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◀
To talk about contextual equivalence of programs, we first need to define a language, which captures what programs can be. Consequently, the choice of language influences the definition of contextual equivalence. We will specifically how this manifests when we reach chapter 3. In this chapter, we will focus on formally defining the language, and showing some facts about it, which will be useful in later chapters.

2.1 The language

We define the language in three parts: the syntax, the typing rules, and finally the dynamics (semantics). The language we will be working with is generally refereed to as *System F*, and we shall do so here as well. However, the features included in this language may be different from other presentations. We note here that we will be working with a Curry-style language, as opposed to a Church-style language. This essentially just means that types are not an intrinsic part of the semantics. Instead, types become a property that programs can have. Thus, we may attempt to evaluate a program even though it has type-errors.

2.1.1 Syntax

The syntax captures exactly all programs that can be written in our language. We define it in Backus-Naur form as follows:

$$
\begin{aligned}
e ::= & () \mid && \text{(unit value)}\\
& x \mid && \text{(variables)}\\
& \overline{n} \mid e + e \mid e - e \mid e \leq e \mid e < e \mid e = e \mid && \text{(integers)}\\
& \mathsf{true} \mid \mathsf{false} \mid \mathsf{if}\ e\ \mathsf{then}\ e\ \mathsf{else}\ e \mid && \text{(booleans)}\\
& (e, e) \mid \mathsf{fst}\ e \mid \mathsf{snd}\ e \mid && \text{(products)}\\
& \mathsf{inj}_1\ e \mid \mathsf{inj}_2\ e \mid \mathsf{match}\ e\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e \mid \mathsf{inj}_2\ x \Rightarrow e\ \mathsf{end} \mid && \text{(sums)}\\
& \lambda x.\ e \mid e\ e && \text{(functions)}\\
& \Lambda\ e \mid e\ \_ && \text{(polymorphism)}\\
v ::= & () \mid \overline{n} \mid \mathsf{true} \mid \mathsf{false} \mid (v, v) \mid \mathsf{inj}_1\ v \mid \mathsf{inj}_2\ v \mid \lambda x.\ e \mid \Lambda\ e && \text{(values)}\\
\tau ::= & \mathsf{Unit} \mid \mathbb{Z} \mid \mathbb{B} \mid \tau \times \tau \mid \tau + \tau \mid \tau \to \tau \mid \forall X.\ \tau && \text{(types)}\\
K ::= & [] \mid K + e \mid v + K \mid K - e \mid v - K \mid K \leq e \mid v \leq K \mid K < e \mid v < K \mid && \text{(evaluation context)}\\
& K = e \mid v = K \mid \mathsf{if}\ K\ \mathsf{then}\ e\ \mathsf{else}\ e \mid (K, e) \mid (v, K) \mid \mathsf{fst}\ K \mid \mathsf{snd}\ K \mid\\
& \mathsf{inj}_1\ K \mid \mathsf{inj}_2\ K \mid \mathsf{match}\ K\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e \mid \mathsf{inj}_2\ x \Rightarrow e\ \mathsf{end} \mid K\ e \mid v\ K \mid K\ \_
\end{aligned}
$$

Thus, all programs in our language can be derived using the rules above. Evaluation contexts will be used when we get to defining the dynamics. Note that programs in our language may not necessarily make "sense". For example, using the rules above, we can construct the program $\mathsf{true} + 1$. To avert this problem, we introduce the typing rules, which will allow us to only talk about well-typed programs.

2.1.2 Typing rules

The typing rules gives us a way to derive the type of a program in our language. They are defined using inference rules as follows:

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

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

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

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

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

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

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

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

$$\text{T-LAM}\ \frac{\Xi \mid \Gamma, x : \tau_1 \vdash e : \tau_2}{\Xi \mid \Gamma \vdash \lambda x.\ e : \tau_1 \to \tau_2} \qquad \text{T-APP}\ \frac{\Xi \mid \Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Xi \mid \Gamma \vdash e_2 : \tau_1}{\Xi \mid \Gamma \vdash e_1\ e_2 : \tau_2}$$

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

2.1.3 Dynamics

Finally, we define the semantics of the language. We do this using a single "step" rule, and several "head-step" rules. The head-step rules govern all possible

reductions, and the "head-step-step" rule tells us where in our program we can apply these reductions.

$$\text{HEAD-STEP-STEP}\ \frac{e \to_h e'}{K[e] \to K[e']}$$

The evaluation context, K, tells us where in our program we may apply the reduction. If we have an expression e, and we have $e = K[f]$, and $f \to_h f'$, then we may conclude that $e \to K[f']$. Now, the head-steps are as follows:

$$\text{E-ADD}\ \overline{n_1} + \overline{n_2} \to_h \overline{n_1 + n_2} \qquad \text{E-SUB}\ \overline{n_1} - \overline{n_2} \to_h \overline{n_1 - n_2} \qquad \text{E-EQ}\ \frac{n_1 = n_2}{\overline{n_1} = \overline{n_2} \to_h \mathsf{true}}$$

$$\text{E-NOT-EQ}\ \frac{n_1 \neq n_2}{\overline{n_1} = \overline{n_2} \to_h \mathsf{false}} \qquad \text{E-LE}\ \frac{n_1 \leq n_2}{\overline{n_1} \leq \overline{n_2} \to_h \mathsf{true}} \qquad \text{E-NOT-LE}\ \frac{n_1 \not\leq n_2}{\overline{n_1} \leq \overline{n_2} \to_h \mathsf{false}}$$

$$\text{E-LT}\ \frac{n_1 < n_2}{\overline{n_1} < \overline{n_2} \to_h \mathsf{true}} \qquad \text{E-NOT-LT}\ \frac{n_1 \not< n_2}{\overline{n_1} < \overline{n_2} \to_h \mathsf{false}} \qquad \text{E-IF-TRUE}\ \mathsf{if\ true\ then}\ e_2\ \mathsf{else}\ e_3 \to_h e_2$$

$$\text{E-IF-FALSE}\ \mathsf{if\ false\ then}\ e_2\ \mathsf{else}\ e_3 \to_h e_3 \qquad \text{E-FST}\ \mathsf{fst}\ (v_1, v_2) \to_h v_1 \qquad \text{E-SND}\ \mathsf{snd}\ (v_1, v_2) \to_h v_2$$

$$\text{E-MATCH-INJ1}\ \mathsf{match}\ (\mathsf{inj}_1\ v)\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e_2 \mid \mathsf{inj}_2\ x \Rightarrow e_3\ \mathsf{end} \to_h e_2[v/x]$$

$$\text{E-MATCH-INJ2}\ \mathsf{match}\ (\mathsf{inj}_2\ v)\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e_2 \mid \mathsf{inj}_2\ x \Rightarrow e_3\ \mathsf{end} \to_h e_3[v/x]$$

$$\text{E-LAM-APP}\ (\lambda x.\ e)\ v \to_h e[v/x] \qquad \text{E-TAPP-TLAM}\ (\Lambda\ e)\ \_ \to_h e$$

At this point, our language has been formally been defined, so next we will note down some facts and results about it.

2.2 Properties of the language

In this section we note down some useful properties of our language, which will help us reason about the concepts in later chapters. First we show some interesting lemmas about the evaluation context.

2.2.1 Evaluation Context

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 $\text{Val}(e)$, as wanted. □

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

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

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

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

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

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

□

Corollary 2.1. $K[e] \to^* v \implies \exists e''.(e \to^* e'') \wedge (\text{Val}(e'') \wedge K[e''] \to^* v)$

Proof. Assuming $K[e] \to^* v$, we must show $\exists e''.(e \to^* e'') \wedge (\mathrm{Val}(e'') \wedge K[e''] \to^* v)$. By lemma 2, we know $\exists e''.(e \to^* e'') \wedge ((\mathrm{Val}(e'') \wedge K[e''] \to^* v) \vee (\neg\mathrm{Val}(e'') \wedge K[e''] = v))$. Now, if $\neg\mathrm{Val}(e'')$ then $\neg\mathrm{Val}(K[e''])$, which one may see simply by inspecting the possible evaluation contexts. Therefore, $K[e''] \neq v$, so $\neg(\neg\mathrm{Val}(e'') \wedge K[e''] = v)$. Thus, we conclude $\exists e''.(e \to^* e'') \wedge (\mathrm{Val}(e'') \wedge K[e''] \to^* v)$. □

2.2.2 substitution

We define a substitution $\gamma = \{x_1 \mapsto v_1, x_2 \mapsto v_2, \dots\}$ inductively as follows:

$$\emptyset(e) = e$$
$$\gamma[x \mapsto v](e) = \gamma(e[v/x])$$

Lemma 3 (Substitution)**.** $\gamma[x \mapsto v](e) = \gamma(e)[v/x]$

Proof. ▶**proper ref**◀ Lemma 10 in notes by Lau. □

2.2.3 Normalization

▶**expand a little on this topic**◀

Theorem 4. *System F is a normalizing language.*

This won't be proved here.

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

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

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

$$\Xi \mid \Gamma \vdash e_1 \approx^{ctx} e_2 : \tau$$
$$\iff$$
$$\Xi \mid \Gamma \vdash e_1 : \tau \quad \wedge \quad \Xi \mid \Gamma \vdash e_2 : \tau \quad \wedge$$
$$\forall C : (\Xi \mid \Gamma \vdash \tau) \Rightarrow (\bullet \mid \bullet \vdash \mathbb{B}), v.(C[e_1] \Downarrow v \iff C[e_2] \Downarrow v)$$

The way we have defined contextual equivalence here means that given two programs, e_1, e_2 that are contextually equivalent, if we plug them into any context, C, which becomes closed and of type $\mathbb{B}$, then one program, $C[e_1]$ terminates to some value v if and only if the other program $C[e_2]$ terminates to the same value, v. Note here that that value must be a boolean, i.e. either true or false.

One may ponder why we have decided to use $\mathbb{B}$ as the type of the plugged programs; there are plenty other types that could have been used instead such as Unit or $\mathbb{Z}$. Why not use those? In fact, we could have used some of them. The important part is that we are able to make the plugged programs elicit differing behaviour, as we will se below. We cannot do this if the plugged programs have type Unit; by theorem 4 we know that both plugged programs terminate, and there is only one value with type Unit, namely (), so both programs will terminate at () regardless of whether or not e_1 and e_2 are behaviourally equivalent. Had we been dealing with a non-terminating language, then we could have used Unit, since it would then have two possible differing behaviours: terminate with (), or not terminate. In that case, we would define contextual equivalence as $C[e_1] \Downarrow \iff C[e_2] \Downarrow$.

Another cause for speculation is whether or not the definition is really strong enough. If both e_1 and e_2 reduce to some values, we really want the values to be behaviourally the same. If e_1 and e_2 are of integer type, for example, then 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 and ensures the above examples, consider two contextually equivalent programs e_1 and e_2, where $\bullet \mid \bullet \vdash e_1 : \tau$ and $\bullet \mid \bullet \vdash e_2 : \tau$. 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 at some value v if and only if the other one does. Let's first consider when $\tau = \mathbb{Z}$. Then consider when C has the form if $[\cdot] = v_1$ then true else false. Here $C[e_1] \Downarrow$ true. 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 $C[e_2] \Downarrow$ false. However, since e_1 and e_2 are contextually equivalent, and $C[e_1]$ reduces to true, then we know that $C[e_2]$ must reduce to

true. 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.

It becomes a little more difficult to reason about when τ is not a base-type, like Unit, $\mathbb{Z}$, 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? To see this, consider the following "congruence" rule:

$$\frac{\Xi \mid \Gamma \vdash f \approx^{ctx} f' : \tau \to \tau' \qquad \Xi \mid \Gamma \vdash t \approx^{ctx} t' : \tau}{\Xi \mid \Gamma \vdash f\ t \approx^{ctx} f'\ t' : \tau'} \text{CNG-CTX-APP}$$

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 application.

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 $\forall C : (\Xi \mid \Gamma \vdash \tau') \Rightarrow (\bullet \mid \bullet \vdash \mathbb{B}), v.(C[f\ t] \Downarrow v \iff C[f'\ t'] \Downarrow v)$. So assume some context, C, of the right type, and some value v. Consider now the context $C[[\cdot]\ t]$. We know that f is contextually equivalent to f', so $\forall v'.C[f\ t] \Downarrow v' \iff C[f'\ t] \Downarrow v'$. Now consider the context $C[f'[\cdot]]$. Since t is contextually equivalent to t', then we know $\forall v''.C[f'\ t] \Downarrow v'' \iff C[f'\ t'] \Downarrow v''$. Now it follows by instantiating both of the prior results with v. Then we have $C[f\ t] \Downarrow v \iff C[f'\ t] \Downarrow v \iff C[f'\ t'] \Downarrow v$, 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

Before giving the alternative definition, we first introduce a few concepts. In the following, we shall let R denote any relation over Ξ, Γ, e, e, τ. I.e. $R \subseteq \Xi \times \Gamma \times e \times e \times \tau$. We also introduce the following notation: $R(\Xi, \Gamma, e, e', \tau) \triangleq$

$\Xi \mid \Gamma \vdash e \approx e' : \tau$, and use them interchangeably.
Now we define two properties that a relation R may have.

Definition 3.2.1 (Adequacy)**.** We say R is an adequate relation when it holds for R that $\bullet \mid \bullet \vdash e \approx e' : \mathbb{B} \implies e \Downarrow v \iff e' \Downarrow v$.

Definition 3.2.2 (Congruency)**.** We say R is a congruence relation (with respect to the typing rules) when it satisfies all the congruence rules that arises from the typing rules.

For instance, the congruence rules for T-unit and T-app would be:

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

The remaining congruence rules can be found in the appendix (A.1).
Now, we may define contextual equivalence in the following way.

Definition 3.2.3. Contextual Equivalence is a relation, $CE \subseteq \Xi \times \Gamma \times e \times e \times \tau$, such that

- all expressions in CE are well-typed
- CE is a congruence relation
- CE is an adequate relation

Finally, it is the coarsest such relation.

CE being the coarsest such relation means that if given a relation R satisfying the three properties, then $\forall \Xi, \Gamma, e, e', \tau. R(\Xi, \Gamma, e, e', \tau) \implies CE(\Xi, \Gamma, e, e', \tau)$.
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 5. $\Xi \mid \Gamma \vdash e \approx^{ctx} e' : \tau \iff CE(\Xi, \Gamma, e, e', \tau)$

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 6. $\Xi \mid \Gamma \vdash e \approx^{ctx} e' : \tau \subseteq \Xi \times \Gamma \times e \times e \times \tau$ *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 $\bullet \mid \bullet \vdash e \approx^{ctx} e' : \mathbb{B}$, we must show $e \Downarrow v \iff e' \Downarrow v$. We know that $\forall C : (\bullet \mid \bullet \vdash \mathbb{B}) \Rightarrow (\bullet \mid \bullet \vdash \mathbb{B}), v'.(C[e] \Downarrow v' \iff C[e'] \Downarrow v')$. So specifically for C being the empty context and v' being v, we get $e \Downarrow v \iff e' \Downarrow v$, 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, Cng-ctx-app. The rest will not be shown here. □

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

Lemma 7 (Reflexivity)**.** *Let R be a congruence relation. Then* $\Xi \mid \Gamma \vdash e : \tau \implies R(\Xi, \Gamma, e, e, \tau)$.

Proof. By induction on the typing derivation of e. It follows immediately from applying the induction hypothesis and applying congruency rules. We show one case here.

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 $Cng - tlam$, we then get $\Xi \mid \Gamma \vdash \Lambda\ e \approx \Lambda\ e : \forall X.\ \tau$, which was what we wanted.

□

Lemma 8. *Let R be a congruence relation, then* $R(\Xi, \Gamma, e, e', \tau) \wedge C : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi' \mid \Gamma' \vdash \tau') \implies R(\Xi', \Gamma', C[e], C[e'], \tau')$

Proof. By induction on C. We show here only a few interesting cases. For each non-base-case we show, the induction hypothesis will be $\forall \Xi, \Gamma, \Xi', \Gamma', e, e', \tau, \tau'.$ $R(\Xi, \Gamma, e, e', \tau) \wedge C' : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi' \mid \Gamma' \vdash \tau') \implies R(\Xi', \Gamma', C'[e], C'[e'], \tau')$, where C' is structurally smaller than C.

case $C = [\cdot]$.
So assuming $R(\Xi, \Gamma, e, e', \tau)$ and $C : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi' \mid \Gamma' \vdash \tau')$, we must show $R(\Xi', \Gamma', C[e], C[e'], \tau')$. We may conclude that $\Xi = \Xi'$, $\Gamma = \Gamma'$, and $\tau = \tau'$, given that $[\cdot] : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi \mid \Gamma \vdash \tau)$. So since $e = [\cdot][e] = C[e]$, and $e' = [\cdot][e'] = C[e']$, we get $R(\Xi', \Gamma', C[e], C[e'], \tau')$, which was what we wanted.

case $C = C' + e''$
So assuming $R(\Xi, \Gamma, e, e', \tau)$ and $C : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi' \mid \Gamma' \vdash \tau')$, we must show $R(\Xi', \Gamma', C[e], C[e'], \tau')$. In this case, $C : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi' \mid \Gamma' \vdash \mathbb{Z})$ and $C' : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi \mid \Gamma \vdash \mathbb{Z})$, meaning $\tau' = \mathbb{Z}$. So we have $R(\Xi, \Gamma, e, e', \mathbb{Z})$, and our goal then is $R(\Xi', \Gamma', C[e], C[e'], \mathbb{Z})$. From the context typing and from rule T-add, we may conclude $\Xi' \mid \Gamma' \vdash e'' : \mathbb{Z}$. So by lemma 7, we know that $\Xi' \mid \Gamma' \vdash e'' \approx e'' : \mathbb{Z}$. And by I.H. we get $R(\Xi', \Gamma', C'[e], C'[e'], \mathbb{Z}) \equiv \Xi' \mid \Gamma' \vdash C'[e] \approx C'[e'] : \mathbb{Z}$. Finally, from the congruence rule cng-add, we get $\Xi' \mid \Gamma' \vdash C'[e] + e'' \approx C'[e'] + e'' : \mathbb{Z} \equiv \Xi' \mid \Gamma' \vdash C[e] \approx C[e'] : \mathbb{Z}$, which was what we wanted.

case $C = \mathsf{fst}\ C'$
Then $C : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi' \mid \Gamma' \vdash \tau_1)$ and $C' : (\Xi \mid \Gamma \vdash \tau \Rightarrow$

$\Xi' \mid \Gamma' \vdash \tau_1 \times \tau_2$), so our goal is $R(\Xi', \Gamma', C[e], C[e'], \tau_1)$. By I.H. we get $\Xi' \mid \Gamma' \vdash C'[e] \approx C'[e'] : \tau_1 \times \tau_2$, and by cng-fst with this, we get $\Xi' \mid \Gamma' \vdash \mathsf{fst}\ C'[e] \approx \mathsf{fst}\ C'[e'] : \tau_1 \equiv \Xi' \mid \Gamma' \vdash C[e] \approx C[e'] : \tau_1$.

case $C = \lambda x.\ C'$

Then $C : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi' \mid \Gamma' \vdash \tau_1 \to \tau_2)$ and $C' : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi' \mid \Gamma', x : \tau_1 \vdash \tau_2)$, so our goal is $R(\Xi', \Gamma', C[e], C[e'], \tau_1 \to \tau_2)$. By I.H. we get $\Xi' \mid \Gamma', x : \tau_1 \vdash C'[e] \approx C'[e'] : \tau_2$, and by cng-lam with this, we get $\Xi' \mid \Gamma' \vdash \lambda x.\ C'[e] \approx \lambda x.\ C'[e'] : \tau_1 \to \tau_2 \equiv \Xi' \mid \Gamma' \vdash C[e] \approx C[e'] : \tau_1 \to \tau_2$.

case $C = C'\ e''$

Then $C : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi' \mid \Gamma' \vdash \tau_2)$ and $C' : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi' \mid \Gamma' \vdash \tau_1 \to \tau_2)$, so our goal is $R(\Xi', \Gamma', C[e], C[e'], \tau_2)$. From the context-typing and T-app, we have $\Xi' \mid \Gamma' \vdash e'' : \tau_1$. By lemma 7, we get $\Xi' \mid \Gamma' \vdash e'' \approx e'' : \tau_1$. Now, by the induction hypothesis, we get $\Xi' \mid \Gamma' \vdash C'[e] \approx C'[e'] : \tau_1 \to \tau_2$. And by cng-app with this, we get $\Xi' \mid \Gamma' \vdash C'[e]\ e'' \approx C'[e']\ e'' : \tau_2 \equiv \Xi' \mid \Gamma' \vdash C[e] \approx C[e] : \tau_2$.

case $C = \Lambda\ C'$

Then $C : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi' \mid \Gamma' \vdash \forall X.\ \tau')$ and $C' : (\Xi \mid \Gamma \vdash \tau \Rightarrow \Xi', X \mid \Gamma' \vdash \tau')$, so our goal is $R(\Xi', \Gamma', C[e], C[e'], \forall X.\ \tau')$. By I.H. we get $\Xi', X \mid \Gamma' \vdash C'[e] \approx C'[e'] : \tau'$, and by cng-tlam with this, we get $\Xi' \mid \Gamma' \vdash \Lambda\ C'[e] \approx \Lambda\ C'[e'] : \forall X.\ \tau' \equiv \Xi' \mid \Gamma' \vdash C[e] \approx C[e'] : \forall X.\ \tau'$.

□

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

Theorem 9. *Let R be a relation satisfying the three properties of definition 3.2.3, then $R(\Xi, \Gamma, e, e', \tau) \implies \Xi \mid \Gamma \vdash e \approx^{ctx} e' : \tau$.*

Proof. So assuming $R(\Xi, \Gamma, e, e', \tau)$, we must show $\Xi \mid \Gamma \vdash e : \tau \ \wedge\ \Xi \mid \Gamma \vdash e' : \tau \ \wedge \forall C : (\Xi \mid \Gamma \vdash \tau) \Rightarrow (\bullet \mid \bullet \vdash \mathbb{B}), v.(C[e] \Downarrow v \iff C[e'] \Downarrow v)$. Again, the well-typedness follows from the fact that all expressions in R are well-typed. So what remains to be shown is that $\forall C : (\Xi \mid \Gamma \vdash \tau) \Rightarrow (\bullet \mid \bullet \vdash \mathbb{B}), v.(C[e] \Downarrow v \iff C[e'] \Downarrow v)$. So assume some context C of the right type and some value v. By lemma 8, we have $R(\bullet, \bullet, C[e], C[e'], \mathbb{B})$. By adequacy of R, we have that $C[e] \Downarrow v \iff C[e'] \Downarrow v$, which was what we wanted. □

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

Proof of theorem 5. By theorem 6 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 9 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◄

4.1 Logical Relations

►explain briefly what logical relations are◄

4.2 Defining the Logical Relations model

well-typedness relation

$$\mathcal{W}[\![\tau]\!]_\rho(v_1, v_2) \triangleq \bullet \mid \bullet \vdash v_1 : \rho_1(\tau) \wedge \bullet \mid \bullet \vdash v_2 : \rho_2(\tau)$$

Value interpretation As part of all relations below, we have that $\mathcal{W}[\![\tau]\!]_\rho(v_1, v_2)$, which we don't write for succinctness.

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

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

$$\begin{aligned} \mathcal{E}[\![\tau]\!]_\rho(e_1, e_2) \triangleq \bullet \mid \bullet \vdash e_1 : \rho_1(\tau) \wedge \bullet \mid \bullet \vdash e_2 : \rho_2(\tau) \wedge e_1 \wedge \\ (\exists v_1, v_2. e_1 \to^* v_1 \wedge e_2 \to^* v_2 \wedge \mathcal{V}[\![\tau]\!]_\rho(v_1, v_2)) \end{aligned}$$

Variable environment interpretation

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

Type environment interpretation

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

Definition 4.2.1 (Logical relation)**.**

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

4.3 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 10 (Cmpt-unit)**.** $\dfrac{}{\Xi \mid \Gamma \vdash () \approx^{LR} () : \mathsf{Unit}}$

Proof. By definition 4.2.1, we must show $\Xi \mid \Gamma \vdash () : \mathsf{Unit} \wedge \Xi \mid \Gamma \vdash () : \mathsf{Unit} \wedge \forall \rho \in \mathcal{D}[\![\Xi]\!], \gamma \in \mathcal{G}[\![\Gamma]\!]_\rho. \mathcal{E}[\![\mathsf{Unit}]\!]_\rho(\gamma_1(()), \gamma_2(()))$. 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 $\mathcal{E}[\![\mathsf{Unit}]\!]_\rho(\gamma_1(()), \gamma_2(())) \equiv \mathcal{E}[\![\mathsf{Unit}]\!]_\rho((), ())$. Thus, we must show three things

1. $\bullet \mid \bullet \vdash () : \rho_1(\mathsf{Unit})$
2. $\bullet \mid \bullet \vdash () : \rho_2(\mathsf{Unit})$
3. $\exists v_1, v_2.() \to^* v_1 \wedge () \to^* v_2 \wedge \mathcal{V}[\![\mathsf{Unit}]\!]_\rho(v_1, v_2)$

Given that $\rho_1(\mathsf{Unit}) = \rho_2(\mathsf{Unit}) = \mathsf{Unit}$, the well-typedness once again simply follows from typing rule T-unit. For (3), we choose () for v_1 and v_2. Then we must show $() \to^* () \wedge () \to^* () \wedge \mathcal{V}[\![\mathsf{Unit}]\!]_\rho((),())$. The first two holds trivially. The last part says we must show $\mathcal{W}[\![\mathsf{Unit}]\!]_\rho((),()) \wedge ((),()) \in \{((),())\}$. The last part trivially holds, and the first part holds by the same argument made above: $\rho_1(\mathsf{Unit}) = \rho_2(\mathsf{Unit}) = \mathsf{Unit}$, and then it follows from T-unit. □

Lemma 11 (Cmpt-add)**.** $$\frac{\Xi \mid \Gamma \vdash e_1 \approx^{LR} e_1' : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 \approx^{LR} e_2' : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 + e_2 \approx^{LR} e_1' + e_2' : \mathbb{Z}}$$

Proof. So assuming

$$\Xi \mid \Gamma \vdash e_1 \approx^{LR} e_1' : \mathbb{Z} \tag{4.1}$$
$$\Xi \mid \Gamma \vdash e_2 \approx^{LR} e_2' : \mathbb{Z} \tag{4.2}$$

we must show $\Xi \mid \Gamma \vdash e_1 + e_2 : \mathbb{Z} \wedge \Xi \mid \Gamma \vdash e_1' + e_2' : \mathbb{Z} \wedge \forall \rho \in \mathcal{D}[\![\Xi]\!], \gamma \in \mathcal{G}[\![\Gamma]\!]_\rho . \mathcal{E}[\![\mathbb{Z}]\!]_\rho(\gamma_1(e_1 + e_2), \gamma_2(e_1' + e_2'))$. From (4.1), we have $\Xi \mid \Gamma \vdash e_1 : \mathbb{Z}$, and from (4.2) we have $\Xi \mid \Gamma \vdash e_2 : \mathbb{Z}$. So by the T-add rule, we get $\Xi \mid \Gamma \vdash e_1 + e_2 : \mathbb{Z}$. The well-typedness of $e_1' + e_2'$ follows in the same way.
So let's assume some $\rho \in \mathcal{D}[\![\Xi]\!]$ and $\gamma \in \mathcal{G}[\![\Gamma]\!]_\rho$. We must show $\mathcal{E}[\![\mathbb{Z}]\!]_\rho(\gamma_1(e_1 + e_2), \gamma_2(e_1' + e_2'))$, which is equivalent to $\mathcal{E}[\![\mathbb{Z}]\!]_\rho(\gamma_1(e_1) + \gamma_1(e_2), \gamma_2(e_1') + \gamma_2(e_2'))$. Unfolding the expression interpretation, what we need to show is the following three points

1. $\bullet \mid \bullet \vdash \gamma_1(e_1) + \gamma_1(e_2) : \rho_1(\mathbb{Z})$
2. $\bullet \mid \bullet \vdash \gamma_2(e_1') + \gamma_2(e_2') : \rho_2(\mathbb{Z})$
3. $\exists v_1, v_2 . \gamma_1(e_1) + \gamma_1(e_2) \to^* v_1 \wedge \gamma_2(e_1') + \gamma_2(e_2') \to^* v_2 \wedge \mathcal{V}[\![\mathbb{Z}]\!]_\rho(v_1, v_2)$

To prove these points, we will need some more information. From the last part of (4.1) instantiated with ρ and γ, we get

$$\bullet \mid \bullet \vdash \gamma_1(e_1) : \rho_1(\mathbb{Z}) \tag{4.3}$$
$$\bullet \mid \bullet \vdash \gamma_2(e_1') : \rho_2(\mathbb{Z}) \tag{4.4}$$
$$\exists v_1, v_2 . \gamma_1(e_1) \to^* v_1 \wedge \gamma_2(e_1') \to^* v_2 \wedge \mathcal{V}[\![\mathbb{Z}]\!]_\rho(v_1, v_2) \tag{4.5}$$

Likewise, from the last part of 4.2 instantiated with ρ and γ, we get

$$\bullet \mid \bullet \vdash \gamma_1(e_2) : \rho_1(\mathbb{Z}) \tag{4.6}$$
$$\bullet \mid \bullet \vdash \gamma_2(e_2') : \rho_2(\mathbb{Z}) \tag{4.7}$$
$$\exists v_1, v_2 . \gamma_1(e_2) \to^* v_1 \wedge \gamma_2(e_2') \to^* v_2 \wedge \mathcal{V}[\![\mathbb{Z}]\!]_\rho(v_1, v_2) \tag{4.8}$$

Now to prove the three points above. Given that $\mathbb{Z}$ is a closed type, $\rho_1(\mathbb{Z}) = \rho_2(\mathbb{Z}) = \mathbb{Z}$. Thus, point 1 follows from (4.3), (4.6), and the T-add rule. Similarly,

point 2 follows from (4.4), (4.7) and the T-add rule.
So all that remains to be shown is point 3. Let's denote the values in (4.5) v_1' and v_2', and the values in (4.8) v_1'' and v_2''. Then we know

$$\gamma_1(e_1) + \gamma_1(e_2) \to^* v_1' + \gamma_1(e_2) \to^* v_1' + v_1'' \tag{4.9}$$

$$\gamma_2(e_1') + \gamma_2(e_2') \to^* v_2' + \gamma_2(e_2') \to^* v_2' + v_2'' \tag{4.10}$$

$$\mathcal{V}[\![\mathbb{Z}]\!]_\rho(v_1', v_2') \wedge \mathcal{V}[\![\mathbb{Z}]\!]_\rho(v_1'', v_2'') \tag{4.11}$$

From (4.11), we know that all our values are integers and $\overline{v_1'} = \overline{v_2'}$, and $\overline{v_1''} = \overline{v_2''}$. Furthermore $\overline{v_1'} + \overline{v_1''} \to \overline{v_1' + v_1''}$, and $\overline{v_2'} + \overline{v_2''} \to \overline{v_2' + v_2''}$. Thus $\gamma_1(e_1) + \gamma_1(e_2) \to^* \overline{v_1' + v_1''}$ and $\gamma_2(e_1') + \gamma_2(e_2') \to^* \overline{v_2' + v_2''}$.
What we need to show is point 3. So now use $\overline{v_1' + v_1''}$ for v_1, and $\overline{v_2' + v_2''}$ for v_2. The first two propositions hold by what we have just shown, so all that remains is to show $\mathcal{V}[\![\mathbb{Z}]\!]_\rho(\overline{v_1' + v_1''}, \overline{v_2' + v_2''})$, which by definition means we have to show $\mathcal{W}[\![\mathbb{Z}]\!]_\rho(\overline{v_1' + v_1''}, \overline{v_2' + v_2''})$ and $\overline{v_1' + v_1''} = \overline{v_2' + v_2''}$. The first part follows simply from the T-int rule. For the second part, note that we know $\overline{v_1'} = \overline{v_2'}$, and $\overline{v_1''} = \overline{v_2''}$, so it follows that $\overline{v_1' + v_1''} = \overline{v_2' + v_1''} = \overline{v_2' + v_2''}$. □

Lemma 12 (Cmpt-fst). $\dfrac{\Xi \mid \Gamma \vdash e \approx^{LR} e' : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \mathsf{fst}\ e \approx^{LR} \mathsf{fst}\ e' : \tau_1}$

Proof. So assuming $\Xi \mid \Gamma \vdash e \approx^{LR} e' : \tau_1 + \tau_2$, we must show (ignoring well-typedness): $\forall \rho \in \mathcal{D}[\![\Xi]\!], \gamma \in \mathcal{G}[\![\Gamma]\!]_\rho . \mathcal{E}[\![\tau_1]\!]_\rho(\gamma_1(\mathsf{fst}\ e), \gamma_2(\mathsf{fst}\ e'))$. So assume some $\rho \in \mathcal{D}[\![\Xi]\!]$, and $\gamma \in \mathcal{G}[\![\Gamma]\!]_\rho$, we must show $\mathcal{E}[\![\tau_1]\!]_\rho(\gamma_1(\mathsf{fst}\ e), \gamma_2(\mathsf{fst}\ e')) \equiv \mathcal{E}[\![\tau_1]\!]_\rho(\mathsf{fst}\ \gamma_1(e), \mathsf{fst}\ \gamma_2(e'))$. By definition of our expression interpretation, we must show

$$\exists v_1, v_2 . \mathsf{fst}\ \gamma_1(e) \to^* v_1 \wedge \mathsf{fst}\ \gamma_2(e') \to^* v_2 \wedge \mathcal{V}[\![\tau_1]\!]_\rho(v_1, v_2) \tag{4.12}$$

Note first that if we instantiate our initial assumption with ρ and γ, we get

$$\exists v_1, v_2 . \gamma_1(e) \to^* v_1 \wedge \gamma_2(e') \to^* v_2 \wedge \mathcal{V}[\![\tau_1 \times \tau_2]\!]_\rho(v_1, v_2) \tag{4.13}$$

Let's instantiate the values in (4.13) as v_1' and v_2'. Then we know

$$\mathsf{fst}\ \gamma_1(e) \to^* \mathsf{fst}\ v_1' \tag{4.14}$$

$$\mathsf{fst}\ \gamma_2(e') \to^* \mathsf{fst}\ v_2' \tag{4.15}$$

$$\mathcal{V}[\![\tau_1 \times \tau_2]\!]_\rho(v_1', v_2') \tag{4.16}$$

By (4.16) we know that $v_1' = (w_{11}, w_{12})$ and $v_2' = (w_{21}, w_{22})$ for some values $w_{11}, w_{12}, w_{21}, w_{22}$. Further, $\mathcal{V}[\![\tau_1]\!]_\rho(w_{11}, w_{21})$ and $\mathcal{V}[\![\tau_2]\!]_\rho(w_{12}, w_{22})$. Now, by (4.14) and E-fst we have $\mathsf{fst}\ \gamma_1(e) \to^* w_{11}$. Likewise, from (4.15) and E-fst, we get $\mathsf{fst}\ \gamma_2(e') \to^* w_{21}$.
Now to prove the goal (4.12). We take w_{11} for v_1 and w_{21} for v_2, so that we must show $\mathsf{fst}\ \gamma_1(e) \to^* w_{11} \wedge \mathsf{fst}\ \gamma_2(e') \to^* w_{21} \wedge \mathcal{V}[\![\tau_1]\!]_\rho(w_{11}, w_{21})$, all of which we have just argued holds. □

Lemma 13 (Cmpt-match)**.**

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

Proof. First, we introduce the following notation: $e = \mathsf{match}\ e_1\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e_2 \mid \mathsf{inj}_2\ x \Rightarrow e_3\ \mathsf{end}$, and $e' = \mathsf{match}\ e_1'\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow e_2' \mid \mathsf{inj}_2\ x \Rightarrow e_3'\ \mathsf{end}$. So we assume

$$\Xi \mid \Gamma \vdash e_1 \approx^{LR} e_1' : \tau_1 + \tau_2 \tag{4.17}$$

$$\Xi \mid \Gamma, x : \tau_1 \vdash e_2 \approx^{LR} e_2' : \tau \tag{4.18}$$

$$\Xi \mid \Gamma, x : \tau_2 \vdash e_3 \approx^{LR} e_3' : \tau \tag{4.19}$$

and must show (ignoring well-typedness) $\forall \rho \in \mathcal{D}[\![\Xi]\!], \gamma \in \mathcal{G}[\![\Gamma]\!]_\rho . \mathcal{E}[\![\tau]\!]_\rho(\gamma_1(e), \gamma_2(e'))$. So assume some $\rho \in \mathcal{D}[\![\Xi]\!]$, and $\gamma \in \mathcal{G}[\![\Gamma]\!]_\rho$, then we must show $\mathcal{E}[\![\tau]\!]_\rho(\gamma_1(e), \gamma_2(e'))$. We again introduce the following notation $f = \mathsf{match}\ \gamma_1(e_1)\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow \gamma_1(e_2) \mid \mathsf{inj}_2\ x \Rightarrow \gamma_1(e_3)\ \mathsf{end}$, and $f' = \mathsf{match}\ \gamma_2(e_1')\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow \gamma_2(e_2') \mid \mathsf{inj}_2\ x \Rightarrow \gamma_2(e_3')\ \mathsf{end}$, and note that $\gamma_1(e) = f$ and $\gamma_2(e') = f'$. Thus, we must show: $\mathcal{E}[\![\tau]\!]_\rho(f, f')$, which by definition corresponds to showing

$$\exists v_1, v_2 . f \to^* v_1 \wedge f' \to^* v_2 \wedge \mathcal{V}[\![\tau]\!]_\rho(v_1, v_2) \tag{4.20}$$

By 4.17 instantiated with ρ, γ, we get

$$\exists v_1, v_2 . \gamma_1(e_1) \to^* v_1 \wedge \gamma_2(e_1') \to^* v_2 \wedge \mathcal{V}[\![\tau_1 + \tau_2]\!]_\rho(v_1, v_2) \tag{4.21}$$

Let's instantiate the values in (4.21) as v_{1_1}' and v_{1_2}', so that we know $\gamma_1(e_1) \to^* v_{1_1}' \wedge \gamma_2(e_1') \to^* v_{1_2}' \wedge \mathcal{V}[\![\tau_1 + \tau_2]\!]_\rho(v_{1_1}', v_{1_2}')$. By definition of the value interpretation, this gives us
$(v_{1_1}', v_{1_2}') \in \{(\mathsf{inj}_1\ w_1, \mathsf{inj}_1\ w_2) \mid \mathcal{V}[\![\tau_1]\!]_\rho(w_1, w_2)\} \vee (v_{1_1}', v_{1_2}') \in \{(\mathsf{inj}_2\ w_1, \mathsf{inj}_2\ w_2) \mid \mathcal{V}[\![\tau_2]\!]_\rho(w_1, w_2)\}$. We do case distinction on the "$\vee$", but show only the first case as they are similar. So assume $(v_{1_1}', v_{1_2}') \in \{(\mathsf{inj}_1\ w_1, \mathsf{inj}_1\ w_2) \mid \mathcal{V}[\![\tau_1]\!]_\rho(w_1, w_2)\}$. This means that $v_{1_1}' = \mathsf{inj}_1\ w_1$ and $v_{1_2}' = \mathsf{inj}_1\ w_2$ for some w_1, w_2, and $\mathcal{V}[\![\tau_1]\!]_\rho(w_1, w_2)$. Now, from 4.18 instantiated with $\rho, \gamma' = \gamma[x \mapsto (w_1, w_2)]$ we get

$$\exists v_1, v_2 . \gamma_1'(e_2) \to^* v_1 \wedge \gamma_2'(e_2') \to^* v_2 \wedge \mathcal{V}[\![\tau]\!]_\rho(v_1, v_2) \tag{4.22}$$

If we instantiate the values in (4.22) as v_{2_1}' and v_{2_2}', then we know that $\gamma_1'(e_2) \to^* v_{2_1}' \wedge \gamma_2'(e_2') \to^* v_{2_2}' \wedge \mathcal{V}[\![\tau]\!]_\rho(v_{2_1}', v_{2_2}')$. By the substitution lemma (lemma 3), we then know:

$$\gamma_1(e_2)[w_1/x] \to^* v_{2_1}' \tag{4.23}$$

$$\gamma_2(e_2')[w_2/x] \to^* v_{2_2}' \tag{4.24}$$

Now let us turn to proving the goal, (4.20). We use v'_{2_1} for v_1 and v'_{2_2} for v_2. Thus, we must show $f \to^* v'_{2_1} \wedge f' \to^* v'_{2_2} \wedge \mathcal{V}[\![\tau]\!]_\rho(v'_{2_1}, v'_{2_2})$. We have the following derivations:
$f \to^*$ match $\mathsf{inj}_1\ w_1$ with $\mathsf{inj}_1\ x \Rightarrow \gamma_1(e_2) \mid \mathsf{inj}_2\ x \Rightarrow \gamma_1(e_3)$ end $\to \gamma_1(e_2)[w_1/x] \to^* v'_{2_1}$, and
$f' \to^*$ match $\mathsf{inj}_1\ w_2$ with $\mathsf{inj}_1\ x \Rightarrow \gamma_2(e'_2) \mid \mathsf{inj}_2\ x \Rightarrow \gamma_2(e'_3)$ end $\to \gamma_2(e'_2)[w_2/x] \to^* v'_{2_2}$.
In both derivations, the first "$\to^*$" follows from the instantiation of (4.21), and the fact that $v'_{1_1} = \mathsf{inj}_1\ w_1$ and $v'_{1_2} = \mathsf{inj}_1\ w_2$. The next "$\to$" holds by E-match-inj1. And finally, the last "$\to^*$" holds by (4.23) and (4.24). Thus the first two parts of the goal are satisfied. The last part of the goal we get from the instantiation of (4.22). □

Lemma 14 (Cmpt-lam). $\dfrac{\Xi \mid \Gamma, x : \tau_1 \ \vdash e \approx^{LR} e' : \tau_2}{\Xi \mid \Gamma \ \vdash \lambda x.\ e \approx^{LR} \lambda x.\ e' : \tau_1 \to \tau_2}$

Proof. So assuming

$$\Xi \mid \Gamma, x : \tau_1 \ \vdash e \approx^{LR} e' : \tau_2 \tag{4.25}$$

we must show (ignoring well-typedness) $\forall \rho \in \mathcal{D}[\![\Xi]\!], \gamma \in \mathcal{G}[\![\Gamma]\!]_\rho . \mathcal{E}[\![\tau_1 \to \tau_2]\!]_\rho(\gamma_1(\lambda x.\ e), \gamma_2(\lambda x.\ e'))$. So assume some $\rho \in \mathcal{D}[\![\Xi]\!]$, and $\gamma \in \mathcal{G}[\![\Gamma]\!]_\rho$, then we must show $\mathcal{E}[\![\tau_1 \to \tau_2]\!]_\rho(\gamma_1(\lambda x.\ e), \gamma_2(\lambda x.\ e')) \equiv \mathcal{E}[\![\tau_1 \to \tau_2]\!]_\rho(\lambda x.\ \gamma_1(e), \lambda x.\ \gamma_2(e'))$, which by definition corresponds to

$$\exists v_1, v_2 . \lambda x.\ \gamma_1(e) \to^* v_1 \wedge \lambda x.\ \gamma_2(e') \to^* v_2 \wedge \mathcal{V}[\![\tau_1 \to \tau_2]\!]_\rho(v_1, v_2) \tag{4.26}$$

Since $\lambda x.\ \gamma_1(e)$ and $\lambda x.\ \gamma_2(e')$ are already values, we can use these for v_1 and v_2. The first two parts are then trivially satisfied, so we only need to show $\mathcal{V}[\![\tau_1 \to \tau_2]\!]_\rho(\lambda x.\ \gamma_1(e), \lambda x.\ \gamma_2(e'))$, which by definition means we have to show $\forall w_1, w_2 . \mathcal{V}[\![\tau_1]\!]_\rho(w_1, w_2) \implies \mathcal{E}[\![\tau_2]\!]_\rho(\gamma_1(e)[w_1/x], \gamma_2(e')[w_2/x])$. So assume some w_1, w_2 such that $\mathcal{V}[\![\tau_1]\!]_\rho(w_1, w_2)$, then we must show $\mathcal{E}[\![\tau_2]\!]_\rho(\gamma_1(e)[w_1/x], \gamma_2(e')[w_2/x])$, which by definition means we have to show $\exists v_1, v_2 . \gamma_1(e)[w_1/x] \to^* v_1 \wedge \gamma_2(e')[w_2/x] \to^* v_2 \wedge \mathcal{V}[\![\tau_2]\!]_\rho(v_1, v_2)$. Now we instantiate (4.25) with $\rho, \gamma' = \gamma[x \mapsto (w_1, w_2)]$, so that we have $\exists v_1, v_2 . \gamma'_1(e) \to^* v_1 \wedge \gamma'_2(e') \to^* v_2 \wedge \mathcal{V}[\![\tau_2]\!]_\rho(v_1, v_2)$. By the substitution lemma (lemma 3), this is equivalent to our goal, so we are done. **►Prettify proof◄** □

Lemma 15 (Cmpt-app). $\dfrac{\Xi \mid \Gamma \ \vdash e_1 \approx^{LR} e'_1 : \tau_1 \to \tau_2 \qquad \Xi \mid \Gamma \ \vdash e_2 \approx^{LR} e'_2 : \tau_1}{\Xi \mid \Gamma \ \vdash e_1\ e_2 \approx^{LR} e'_1\ e'_2 : \tau_2}$

Proof. **►do proof◄** □

Lemma 16 (Cmpt-Tlam). $\dfrac{\Xi, X \mid \Gamma \ \vdash e \approx^{LR} e' : \tau}{\Xi \mid \Gamma \ \vdash \Lambda\ e \approx^{LR} \Lambda\ e' : \forall X.\ \tau}$

Proof. ▶**write in proof**◀ □

Lemma 17 (Cmpt-Tapp)**.** $$\frac{\Xi \mid \Gamma \vdash e \approx^{LR} e' : \forall X.\, \tau}{\Xi \mid \Gamma \vdash e\ \_ \approx^{LR} e'\ \_ : \tau[\tau'/X]}$$

Proof. ▶**proper ref**◀ Similar proof given in notes by Lau □

4.4 Properties of LR

Lemma 18 (Adequacy of LR)**.** $\bullet \mid \bullet \vdash e \approx^{LR} e' : \mathsf{Unit} \implies e \Downarrow \iff e' \Downarrow$

Proof. ▶**do proof**◀ □

Theorem 19. $\Xi \mid \Gamma \vdash e \approx^{LR} e' : \tau \implies \Xi \mid \Gamma \vdash e \approx^{ctx} e' : \tau$

Proof. ▶**write properly**◀ LR is a congruence relation, LR is an adequate relation, and all expressions in it are well-formed by definition. It thusly follows from theorem 9. □

Theorem 20 (Fundamental Theorem)**.** $\Xi \mid \Gamma \vdash e : \tau \implies \Xi \mid \Gamma \vdash e \approx^{LR} e : \tau$

Proof. Since LR is a congruence relation, it follows directly from lemma 7. □

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

$$\textsc{Cng-var}\ \frac{(x:\tau)\in\Gamma}{\Xi \mid \Gamma \vdash x \approx x : \tau} \qquad \textsc{Cng-unit}\ \frac{}{\Xi \mid \Gamma \vdash () \approx () : \mathsf{Unit}} \qquad \textsc{Cng-int}\ \frac{}{\Xi \mid \Gamma \vdash \overline{n} \approx \overline{n} : \mathbb{Z}}$$

$$\textsc{Cng-add}\ \frac{\Xi \mid \Gamma \vdash e_1 \approx e_1' : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 \approx e_2' : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 + e_2 \approx e_1' + e_2' : \mathbb{Z}}$$

$$\textsc{Cng-sub}\ \frac{\Xi \mid \Gamma \vdash e_1 \approx e_1' : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 \approx e_2' : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 - e_2 \approx e_1' - e_2' : \mathbb{Z}}$$

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

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

$$\textsc{Cng-eq}\ \frac{\Xi \mid \Gamma \vdash e_1 \approx e_1' : \mathbb{Z} \qquad \Xi \mid \Gamma \vdash e_2 \approx e_2' : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 = e_2 \approx e_1' = e_2' : \mathbb{B}} \qquad \textsc{Cng-true}\ \frac{}{\Xi \mid \Gamma \vdash \mathsf{true} \approx \mathsf{true} : \mathbb{B}}$$

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

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

$$\textsc{Cng-pair}\ \frac{\Xi \mid \Gamma \vdash e_1 \approx e_1' : \tau_1 \qquad \Xi \mid \Gamma \vdash e_2 \approx e_2' : \tau_2}{\Xi \mid \Gamma \vdash (e_1, e_2) \approx (e_1', e_2') : \tau_1 \times \tau_2} \qquad \textsc{Cng-fst}\ \frac{\Xi \mid \Gamma \vdash e \approx e' : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \mathsf{fst}\ e \approx \mathsf{fst}\ e' : \tau_1}$$

$$\textsc{Cng-snd}\ \frac{\Xi \mid \Gamma \vdash e \approx e' : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \mathsf{snd}\ e \approx \mathsf{snd}\ e' : \tau_2} \qquad \textsc{Cng-inj1}\ \frac{\Xi \mid \Gamma \vdash e \approx e' : \tau_1}{\Xi \mid \Gamma \vdash \mathsf{inj}_1\ e \approx \mathsf{inj}_1\ e' : \tau_1 + \tau_2}$$

$$\textsc{Cng-inj2}\ \frac{\Xi \mid \Gamma \vdash e \approx e' : \tau_2}{\Xi \mid \Gamma \vdash \mathsf{inj}_2\ e \approx \mathsf{inj}_2\ e' : \tau_1 + \tau_2}$$

CNG-MATCH

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

CNG-LAM

$$\frac{\Xi \mid \Gamma, x : \tau_1 \vdash e \approx e' : \tau_2}{\Xi \mid \Gamma \vdash \lambda x.\ e \approx \lambda x.\ e' : \tau_1 \to \tau_2}$$

CNG-APP

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

CNG-TLAM

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

CNG-TAPP

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