

---

# Relational Reasoning (Relationel ræsonnement)

Mathias Pedersen, 201808137

---

Bachelor Report (15 ECTS) in Computer Science

Advisor: Amin Timany

Department of Computer Science, Aarhus University

September 2021



AARHUS  
UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE



# Abstract

► in English... ◄

*Mathias Pedersen,  
Aarhus, September 2021.*



# Contents

<b>Abstract</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Definition of Language</b>	<b>3</b>
<b>3 Contextual Equivalence</b>	<b>9</b>
<b>4 Logical Relations for Contextual Equivalence</b>	<b>11</b>
<b>5 Examples of Application of Contextual Equivalence</b>	<b>13</b>
<b>6 Comparison to Other Work and Ideas for Future Work</b>	<b>15</b>
<b>7 Conclusion</b>	<b>17</b>
<b>Acknowledgments</b>	<b>19</b>
<b>Bibliography</b>	<b>21</b>
<b>A The Technical Details</b>	<b>23</b>



# Chapter 1

## Introduction

►motivate and explain the problem to be addressed◄

►example of a citation: [1]◄ ►get your bibtex entries from <https://dblp.org/>◄





## Chapter 2

# Definition of Language

►create draft◄

Syntax

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

### Typing rules

$\frac{\text{T-VAR} \quad (x : \tau) \in \Gamma}{\Xi \mid \Gamma \vdash x : \tau}$	$\frac{\text{T-UNIT}}{\Xi \mid \Gamma \vdash () : \text{Unit}}$	$\frac{\text{T-INT}}{\Xi \mid \Gamma \vdash \bar{n} : \mathbb{Z}}$
$\frac{\text{T-ADD} \quad \Xi \mid \Gamma \vdash e_1 : \mathbb{Z} \quad \Xi \mid \Gamma \vdash e_2 : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 + e_2 : \mathbb{Z}}$	$\frac{\text{T-SUB} \quad \Xi \mid \Gamma \vdash e_1 : \mathbb{Z} \quad \Xi \mid \Gamma \vdash e_2 : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 - e_2 : \mathbb{Z}}$	
$\frac{\text{T-LE} \quad \Xi \mid \Gamma \vdash e_1 : \mathbb{Z} \quad \Xi \mid \Gamma \vdash e_2 : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 \leq e_2 : \mathbb{B}}$	$\frac{\text{T-LT} \quad \Xi \mid \Gamma \vdash e_1 : \mathbb{Z} \quad \Xi \mid \Gamma \vdash e_2 : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 < e_2 : \mathbb{B}}$	
$\frac{\text{T-EQ} \quad \Xi \mid \Gamma \vdash e_1 : \mathbb{Z} \quad \Xi \mid \Gamma \vdash e_2 : \mathbb{Z}}{\Xi \mid \Gamma \vdash e_1 = e_2 : \mathbb{B}}$	$\frac{\text{T-TRUE}}{\Xi \mid \Gamma \vdash \text{true} : \mathbb{B}}$	$\frac{\text{T-FALSE}}{\Xi \mid \Gamma \vdash \text{false} : \mathbb{B}}$
$\frac{\text{T-IF} \quad \Xi \mid \Gamma \vdash e_1 : \mathbb{B} \quad \Xi \mid \Gamma \vdash e_2 : \tau \quad \Xi \mid \Gamma \vdash e_3 : \tau}{\Xi \mid \Gamma \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : \tau}$		
$\frac{\text{T-PAIR} \quad \Xi \mid \Gamma \vdash e_1 : \tau_1 \quad \Xi \mid \Gamma \vdash e_2 : \tau_2}{\Xi \mid \Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2}$	$\frac{\text{T-FST} \quad \Xi \mid \Gamma \vdash e : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \text{fst } e : \tau_1}$	$\frac{\text{T-SND} \quad \Xi \mid \Gamma \vdash e : \tau_1 \times \tau_2}{\Xi \mid \Gamma \vdash \text{snd } e : \tau_2}$
$\frac{\text{T-INJ1} \quad \Xi \mid \Gamma \vdash e : \tau_1}{\Xi \mid \Gamma \vdash \text{inj}_1 e : \tau_1 + \tau_2}$	$\frac{\text{T-INJ2} \quad \Xi \mid \Gamma \vdash e : \tau_2}{\Xi \mid \Gamma \vdash \text{inj}_2 e : \tau_1 + \tau_2}$	
$\frac{\text{T-MATCH} \quad \Xi \mid \Gamma \vdash e_1 : \tau_1 + \tau_2 \quad \Xi \mid \Gamma, x : \tau_1 \vdash e_2 : \tau \quad \Xi \mid \Gamma, x : \tau_2 \vdash e_3 : \tau}{\Xi \mid \Gamma \vdash \text{match } e_1 \text{ with } \text{inj}_1 x \Rightarrow e_2 \mid \text{inj}_2 x \Rightarrow e_3 \text{ end} : \tau}$		
$\frac{\text{T-REC} \quad \Xi \mid \Gamma, f : \tau_1 \rightarrow \tau_2, x : \tau_1 \vdash e : \tau_2}{\Xi \mid \Gamma \vdash \text{rec } f(x) := e : \tau_1 \rightarrow \tau_2}$	$\frac{\text{T-APP} \quad \Xi \mid \Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \quad \Xi \mid \Gamma \vdash e_2 : \tau_1}{\Xi \mid \Gamma \vdash e_1 e_2 : \tau_2}$	
$\frac{\text{T-TLAM} \quad \Xi, X \mid \Gamma \vdash e : \tau}{\Xi \mid \Gamma \vdash \Lambda e : \forall X. \tau}$	$\frac{\text{T-TAPP} \quad \Xi \mid \Gamma \vdash e : \forall X. \tau}{\Xi \mid \Gamma \vdash e \_ : \tau[\tau'/X]}$	

## Dynamics

$$\begin{array}{c}
\text{HEAD-STEP-STEP} \\
\frac{e \rightarrow_h e'}{K[e] \rightarrow K[e']} \\
\\
\begin{array}{ccc}
\text{E-ADD} & \text{E-SUB} & \text{E-EQ} \\
\frac{}{\overline{n_1} + \overline{n_2} \rightarrow_h \overline{n_1 + n_2}} & \frac{}{\overline{n_1} - \overline{n_2} \rightarrow_h \overline{n_1 - n_2}} & \frac{n_1 = n_2}{\overline{n_1} = \overline{n_2} \rightarrow_h \text{true}} \\
\\
\begin{array}{cccc}
\text{E-NOT-EQ} & \text{E-LE} & \text{E-NOT-LE} & \text{E-LT} \\
\frac{n_1 \neq n_2}{\overline{n_1} = \overline{n_2} \rightarrow_h \text{false}} & \frac{n_1 \leq n_2}{\overline{n_1} \leq \overline{n_2} \rightarrow_h \text{true}} & \frac{n_1 \not\leq n_2}{\overline{n_1} \leq \overline{n_2} \rightarrow_h \text{false}} & \frac{n_1 < n_2}{\overline{n_1} < \overline{n_2} \rightarrow_h \text{true}} \\
\\
\text{E-NOT-LT} & \text{E-IF-TRUE} & \text{E-IF-FALSE} \\
\frac{n_1 \not< n_2}{\overline{n_1} < \overline{n_2} \rightarrow_h \text{false}} & \text{if true then } e_2 \text{ else } e_3 \rightarrow_h e_2 & \text{if false then } e_2 \text{ else } e_3 \rightarrow_h e_3 \\
\\
\begin{array}{cc}
\text{E-FST} & \text{E-SND} \\
\text{fst}(v_1, v_2) \rightarrow_h v_1 & \text{snd}(v_1, v_2) \rightarrow_h v_2 \\
\\
\text{E-MATCH-INJ1} \\
\text{match}(\text{inj}_1 v) \text{ with } \text{inj}_1 x \Rightarrow e_2 \mid \text{inj}_2 x \Rightarrow e_3 \text{ end} \rightarrow_h e_2[v/x] \\
\\
\text{E-MATCH-INJ2} \\
\text{match}(\text{inj}_2 v) \text{ with } \text{inj}_1 x \Rightarrow e_2 \mid \text{inj}_2 x \Rightarrow e_3 \text{ end} \rightarrow_h e_3[v/x] \\
\\
\begin{array}{cc}
\text{E-REC-APP} & \text{E-TAPP-TLAM} \\
(\text{rec } f(x) := e)v \rightarrow_h e[\text{rec } f(x) := e/f][v/x] & (\Lambda e) \_ \rightarrow_h e
\end{array}
\end{array}
\end{array}$$

**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 for case E-ADD.

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.  $\square$

**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

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

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

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

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:

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

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:

- $K = H$  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

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

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

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

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

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

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

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

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  $\text{Val}(e)$ . Now, pick  $e$  for  $e''$  in 2.1. We must show

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

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

□



## Chapter 3

# Contextual Equivalence

►draft◄

►fix unfinished text passage - start◄

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

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

Now if  $\tau$  was `bool` instead, the context  $\text{if } \square \text{ then } () \text{ else } \omega$  would suffice in showing that  $v_2 = v_1$ .

►fix unfinished text passage - end◄





## **Chapter 4**

# **Logical Relations for Contextual Equivalence**

►draft◄



## **Chapter 5**

# **Examples of Application of Contextual Equivalence**

►draft◄



## **Chapter 6**

# **Comparison to Other Work and Ideas for Future Work**

►draft◄



## Chapter 7

# Conclusion

►conclude on the problem statement from the introduction◄





# Acknowledgments





# Bibliography

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



## **Appendix A**

# **The Technical Details**

