Relational Reasoning
(Relationel ræsonnement)

Mathias Pedersen, 201808137

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

AARHUS
UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Abstract

►in English. . . ◄

Mathias Pedersen,
Aarhus, December 2021.

Contents

Chapter 1

Introduction

►motivate and explain the problem to be addressed◄
►get your bibtex entries from `https://dblp.org/`◄

Chapter 2

Definition of Language

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 see 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.

2.1.1 Syntax

The syntax captures exactly all programs that can be written in our language – in other words, it governs the "forms" our programs can take. We define it using a context-free grammar whose production rules are in Backus-Naur form:

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

►**question, is only 'e' part of syntax**◄ All programs in our language can be derived using the rules above. We shall also define a couple more concepts: values, types, and evaluation contexts.

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

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

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

Evaluation contexts will be used when we get to defining the dynamics. In general, contexts like this are simply programs with a "hole". That hole can be plugged by some expressions e, and in that case we write $K[e]$. We will see another example of a context in chapter 3.
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 give 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.

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

The evaluation context, K, is the part that 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:

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

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

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

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

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

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

$$\textsc{E-lam-app}\ (\lambda x.\ e)\ v \to_h e[v/x] \qquad \textsc{E-tapp-tlam}\ (\Lambda\ e)\ \_ \to_h e$$

To strike home the workings of evaluation context, we give here an example. Let's say we want to show that $(\lambda x.\ x)\ 2 + 3 \to (\lambda x.\ x)\ 5$. Using the evaluation context $K = (\lambda x.\ x)\ []$, which is a valid evaluation context, we may conclude using the head-step-step rule that $K[2+3] \to K[5]$, if $2 + 3 \to_h 5$. By the E-add head-step rule, we know that $2 + 3 \to_h 5$, thus we have that $(\lambda x.\ x)\ 2 + 3 \equiv K[2 + 3] \to K[5] \equiv (\lambda x.\ x)\ 5$. Further, using $K = []$ and the headstep rule E-lam-app, we may conclude $(\lambda x.\ x)\ 5 \to x[5/x] \equiv 5$. We can hide the middle step, and simply write $(\lambda x.\ x)\ 2 + 3 \to^2 5$, meaning that we perform two steps to reach 5. We also introduce the notation $e \to^* e'$, which means that e takes zero or more steps to reach e'. And if $e \to^* v$, then we shall write $e \Downarrow v$.
In the majority of the report, we will leave the evaluation contexts implicit and simply refer to the head-steps when justifying reductions.

At this point, our language has been formally 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. $K[e] \rightarrow_h e' \wedge \neg(K = []) \implies \text{Val}(e)$

Proof. This can be shown by doing case distinction on the head-step $K[e] \rightarrow_h e'$. 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)**.** $K[e] \rightarrow^* e' \implies \exists e''.(e \rightarrow^* e'') \wedge ((\text{Val}(e'') \wedge K[e''] \rightarrow^* e') \vee (\neg\text{Val}(e'') \wedge K[e''] = e'))$

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

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

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

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

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

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

  - $\text{Val}(e)$. We have that $K[e] \rightarrow^0 e'$, so $K[e] \rightarrow^* e'$. Thus, we have $\text{Val}(e) \wedge K[e] \rightarrow^* e'$, 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 $K[e] \rightarrow^{m+1} e'$. By the Induction Hypothesis, we have:

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

Split the evaluation, $K[e] \rightarrow^{m+1} e'$, up, so that $K[e] \rightarrow g \wedge g \rightarrow^{m} e'$. 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 $h \rightarrow_h h'$. 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 $e \rightarrow_h h'$. Furthermore, $K[h'] \rightarrow^{m} e'$. Instantiate I.H. with this to get

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

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

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

  We know that $e \rightarrow h'$, as $e \rightarrow_h h'$, and by 2.3, we know that $h' \rightarrow^* f''$, so that $e \rightarrow^* f''$. 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 $K[E[h']] \rightarrow^{m} e'$. Instantiate I.H. with this to get

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

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

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

  We know that $E[h] \rightarrow E[h']$, as $h \rightarrow_h h'$, and since $e = E[h]$, then $e \rightarrow E[h']$. By 2.4, we know that $E[h'] \rightarrow^* f''$, so that $e \rightarrow^* f''$. The second part follows directly from 2.4.
- $K = H[E[]]$ and $E[e] = h$. Here we have $h = E[e]$, so $E[e] \rightarrow_h h'$. 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

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

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

□

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

2.2.2 Substitution

We will not define formally how a substitution $e[e'/x]$ works here, but refer instead to types and programming languages by benjamin c pierce (section 5.3)►**proper ref**◄, who explains how to formally define substitutions so that they are *capture-avoiding*. However, we will be needing a way to close off open expressions later, so we shall define here an extended substitution $\gamma = \{x_1 \mapsto v_1, x_2 \mapsto v_2, \dots\}$ inductively as follows:

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

The following lemma will be useful in later proofs.

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

Proof. Similar to Lemma 10 in [1]. □

2.2.3 Type Safety, Normalisation, and Determinacy

We note here a few interesting results about System F. They will not be proved here, as it requires a substantial amount of work.
Firstly, System F is a type-safe language. Essentially, this means that well-typed programs don't get "stuck", in the sense that either we can take a step, or we have reduced to a value. We may state this formally as follows

Theorem 4. $\bullet \mid \bullet \vdash e : \tau \wedge e \to^* e' \implies \text{Val}(e') \vee \exists e''.e' \to e''$

Secondly, every derivation of an expression in System F will at some point terminate. This property is called "normalisation".

Theorem 5. *System F is a normalising language.*

Together with type-safety, this implies that all closed and well-typed expressions will terminate at some value. ►**is this correct?**◄ Finally, System F is a deterministic language.

Theorem 6. *If* $e \to e_1$ *and* $e \to e_2$ *then* $e_1 = e_2$

A consequence of this is the following corollary.

Corollary 6.1. $e \Downarrow v_1 \wedge e \Downarrow v_2 \implies v_1 = v_2$

A proof of this can be found in appendix B at B.1

Chapter 3

Contextual Equivalence

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 sure whether you can justify the refactoring – the implementations might show differing behaviour.
What you want to show, then, is that your two stack implementations *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*.
Contextual Equivalence can be used to show this along with other, "smaller" properties such as idempotency and hoisting of functions. Both of which we will be proving in chapter 5.
In this chapter 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

In order to define contextual equivalence, we will first need to define the notion of a context. As with the evaluation context, this will simply be a program with exactly one hole in it, which may be plugged by some expression.

Definition 3.1.1 (Context)**.** A context C is anything that may be derived using

the following CFG.

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

Unlike the evaluation context, the hole can here be *anywhere* in the program. This is due to how we will be defining contextual equivalence. Intuitively, we only want to consider two expressions contextually equivalent, when we can plug them into *any* program with a hole, and they elicit similar behaviour.
Next, we shall define "Context Typing". The type of a context will capture the type of the hole, and the type of the plugged program. We define it as an inference rule:

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

We can read this as: the context C takes an expression e (the expression which plugs the hole) of type τ under Ξ, Γ, and outputs an expression $C[e]$ (the plugged program) of type τ' under Ξ', Γ'.
With this defined, we are now ready to define contextual equivalence. We first state the definition, and then give some intuition as to why this definition makes sense.

Definition 3.1.2 (Contextual Equivalence)**.** We define contextual equivalence of two expressions e_1 and e_2 at type τ under Ξ and Γ as

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

This says that two expressions are contextually equivalent when they are well typed and we can plug them into any context, which closes both expressions and makes the plugged programs have type $\mathbb{B}$, and the plugged programs run down to the same value (thus, either true or false). In other words, 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 Boolean v if and only if the other program $C[e_2]$ terminates to the same Boolean, v.
To help demystify this definition, we will discuss a few points. Firstly, 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? The answer is that using $\mathbb{B}$ is "strong enough". We will discuss what this means in a moment. But we could in fact have used some other type like $\mathbb{Z}$ or even a composite type like $\mathbb{B} \times \mathbb{Z}$. The important part is that we are able to make the plugged programs elicit differing behaviour. We cannot do this if the plugged programs have type Unit; by theorem 4 and 5 we know that both plugged programs terminate at a value **►is this correct?◄**, and there is only one value with type Unit, namely (), so both programs will terminate at () regardless of what e_1 and e_2 looks like. 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 $C[e_1] \Downarrow \iff C[e_2] \Downarrow$.

Now, let's discuss why our definition is strong enough, and what that means. Firstly, when talking about contextual equivalence, we are not interested in whether or not expressions are syntactically the same – we only care about their observable behaviour. So let's say that two contextually equivalent expressions e_1 and e_2 run down to some values, then those values should be behaviourally the same. We have not yet defined formally what we mean by "behaviourally the same". However, intuitively, if the expressions are of integer type, for example, then the two values should be the same number. And for product types, the first value in each pair should be behaviourally the same, and likewise with the second value in each pair. Our definition of contextual equivalence ensures this. To see why this is the case, 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 the expressions are of integer type (so $\tau = \mathbb{Z}$). Then consider when C has the form $[\cdot] = v_1$. Here $C[e_1] \Downarrow \mathsf{true}$. But what about $C[e_2]$? If $v_2 \neq v_1$, then our evaluation rule E-not-eq tells us that $C[e_2] \Downarrow \mathsf{false}$. 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, 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:

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

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 outputs 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 state similar congruence rules 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 relations model 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 will be working with relations $R \subseteq \text{TENV} \times \text{VENV} \times \text{EXPR} \times \text{EXPR} \times \text{TYPE}$, and R will refer to any such relation. TENV is simply the set of all type environments, so $\Xi \in \text{TENV}$. Likewise with the other sets. 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:

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

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

Definition 3.2.3. Contextual Equivalence is a relation, $CE \subseteq \text{TENV} \times \text{VENV} \times \text{EXPR} \times \text{EXPR} \times \text{TYPE}$, 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 3.1.2:

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

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 8. $\cdot \mid \cdot \vdash \cdot \approx^{ctx} \cdot : \cdot \subseteq \text{TENV} \times \text{VENV} \times \text{EXPR} \times \text{EXPR} \times \text{TYPE}$ *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 9 (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 $R(\Xi, \Gamma, \Lambda\ e, \Lambda\ e, \forall X.\ \tau)$. Our induction hypothesis is $R((\Xi, X), \Gamma, e, e, \tau)$ which is equivalent to $\Xi, X \mid \Gamma \ \vdash e \approx e : \tau$. 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 10. *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 9, 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 9, 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 11. *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 10, 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 7. By theorem 8 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 11 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

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

Expression interpretation

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

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 12 (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 13 (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 14 (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 15 (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:
$f \to^* \mathsf{match}\ \mathsf{inj}_1\ w_1\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow \gamma_1(e_2) \mid \mathsf{inj}_2\ x \Rightarrow \gamma_1(e_3)\ \mathsf{end} \to \gamma_1(e_2)[w_1/x] \to^* v'_{2_1}$, and
$f' \to^* \mathsf{match}\ \mathsf{inj}_1\ w_2\ \mathsf{with}\ \mathsf{inj}_1\ x \Rightarrow \gamma_2(e'_2) \mid \mathsf{inj}_2\ x \Rightarrow \gamma_2(e'_3)\ \mathsf{end} \to \gamma_2(e'_2)[w_2/x] \to^* v'_{2_2}$.
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 16 (Cmpt-lam)**.** $$\frac{\Xi \mid \Gamma, x : \tau_1 \ \vdash e \approx^{LR} e' : \tau_2}{\Xi \mid \Gamma \ \vdash \lambda x.\ e \approx^{LR} \lambda x.\ e' : \tau_1 \to \tau_2}$$

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

Proof. So we assume

$$\Xi \mid \Gamma \ \vdash e_1 \approx^{LR} e'_1 : \tau_1 \to \tau_2 \tag{4.27}$$
$$\Xi \mid \Gamma \ \vdash e_2 \approx^{LR} e'_2 : \tau_1 \tag{4.28}$$

Then we must show $\forall \rho \in \mathcal{D}[\![\Xi]\!], \gamma \in \mathcal{G}[\![\Gamma]\!]_\rho . \mathcal{E}[\![\tau_2]\!]_\rho(\gamma_1(e_1\ e_2), \gamma_2(e_1'\ e_2'))$. So we assume some $\rho \in \mathcal{D}[\![\Xi]\!], \gamma \in \mathcal{G}[\![\Gamma]\!]_\rho$, and our goal now is $\mathcal{E}[\![\tau_2]\!]_\rho(\gamma_1(e_1\ e_2), \gamma_2(e_1'\ e_2')) \equiv \mathcal{E}[\![\tau_2]\!]_\rho(\gamma_1(e_1)\ \gamma_1(e_2), \gamma_2(e_1')\ \gamma_2(e_2'))$, which by definition means we have to show

$$\exists v_1, v_2 . \gamma_1(e_1)\ \gamma_1(e_2) \to^* v_1 \wedge \gamma_2(e_1')\ \gamma_2(e_2') \to^* v_2 \wedge \mathcal{V}[\![\tau_2]\!]_\rho(v_1, v_2) \tag{4.29}$$

We instantiate both (4.27) and (4.28) with ρ, γ, and expand the definitions to get

$$\exists v_1', v_2' . \gamma_1(e_1) \to^* v_1' \wedge \gamma_2(e_1') \to^* v_2' \wedge \mathcal{V}[\![\tau_1 \to \tau_2]\!]_\rho(v_1', v_2') \tag{4.30}$$
$$\exists v_1'', v_2'' . \gamma_1(e_2) \to^* v_1'' \wedge \gamma_2(e_2') \to^* v_2'' \wedge \mathcal{V}[\![\tau_1]\!]_\rho(v_1'', v_2'') \tag{4.31}$$

Instantiating all quantifications with the same name they quantify over, we get from the definition of the value interpretation for function-type that $v_1' = \lambda x.\ e_1'$ and $v_2' = \lambda x.\ e_2'$, for some e_1' and e_2'. Further,

$$\forall w_1, w_2 . \mathcal{V}[\![\tau_1]\!]_\rho(w_1, w_2) \implies \mathcal{E}[\![\tau_2]\!]_\rho(e_1'[w_1/x], e_2'[w_2/x]) \tag{4.32}$$

We instantiate this with v_1'' and v_2''. The antecedent is satisfied by the last part of (4.31), so we know $\mathcal{E}[\![\tau_2]\!]_\rho(e_1'[v_1''/x], e_2'[v_2''/x])$, which by definition means

$$\exists v_{f_1}, v_{f_2} . e_1'\big[v_1''/x\big] \to^* v_{f_1} \wedge e_2'\big[v_2''/x\big] \to^* v_{f_2} \wedge \mathcal{V}[\![\tau_2]\!]_\rho(v_{f_1}, v_{f_2}) \tag{4.33}$$

Now we turn to showing the goal, (4.29). We use v_{f_1} for v_1 and v_{f_2} for v_2, so that we must show: $\gamma_1(e_1)\ \gamma_1(e_2) \to^* v_{f_1} \wedge \gamma_2(e_1')\ \gamma_2(e_2') \to^* v_{f_2} \wedge \mathcal{V}[\![\tau_2]\!]_\rho(v_{f_1}, v_{f_2})$. We have the following derivations:

$$\gamma_1(e_1)\ \gamma_1(e_2) \to^* \lambda x.\ e_1'\ \gamma_1(e_2) \to^* \lambda x.\ e_1'\ v_1'' \to e_1'\big[v_1''/x\big] \to^* v_{f_1} \tag{4.34}$$
$$\gamma_2(e_1')\ \gamma_2(e_2') \to^* \lambda x.\ e_2'\ \gamma_2(e_2') \to^* \lambda x.\ e_2'\ v_2'' \to e_2'\big[v_2''/x\big] \to^* v_{f_2} \tag{4.35}$$

Here, in both derivations, the first "$\to^*$" follows by (4.30) and the fact that $v_1' = \lambda x.\ e_1'$ and $v_2' = \lambda x.\ e_2'$. The second "$\to^*$" follows by (4.31). The next "$\to$" follows by rule E-lam-app, and the final "$\to^*$" follows from (4.33). Thus, the first two parts of what we have to show follows from (4.34) and (4.35) respectively, and the last part follows from the last part of (4.33). □

Lemma 18 (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. Assuming

$$\Xi, X \mid \Gamma\ \vdash e \approx^{LR} e' : \tau \tag{4.36}$$

we must show $\forall \rho \in \mathcal{D}[\![\Xi]\!], \gamma \in \mathcal{G}[\![\Gamma]\!]_\rho . \mathcal{E}[\![\forall X.\ \tau]\!]_\rho(\gamma_1(\Lambda\ e), \gamma_2(\Lambda\ e'))$. So assume some $\rho \in \mathcal{D}[\![\Xi]\!]$ and $\gamma \in \mathcal{G}[\![\Gamma]\!]_\rho$, by the definition of expression interpretation, our goal is now to show

$$\exists v_1, v_2 . \Lambda\ \gamma_1(e) \to^* v_1 \wedge \Lambda\ \gamma_2(e') \to^* v_2 \wedge \mathcal{V}[\![\forall X.\ \tau]\!]_\rho(v_1, v_2) \tag{4.37}$$

Note here that we have pushed the variable substitution inside the Λ-abstraction. Here we can simply use $\Lambda\ \gamma_1(e)$ for v_1 and $\Lambda\ \gamma_2(e')$ for v_2, as they are already values. Thus, we must show $\mathcal{V}[\![\forall X.\ \tau]\!]_\rho(\Lambda\ \gamma_1(e), \Lambda\ \gamma_2(e'))$, which by definition means we have to show $\forall \tau_1, \tau_2, R \in Rel[\tau_1, \tau_2].\mathcal{E}[\![\tau]\!]_{\rho'}(\gamma_1(e), \gamma_2(e'))$, where $\rho' = \rho[X \mapsto (\tau_1, \tau_2, R)]$. So assume some types τ_1, τ_2, and a relation $R \in Rel[\tau_1, \tau_2]$, then our goal is

$$\mathcal{E}[\![\tau]\!]_{\rho'}(\gamma_1(e), \gamma_2(e')) \tag{4.38}$$

We now instantiate (4.36) with ρ' and γ. Note that $\gamma \in \mathcal{G}[\![\Gamma]\!]_{\rho'}$ by corollary 30.1. Thus, we get $\mathcal{E}[\![\tau]\!]_{\rho'}(\gamma_1(e), \gamma_2(e'))$, which is exactly what we had to show. □

Lemma 19 (Cmpt-Tapp)**.** $\dfrac{\Xi \mid \Gamma \ \vdash e \approx^{LR} e' : \forall X.\ \tau}{\Xi \mid \Gamma \ \vdash e\ \_ \approx^{LR} e'\ \_ : \tau[\tau'/X]}$

Proof. Similar to proof given in [1]. ►**If space, do proof.**◄ □

4.4 Properties of LR

Lemma 20 (Adequacy of LR)**.** $\bullet \mid \bullet \ \vdash e \approx^{LR} e' : \mathbb{B} \implies e \Downarrow v \iff e' \Downarrow v$

Proof. Assuming $\bullet \mid \bullet \ \vdash e \approx^{LR} e' : \mathbb{B}$, we must show $e \Downarrow v \iff e' \Downarrow v$. By the definition of LR, we have that $\forall \rho \in \mathcal{D}[\![\bullet]\!], \gamma \in \mathcal{G}[\![\bullet]\!]_\rho.\mathcal{E}[\![\mathbb{B}]\!]_\rho(\gamma_1(e), \gamma_2(e'))$. Given our definitions of $\mathcal{D}$ and $\mathcal{G}$, we can only instantiate it with the empty relational type substitution and the empty variable substitution. Doing this, we get $\mathcal{E}[\![\mathbb{B}]\!]_\emptyset(\emptyset(e), \emptyset(e')) \equiv \mathcal{E}[\![\mathbb{B}]\!]_\emptyset(e, e')$, which by definition means we have $\exists v_1, v_2.e \to^* v_1 \wedge e' \to^* v_2 \wedge \mathcal{V}[\![\mathbb{B}]\!]_\emptyset(v_1, v_2)$. Instantiating these values, we may conclude that $e \Downarrow v_1$, $e' \Downarrow v_2$. Now, our goal is to show $e \Downarrow v \iff e' \Downarrow v$. Both sides are similar, so we show just the " $\implies$ " direction. So we assume $e \Downarrow v$, and must show $e' \Downarrow v$. By corollary 6.1, we may conclude that $v_1 = v$, so that our goal now is to show $e' \Downarrow v_1$. Given we know that $e' \Downarrow v_2$, then it suffices to show that $v_1 = v_2$. Since $\mathcal{V}[\![\mathbb{B}]\!]_\emptyset(v_1, v_2)$, then either $v_1 = v_2 = \mathsf{true}$ or $v_1 = v_2 = \mathsf{false}$. In any case, $v_1 = v_2$, so we are done. □

Lemma 21 (Congruency of LR)**.** *LR is a congruence relation*

Proof. Follows immediately from the compatibility lemmas. □

Theorem 22. $\Xi \mid \Gamma \ \vdash e \approx^{LR} e' : \tau \implies \Xi \mid \Gamma \ \vdash e \approx^{ctx} e' : \tau$

Proof. We now know that LR is an adequate relation (lemma 20) and a congruence relation (lemma 21). Further, it follows directly from the definition of LR, that all expressions in the relation are well-typed. Thus, by theorem 11, we may conclude $\Xi \mid \Gamma \ \vdash e \approx^{LR} e' : \tau \implies \Xi \mid \Gamma \ \vdash e \approx^{ctx} e' : \tau$. □

Theorem 23 (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 (lemma 21), it follows directly from lemma 9. □

Chapter 5

Examples of Application of Contextual Equivalence

►draft◄
With our logical relations model defined and the necessary properties of it proved, we are ready to put it to use on some interesting examples. Some of the theorems will involve contextual equivalence directly, but as we will see, we can even use it in theorems where it isn't mentioned! This is thanks to the fundamental theorem.

5.1 Identity

The first example we shall explore is that of identity; if an expression has a certain type, then we may prove that it behaves just as the identity function. We shall actually show two theorems. The first one is weaker, in the sense that there are some languages (fx ones with side-effects), where only the first theorem holds.

5.1.1 Identity: Reduction

Informally, the first theorem says that an expression e, which given a value of any type τ outputs an expression of the same type τ, will always output the value it was given as input. The intuition for why this is, is that since the expression works for any type, it cannot do anything specific to the value given. Since it must output an expression of the same type, and it doesn't know in advance the type of the input, it only has one option: to return the value it was given.
►question: how does this relate to theorems for free?◄

Theorem 24. $\bullet \mid \bullet \vdash e : \forall X.\, X \to X \land \bullet \mid \bullet \vdash v : \tau \implies (e\ \_)\ v \to^* v$

Proof. So assume

$$\bullet \mid \bullet \vdash e : \forall X.\, X \to X \tag{5.1}$$

$$\bullet \mid \bullet \vdash v : \tau \tag{5.2}$$

Then we must show $(e\ \_)\ v \to^* v$. From 5.1 and by the fundamental theorem (theorem 23) we know $\bullet \mid \bullet \vdash e \approx^{LR} e : \forall X.\ X \to X$ from which we get $\forall \rho \in \mathcal{D}[\![\bullet]\!], \gamma \in \mathcal{G}[\![\bullet]\!]_\rho . \mathcal{E}[\![\forall X.\ X \to X]\!]_\rho(\gamma_1(e), \gamma_2(e))$. By the definition of the type interpretation and value environment interpretation, we only have one choice of instantiation, so we instantiate both with the empty relational substitution, $\emptyset$. Thus we get $\mathcal{E}[\![\forall X.\ X \to X]\!]_\emptyset(e, e)$. And from this, we may conclude $\exists v_1, v_2 . e \to^* v_1 \wedge e \to^* v_2 \wedge \mathcal{V}[\![\forall X.\ X \to X]\!]_\emptyset(v_1, v_2)$. By determinacy (specifically, corollary 6.1) these values must be the same, and from the value interpretation we further know that the value must be a type abstraction. Thus $e \to^* \Lambda\ e'$ and $\forall \tau_1, \tau_2, R \in Rel[\tau_1, \tau_2] . \mathcal{E}[\![X \to X]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]}(e', e')$. Here we use τ for τ_1 and τ_2, and the relation $R' = \{(v, v)\}$ for R, giving us $\mathcal{E}[\![X \to X]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R')]}(e', e')$, meaning $\exists v_1', v_2' . e' \to^* v_1' \wedge e' \to^* v_2' \wedge \mathcal{V}[\![X \to X]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R')]}(v_1', v_2')$. Thus $e' \to^* \lambda x.\ e''$ and $\forall w_1, w_2 . \mathcal{V}[\![X]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R')]}(w_1, w_2) \implies \mathcal{E}[\![X]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R')]}(e''[w_1/x], e''[w_2/x])$. Using v for both w_1 and w_2, we get $\mathcal{E}[\![X]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R')]}(e''[v/x], e''[v/x])$ which means $\exists v_{1f}, v_{2f} . e''[v/x] \to^* v_{1f} \wedge e''[v/x] \to^* v_{2f} \wedge \mathcal{V}[\![X]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R')]}(v_{1f}, v_{2f})$, which again means $e''[v/x] \to^* v_f$ and $(v_f, v_f) \in R'$, but since R' is a singleton, then we must have $v_f = v$. From the above, we have the following derivation

$$(e\ \_)\ v \to^* (\Lambda\ e'\ \_)\ v \to e'\ v \to^* (\lambda x.\ e'')\ v \to e''[v/x] \to^* v_f = v \tag{5.3}$$

The first "$\to^*$" follows from $e \to^* \Lambda\ e'$. The following step holds by E-tapp-tlam. The next "$\to^*$" follows by $e' \to^* \lambda x.\ e''$. The next step holds by E-lam-app. and the final "$\to^*$" follows by $e''[v/x] \to^* v_f$. Thus we have know shown $(e\ \_)\ v \to^* v$. □

5.1.2 Identity: Contextual Equivalence

The next theorem shows that any expression of type $\forall X.\ X \to X$ may as well be substituted with the simple, generic identity function.

Theorem 25. $\Xi \mid \Gamma \vdash e : \forall X.\ X \to X \implies \Xi \mid \Gamma \vdash e \approx^{ctx} \Lambda\ \lambda x.\ x : \forall X.\ X \to X$

Proof. The majority of this proof is simply unfolding definitions; one side of the coin will be working on the goal, and the other side will be working on the fundamental theorem. So we shall go fairly quickly through this.
So assuming $\Xi \mid \Gamma \vdash e : \forall X.\ X \to X$, we must show $\Xi \mid \Gamma \vdash e \approx^{ctx} \Lambda\ \lambda x.\ x : \forall X.\ X \to X$. We shall do this by showing that the two expressions are logically related. Then contextual equivalence follows from theorem 22. So we assume some $\rho \in \mathcal{D}[\![\Xi]\!]$ and $\gamma \in \mathcal{G}[\![\Gamma]\!]_\rho$. Then we must show $\exists v_1, v_2 . \gamma_1(e) \to^* v_1 \wedge \Lambda\ \lambda x.\ x \to^* v_2 \wedge \mathcal{V}[\![\forall X.\ X \to X]\!]_\rho(v_1, v_2)$. By the fundamental theorem on our initial assumption and by instantiating this with ρ and γ, we may conclude

$\bullet \mid \bullet \vdash \gamma_2(e) : \forall X.\ X \to X$ and $\gamma_1(e) \to^* \Lambda\ e_1$ and $\gamma_2(e) \to^* \Lambda\ e_2$ and

$$\forall \tau_1, \tau_2, R \in Rel[\tau_1, \tau_2].\mathcal{E}[\![X \to X]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]}(e_1, e_2) \tag{5.4}$$

Now use $\Lambda\ e_1$ for v_1 and $\Lambda\ \lambda x.\ x$ for v_2 in our goal. Then we must show $\mathcal{V}[\![\forall X.\ X \to X]\!]_\rho(\Lambda\ e_1, \Lambda\ \lambda x.\ x)$. So assume some τ_1, τ_2, and $R \in Rel[\tau_1, \tau_2]$, then we must show $\exists v_1', v_2'.e_1 \to^* v_1' \wedge \lambda x.\ x \to^* v_2' \wedge \mathcal{V}[\![X \to X]\!]_{\rho'}(v_1', v_2')$, where $\rho' = \rho[X \mapsto (\tau_1, \tau_2, R)]$. Now instantiate (5.4) with τ_1, τ_2, R. Then we know $e_1 \to^* \lambda x.\ e_1'$, and $e_2 \to^* \lambda x.\ e_2'$, and

$$\forall w_1, w_2.\mathcal{V}[\![X]\!]_{\rho'}(w_1, w_2) \implies \mathcal{E}[\![X]\!]_{\rho'}(e_1'[w_1/x], e_2'[w_2/x]) \tag{5.5}$$

We then use $\lambda x.\ e_1'$ for v_1' and $\lambda x.\ x$ for v_2' in our goal. Then we have to show $\mathcal{V}[\![X \to X]\!]_{\rho'}(\lambda x.\ e_1', \lambda x.\ x)$. So assume some w_1 and w_2, such that $\mathcal{V}[\![X]\!]_{\rho'}(w_1, w_2)$, then it suffices to show $\exists v_{1f}, v_{2f}.e_1'[w_1/x] \to^* v_{1f} \wedge w_2 \to^* v_{2f} \wedge \mathcal{V}[\![X]\!]_{\rho'}(v_{1f}, v_{2f})$. Finally, instantiate (5.5) with w_1 and w_2. Then we know $e_1'[w_1/x] \to^* v_{1f}'$, and $e_2'[w_2/x]) \to^* v_{2f}'$, and $\mathcal{V}[\![X]\!]_{\rho'}(v_{1f}', v_{2f}')$. Thus, by the definition of our value interpretation, $(v_{1f}', v_{2f}') \in R$. Now use v_{1f}' for v_{1f} and w_2 for v_{2f} in our goal. Then we must show $(v_{1f'}, w_2) \in R$. To show this, we need to do some arguing about the reduction $(\gamma_2(e)\ \_)\ w_2$. We have

$$(\gamma_2(e)\ \_)\ w_2 \to^* (\Lambda\ e_2\ \_)\ w_2 \to e_2\ w_2 \to^* (\lambda x.\ e_2')\ w_2 \to e_2'[w_2/x] \to^* v_{2f}'$$

Since $\bullet \mid \bullet \vdash \gamma_2(e) : \forall X.\ X \to X$ then by theorem 24 we may conclude $(\gamma_2(e)\ \_)\ w_2 \to^* w_2$. Thus by determinacy of System F (specifically, corollary 6.1), we have that $v_{2f}' = w_2$. Since we know $(v_{1f}', v_{2f}') \in R$, then it follows that $(v_{1f}', w_2) \in R$, which was what we had to show. □

5.2 Empty type

►explain briefly what the empty type is◄

Theorem 26. $\nexists e.\bullet \mid \bullet \vdash e : \forall X.\ X$

Proof. Assume for the sake of contradiction that $\exists e.\bullet \mid \bullet \vdash e : \forall X.\ X$. Then, by the fundamental theorem (theorem 23), we know that $\bullet \mid \bullet \vdash e \approx^{LR} e : \forall X.\ X$, which means that $\forall \rho \in \mathcal{D}[\![\bullet]\!], \gamma \in \mathcal{G}[\![\bullet]\!]_\rho.\mathcal{E}[\![\forall X.\ X]\!]_\rho(\gamma_1(e), \gamma_2(e))$. Here, pick the empty relational substitutions, so that $\mathcal{E}[\![\forall X.\ X]\!]_\emptyset(e, e)$. Which by definition means $\exists v_1, v_2.e \to^* v_1 \wedge e \to^* v_2 \wedge \mathcal{V}[\![\forall X.\ X]\!]_\emptyset(v_1, v_2)$. From this we may conclude that $e \to^* \Lambda\ e'$, and $\forall \tau_1, \tau_2, R \in Rel[\tau_1, \tau_2].\mathcal{E}[\![X]\!]_{[X \mapsto (\tau_1, \tau_2, R)]}(e', e')$. Instantiate this with any two types (for example $\mathbb{Z}$), and the empty relation, $R = \emptyset$. Then we have $\mathcal{E}[\![X]\!]_{[X \mapsto (\mathbb{Z}, \mathbb{Z}, \emptyset)]}(e', e')$, from which we know $\exists v_1, v_2.e' \to^* v_1 \wedge e' \to^* v_2 \wedge \mathcal{V}[\![X]\!]_{[X \mapsto (\mathbb{Z}, \mathbb{Z}, \emptyset)]}(v_1, v_2)$. By the definition of the value interpretation for type variables, the last part gives us $(v_1, v_2) \in \emptyset$, which is a contradiction. □

5.3 Idempotency

For the next three theorems, we shall introduce let-expressions, which will just be syntactic sugar for a function application: $\mathsf{let}\ x = e\ \mathsf{in}\ e' \triangleq (\lambda x.\ e')\ e$.

Idempotency in general is the idea that doing something multiple times is equivalent to doing it once. In our theorem below, it corresponds to the fact that evaluating the same expression multiple times does not yield different behaviour compared to just evaluating it once, and using the value you get in its place.

Theorem 27. *If* $\Xi \mid \Gamma \vdash e : \tau$ *then* $\Xi \mid \Gamma \vdash \mathsf{let}\ x = e\ \mathsf{in}\ (x, x) \approx^{ctx} (e, e) : \tau \times \tau$

Proof. We will show the theorem by showing that the two expressions are logically related. Then, by theorem 22, it follows that they are also contextually equivalent. So assume $\Xi \mid \Gamma \vdash e : \tau$, then we will show $\Xi \mid \Gamma \vdash \mathsf{let}\ x = e\ \mathsf{in}\ (x, x) \approx^{LR} (e, e) : \tau \times \tau$. By definition of LR, we must show two things. First that both expressions are well-typed under Ξ and Γ. This follows fairly easily from the assumption and by applying the typing rules, so we will not explain it here. Second that, when the expressions and types are closed, they are in the expression interpretation. So assume some $\rho \in \mathcal{D}[\![\Xi]\!]$ and $\gamma \in \mathcal{G}[\![\Gamma]\!]_\rho$. Then we must show $\mathcal{E}[\![\tau \times \tau]\!]_\rho(\gamma_1(\mathsf{let}\ x = e\ \mathsf{in}\ (x, x)), \gamma_2((e, e)))$, which by the definition of the expression interpretation means we have to show

- $\mid \bullet \vdash \mathsf{let}\ x = \gamma_1(e)\ \mathsf{in}\ (x, x) : \rho_1(\tau \times \tau)$ (5.6)
- $\mid \bullet \vdash (\gamma_2(e), \gamma_2(e)) : \rho_2(\tau \times \tau)$ (5.7)

$$\exists v_1, v_2.\mathsf{let}\ x = \gamma_1(e)\ \mathsf{in}\ (x, x) \to^* v_1 \wedge (\gamma_2(e), \gamma_2(e)) \to^* v_2 \wedge \mathcal{V}[\![\tau \times \tau]\!]_\rho(v_1, v_2) \tag{5.8}$$

From the initial assumption and the fundamental theorem (theorem 23) we have that $\Xi \mid \Gamma \vdash e \approx^{LR} e : \tau$, which, when we instantiate with ρ and γ, gives us

- $\mid \bullet \vdash \gamma_1(e) : \rho_1(\tau)$ (5.9)
- $\mid \bullet \vdash \gamma_2(e) : \rho_2(\tau)$ (5.10)

$$\exists v_1', v_2'.\gamma_1(e) \to^* v_1' \wedge \gamma_2(e) \to^* v_2' \wedge \mathcal{V}[\![\tau]\!]_\rho(v_1', v_2') \tag{5.11}$$

From (5.9) and (5.10), we may prove the first two parts of the goal (5.6) and (5.7). So let's turn our attention the last part (5.8).

$$\mathsf{let}\ x = \gamma_1(e)\ \mathsf{in}\ (x, x) \equiv (\lambda x.\ (x, x))\ \gamma_1(e) \to^* (\lambda x.\ (x, x))\ v_1' \to (x, x)\left[v_1'/x\right] \equiv (v_1', v_1') \tag{5.12}$$

The first equivalence simply de-sugars the let expression. The next "$\to^*$" follows from (5.11). The next step is simply a lambda reduction, and the final equivalence is just performing the substitution.

$$(\gamma_2(e), \gamma_2(e)) \to^* (v_2', \gamma_2(e)) \to^* (v_2', v_2') \tag{5.13}$$

Both "$\to^*$" follows from (5.11). We here use (v_1', v_1') for v_1 and (v_2', v_2') for v_2. Then we must show

$$\mathcal{V}[\![\tau \times \tau]\!]_\rho((v_1', v_1'), (v_2', v_2')) \tag{5.14}$$

By the definition of the value interpretation for product type, it suffices to show well-typedness and $\mathcal{V}[\![\tau]\!]_\rho(v_1', v_2') \wedge \mathcal{V}[\![\tau]\!]_\rho(v_1', v_2')$. Both of which follows from the last part of (5.11) **►prettify proof◄** □

5.4 Commutativity

In short, this commutativity theorem states that the order of evaluation of expressions in tuples is unimportant. Note the order of evaluation in the theorem below. In the left expression, e_2 is reduced first. And in the right, e_1 is reduced first.

Theorem 28. *If* $\Xi \mid \Gamma \vdash e_1 : \tau_1$ *and* $\Xi \mid \Gamma \vdash e_2 : \tau_2$ *then*
$\Xi \mid \Gamma \vdash \mathsf{let}\ x = e_2\ \mathsf{in}\ (e_1, x) \approx^{ctx} (e_1, e_2) : \tau_1 \times \tau_2$

Proof. As in the previous proofs, we shall show this by showing the they are logically related. So assume $\Xi \mid \Gamma \vdash e_1 : \tau_1$ and $\Xi \mid \Gamma \vdash e_2 : \tau_2$. We will ignore showing well-typedness here, so assume some $\rho \in \mathcal{D}[\![\Xi]\!]$ and $\gamma \in \mathcal{G}[\![\Gamma]\!]_\rho$, then we must show $\mathcal{E}[\![\tau_1 \times \tau_2]\!]_\rho(\gamma_1(\mathsf{let}\ x = e_2\ \mathsf{in}\ (e_1, x)), \gamma_2((e_1, e_2)))$, which by the definition of the expression interpretation means we have to show

$$\exists v_1, v_2.\mathsf{let}\ x = \gamma_1(e_2)\ \mathsf{in}\ (\gamma_1(e_1), x) \to^* v_1 \wedge (\gamma_2(e_1), \gamma_2(e_2)) \to^* v_2 \wedge \mathcal{V}[\![\tau_1 \times \tau_2]\!]_\rho(v_1, v_2) \tag{5.15}$$

From the initial assumptions and the fundamental theorem (theorem 23) we have that $\Xi \mid \Gamma \vdash e_1 \approx^{LR} e_1 : \tau_1$ and $\Xi \mid \Gamma \vdash e_2 \approx^{LR} e_2 : \tau_2$, which, when we instantiate with ρ and γ, gives us

$$\exists v_{1f_1}, v_{1f_2}.\gamma_1(e_1) \to^* v_{1f_1} \wedge \gamma_2(e_1) \to^* v_{1f_2} \wedge \mathcal{V}[\![\tau_1]\!]_\rho(v_{1f_1}, v_{1f_2}) \tag{5.16}$$

$$\exists v_{2f_1}, v_{2f_2}.\gamma_1(e_2) \to^* v_{2f_1} \wedge \gamma_2(e_2) \to^* v_{2f_2} \wedge \mathcal{V}[\![\tau_2]\!]_\rho(v_{2f_1}, v_{2f_2}) \tag{5.17}$$

Now we will instantiate our goal (5.15). Use (v_{1f_1}, v_{2f_1}) for v_1 and (v_{1f_2}, v_{2f_2}) for v_2. Thus, we must show

$$\mathsf{let}\ x = \gamma_1(e_2)\ \mathsf{in}\ (\gamma_1(e_1), x) \to^* (v_{1f_1}, v_{2f_1}) \tag{5.18}$$

$$(\gamma_2(e_1), \gamma_2(e_2)) \to^* (v_{1f_2}, v_{2f_2}) \tag{5.19}$$

$$\mathcal{V}[\![\tau_1 \times \tau_2]\!]_\rho((v_{1f_1}, v_{2f_1}), (v_{1f_2}, v_{2f_2})) \tag{5.20}$$

The derivations are fairly straightforward to show. For the first one (5.18) we have

$$\begin{aligned}&\mathsf{let}\ x = \gamma_1(e_2)\ \mathsf{in}\ (\gamma_1(e_1), x) \equiv (\lambda x.\ (\gamma_1(e_1), x))\ \gamma_1(e_2) \to^* \\ &\quad (\lambda x.\ (\gamma_1(e_1), x))\ v_{2f_1} \to (\gamma_1(e_1), x)[v_{2f_1}/x] \equiv (\gamma_1(e_1), v_{2f_1}) \to^* (v_{1f_1}, v_{2f_1})\end{aligned}$$

For the second one (5.19) we have

$$(\gamma_2(e_1), \gamma_2(e_2)) \to^* (v_{1f_2}, \gamma_2(e_2)) \to^* (v_{1f_2}, v_{2f_2})$$

Finally, (5.20) follows from the last parts of (5.16) and (5.17). ▶**prettify proof**◀ □

5.5 Lam hoisting

The final theorem we shall show is one which is useful, were we to write a compiler for System F. In some compilers, there is a phase which performs "hoisting". Hoisting is the act of lifting invariant declarations out of loops or functions. We don't have loops in our language, so the theorem below shows that it is safe to hoist declarations (let-expressions) out of lambda abstractions.

Theorem 29. *If* $\Xi \mid \Gamma \vdash e_1 : \tau$ *and* $\Xi \mid \Gamma, y : \tau, x : \tau_1 \vdash e_2 : \tau_2$ *then* $\Xi \mid \Gamma \vdash \mathsf{let}\ y = e_1\ \mathsf{in}\ \lambda x.\ e_2 \approx^{ctx} \lambda x.\ \mathsf{let}\ y = e_1\ \mathsf{in}\ e_2 : \tau_1 \to \tau_2$

Proof. So assuming

$$\Xi \mid \Gamma \vdash e_1 : \tau \tag{5.21}$$

$$\Xi \mid \Gamma, y : \tau, x : \tau_1 \vdash e_2 : \tau_2 \tag{5.22}$$

it suffices to show $\Xi \mid \Gamma \vdash \mathsf{let}\ y = e_1\ \mathsf{in}\ \lambda x.\ e_2 \approx^{LR} \lambda x.\ \mathsf{let}\ y = e_1\ \mathsf{in}\ e_2 : \tau_1 \to \tau_2$. Using our two assumptions (5.21 and 5.22), along with the typing rules T-lam and T-app, we may conclude

$$\Xi \mid \Gamma \vdash \mathsf{let}\ y = e_1\ \mathsf{in}\ \lambda x.\ e_2 : \tau_1 \to \tau_2 \tag{5.23}$$

$$\Xi \mid \Gamma \vdash \lambda x.\ \mathsf{let}\ y = e_1\ \mathsf{in}\ e_2 : \tau_1 \to \tau_2 \tag{5.24}$$

So assume some $\rho \in \mathcal{D}[\![\Xi]\!]$ and $\gamma \in \mathcal{G}[\![\Gamma]\!]_\rho$, then we must show $\mathcal{E}[\![\tau_1 \to \tau_2]\!]_\rho(\mathsf{let}\ y = \gamma_1(e_1)\ \mathsf{in}\ \lambda x.\ \gamma_1(e_2), \lambda x.\ \mathsf{let}\ y = \gamma_2(e_1)\ \mathsf{in}\ \gamma_2(e_2))$. The closedness and well-typedness follows from applying the fundamental theorem on the above judgements (5.23 and 5.24), and instantiating with ρ and γ. So what remains to be shown is

$$\begin{aligned}&\exists v_1, v_2.(\mathsf{let}\ y = \gamma_1(e_1)\ \mathsf{in}\ \lambda x.\ \gamma_1(e_2)) \to^* v_1 \wedge \\ &\qquad (\lambda x.\ \mathsf{let}\ y = \gamma_2(e_1)\ \mathsf{in}\ \gamma_2(e_2)) \to^* v_2 \wedge \mathcal{V}[\![\tau_1 \to \tau_2]\!]_\rho(v_1, v_2)\end{aligned} \tag{5.25}$$

From the fundamental theorem on (5.21) instantiated with ρ and γ, we know that

$$\gamma_1(e_1) \to^* v_{y_1} \wedge \gamma_2(e_1) \to^* v_{y_2} \wedge \mathcal{V}[\![\tau]\!]_\rho(v_{y_1}, v_{y_2}) \tag{5.26}$$

for some values v_{y_1}, v_{y_2}. We also get that $\bullet \mid \bullet \vdash \gamma_2(e_1) : \rho_1(\tau)$. In our goal (5.25), we use $\lambda x.\ \gamma_1(e_2)[v_{y_1}/y]$ for v_1 and $\lambda x.\ \mathsf{let}\ y = \gamma_2(e_1)\ \mathsf{in}\ \gamma_2(e_2)$ for v_2. One may see that the first derivation holds by desugaring the let expression into a function application, then using $\gamma_1(e_1) \to^* v_{y_1}$ from (5.26) and the E-lam-app rule to conclude

$$(\mathsf{let}\ y = \gamma_1(e_1)\ \mathsf{in}\ \lambda x.\ \gamma_1(e_2)) \equiv (\lambda y.\ (\lambda x.\ \gamma_1(e_2)))\ \gamma_1(e_1) \to^* (\lambda y.\ (\lambda x.\ \gamma_1(e_2)))\ v_{y_1} \to \lambda x.\ \gamma_1(e_2)[v_{y_1}/y]$$

The second derivation holds trivially, so all we need to show is

$$\mathcal{V}[\![\tau_1 \to \tau_2]\!]_\rho(\lambda x.\ \gamma_1(e_2)[v_{y_1}/y], \lambda x.\ \mathsf{let}\ y = \gamma_2(e_1)\ \mathsf{in}\ \gamma_2(e_2)) \tag{5.27}$$

So we assume some w_1, w_2, and $\mathcal{V}[\![\tau_1]\!]_\rho(w_1, w_2)$, and show

$$\exists v_1, v_2.(\gamma_1(e_2)[v_{y_1}/y])[w_1/x] \to^* v_1 \wedge (\mathsf{let}\ y = \gamma_2(e_1)\ \mathsf{in}\ \gamma_2(e_2))[w_2/x] \to^* v_2 \wedge \mathcal{V}[\![\tau_2]\!]_\rho(v_1, v_2) \tag{5.28}$$

We now use the fundamental theorem on (5.22) and instantiate it with ρ and $\gamma' = \gamma[y \mapsto (v_{y_1}, v_{y_2})][x \mapsto (w_1, w_2)]$. Note that this is fine as both variables map to pairs of values that are in the value interpretation at their respective types. From this we get

$$\gamma'_1(e_2) \to^* v_{f_1} \wedge \gamma'_2(e_2) \to^* v_{f_2} \wedge \mathcal{V}[\![\tau_2]\!]_\rho(v_{f_1}, v_{f_2}) \tag{5.29}$$

For some values v_{f_1}, v_{f_2}. Finally, in our goal (5.28), we use v_{f_1} for v_1, and v_{f_2} for v_2. Firstly, $\mathcal{V}[\![\tau_2]\!]_\rho(v_{f_1}, v_{f_2})$ follows directly from (5.29), so let's show that the derivations hold. The first one holds by

$$\begin{aligned}
(\gamma_1(e_2)[v_{y_1}/y])[w_1/x] &\equiv \gamma_1[y \mapsto v_{y_1}](e_2)[w_1/x] && \text{(substitution lemma)} \\
&\equiv \gamma_1[y \mapsto v_{y_1}][x \mapsto w_1](e_2) && \text{(substitution lemma)} \\
&\equiv \gamma'_1(e_2) \\
&\to^* v_{f_1} && \text{(from (5.29))}
\end{aligned}$$

The second derivation holds by

$$\begin{aligned}
(\mathsf{let}\ y = \gamma_2(e_1)\ \mathsf{in}\ \gamma_2(e_2))[w_2/x] &\equiv \mathsf{let}\ y = \gamma_2(e_1)[w_2/x]\ \mathsf{in}\ \gamma_2(e_2)[w_2/x] \\
&\equiv \mathsf{let}\ y = \gamma_2(e_1)\ \mathsf{in}\ \gamma_2(e_2)[w_2/x] \\
&\to^* \mathsf{let}\ y = v_{y_2}\ \mathsf{in}\ \gamma_2(e_2)[w_2/x] \\
&\to (\gamma_2(e_2)[w_2/x])[v_{y_2}/y] \\
&\equiv \gamma_2[x \mapsto w_2](e_2)[v_{y_2}/y] \\
&\equiv \gamma_2[x \mapsto w_2][y \mapsto v_{y_2}](e_2) \\
&\equiv \gamma_2'(e_2) \\
&\to^* v_{f_2}
\end{aligned}$$

In the first equivalence, we simply push in the substitution. The next equivalence holds as $\gamma_2(e_1)$ is a closed expression. Next we use the fact that $\gamma_2(e_1) \to^* v_{y_2}$. Following that, the let-expression desugars into a function application where both expressions are values, so we use E-lam-app to take a step. Then we apply the substitution lemma twice, and the last step holds by (5.29). □

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] Lau Skorstengaard. An introduction to logical relations. *CoRR*, abs/1907.11133, 2019.

Appendix A

Congruence Rules for Contextual Equivalence

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\textsc{Cng-Tlam}\ \frac{\Xi, X \mid \Gamma \vdash e \approx e' : \tau}{\Xi \mid \Gamma \vdash \Lambda\ e \approx \Lambda\ e' : \forall X.\ \tau} \qquad \textsc{Cng-Tapp}\ \frac{\Xi \mid \Gamma \vdash e \approx e' : \forall X.\ \tau}{\Xi \mid \Gamma \vdash e\ \_ \approx e'\ \_ : \tau\left[\tau'/X\right]}$$

Appendix B

Omitted proofs

B.1 Proof of corollary 6.1

Proof. We proceed by induction on the number of steps of the evaluation $e \Downarrow v_1$.

- Base Case $n = 0$. Since $e \to^0 v_1$, then $e = v_1$, meaning that $v_1 \Downarrow v_2$, which implies that $v_1 = v_2$.
- Inductive Step $n = m + 1$. Our induction hypothesis is $\forall e', v_1', v_2'.e' \to^m v_1' \wedge e' \Downarrow v_2' \implies v_1' = v_2'$. Now, in this case we have that $e \to^{m+1} v_1$. We split this derivation up as follows: $e \to e_1$ and $e_1 \to^m v_1$, for some e_1. Note that this implies that e is not a value. Thus, we may conclude that the evaluation, $e \to^* v_2$, uses at least one step. We split this up as well so that $e \to e_2$ and $e_2 \to^* v_2$, for some e_2. By determinacy (theorem 6), we know that $e_1 = e_2$. Thus $e_1 \to^* v_2$. We may now apply the induction hypothesis, using e_1 for e', v_1 for v_1', and v_2 for v_2', and conclude that $v_1 = v_2$.

□

Appendix C

Additional Results for the Logical Relations Model

Lemma 30. $X \notin \text{dom}(\rho) \wedge X \notin \text{free}(\tau) \implies \forall \tau_1, \tau_2, R \in Rel[\tau_1, \tau_2].\mathcal{V}[\![\tau]\!]_\rho(v_1, v_2) \iff \mathcal{V}[\![\tau]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]}(v_1, v_2)$

Proof. The proof won't be carried out here, but it follows by induction on τ. □

Corollary 30.1. $X \notin \text{dom}(\rho) \wedge X \notin \text{free}(\Gamma) \implies \forall \tau_1, \tau_2, R \in Rel[\tau_1, \tau_2].\mathcal{G}[\![\Gamma]\!]_\rho = \mathcal{G}[\![\Gamma]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]}$

Proof. Assuming $X \notin \text{dom}(\rho)$, $X \notin \text{free}(\Gamma)$, and some τ_1, τ_2, and $R \in Rel[\tau_1, \tau_2]$, we must show $\mathcal{G}[\![\Gamma]\!]_\rho = \mathcal{G}[\![\Gamma]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]}$. We proceed by induction on Γ. The base case is trivial: $\mathcal{G}[\![\bullet]\!]_\rho = \mathcal{G}[\![\bullet]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]} = \{\emptyset\}$. So let's show the inductive step. Our induction hypothesis is $\mathcal{G}[\![\Gamma]\!]_\rho = \mathcal{G}[\![\Gamma]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]}$, and we must show $\mathcal{G}[\![\Gamma, x : \tau]\!]_\rho = \mathcal{G}[\![\Gamma, x : \tau]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]}$, which is equivalent to showing $\gamma \in \mathcal{G}[\![\Gamma, x : \tau]\!]_\rho \iff \gamma \in \mathcal{G}[\![\Gamma, x : \tau]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]}$, for all γ. Both directions of the double implication are similar, so we show just the " $\implies$ " direction. So assuming $\gamma \in \mathcal{G}[\![\Gamma, x : \tau]\!]_\rho$, we must show $\gamma \in \mathcal{G}[\![\Gamma, x : \tau]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]}$. From $\gamma \in \mathcal{G}[\![\Gamma, x : \tau]\!]_\rho$ we may conclude that $\gamma = \gamma'[x \mapsto (v_1, v_2)]$, and that $\gamma' \in \mathcal{G}[\![\Gamma]\!]_\rho$ and $\mathcal{V}[\![\tau]\!]_\rho(v_1, v_2)$. From this and lemma 30 we know that $\mathcal{V}[\![\tau]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]}(v_1, v_2)$, and from our induction hypothesis, we get $\gamma' \in \mathcal{G}[\![\Gamma]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]}$. So by the definition of $\mathcal{G}$, we may conclude $\gamma \in \mathcal{G}[\![\Gamma, x : \tau]\!]_{\rho[X \mapsto (\tau_1, \tau_2, R)]}$. □

Lemma 31 (Substitution lemma LR)**.** *If* $\Xi \mid \Gamma \vdash e : \tau$ *and* $\rho \in \mathcal{D}[\![\Xi]\!]$ *and* $\gamma \in \mathcal{G}[\![\Gamma]\!]_\rho$*, then* $\bullet \mid \bullet \vdash \gamma_1(e) : \rho_1(\tau)$ *and* $\bullet \mid \bullet \vdash \gamma_2(e) : \rho_2(\tau)$

Proof. This follows fairly easily from the fundamental theorem: by the fundamental theorem on $\Xi \mid \Gamma \vdash e : \tau$, we have $\forall \rho \in \mathcal{D}[\![\Xi]\!], \gamma \in \mathcal{G}[\![\Gamma]\!]_\rho.\mathcal{E}[\![\tau]\!]_\rho(\gamma_1(e), \gamma_2(e))$. Using our ρ and γ, we get $\mathcal{E}[\![\tau]\!]_\rho(\gamma_1(e), \gamma_2(e))$. By definition of the expression interpretation, this means that $\bullet \mid \bullet \vdash \gamma_1(e) : \rho_1(\tau)$ and $\bullet \mid \bullet \vdash \gamma_2(e) : \rho_2(\tau)$, which was what we wanted. □