \overline{n_1} + \overline{n_2}$, $K = [] + \overline{n_2}$ and $e = \overline{n_1}$, or $K = \overline{n_1} + []$ and $e = \overline{n_2}$. The first case raises a contradiction as we have assumed $\neg(K = [])$. In the remaining two cases, we may conclude $\mathrm{Val}(e)$, as wanted. □

Lemma 2 (Evaluation under Context)**.** $K[e] \to^* e' \implies \exists e''.(e \to^* e'') \wedge ((\mathrm{Val}(e'') \wedge K[e''] \to^* e') \vee (\neg\mathrm{Val}(e'') \wedge K[e''] = e'))$

Proof. So assuming $K[e] \to^* e'$, we must show

$$\exists e''.(e \to^* e'') \wedge \big((\mathrm{Val}(e'') \wedge K[e''] \to^* e') \vee (\neg\mathrm{Val}(e'') \wedge K[e''] = e')\big) \tag{2.1}$$

We proceed by induction on the number of steps in the evaluation $K[e] \to^* e'$. Let n denote the number of steps taken, so that $K[e] \to^n e'$.

- Base Case $n = 0$. In this case, we have that $K[e] \to^0 e'$, which means that $K[e] = e'$. Now use e for e'' in 2.1. We must show

$$(e \to^* e) \wedge \big((\text{Val}(e) \wedge K[e] \to^* e') \vee (\neg\text{Val}(e) \wedge K[e] = e')\big)$$

  Trivially, $e \to^* e$. For the second part, we proceed by case distinction on $\text{Val}(e)$.

  - $\text{Val}(e)$. We have that $K[e] \to^0 e'$, so $K[e] \to^* e'$. Thus, we have $\text{Val}(e) \wedge K[e] \to^* e'$, which matches the left part of the "$\vee$".
  - $\neg\text{Val}(e)$. We know that $K[e] = e'$, so we have $\neg\text{Val}(e) \wedge K[e] = e'$, which matches the right part of the "$\vee$".

- Inductive Step $n = m + 1$. Now we have $K[e] \to^{m+1} e'$. By the Induction Hypothesis, we have:

$$\forall F, f, f'.F[f] \to^m f' \implies \exists f''.(f \to^* f'') \wedge \big((\text{Val}(f'') \wedge F[f''] \to^* f') \vee (\neg\text{Val}(f'') \wedge K[f''] = f')\big) \quad (2.2)$$

  Split the evaluation, $K[e] \to^{m+1} e'$, up, so that $K[e] \to g \wedge g \to^m e'$. Looking at our dynamics, we must have that $K[e] = H[h]$, and $g = H[h']$, for some evaluation context H, and expressions h, h', and $h \to_h h'$. There are now three possible cases. Either $e = h$, e is a superexpression of h, or e is a subexpression of h. We will consider each in turn.

  - $K = H$ and $e = h$.
    Then $g = H[h'] = K[h']$, and $e \to_h h'$. Furthermore, $K[h'] \to^m e'$. Instantiate I.H. with this to get

$$\exists f''.(h' \to^* f'') \wedge \big((\text{Val}(f'') \wedge K[f''] \to^* e') \vee (\neg\text{Val}(f'') \wedge K[f''] = e')\big) \quad (2.3)$$

    Call this quantified expression for f'', and use it for e'' in 2.1. We must then show

$$(e \to^* f'') \wedge \big((\text{Val}(f'') \wedge K[f''] \to^* e') \vee (\neg\text{Val}(f'') \wedge K[f''] = e')\big)$$

    We know that $e \to h'$, as $e \to_h h'$, and by 2.3, we know that $h' \to^* f''$, so that $e \to^* f''$. The second part follows directly from 2.3.
  - $K[E[]] = H$ and $e = E[h]$.
    Then $g = H[h'] = K[E[]][h'] = K[E[h']]$, thus $K[E[h']] \to^m e'$. Instantiate I.H. with this to get

$$\exists f''.(E[h'] \to^* f'') \wedge \big((\text{Val}(f'') \wedge K[f''] \to^* e') \vee (\neg\text{Val}(f'') \wedge K[f''] = e')\big) \quad (2.4)$$

Call this quantified expression for f'', and use it for e'' in 2.1. We must then show

$$(e \to^* f'') \wedge \big((\mathrm{Val}(f'') \wedge K[f''] \to^* e') \vee (\neg\mathrm{Val}(f'') \wedge K[f''] = e')\big)$$

We know that $E[h] \to E[h']$, as $h \to_h h'$, and since $e = E[h]$, then $e \to E[h']$. By 2.4, we know that $E[h'] \to^* f''$, so that $e \to^* f''$. The second part follows directly from 2.4.

– $K = H[E[]]$ and $E[e] = h$. Here we have $h = E[e]$, so $E[e] \to_h h'$. Note that E is not the empty evaluation context, as otherwise, we would be in case 1. So by lemma 1, we know that $\mathrm{Val}(e)$. Now, pick e for e'' in 2.1. We must show

$$(e \to^* e) \wedge \big((\mathrm{Val}(e) \wedge K[e] \to^* e') \vee (\neg\mathrm{Val}(e) \wedge K[e] = e')\big)$$

Trivially, $e \to^* e$. We also have that $\mathrm{Val}(e)$, and since $K[e] \to^{m+1} e'$, then $K[e] \to^* e'$.

□

Chapter 3

Contextual Equivalence

►**draft**◄
Imagine you are writing a larger program, and as part of that program, you need to use a stack. So as part of your program, you implement a stack. However, your implementation is naive, and not efficient. Thus you implement a new stack, that is more complex, but also more efficient. You would of course want to use the more efficient stack implementation in your larger program, but since it is more complex, you are not really sure whether you can justify the refactoring – the implementations might show differing behaviour.
What you want to show, then, is that your two stack implementation *behaves* the same. In other words, no matter which *context* you use your two stack implementations in, they will always behave the same as part of the context. This is intuitively what we want to define with *Contextual Equivalence*.
In this section we give two equivalent definitions of what it means for programs to be contextually equivalent. The first definition will be more intuitive and instructive, while the second will give a deeper insight into the theoretical underpinnings of contextual equivalence.

3.1 Definition of Contextual Equivalence

Context

$$
\begin{aligned}
C ::= & [\cdot] \mid C + e \mid e + C \mid C - e \mid e - C \mid C \leq e \mid e \leq C \mid C < e \mid e < C \mid C = e \mid e = C \mid \\
& \mathsf{if}\ C\ \mathsf{then}\ e\ \mathsf{else}\ e \mid \mathsf{if}\ e\ \mathsf{then}\ C\ \mathsf{else}\ e \mid \mathsf{if}\ e\ \mathsf{then}\ e\ \mathsf{else}\ C \mid (C, e) \mid (e, C) \mid \\
& \mathsf{fst}\ C \mid \mathsf{snd}\ C \mid \mathsf{inj}_1\ C \mid \mathsf{inj}_2\ C \mid \mathsf{match}\ C\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e \mid \mathsf{inj}_2\ x \Rightarrow e\ \mathsf{end} \mid \\
& \mathsf{match}\ e\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow C \mid \mathsf{inj}_2\ x \Rightarrow e\ \mathsf{end} \mid \\
& \mathsf{match}\ e\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e \mid \mathsf{inj}_2\ x \Rightarrow C\ \mathsf{end} \mid \mathsf{rec}\ f(x) := C \mid C\ e \mid e\ C \mid \Lambda\ C \mid C\ \_
\end{aligned}
$$

Context Typing

$$\frac{\Xi \mid \Gamma\ \vdash e : \tau \qquad \Xi' \mid \Gamma'\ \vdash C[e] : \tau'}{C : (\Xi \mid \Gamma\ \vdash \tau) \Rightarrow (\Xi' \mid \Gamma'\ \vdash \tau')}\ \text{T-CTX}$$

Read as: the context takes an expression of type τ under Ξ, Γ, and outputs an expression of type τ' under Ξ', Γ'.

Definition 3.1.1 (Contextual Equivalence)**.** We define contextual equivalence of two expressions e_1 and e_2 at type τ under Ξ and Γ as

$$\Xi \mid \Gamma \vdash e_1 \approx^{ctx} e_2 : \tau$$

$$\iff$$

$$\Xi \mid \Gamma \vdash e_1 : \tau \quad \wedge \quad \Xi \mid \Gamma \vdash e_2 : \tau \quad \wedge$$

$$\forall C : (\Xi \mid \Gamma \vdash \tau) \Rightarrow (\bullet \mid \bullet \vdash \mathsf{Unit}).(C[e_1] \Downarrow \iff C[e_2] \Downarrow)$$

We define CE so that two programs are CE when under all Contexts, one program terminates iff the other does. One may ponder whether this definition is really strong enough. If both terminate, we really want the result of the two programs to have values that are behaviourally the same, otherwise, we can certainly put them in a context, where they have different behaviour. For integer values, for example, this means that the two values must be the same number, and for product types, this means that the first value in each pair should be behaviourally the same, and likewise with the second value in each pair. To give an intuition as to why this definition is strong enough, consider two contextually equivalent programs e_1 and e_2, where $\bullet \mid \bullet \vdash e_1 : \tau$ and $\bullet \mid \bullet \vdash e_2 : \tau$. Note first that if both e_1 and e_2 don't terminate, then they will be behaviourally the same; if we plug them into any context, and the context makes us evaluate the expressions, both will not terminate. And if the context doesn't make us evaluate the expressions, then they have no influence in the run of the programs, so they will behave the same.

Now consider what happens if e_1 terminates with some value v_1. Can we then guarantee that e_2 also terminates with some value v_2, and v_1 and v_2 behave the same? Since e_1 and e_2 are contextually equivalent, then we can put them into any context, C, and one will terminate if and only if the other one does. Let's assume that $\tau = \mathbb{Z}$. Then consider when C has the form $\mathsf{if}\ [\cdot] = v_1\ \mathsf{then}\ ()\ \mathsf{else}\ \omega$. Here $C[e_1] \Downarrow ()$. But what about $C[e_2]$? If $v_2 \neq v_1$, then our evaluation rules tell us that we will take the else branch, and hence not terminate. However, since e_1 and e_2 are contextually equivalent, and $C[e_1]$ terminates, then we know that $C[e_2]$ must also terminate. Hence it is not the case that $v_2 \neq v_1$, thus $v_2 = v_1$. So if our two programs of type integer are contextually equivalent, and they both don't run forever, then they must both evaluate to the same number.

Now if τ was $\mathbb{B}$ instead, the context $\mathsf{if}\ [\cdot]\ \mathsf{then}\ ()\ \mathsf{else}\ \omega$ would suffice in showing that $v_2 = v_1$.

It becomes a little more difficult to reason about when τ is not a base-type, like $\mathbb{Z}$ or $\mathbb{B}$. Take function type, for instance. If e_1 and e_2 both evaluate down to functions, how do we know that those functions are behaviourally equivalent (in the sense of functional extensionality)? To see this, consider the following "congruence" rule:

$$\frac{\Xi \mid \Gamma \vdash f \approx^{ctx} f' : \tau \to \tau' \qquad \Xi \mid \Gamma \vdash t \approx^{ctx} t' : \tau}{\Xi \mid \Gamma \vdash f\ t \approx^{ctx} f'\ t' : \tau'}\ \textsc{Congruence-function-app}$$

This essentially gives us what we want: if we have two contextually equivalent functions, then, as long as we give them contextually equivalent inputs, the output will also be contextually equivalent. That output may be another function, but then this rule applies again to that output, and so on. In other words, we can keep on applying this rule until we at some point get a base type, like $\mathbb{Z}$, and at that point, we know that the two numbers will be the same, by the argument made above.
Of course, it may happen that the output is of another non-base type, such as type abstraction. But we may do something similar for all other non-base types. However proving all of them is quite tedious, so we will here only prove the congruence rule for function types.

Proof. By the two hypotheses of the rule, we know that $\Xi \mid \Gamma \vdash f : \tau \to \tau'$ and $\Xi \mid \Gamma \vdash t : \tau$. So by the T-APP rule, we may conclude $\Xi \mid \Gamma \vdash f\ t : \tau'$. Likewise for the prime variants. So all that remains to be shown is that $\forall C : (\Xi \mid \Gamma \vdash \tau') \Rightarrow (\bullet \mid \bullet \vdash \mathsf{Unit}).(C[f\ t] \Downarrow \iff C[f'\ t'] \Downarrow)$. So assume some context, C, of the right type. Consider now the context $C[[\cdot]\ t]$. We know that f is contextually equivalent to f', so $C[f\ t] \Downarrow \iff C[f'\ t] \Downarrow$. Now consider the context $C[f'[\cdot]]$. Since t is contextually equivalent to t', then we know $C[f'\ t] \Downarrow \iff C[f'\ t'] \Downarrow$. In other words, we have that $C[f\ t] \Downarrow \iff C[f'\ t] \iff C[f'\ t'] \Downarrow$, which was what we wanted. □

The way we have defined contextual equivalence is quite instructive and intuitive. However, one may define contextual equivalence in another, equivalent way, which gives much insight into contextual equivalence, and will make working with the logical relation later more intuitive. We will explore this alternative definition in the next section.

3.2 Alternative Definition of Contextual Equivalence

We may define contextual equivalence in the following way

Definition 3.2.1. Contextual Equivalence is a relation, $R \subseteq \Xi \times \Gamma \times e \times e \times \tau$, such that

- all expressions in R are well-typed
- R is a congruence relation with respect to the typing rules
- R is an adequate relation

Finally, it is the coarsest such relation. We shall denote this coarsest relation CE, and R will refer to any other relation satisfying the above three properties.

We introduce the following notation: $R(\Xi, \Gamma, e, e', \tau) \triangleq \Xi \mid \Gamma \vdash e \approx e' : \tau$,and use them interchangeably.
That CE is a congruence relation with respect to the typing rules means that it

satisfies all the congruence rules that arises from the typing rules in the same manner CONGRUENCE-FUNCTION-APP did. For instance, the congruence rules for T-unit and T-app would be:

$$\textsc{Congruence-unit}\ \frac{}{\Xi \mid \Gamma \vdash () \approx () : \mathsf{Unit}} \qquad \textsc{Congruence-type-app}\ \frac{\Xi \mid \Gamma \vdash e \approx e' : \forall X.\, \tau}{\Xi \mid \Gamma \vdash e\ \_ \approx e'\ \_ : \tau\left[\tau'/X\right]}$$

The remaining congruence rules can be found in the appendix (A.1).
The next part, CE being an adequate relation, essentially just means that it satisfies $\bullet \mid \bullet \vdash e \approx e' : \mathsf{Unit} \implies e \Downarrow \iff e' \Downarrow$.
Finally, the fact that CE is the coarsest such relation simply means the following: $\forall \Xi, \Gamma, e, e', \tau. R(\Xi, \Gamma, e, e', \tau) \implies CE(\Xi, \Gamma, e, e', \tau)$.
We of course have to show that this definition of Contextual Equivalence is actually equivalent to the one given in definition 3.1.1:

Theorem 3. $\Xi \mid \Gamma \vdash e \approx^{ctx} e' : \tau \iff CE(\Xi, \Gamma, e, e', \tau)$

We will show this in two steps, where each step essentially corresponds to one way of the double implication. Each step will be phrased as a theorem.

Theorem 4. $\Xi \mid \Gamma \vdash e \approx^{ctx} e' : \tau \subseteq \Xi \times \Gamma \times e \times e \times \tau$ *is a congruence relation, an adequate relation, and all expressions in the relation are well-typed.*

Proof. First, the well-typedness follows directly from the definition of contextual equivalence.
Second, let's show that it is an adequate relation. So assuming $\bullet \mid \bullet \vdash e \approx^{ctx} e' :$ Unit,we must show $e \Downarrow \iff e' \Downarrow$. We know that $\forall C : (\bullet \mid \bullet \vdash \mathsf{Unit}) \Rightarrow (\bullet \mid \bullet \vdash \mathsf{Unit}).(C[e] \Downarrow \iff C[e'] \Downarrow)$. So specifically for C being the empty context, we get $e \Downarrow \iff e' \Downarrow$, which was what we wanted.
To show that it is a congruence relation, we must go through all the congruence rules, and show that they hold. We did the one for function application above, CONGRUENCE-FUNCTION-APP. The rest will not be shown here. □

To prove the next theorem, we will need to show two lemmas.

Lemma 5 (Reflexivity of R)**.** $\Xi \mid \Gamma \vdash e : \tau \implies R(\Xi, \Gamma, e, e, \tau)$

Proof. By induction on the typing derivation $\Xi \mid \Gamma \vdash e : \tau$. We will only do a few cases here, as they are all very similar.

case T-var.
Assuming $\Xi \mid \Gamma \vdash x : \tau$ and $(x : \tau) \in \Gamma$, we must show $R(\Xi, \Gamma, x, x, \tau)$. Since $\Gamma(x) = \tau$, this follows from the congruency rule $Cgr - var$?? $Cgr - var$.

case T-unit.
Assuming $\Xi \mid \Gamma \vdash () : \mathsf{Unit}$, we must show $R(\Xi, \Gamma, (), (), \mathsf{Unit})$. This follows directly from the congruency rule $Cgr - unit$.

case T-add.

Assuming $\Xi \mid \Gamma \vdash e_1 + e_2 : \mathbb{Z}$, $\Xi \mid \Gamma \vdash e_1 : \mathbb{Z}$, and $\Xi \mid \Gamma \vdash e_2 : \mathbb{Z}$, we must show $R(\Xi, \Gamma, e_1 + e_2, e_1 + e_2, \mathbb{Z})$. By our induction hypotheses, we have $\Xi \mid \Gamma \vdash e_1 \approx e_1 : \mathbb{Z}$ and $\Xi \mid \Gamma \vdash e_2 \approx e_2 : \mathbb{Z}$. Now, the congruency rule $Cgr - add$ gives us $\Xi \mid \Gamma \vdash e_1 + e_2 \approx e_1 + e_2 : \mathbb{Z}$, which was what we wanted.

case T-tlam.

Assuming $\Xi \mid \Gamma \vdash \Lambda\ e : \forall X.\ \tau$ and $\Xi, X \mid \Gamma \vdash e : \tau$, we must show $R(\Xi, \Gamma, \Lambda\ e, \Lambda\ e, \forall X.\ \tau)$.By our induction hypothesis, we get $\Xi, X \mid \Gamma \vdash e \approx e : \tau$. By the congruency rule $Cgr{-}tlam$, we get $\Xi \mid \Gamma \vdash \Lambda\ e \approx \Lambda\ e : \forall X.\ \tau$, which was what we wanted.

□

Lemma 6. $R(\Xi, \Gamma, e, e', \tau) \ \wedge \ C : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi' \mid \Gamma' \vdash \tau') \implies R(\Xi', \Gamma', C[e], C[e'], \tau')$

Proof. ▶**do proof**◀ □

With this lemma proved, we are ready to prove the next theorem.

Theorem 7. $R(\Xi, \Gamma, e, e', \tau) \implies \Xi \mid \Gamma \vdash e \approx^{ctx} e' : \tau$.

Proof. So assuming $R(\Xi, \Gamma, e, e', \tau)$, we must show $\Xi \mid \Gamma \vdash e : \tau \ \wedge \ \Xi \mid \Gamma \vdash e' : \tau \ \wedge$
$\forall C : (\Xi \mid \Gamma \vdash \tau) \Rightarrow (\bullet \mid \bullet \vdash \mathsf{Unit}).(C[e] \Downarrow \iff C[e'] \Downarrow)$. Again, the well-typedness follows from the fact that all expressions in R are well-typed. So what remains to be shown is that $\forall C : (\Xi \mid \Gamma \vdash \tau) \Rightarrow (\bullet \mid \bullet \vdash \mathsf{Unit}).(C[e] \Downarrow \iff C[e'] \Downarrow)$. So assume some context C of the right type. By lemma 6, we have $R(\bullet, \bullet, C[e], C[e'], \mathsf{Unit})$. By adequacy of R, we have that $C[e] \Downarrow \iff C[e'] \Downarrow$, which was what we wanted. □

Now, the proof of the two definitions of contextual equivalence being equivalent follows easily:

Proof of theorem 3. By theorem 4 and from the fact that CE is the coarsest relation satisfying the three properties, it follows that $\Xi \mid \Gamma \vdash e \approx^{ctx} e' : \tau \implies CE(\Xi, \Gamma, e, e', \tau)$.
By theorem 7 and from the fact that CE is one such relation R, it follows that $CE(\Xi, \Gamma, e, e', \tau) \implies \Xi \mid \Gamma \vdash e \approx^{ctx} e' : \tau$ □

Chapter 4

Logical Relations Model for Contextual Equivalence

▶**draft**◀

Logical relations model

Value interpretation

$$
\begin{aligned}
\mathcal{V}[\![\tau]\!]_\rho(v_1, v_2) &\triangleq \bullet \mid \bullet \vdash v_1 : \rho_1(\tau) \wedge \bullet \mid \bullet \vdash v_2 : \rho_2(\tau) \wedge \ldots \\
\mathcal{V}[\![\mathsf{Unit}]\!]_\rho(v_1, v_2) &\triangleq (v_1, v_2) \in \{((), ())\} \\
\mathcal{V}[\![\mathbb{Z}]\!]_\rho(v_1, v_2) &\triangleq (v_1, v_2) \in \{(\overline{n}, \overline{n}) \mid \overline{n} \in \mathbb{Z}\} \\
\mathcal{V}[\![\mathbb{B}]\!]_\rho(v_1, v_2) &\triangleq (v_1, v_2) \in \{(\mathsf{true}, \mathsf{true}), (\mathsf{false}, \mathsf{false})\} \\
\mathcal{V}[\![\tau_1 \times \tau_2]\!]_\rho(v_1, v_2) &\triangleq (v_1, v_2) \in \{((w_{11}, w_{12}), (w_{21}, w_{22})) \mid \mathcal{V}[\![\tau_1]\!]_\rho(w_{11}, w_{21}) \wedge \mathcal{V}[\![\tau_2]\!]_\rho(w_{12}, w_{22})\} \\
\mathcal{V}[\![\tau_1 + \tau_2]\!]_\rho(v_1, v_2) &\triangleq (v_1, v_2) \in \{(\mathsf{inj}_1\ w_1, \mathsf{inj}_1\ w_2) \mid \mathcal{V}[\![\tau_1]\!]_\rho(w_1, w_2)\} \vee \\
&\qquad (v_1, v_2) \in \{(\mathsf{inj}_2\ w_1, \mathsf{inj}_2\ w_2) \mid \mathcal{V}[\![\tau_2]\!]_\rho(w_1, w_2)\} \\
\mathcal{V}[\![\tau_1 \to \tau_2]\!]_\rho(v_1, v_2) &\triangleq (v_1, v_2) \in \{(\mathsf{rec}\ f_1(x_1) := e_1, \mathsf{rec}\ f_2(x_2) := e_2) \mid \forall w_1, w_2. \mathcal{V}[\![\tau_1]\!]_\rho(w_1, w_2) \\
&\qquad \implies \mathcal{E}[\![\tau_2]\!]_\rho(e_1[w_1/x_1], e_2[w_2/x_2])\} \\
\mathcal{V}[\![\forall X.\ \tau]\!]_\rho(v_1, v_2) &\triangleq (v_1, v_2) \in \{(\Lambda\ e_1, \Lambda\ e_2) \mid \forall \tau_1, \tau_2, R \in Rel[\tau_1, \tau_2]. \\
&\qquad \mathcal{E}[\![\tau]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]}(e_1[\tau_1/X], e_2[\tau_2/X])\} \\
\mathcal{V}[\![X]\!]_\rho(v_1, v_2) &\triangleq (v_1, v_2) \in \rho_R[X]
\end{aligned}
$$

Relational substitution

$$
Rel[\tau_1, \tau_2] \triangleq \left\{ R \;\middle|\; \begin{array}{l} R \in \mathcal{P}(Val \times Val) \wedge \bullet \mid \bullet \vdash \tau_1 \wedge \bullet \mid \bullet \vdash \tau_1 \wedge \\ \forall (v_1, v_2) \in R. \bullet \mid \bullet \vdash v_1 : \tau_1 \wedge \bullet \mid \bullet \vdash v_2 : \tau_2 \end{array} \right\}
$$

Expression interpretation

$$
\begin{aligned}
\mathcal{E}[\![\tau]\!]_\rho(e_1, e_2) \triangleq \bullet \mid \bullet \vdash e_1 : \rho_1(\tau) \wedge \bullet \mid \bullet \vdash e_2 : \rho_2(\tau) \wedge e_1 \Downarrow &\iff e_2 \Downarrow \wedge \\
(\forall v_1, v_2. e_1 \Downarrow v_1 \wedge e_2 \Downarrow v_2 &\implies \mathcal{V}[\![\tau]\!]_\rho(v_1, v_2))
\end{aligned}
$$

Variable environment interpretation

$$\mathcal{G}[\![\bullet]\!]_\rho \triangleq \{\emptyset\}$$
$$\mathcal{G}[\![\Gamma, x:\tau]\!]_\rho \triangleq \{\gamma[x \mapsto (v_1, v_2)] \mid \gamma \in \mathcal{G}[\![\Gamma]\!]_\rho \wedge \mathcal{V}[\![\tau]\!]_\rho(v_1, v_2)\}$$

Type environment interpretation

$$\mathcal{D}[\![\bullet]\!] \triangleq \{\emptyset\}$$
$$\mathcal{D}[\![\Xi, X]\!] \triangleq \{\rho[X \mapsto (\tau_1, \tau_2, R)] \mid \rho \in \mathcal{D}[\![\Xi]\!] \wedge R \in Rel[\tau_1, \tau_2]\}$$

Definition 4.0.1 (Logical relation)**.**

$$\Xi \mid \Gamma \vdash e \approx^{LR} e' : \tau$$
$$\iff$$
$$\Xi \mid \Gamma \vdash e : \tau \ \wedge \ \Xi \mid \Gamma \vdash e' : \tau \ \wedge$$
$$\forall \rho \in \mathcal{D}[\![\Xi]\!], \gamma \in \mathcal{G}[\![\Gamma]\!]_\rho . \mathcal{E}[\![\tau]\!]_\rho(\rho_1(\gamma_1(e)), \rho_2(\gamma_2(e'))))$$

Compatibility lemmas Essentially just congruency rules as stated in appendix (but with LR being the relation). We have to prove these (and adequacy) to show that LR ==> CE, meaning we can state theorems in terms of LR instead of CE, and then get CE as a consequence.

Lemma 8 (Cmpt-unit)**.** $\dfrac{}{\Xi \mid \Gamma \vdash () \approx^{LR} () : \mathsf{Unit}}$

Proof. By definition 4.0.1, we must show $\Xi \mid \Gamma \vdash () : \mathsf{Unit} \wedge \Xi \mid \Gamma \vdash () : \mathsf{Unit} \wedge \forall \rho \in \mathcal{D}[\![\Xi]\!], \gamma \in \mathcal{G}[\![\Gamma]\!]_\rho . \mathcal{E}[\![\mathsf{Unit}]\!]_\rho(\rho_1(\gamma_1(()))), \rho_2(\gamma_2(()))))$. The well-typedness simply follows by the typing rule T-unit.
So assume some $\rho \in \mathcal{D}[\![\Xi]\!]$ and $\gamma \in \mathcal{G}[\![\Gamma]\!]_\rho$, we must show
$\mathcal{E}[\![\mathsf{Unit}]\!]_\rho(\rho_1(\gamma_1(()))), \rho_2(\gamma_2(())))) \equiv \mathcal{E}[\![\mathsf{Unit}]\!]_\rho((), ())$. Thus, we must show four things

1. $\bullet \mid \bullet \vdash () : \rho_1(\mathsf{Unit})$
2. $\bullet \mid \bullet \vdash () : \rho_2(\mathsf{Unit})$
3. $() \Downarrow \iff () \Downarrow$
4. $\forall v_1, v_2 . () \Downarrow v_1 \wedge () \Downarrow v_2 \implies \mathcal{V}[\![\mathsf{Unit}]\!]_\rho(v_1, v_2))$

Given that $\rho_1(\mathsf{Unit}) = \rho_2(\mathsf{Unit}) = \mathsf{Unit}$, the well-typedness once again simply follows from typing rule T-unit. (3) holds trivially. So let's show (4). Assume some v_1, v_2, and $() \Downarrow v_1$ and $() \Downarrow v_2$, then we must show $\mathcal{V}[\![\mathsf{Unit}]\!]_\rho(v_1, v_2)$. Since () is already a value, we have $v_1 = v_2 = ()$, meaning we must show $\mathcal{V}[\![\mathsf{Unit}]\!]_\rho((), ())$. Once again, the well-typedness holds by T-unit, and we have that $((), ()) \in \{((), ())\}$, so we are done. □

Lemma 9 (Cmpt-add)**.** $\dfrac{\Xi \mid \Gamma \vdash e_1 \approx^{LR} e_1' : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 \approx^{LR} e_2' : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 + e_2 \approx^{LR} e_1' + e_2' : \mathbb{Z}}$

Proof. ►**Write in proof**◄ □

Chapter 5

Examples of Application of Contextual Equivalence

►draft◄

Chapter 6

Comparison to Other Work and Ideas for Future Work

►draft◄

Chapter 7

Conclusion

►conclude on the problem statement from the introduction◄

Acknowledgments

►. . . ◄

Bibliography

[1] Aske Simon Christensen, Anders Møller, and Michael I. Schwartzbach. Precise analysis of string expressions. In Radhia Cousot, editor, *Static Analysis, 10th International Symposium, SAS 2003, San Diego, CA, USA, June 11-13, 2003, Proceedings*, volume 2694 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003.

Appendix A

The Technical Details

►. . . ◄

A.1 Congruence Rules for Alternative Definition of Contextual Equivalence

CGR-VAR ▶CORRECT?◀
$$\frac{(x:\tau) \in \Gamma}{\Xi \mid \Gamma \vdash x \approx x : \tau}$$

CGR-VAR2 ▶OR THIS?◀
$$\frac{}{\Xi \mid \Gamma \vdash x \approx x : \Gamma(x)}$$

CGR-UNIT
$$\frac{}{\Xi \mid \Gamma \vdash () \approx () : \mathsf{Unit}}$$

CGR-INT
$$\frac{}{\Xi \mid \Gamma \vdash \overline{n} \approx \overline{n} : \mathbb{Z}}$$

CGR-ADD
$$\frac{\Xi \mid \Gamma \vdash e_1 \approx e_1' : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 \approx e_2' : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 + e_2 \approx e_1' + e_2' : \mathbb{Z}}$$

CGR-SUB
$$\frac{\Xi \mid \Gamma \vdash e_1 \approx e_1' : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 \approx e_2' : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 - e_2 \approx e_1' - e_2' : \mathbb{Z}}$$

CGR-LE
$$\frac{\Xi \mid \Gamma \vdash e_1 \approx e_1' : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 \approx e_2' : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 \leq e_2 \approx e_1' \leq e_2' : \mathbb{B}}$$

CGR-LT
$$\frac{\Xi \mid \Gamma \vdash e_1 \approx e_1' : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 \approx e_2' : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 < e_2 \approx e_1' < e_2' : \mathbb{B}}$$

CGR-EQ
$$\frac{\Xi \mid \Gamma \vdash e_1 \approx e_1' : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 \approx e_2' : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 = e_2 \approx e_1' = e_2' : \mathbb{B}}$$

CGR-TRUE
$$\frac{}{\Xi \mid \Gamma \vdash \mathsf{true} \approx \mathsf{true} : \mathbb{B}}$$

CGR-FALSE
$$\frac{}{\Xi \mid \Gamma \vdash \mathsf{false} \approx \mathsf{false} : \mathbb{B}}$$

CGR-IF
$$\frac{\Xi \mid \Gamma \vdash e_1 \approx e_1' : \mathbb{B} \qquad \Xi \mid \Gamma \vdash e_2 \approx e_2' : \tau \qquad \Xi \mid \Gamma \vdash e_3 \approx e_3' : \tau}{\Xi \mid \Gamma \vdash \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \approx \mathsf{if}\ e_1'\ \mathsf{then}\ e_2'\ \mathsf{else}\ e_3' : \tau}$$

CGR-PAIR
$$\frac{\Xi \mid \Gamma \vdash e_1 \approx e_1' : \tau_1 \qquad \Xi \mid \Gamma \vdash e_2 \approx e_2' : \tau_2}{\Xi \mid \Gamma \vdash (e_1, e_2) \approx (e_1', e_2') : \tau_1 \times \tau_2}$$

CGR-FST
$$\frac{\Xi \mid \Gamma \vdash e \approx e' : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \mathsf{fst}\ e \approx \mathsf{fst}\ e' : \tau_1}$$

CGR-SND
$$\frac{\Xi \mid \Gamma \vdash e \approx e' : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \mathsf{snd}\ e \approx \mathsf{snd}\ e' : \tau_2}$$

CGR-INJ1
$$\frac{\Xi \mid \Gamma \vdash e \approx e' : \tau_1}{\Xi \mid \Gamma \vdash \mathsf{inj}_1\ e \approx \mathsf{inj}_1\ e' : \tau_1 + \tau_2}$$

CGR-INJ2
$$\frac{\Xi \mid \Gamma \vdash e \approx e' : \tau_2}{\Xi \mid \Gamma \vdash \mathsf{inj}_2\ e \approx \mathsf{inj}_2\ e' : \tau_1 + \tau_2}$$

CGR-MATCH

$$\frac{\Xi \mid \Gamma \vdash e_1 \approx e_1' : \tau_1 + \tau_2 \qquad \Xi \mid \Gamma, x : \tau_1 \vdash e_2 \approx e_2' : \tau \qquad \Xi \mid \Gamma, x : \tau_1 \vdash e_3 \approx e_3' : \tau}{\Xi \mid \Gamma \vdash \mathsf{match}\ e_1\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e_2 \mid \mathsf{inj}_2\ x \Rightarrow e_3\ \mathsf{end} \approx \mathsf{match}\ e_1'\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e_2' \mid \mathsf{inj}_2\ x \Rightarrow e_3'\ \mathsf{end} : \tau}$$

CGR-REC

$$\frac{\Xi \mid \Gamma, f : \tau_1 \to \tau_2, x : \tau_1 \vdash e \approx e' : \tau_2}{\Xi \mid \Gamma \vdash \mathsf{rec}\ f(x) := e \approx \mathsf{rec}\ f(x) := e' : \tau_1}$$

CGR-APP

$$\frac{\Xi \mid \Gamma \vdash e_1 \approx e_1' : \tau_1 \to \tau_2 \qquad \Xi \mid \Gamma \vdash e_2 \approx e_2' : \tau_1}{\Xi \mid \Gamma \vdash e_1\ e_2 \approx e_1'\ e_2' : \tau_2}$$

CGR-TLAM

$$\frac{\Xi, X \mid \Gamma \vdash e \approx e' : \tau}{\Xi \mid \Gamma \vdash \Lambda\ e \approx \Lambda\ e' : \forall X.\ \tau}$$

CGR-TAPP

$$\frac{\Xi \mid \Gamma \vdash e \approx e' : \forall X.\ \tau}{\Xi \mid \Gamma \vdash e\ \_ \approx e'\ \_ : \tau\left[\tau'/X\right]}$$