Extension of a Language Combining Proofs and Programs with Polymorphic Types

A thesis submitted
in Partial Fulfillment of the Requirements
for the Degree of

Master of Technology

by

Ankit Kumar

to the

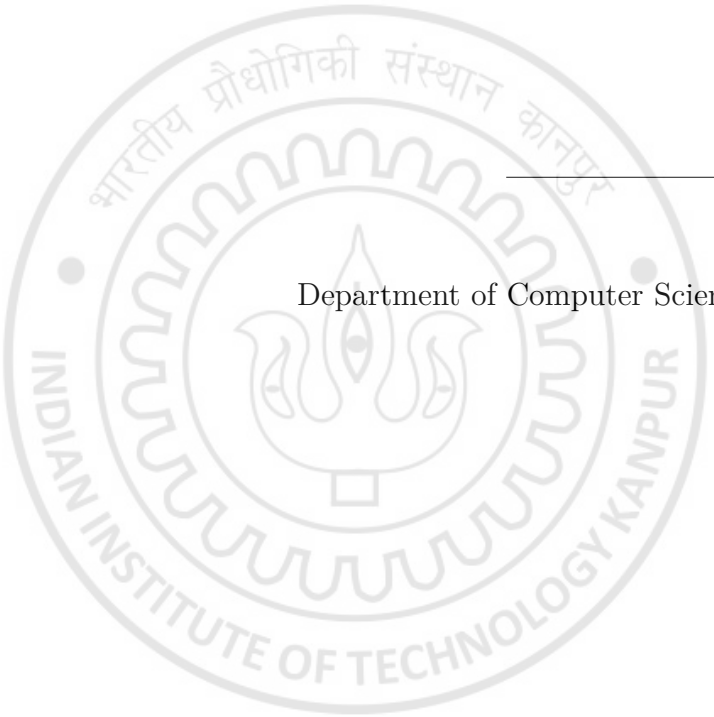
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

May, 2017

भारतीय प्रौद्योगिकी संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

CERTIFICATE

It is certified that the work contained in the thesis titled **Extension of a Language Combining Proofs and Programs with Polymorphic Types**, by **Ankit Kumar**, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.



Dr Anil Seth
Department of Computer Science & Engineering
IIT Kanpur

May, 2017

भारतीय प्रौद्योगिकी संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

DECLARATION

The results reported in chapter 5 are a joint work with my thesis supervisor Dr. Anil Seth.

The overall development of section 5.2 is due to him. In particular, notion of SIRC (definition 5.5) is his. Formulation of lemma 5.7, 5.8, 5.9 are due to him. Lemma 5.11 and its proof is by him. Definition 5.12 and formulation of Theorem 5.13 are due to him.

I have worked out various cases in the proof of Theorem 5.13 as well as parts of proofs of some of the lemma above.

भारतीय प्रौद्योगिकी संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

ABSTRACT

Name of student: **Ankit Kumar** Roll no: **15111010**

Degree for which submitted: **Master of Technology**

Department: **Computer Science & Engineering**

Thesis title: **Extension of a Language Combining Proofs and Programs with Polymorphic Types**

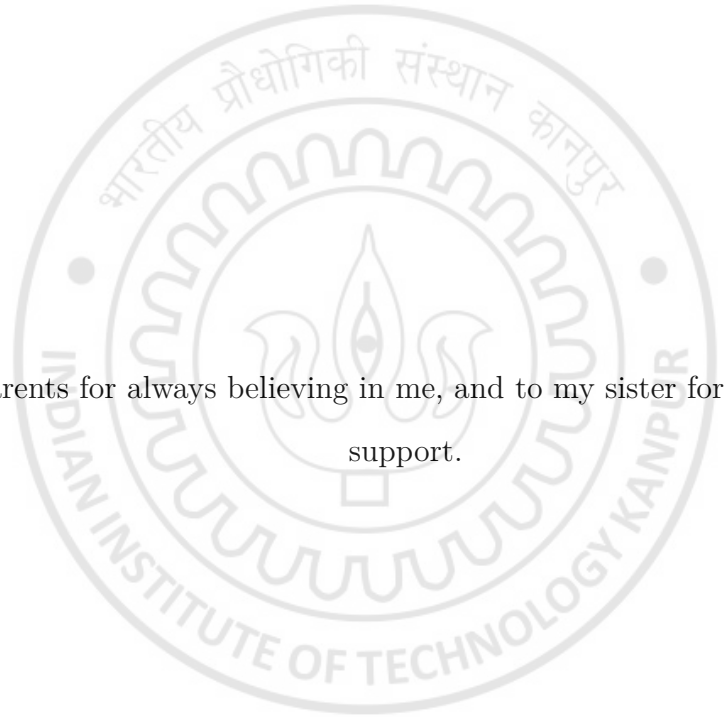
Name of Thesis Supervisor: **Dr Anil Seth**

Month and year of thesis submission: **May, 2017**

In their effort to create a practical dependently typed language, Casinghino, Sjoberg and Weirich gave the type system of an example language L^θ in their MSFP 2012 paper. This language has 2 fragments, logical and programmatic. The logical fragment is consistent but restrictive in that it allows the coding of only terminating programs and hence can be used to prove properties whereas the programmatical fragment is inconsistent, but turing complete, allowing full computational power to the programmer. Interaction between the two fragments is limited by the typing rules. In this paper, they have proved the soundness of the logical fragment of this type system.

In this thesis, we extend the type system of the MSFP paper above with second order polymorphism (as in system F) and show normalization for logical fragment and type safety for the programmatic fragment.

भारतीय प्रौद्योगिकी संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY KANPUR



To my parents for always believing in me, and to my sister for her continuous support.

भारतीय प्रौद्योगिकी संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

Acknowledgements

I would like to thank my thesis advisor Dr. Anil Seth for his guidance and patience throughout this project. I would also like to thank Prof. Piyush Kurur for introducing me to proving theorems with proof assistants.

Thanks must also go to Chris Casinghino, with whom I had detailed correspondence to understand some fine details in his MSFP 2012 paper [1] which helped me a lot in getting started with this project.

And lastly, I thank the faculty, staff and friends at Department of Computer Science and Engineering, IIT Kanpur, who have helped make this a memorable journey.

भारतीय प्रौद्योगिकी संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

Contents

List of Figures

भारतीय प्रौद्योगिकी संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

Chapter 1

Introduction

1.1 Motivation

Any language designed for proof assistance needs to have a consistent type system. A consistent type system is one in which some types are inhabited while some are uninhabited, or by the Curry Howard correspondence, some properties can be proved in this system, while some can't.

Looping terms pose a problem in such systems. Consider an example from [8], where for an arbitrary proposition P, we can write

```
Fixpoint bad (u : unit) : P := bad u.
```

In the above code in Coq language, a fixpoint over the whole expression has been defined (:= stands for assignment), and assigned the type P (by :). **bad tt** would trivially give us the proof of any P, even **False**. This is because if the type checker is unable to infer the type of an expression, the expression is given a minimal type (let's say $\perp$). This is why, only total function definitions are accepted by proof assistants like Coq or Agda.

However, such limitations on the language (disallowing infinite computations), would be too restrictive for it to be of any practical use. The L^{θ} language proposed by Casinghino et al. in their MSFP 2012 paper [1] is an effort to allow programmers to code in a haskell like language, with no restrictions and benefit from the ability

to prove properties about their code as well. Programs which may not terminate are tagged with **prog** whereas programs that represent proofs and terminate are tagged with **log**.

1.2 L^θ examples

Consider the program shown below.

```
prog div : Nat -> Nat -> Nat
prog div n m = if n < m then 0 else 1 + (div (n - m) m)
```

This code loops forever if m is passed a value of 0, hence, this program has been annotated as prog. A programmer would like to prove properties about the program he has written, such as :

```
log div_le : (n:Nat) -> (m:Nat) -> (eq m 0 = False) -> (le (div n m) n = True)
```

i.e. if the divisor is not 0, then the result is less than the dividend. Although this example uses dependent types such as (eq m 0 = False), the L^θ language is only simply typed and thus the above example can't be coded in it. However it is complex enough to exhibit the difficulties faced in proving normalization, which one would encounter as one further scales up.

Programs can also return proofs as witness to the correctness of the result they are returning. Consider

```
prog solver : (f : Formula) -> Maybe ((Satisfiable f) @ L)
```

In the above example, a SAT solver takes as input a formula, and optionally produces a proof which is satisfiable. We know the returned term is provably satisfiable as it has been type checked in the L fragment, even though the solver itself is written in the P fragment.

1.3 Programming in LP^{θ}

With polymorphic types, the language becomes very expressive, and we can do a lot more, than we could earlier. Few examples are shown below.

```
prog map : forall a,b.(a -> b)@L -> [a] -> [b]
```

A polymorphic map function, which requires a proof that the function being mapped is total, to ensure that all terms are mapped.

```
prog sort : forall t.((t,t) -> Bool)@L -> [t] ->[t]
```

A polymorphic sort function that requires the comparison function passed to be provably total.

1.4 Our Contribution

We have extended the language L^{θ} with second order polymorphism, which allows variables ranging over the class of all types, thus making the language highly expressive. Subsequently we have shown normalization for the logical fragment and type safety for the programmatic fragment.

भारतीय प्रौद्योगिकी संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

Chapter 2

Step Indexed Logical Relations

2.1 Declaration

The contents of this chapter are from Amal Ahmed's lectures on logical relations at the Oregon Programming Languages Summer School 2015 [10]

2.2 Logical Relations

The term "logical relations" was first introduced by Gordon Plotkin in his memorandum *Lambda- definability and logical relations* in 1973. However the first use of the idea of a logical relation is usually attributed to Tait's 1967 JSL paper *"Intensional interpretation of functionals of finite type I*, where he used it to prove strong normalization of System T.

The analogy between types and logics, also called the Curry Howard isomorphism allows us to view types as logical formulae (times as conjunction, $\rightarrow$ as implication etc..). Logical Relations are named so because (i) they are relations and

(ii) which treat type constructors as logical connectives, hence "Logical".

Logical relations are proof techniques that can be used to prove properties about languages like normalization, type safety, program equivalence and are closed under elimination. Hence they can be used to prove properties which are not closed under

elimination, such as termination in the Simply Typed Lambda Calculus (STLC).

Mathematically an n-ary logical relation is a family $R = \{R_\theta\}_{\theta \in Types}$ of n-ary relations such that $R_\theta \subseteq \underbrace{[\![\theta]\!] \times \cdots \times [\![\theta]\!]}_{\text{n times}}$ for any θ and

$$R_{\theta_1 \to \theta_2}(f_1, ..., f_n) \Leftrightarrow \forall (d_1, ..., d_n) \in [\![\theta_1]\!]^n,\ if\ R_{\theta_1}()\ then\ R_{\theta_2}(f_1(d_1), ..., f_n(d_n)).$$

2.3 Strong Normalization and Type Safety of STLC

In order to motivate the usage of logical relations, we look at the following use case, where we need to prove that all typed terms of the Simply Typed Lambda Calculus terminate.

Types: $\tau := bool \mid \tau \to \tau$

Expressions: $e := x \mid true \mid false \mid \lambda x : \tau.e \mid ee$

Values: $v := true \mid false \mid \lambda x : \tau.e$

Evaluation Contexts: $E := [] \mid Ee \mid vE \mid$

Operational Semantics: $\dfrac{e \longrightarrow e'}{E[e] \longrightarrow E[e]'}$ $\qquad$ $\dfrac{}{(\lambda x : \tau.e)v \longrightarrow [v/x]e}$

Typing Contexts: $\Gamma \vdash \bullet \mid \Gamma, x : \tau$

Typing Rules: $\dfrac{}{\Gamma \vdash true : bool}$ T-TRUE $\qquad$ $\dfrac{}{\Gamma \vdash false : bool}$ T-FALSE

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}\text{T-VAR} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau_2}\text{T-APP}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1.e : \tau_1 \to \tau_2}\text{T-ABS}$$

Figure 2.1: Simply Typed Lambda Calculus Syntax

Theorem 2.1 *(Strong Normalization) If* $\bullet \vdash e : \tau$ *then* $e \Downarrow$ *where* $e \Downarrow v \triangleq e \rightsquigarrow^* v$, *where* v *is a value and* $e \Downarrow \triangleq \exists v(e \rightsquigarrow^* v)$.

Proof attempt 1: Proof by induction on the typing derivation.

T-True $\bullet \vdash true : bool$: true is a value already

T-Abs $\bullet \vdash \lambda x : \tau_1.e : \tau_1 \rightarrow \tau_2$: lambda expressions are values already.

T-App $\dfrac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau_2}$

By induction hypothesis, we have $e_1 \Downarrow$ and $e_2 \Downarrow$. We need to show that $e_1 e_2 \Downarrow$.

Consider the reduction steps for application:

$e_1 e_2 \rightarrow {}^*(\lambda x : \tau_2.e')e_2 \rightarrow {}^*(\lambda x : \tau_2.e')v_2 \rightarrow {}^*[v_2/x]e'$

We have no knowledge about e' as our induction hypothesis is weak.

2.3.1 Unary Predicate for Strong Normalization

We define $SN_\tau(e)$ as a unary predicate that accepts only those terms of type τ that reduce to a value. Hence

$SN_{bool}(e) \Leftrightarrow \bullet \vdash e : bool \wedge e \Downarrow$

$SN_{\tau_1 \rightarrow \tau_2}(e) \Leftrightarrow \bullet \vdash e : \tau_1 \rightarrow \tau_2 \wedge e \Downarrow \wedge \forall e'(SN_{\tau_1}(e') \Rightarrow SN_{\tau_2}(ee'))$

The predicate is well founded as it is defined over the structure of types.

Proof of normalization splits into 2 parts:

1. $\bullet \vdash e : \tau \Rightarrow SN_\tau(e)$
2. $SN_\tau(e) \Rightarrow e \Downarrow$

We generalize the first statement to account for open terms (which may contain term variables).

If $\Gamma \vdash e : \tau \wedge \rho \vDash \Gamma$ **then** $SN_\rho(\rho(e))$ where ρ is defined as the substitution mapping $\rho = \{x_1 \longrightarrow v_1, ..., x_n \longrightarrow v_n\}$. $\rho \vDash \Gamma$ means that the substitution ρ satisfies the type environment Γ and is defined as

$\rho \vDash \Gamma \triangleq dom(\rho) = dom(\Gamma) \wedge \forall x \in dom(\rho), SN_{\Gamma x}(\rho(x))$.

Proof depends on 2 lemmas, which are mentioned below :

Lemma 2.2 *(Substitution Lemma) If* $\Gamma \vdash e : \tau$ *and* $\gamma \vDash \Gamma$*, then* $\bullet \vdash \gamma(e) : \tau$

Lemma 2.3 *(Forward/Backward reduction Preserve SN) Let* $\bullet \vdash e : \tau$ *and* $e \rightsquigarrow e'$*, then*

1. *if* $SN(e)$, *then* $SN(e')$

2. *if* $SN(e')$, *then* $SN(e)$

We are now ready to prove the Strong Normalization theorem.

The first part is proved by induction on the typing derivation. Let us examine the T-App case, where we got stuck in the earlier proof attempt:

T-APP $\dfrac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau_2}$

Given $\rho \vDash \Gamma$, prove that $SN_{\tau_2}(\rho(e_1 e_2))$.

Proof: By induction hypothesis, we have $SN_{\tau_1 \rightarrow \tau_2}(\rho(e_1))$ and $SN_{\tau_1}(\rho(e_2))$. We need to show that $SN_{\tau_2}(e_1 e_2)$.

From definition of $SN_{\tau_1 \rightarrow \tau_2}(\rho(e_1))$, we have $\bullet \vdash \rho(e_1) : \tau_1 \rightarrow \tau_2 \wedge \rho(e_1) \Downarrow \wedge \forall e'(SN_{\tau_1}(e') \Rightarrow SN_{\tau_2}(\rho(e)e'))$. Instantiating e' with $\rho(e_2)$, which we know from induction hypothesis to be $SN_{\tau_1}(\rho(e_2))$, we get $SN_{\tau_2}(\rho(e_1 e_2))$. □

2.3.2 Type Safety with logical relations

Type safety for STLC is defined by 2 lemmas :

Lemma 2.4 *(Progress) If* $\bullet \vdash e : \tau$, *then* $Val(e)$ *or* $\exists e'(e \rightarrow e')$.

Generally we prove progress by induction on the typing derivation.

Lemma 2.5 *(Preservation) If* $\bullet \vdash e : \tau$, *and* $e \rightarrow e'$ *then* $\bullet \vdash e' : \tau$

Generally we prove preservation by induction on the evaluation.

In order to prove type safety with logical predicates, we define the value and expression interpretation of types:

$\mathcal{V}[\![bool]\!] = \{true, false\}$

$\mathcal{V}[\![\tau_1 \rightarrow \tau_2]\!] = \{\lambda x : \tau_1.e \mid \forall v \in \mathcal{V}[\![\tau_1]\!]([v/x]e \in \mathcal{E}[\![\tau_2]\!])\}$

Expression interpretations are defined as:

$\mathcal{E}[\![\tau]\!] = \{e \mid \forall e'(e \rightsquigarrow^* e') \wedge irred(e') \Rightarrow e' \in \mathcal{V}[\![\tau]\!]\}$ where $irred(e) \triangleq \nexists e'.e \rightarrow e'$

Again, we define a predicate $safe$ such that it contains only those terms that either reduce to a value or step to another term i.e.

$safe(e) \triangleq \forall e'(e \rightarrow {}^* e') \Rightarrow Val(e') \vee \exists e''(e' \rightarrow e'')$

We also define the interpretation of the typing environments as follows:

$\mathcal{G}[\![\bullet]\!] = \{\phi\}$

$\mathcal{G}[\![\Gamma, x : \tau]\!] = \{\rho[x \rightarrow v] \mid \rho \in \mathcal{G}[\![\Gamma]\!] \wedge v \in \mathscr{V}[\![\tau]\!]\}$

In order to prove type safety, we will also require a substitution lemma:

Lemma 2.6 *(Substitution) Let e be syntactically well-formed term, v be a value and ρ be a substitution that map term variables to closed values, and let $x \not\in dom\rho$, then* $\rho[x \rightarrow v](e) = [x/v]\rho(e)$

Proof: By induction on the size of ρ □

We are now ready to prove type safety, in two steps:

1. $\bullet \vdash e : \tau \Rightarrow e \in \mathcal{E}[\![\tau]\!]$, which on generalizing becomes

   If $\Gamma \vdash e : \tau$ **then** $\Gamma \vDash e : \tau$, where we define semantic type safety as:

   $\Gamma \vDash e : \tau \triangleq \forall \rho \in \mathcal{G}[\![\Gamma]\!].\rho(e) \in \mathcal{E}[\![\tau]\!]$

2. $\bullet \vdash e : \tau \Rightarrow safe(e)$

Proof: First part is proved by induction on the typing judgement. We consider an interesting case: **T-ABS** $\dfrac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1.e : \tau_1 \rightarrow \tau_2}$

Given $\rho \in \mathcal{G}[\![\Gamma]\!]$, we need to show that

$\Gamma \vDash \rho(\lambda x : \tau_1.e) \in \mathcal{E}[\![\tau_1 \rightarrow \tau_2]\!] \equiv \lambda x : \tau_1.\rho(e) \in \mathcal{E}[\![\tau_1 \rightarrow \tau_2]\!]$.

Let $\lambda x : \tau_1.\rho(e) \rightsquigarrow^* e' \wedge irred(e')$. Now our proof obligation is to show that $e' \in \mathcal{V}[\![\tau_1 \rightarrow \tau_2]\!]$. Now, as $\lambda x : \tau_1.\rho(e)$ is a value, it is by definition irreducible, and hence $\lambda x : \tau_1.\rho(e) = e'$.

So, we need to show that $\lambda x : \tau_1.\rho(e) \in \mathcal{V}[\![\tau_1 \rightarrow \tau_2]\!]$. Or, for a $v \in \mathcal{V}[\![\tau_1]\!]$, we need to show that $[v/x]\rho(e) \in \mathcal{E}[\![\tau_2]\!]$.

By induction hypothesis, we have: $\Gamma, x : \tau_1 \vDash e : \tau_2$

Instantiating with a substitution $\rho[x \rightarrow v]$, we get $\rho[x \rightarrow v] \in \mathcal{G}[\![\Gamma]\!]$ (as by I.H. $\rho \in \mathcal{G}[\![\Gamma]\!]$ and $v \in \mathcal{V}[\![\tau_1]\!]$).

Due to this instantiation $\rho[x \rightarrow v](e) \in \mathcal{E}[\![\tau_2]\!] \equiv [v/x]\rho(e)$ (by Substitution lemma).

□

2.4 Step Indexed Logical Relations

In the previous section we observed how logical relations help us when we get stuck due to weak hypotheses. We will now see whether logical relations alone can suffice for all cases.

2.4.1 Recursive types and Step Indexing in STLC

We extend our previous definition of STLC (2.1) with recursive type $\mu\alpha.\tau$

$$
\begin{array}{ll}
\tau := & \ldots \mid \mu\alpha.\tau \\
e := & \ldots \mid fold\ e \mid unfold\ e \\
v := & \ldots \mid fold\ v \\
E := & \ldots \mid fold\ E \mid unfold\ E
\end{array}
$$

$$
\frac{}{unfold(fold\ v) \rightarrow v} \qquad \frac{\Gamma \vdash e : [\mu\alpha.\tau/\alpha]\tau}{fold\ e : \mu\alpha.\tau}\text{T-FOLD}
$$

$$
\frac{\Gamma \vdash e : \tau[\mu\alpha.\tau]}{unfold\ e : [\mu\alpha.\tau/\alpha]\tau}\text{T-UNFOLD}
$$

Figure 2.2: STLC syntax extension by recursive type

Similarly, extending our value interpretations with the recursive type, we get

$\mathcal{V}[\![\mu\alpha.\tau]\!] = \{fold\ v \mid unfold(fold\ v) \in \mathcal{V}[\![\mu\alpha.\tau/\alpha]\!]\}$

$\equiv \mathcal{V}[\![\mu\alpha.\tau]\!] = \{fold\ v \mid v \in \mathcal{V}[\![[\mu\alpha.\tau/\alpha]\tau]\!]\}$

Immediately we see there is a problem. Our value interpretation is no longer well

founded, since it was defined by induction on types. But now, for this interpretation, a smaller type $\mu\alpha.\tau$ is now depending on a larger type $[\mu\alpha.\tau/\alpha]\tau$.

This problem is solved by adding an index k to the value interpretations as shown:

$\mathcal{V}_k[\![bool]\!] = \{true, false\}$

$\mathcal{V}_k[\![\tau_1 \rightarrow \tau_2]\!] = \{\lambda x : \tau_1.e \mid \forall j \leq k(\forall v \in \mathcal{V}_j[\![\tau_1]\!]([v/x]e \in \mathcal{E}_j[\![\tau_2]\!]))\}$

Intuitively, we are making sure that for the next k steps for which an expression e runs, its context can not detect that the type of e is not τ. However, this surity will vanish after the said k steps of run have elapsed. Using step indexing, we now give a new value interpretation for the recursive type

$\mathcal{V}_k[\![\mu\alpha.\tau]\!] = \{fold\ v \mid \forall j \leq k(v \in \mathcal{V}_j[\![[\mu\alpha.\tau/\alpha]\tau]\!])\}$

Thus, we now base our well foundedness on the decreasing pair (k, τ). The updated expression interpretation

$\mathcal{E}_k[\![\tau]\!] = \{e \mid \forall j \leq k(\forall e'(e \rightsquigarrow^j e') \wedge irred(e') \Rightarrow e' \in \mathcal{V}_{k-j}[\![\tau]\!])\}$

In effect, the expression interpretation "counts" the number of steps elapsed in the reduction of expression e to e'.

We will need the Monotonicity lemma, in order to prove the type safety theorem.

Lemma 2.7 *(Monotonicity) If $v \in \mathcal{V}_k[\![\tau]\!]$ and $j \leq k$, then $v \in \mathcal{V}_j[\![\tau]\!]$.*

Proof: By case on τ. We choose the recursive type.

$\tau = \mu\alpha.\tau$

Let $v \in \mathcal{V}_k[\![\mu\alpha.\tau]\!]$ and $j \leq k$. Then we need to show that $v \in \mathcal{V}_j[\![\mu\alpha.\tau]\!]$.

From definition of membership of v in its type interpretation, $\exists v'$ s.t. $v = fold\ v'$.

Choose $i < j$, thus $i < k$. Then we need to show that $v' \in \mathcal{V}_i[\![[\mu\alpha.\tau/\alpha]\tau]\!]$.

Then by induction hypothesis on v', $\forall n < k(v' \in \mathcal{V}_n[\![[\mu\alpha.\tau/\alpha]\tau]\!])$.

Instantiate with i in I.H. for v', we get $v' \in \mathcal{V}_i[\![[\mu\alpha.\tau/\alpha]\tau]\!]$. □

Our interpretation of tying environments is augmented with step-indexing

$\mathcal{G}_k[\![\bullet]\!] = \{\phi\}$

$\mathcal{G}_k[\![\Gamma, x : \tau]\!] = \{\rho[x \rightarrow v] \mid \rho \in \mathcal{G}_k[\![\Gamma]\!] \wedge v \in \mathscr{V}_k[\![\tau]\!]\}$

thus changing the definition of semantic type safety to

$\Gamma \vDash e : \tau \triangleq \forall k \geq 0(\forall \rho \in \mathcal{G}_k[\![\Gamma]\!](\rho(e) \in \mathcal{E}_k[\![\tau]\!]))$

We now prove the first part of our type safety theorem.

Proof: By induction over the typing judgement. We consider one case.

T-FOLD $\dfrac{\Gamma \vdash e : [\mu\alpha.\tau/\alpha]\tau}{fold\ e : \mu\alpha.\tau}$

Given $\rho \in \mathcal{G}_k[\![\mu\alpha.\tau]\!]$, we need to show that

$\rho(fold\ e) \in \mathcal{E}_k[\![\mu\alpha.\tau]\!] \equiv fold\ \rho(e) \in \mathcal{E}_k[\![\mu\alpha.\tau]\!]$.

Let $fold\ e \rightsquigarrow^j e' \wedge irred(e')$ for some $j \leq k$.

Then we need to show that $e' \in \mathcal{V}_{k-j}[\![\mu\alpha.\tau]\!]$.

As $fold\ \rho(e)$ reduced down to something irreducible and the operational semantics of STLC are deterministic, $\rho(e)$ must also have reduced to some irreducible value. So, $\rho(e) \rightsquigarrow^{j_1} e_1$. Hence $e' = fold\ e_1 \wedge j_1 \leq j$. Our proof obligation is now to prove that $e_1 \in \mathcal{E}_{k-j}[\![[\mu\alpha.\tau/\alpha]\tau]\!]$.

By induction hypothesis, $\rho(e) \in \mathcal{C}_k[\![[\mu\alpha.\tau/\alpha]\tau]\!]$. Let $\rho(e) \rightsquigarrow^{j_2} e_2$ s.t. $irred(e_2)$. As there can only be one term $\rho(e)$ can reduce to, hence $e_2 = e_1 \wedge j_2 = j_1$. Hence, $e_1 \in \mathcal{E}_{k-j}[\![[\mu\alpha.\tau/\alpha]\tau]\!]$.

□

Chapter 3

L^{θ} language

3.1 Declaration

The contents of this chapter are from Casinghino et al's MSFP 2012 paper "Step-Indexed Normalization for a Language with General Recursion" [1].

3.2 Language Definition

The language is a variant of the simply-typed call-by-value lambda calculus extended with recursive types and general recursion. Using anew feature called *consistency classifiers* θ the typing judgements divide the language into 2 fragments. The logical fragment ($\theta = L$) is the simply typed lambda calculus with unit and sum types. Hence, all terms in this fragment are normalizing. The programmatic fragment ($\theta = P$) adds general recursion and recursive types. Programmatic fragment is a strict superset of the logical fragment i.e.
If $\Gamma \vdash^{L} t : A$ then $\Gamma \vdash^{P} t : A$ also.

3.3 Typing Judgement

Due to interaction between the 2 fragments, terms can have sub-terms from both fragments. In order to mark such transitioned terms, the language introduces *box a*

Types	$A,B ::=$	$\mathsf{Unit} \mid A \to^{\theta} B \mid A + B \mid A@\theta \mid \alpha \mid \mu\,\alpha.A$
Terms	$a,b ::=$	$x \mid \mathsf{rec} f\,x.a \mid a\,b \mid \mathsf{box}\,a \mid \mathsf{unbox}\,x = a\,\mathsf{in}\,b$ $\mid () \mid \mathsf{inl}\,a \mid \mathsf{inr}\,a \mid \mathsf{case}\,a\,\mathsf{of}\,\{\mathsf{inl}\,x \Rightarrow a_1; \mathsf{inr}\,x \Rightarrow a_2\} \mid \mathsf{roll}\,a \mid \mathsf{unroll}\,a$
Language Classifiers	$\theta ::=$	$\mathsf{L} \mid \mathsf{P}$
Environments	$\Gamma ::=$	$\cdot \mid \Gamma, x :^{\theta} A$
Values	$v ::=$	$x \mid () \mid \mathsf{inl}\,v \mid \mathsf{inr}\,v \mid \mathsf{rec} f\,x.a \mid \mathsf{box}\,v \mid \mathsf{roll}\,v$

Syntactic Abbreviation:

$$\lambda x.a \triangleq \mathsf{rec} f\,x.a \qquad \text{when } f \notin \mathrm{FV}(a)$$

Figure 3.1: L^{θ} language syntax

term and its corresponding $a@\theta$ type.

$\Gamma \vdash^{\theta} box\ a : A@\theta'$ means that the fragment θ safely observes the type of a as A in θ' fragment. The function type $A@\theta \to B$ means that the function expects an argument from θ fragment. The *box* a term internalizes the typing judgement. It is checked by 3 rules depending on circumstances in which fragments safely talk about each other. TBOXP says that P fragment can observe that a has type A in any fragment θ. TBOXL checks that if a has type A in L fragment then *box* a has type $A@\theta$ for any θ (as P subsumes L fragment). TBOXLV checks that a term coming from P fragment must be a value to ensure normalization. Still recursive functions (which are values) are allowed in the L fragment.

TUNBOX resembles a "let". The term $unbox\ x = a\ in\ b$ typechecks if a has the type $A@\theta'$ and b has a free variable of type A in fragment θ'.

Functions are typechecked by 2 rules, TLAM and TREC. TLAM checks non recursive (lambda functions) in the L fragment. TREC checks potentially recursive functions in the P fragment.

TAPP makes use of the internalized typing judgement. It essentially limits application by requiring that for application ab in fragment θ, it checks that a has some function type $A@\theta' \to B$, then checks whether *box* b can be given the type $A@\theta'$ in the current fragment. One effect of this rule is to ensure that programmatic arguments to logical functions are values, thus ensuring termination.

TUNIT, TINL and TINR deal with the introduction forms for the unit and sum base types.

$\boxed{\Gamma \vdash^{\theta} a : A}$

$$\frac{x :^{\theta} A \in \Gamma}{\Gamma \vdash^{\theta} x : A}\ \textsc{TVar} \qquad \frac{\Gamma, x :^{\theta} A \vdash^{\mathsf{L}} b : B}{\Gamma \vdash^{\mathsf{L}} \lambda x.b : A^{\theta} \to B}\ \textsc{TLam} \qquad \frac{\Gamma, y :^{\theta} A, f :^{\mathsf{P}} A^{\theta} \to B \vdash^{\mathsf{P}} a : B}{\Gamma \vdash^{\mathsf{P}} \mathsf{rec} f\, y.a : A^{\theta} \to B}\ \textsc{TRec}$$

$$\frac{\Gamma \vdash^{\theta} a : A}{\Gamma \vdash^{\mathsf{P}} \mathsf{box} a : A@\theta}\ \textsc{TBoxP} \qquad \frac{\Gamma \vdash^{\mathsf{L}} a : A}{\Gamma \vdash^{\mathsf{L}} \mathsf{box} a : A@\theta}\ \textsc{TBoxL} \qquad \frac{\Gamma \vdash^{\mathsf{P}} v : A}{\Gamma \vdash^{\mathsf{L}} \mathsf{box} v : A@\mathsf{P}}\ \textsc{TBoxLV}$$

$$\frac{\Gamma \vdash^{\theta} a : A@\theta' \quad \Gamma, x :^{\theta'} A \vdash^{\theta} b : B}{\Gamma \vdash^{\theta} \mathsf{unbox} x = a \mathsf{in} b : B}\ \textsc{TUnbox} \qquad \frac{\Gamma \vdash^{\theta} a : A^{\theta'} \to B \quad \Gamma \vdash^{\theta} \mathsf{box} b : A@\theta'}{\Gamma \vdash^{\theta} a\, b : B}\ \textsc{TApp} \qquad \frac{}{\Gamma \vdash^{\theta} () : \mathsf{Unit}}\ \textsc{TUnit}$$

$$\frac{\Gamma \vdash^{\mathsf{L}} a : A}{\Gamma \vdash^{\mathsf{P}} a : A}\ \textsc{TSub} \qquad \frac{\Gamma \vdash^{\mathsf{P}} v : A \quad \mathsf{FO}(A)}{\Gamma \vdash^{\mathsf{L}} v : A}\ \textsc{TFOVal} \qquad \frac{\Gamma \vdash^{\theta} a : A}{\Gamma \vdash^{\theta} \mathsf{inl} a : A + B}\ \textsc{TInl}$$

$$\frac{\Gamma \vdash^{\theta} b : B}{\Gamma \vdash^{\theta} \mathsf{inr} b : A + B}\ \textsc{TInr} \qquad \frac{\Gamma \vdash^{\theta} \mathsf{box} a : (A_1 + A_2)@\theta' \quad \Gamma, x :^{\theta'} A_1 \vdash^{\theta} b_1 : B \quad \Gamma, x :^{\theta'} A_2 \vdash^{\theta} b_2 : B}{\Gamma \vdash^{\theta} \mathsf{case}\ a\ \mathsf{of}\ \{\mathsf{inl}\, x \Rightarrow b_1; \mathsf{inr}\, x \Rightarrow b_2\} : B}\ \textsc{TCase}$$

$$\frac{\Gamma \vdash^{\mathsf{P}} a : [\mu\alpha.A/\alpha]A}{\Gamma \vdash^{\mathsf{P}} \mathsf{roll} a : \mu\alpha.A}\ \textsc{TRoll} \qquad \frac{\Gamma \vdash^{\mathsf{P}} a : \mu\alpha.A}{\Gamma \vdash^{\mathsf{P}} \mathsf{unroll} a : [\mu\alpha.A/\alpha]A}\ \textsc{TUnroll}$$

$\boxed{\mathsf{FO}(A)}$

$$\frac{}{\mathsf{FO}(\mathsf{Unit})}\ \textsc{FOUnit} \qquad \frac{\mathsf{FO}(A) \quad \mathsf{FO}(B)}{\mathsf{FO}(A + B)}\ \textsc{FOSum} \qquad \frac{}{\mathsf{FO}(A@\theta)}\ \textsc{FOAt}$$

Figure 3.2: L^{θ} typing rules

TSUB ensures subsumption of L by P fragment. TFOVAL allows programmatic values, but not computations, to be used in the L fragment. Such terms have first order types, signified by the judgement FO.

Notice that the box type $A@\theta$ is a first order type for any A. The programmatic fragment is permitted to compute logical values, including logical function values, and pass them back to the logical fragment.

TROLL and TUNROLL are used to check iso-recursive types. Recursive types aren't allowed in the L fragment as they introduce non-termination.

3.4 Operational Semantics

Fig 3.3 gives the operational semantics for this language. Standard call-by-value evaluation contexts and small-step reduction relation are used. Multi-step reduction relation is indexed by a natural number, which will be useful in the step indexed logical relations proof method.

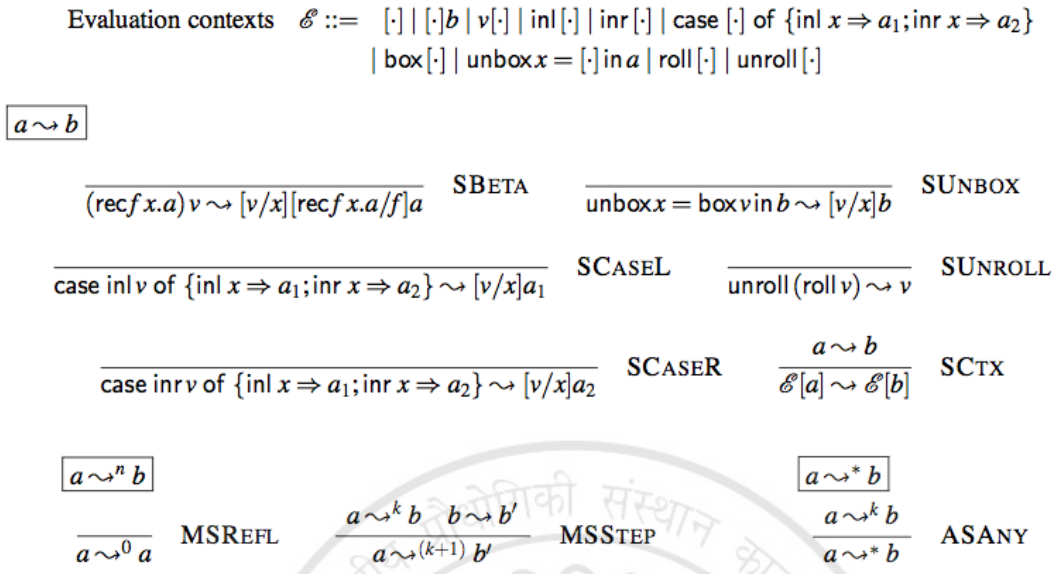


Figure 3.3: L^θ operational semantics

3.5 Metatheory

3.5.1 Type Safety

Type safety is proved via syntactic progress and preservation theorems [9]. The progress result is direct by induction on typing derivations, using appropriate canonical forms lemmas.

Theorem 3.1 *(Progress) If* $\bullet \vdash^\theta a : A$ *then either* a *is a value or* $a \rightsquigarrow a$ *for some* a.

For preservation, a substitution lemma is required. Because variables are values and L^θ includes a value restriction (in the TBOXLV rule), the substitution lemma is proved only for values.

Lemma 3.2 *(Substitution) If* $\Gamma, x : \theta B \vdash^\theta a : A$ *and* $\Gamma \vdash^\theta v : B$, *then* $\Gamma \vdash^\theta [v/x]a : A$.

Since a call-by-value operational semantics is used, this value substitution lemma is enough to prove preservation.

Theorem 3.3 *(Preservation) If* $\Gamma \vdash^\theta a : A$ *and* $a \rightsquigarrow a'$, *then* $\Gamma \vdash^\theta a' : A$.

3.5.2 Normalization

The standard tait-girard reducibility method for proof of normalization of the L fragment will fail due to

- The TFOVAL typing rule won't pass through, if we are interpreting types only for the L fragment, as its premise doesn't type check in the L fragment, hence no induction hypothesis!
- Even if interpretations for both the fragments are defined, the interpretation for the recursive type $\mu\alpha.A$ ruins the well foundedness of the interpretations, as its elimination causes the type to depend on a larger type.

As observed in Ch 4, we have a solution for this problem, in the form of step indexed logical relations. We define a step indexed value interpretation of types as shown in fig.3.4. Intuitively, step indexed logical relations describe partial correctness

$$
\begin{array}{lcl}
\mathscr{V}[\![\mathsf{Unit}]\!]^{\theta}_{k} & = & \{()\} \\
\mathscr{V}[\![A+B]\!]^{\theta}_{k} & = & \{\mathsf{inl}\, v \mid v \in \mathscr{V}[\![A]\!]^{\theta}_{k}\} \cup \{\mathsf{inr}\, v \mid v \in \mathscr{V}[\![B]\!]^{\theta}_{k}\} \\
\mathscr{V}[\![A@\theta']\!]^{\theta}_{k} & = & \{\mathsf{box}\, v \mid v \in \mathscr{V}[\![A]\!]^{\theta'}_{k}\} \\
\mathscr{V}[\![A^{\theta'} \to B]\!]^{\mathsf{L}}_{k} & = & \{\mathsf{rec} f\, x.a \mid \cdot \vdash^{\mathsf{L}} \mathsf{rec} f\, x.a : A^{\theta'} \to B \\
 & & \qquad \text{and } \forall j \leq k, \text{ if } v \in \mathscr{V}[\![A]\!]^{\theta'}_{j} \text{ then } [v/x]a \in \mathscr{C}[\![B]\!]^{\mathsf{L}}_{j}\} \\
\mathscr{V}[\![A^{\theta'} \to B]\!]^{\mathsf{P}}_{k} & = & \{\mathsf{rec} f\, x.a \mid \cdot \vdash^{\mathsf{P}} \mathsf{rec} f\, x.a : A^{\theta'} \to B \\
 & & \qquad \text{and } \forall j < k, \text{ if } v \in \mathscr{V}[\![A]\!]^{\theta'}_{j} \text{ then } [v/x][\mathsf{rec} f\, x.a/f]a \in \mathscr{C}[\![B]\!]^{\mathsf{P}}_{j}\} \\
\mathscr{V}[\![\mu\, \alpha.A]\!]^{\mathsf{L}}_{k} & = & \emptyset \\
\mathscr{V}[\![\mu\, \alpha.A]\!]^{\mathsf{P}}_{k} & = & \{\mathsf{roll}\, v \mid \cdot \vdash^{\mathsf{P}} \mathsf{roll}\, v : \mu\, \alpha.A \text{ and } \forall j < k, v \in \mathscr{V}[\![[\mu\, \alpha.A/\alpha]A]\!]^{\mathsf{P}}_{j}\}
\end{array}
$$

$$
\begin{array}{lcl}
\mathscr{C}[\![A]\!]^{\mathsf{P}}_{k} & = & \{a \mid \cdot \vdash^{\mathsf{P}} a : A \text{ and } \forall j \leq k, \text{ if } a \rightsquigarrow^{j} v \text{ then } v \in \mathscr{V}[\![A]\!]^{\mathsf{P}}_{(k-j)}\} \\
\mathscr{C}[\![A]\!]^{\mathsf{L}}_{k} & = & \{a \mid \cdot \vdash^{\mathsf{L}} a : A \text{ and } a \rightsquigarrow^{*} v \in \mathscr{V}[\![A]\!]^{\mathsf{L}}_{k}\}
\end{array}
$$

Figure 3.4: L^{θ} value and computation interpretation

properties - terms are certified to be well behaved for a finite number of steps, hence they are used to prove type safety or program equivalence. However, here a hybrid approach has been used. Step counting is implemented only for terms in the P fragment, and not in L fragment, where normalization needs to be proved. The descending well-foundedness of this definition can be formalized as a lexicographically

ordered triple (k,A,$\mathcal{I}$), where k is the index, A is the type and $\mathcal{I}$ is one of $\mathscr{C}$ or $\mathscr{V}$ with $\mathscr{V} < \mathscr{C}$.

Definition 3.4 *(Well formed substitution)* $\Gamma \vDash_k \rho$ *when* $x :^\theta A \in \Gamma$ *implies* $\rho x \in \mathscr{V}[\![A]\!]^\theta_k$.

3 more lemmas are needed in the proof of the soundness theorem, given below.

Lemma 3.5 *(Downward Closure) For any A and* θ*, if* $j \leq k$ *then* $\mathscr{V}[\![A]\!]^\theta_k \subseteq \mathscr{V}[\![A]\!]^\theta_j$ *and* $\mathscr{C}[\![A]\!]^\theta_k \subseteq \mathscr{C}[\![A]\!]^\theta_j$.

This is the standard downward closure lemma needed in step indexed logical relational proofs.

Lemma 3.6 *(FO agreement) If FO(A), then* $\mathscr{V}[\![A]\!]^L_k = \mathscr{V}[\![A]\!]^P_k$

This lemma suggests that the 2 fragments agree on first order types.

Lemma 3.7 *(subsumption) For any A and k,* $\mathscr{V}[\![A]\!]^L_k \subseteq \mathscr{V}[\![A]\!]^P_k$ *and* $\mathscr{C}[\![A]\!]^L_k \subseteq \mathscr{C}[\![A]\!]^P_k$

This lemma captures the idea that the P fragment subsumes the L fragment.

The soundness and normalization theorem are then stated as follows.

Theorem 3.8 *(Soundness) If* $\Gamma \vdash^\theta a : A$ *and* $\Gamma \vDash_k \rho$*, then* $\rho a \in \mathscr{C}[\![A]\!]^\theta_k$.

Proof: Follows by induction on typing judgement, exactly as shown in chapter 4. □

Lemma 3.9 *(Normalization) If* $\bullet \vdash^L a : A$ *then* $\exists v$ *s.t.* $Val(v) \wedge a \rightsquigarrow^* v$.

Chapter 4

Reducibility Candidates

4.1 Declaration

The contents of this chapter are from Girard et al's Proofs and Types book [2].

4.2 Introduction

System F is STLC extended with the $\forall$ type. It allows us to define type constructors i.e. functions that take a type as input and return another type as output. This type system is elegant, given how simply it is defined, yet how powerful and expressive it is. Using only the constructs of this simple language, one can define integers, bool, pairs, complex data types and even existential types!

If we naively try to extend Tait's proof of normalization of the STLC to System F, we will soon hit a road block. We want to say that a term t of type $\forall X.T$ is reducible iff for all types U, tU is reducible of type $T[U/X]$. But U could be any type, even the uninhabited type $\forall\alpha.\alpha$! This requirement to know about the meaning of reducibility of all types U, in order to define reducibility for a single type prohibits any progress towards normalization using this approach.

4.3 Reducibility Candidates

The above problem can be solved using Girard's "Candidats De Reductibilit´e" or "Reducibility Candidates". A reducibility candidate of type U is a set R of terms of type U satisfying the following conditions

- CR1 If $t \in R$, then t is strongly normalizing.
- CR2 If $t \in R$, and $t \rightsquigarrow t'$ then $t' \in R$.
- CR3 If t is neutral, and whenever we convert a redex of t, we geta term $t' \in R$, then $t \in R$.

Neutral terms are one of the following: x, tu, tU. i.e. they don't contain any abstraction.

If R and S are reducibility candidates of types U and V, we define a set $R \to S$ of terms of type $U \to V$ by

$t \in R \to S$ iff $\forall u(u \in R \Rightarrow tU \in S)$. This set is proved to be a reducibility candidate of type $U \to V$ by induction on length of the reduction sequence of the term t as given in 6.3.3 in [2].

4.4 Parametric Reducibility

In order to deal with types containing type variables, we require the notion of reducibility with parameters. Let $T[X]$ be a type, where $\underline{X}$ contains atleast all the free type variables of type T. Let $\underline{U}$ be a sequence of types, of the same length as $\underline{X}$, then we can define by substituting simultaneously a type $T[\underline{U}/\underline{X}]$. Now let $\underline{R}$ be a sequence of reducibility candidates of respective types, then we can define a set $RED_T[\underline{R}/\underline{X}]$ (parametric reducibility) of terms of type $T[\underline{U}/\underline{X}]$ as follows:

- If $T = X_i$, then $RED_T[\underline{R}/\underline{X}] = R_i$;
- If $T = V \to W$, then $RED_T[\underline{R}/\underline{X}] = RED_V[\underline{R}/\underline{X}] \to RED_W[\underline{R}/\underline{X}]$;

- If $T = \forall Y.W$ then $RED_T[\underline{R}/\underline{X}]$ is the set of terms of type $T[\underline{U}/\underline{X}]$ s.t., for every type V and its reducibility candidate S, $tV \in RED_W[\underline{R}, S/\underline{X}, Y]$

Lemma 4.1 *(RED is a reducibility candidate)* $RED_T[\underline{R}/\underline{X}]$ *is a reducibility candidate of type* $T[\underline{U}/\underline{X}]$.

Proof: By induction on type T. □

Lemma 4.2 *(Substitution)* $RED_{T[V/Y]}[\underline{R}/\underline{X}] = RED_T[\underline{R}, RED_V[\underline{R}/\underline{X}]/\underline{X}, Y]$

Proof: By induction on type T. □

Lemma 4.3 *(Universal Abstraction) If for every type* V *and candidate* S, $w[V/Y] \in RED_W[\underline{R}/\underline{X}]/\underline{X}, Y]$, *then* $\Lambda Y.w \in RED_{\forall Y.W}[\underline{R}/\underline{X}]$

Proof: By induction on length of the reduction sequence of the term w. □

Lemma 4.4 *(Universal Application) If* $t \in RED_{\forall Y.W}[\underline{R}/\underline{X}]$, *then* $tV \in RED_{W[V/Y]}[\underline{R}/\underline{X}]$ *for every type* V.

Proof: By induction hypothesis, $tV \in RED_W[\underline{R}, S/\underline{X}, Y]$ for any candidate S. If we take $S = RED_V[\underline{R}/\underline{X}]$, the result follows from 4.2. □

Theorem 4.5 *(Reducibility Theorem) Let* t *be a term of type* T. *Let all the free variables of* t *are* $x_1, .., x_n$ *of types* $U_1, .., U_n$, *and all the free type variables of* T, $U_1, .., U_n$ *are among* $X_1, .., X_m$. *If* $R_1, .., R_m$ *are reducibility candidates of types* $V_1, .., V_m$ *and* $u_1, .., u_n$ *are terms of types* $U_1[\underline{V}/\underline{X}], .., U_n[\underline{V}/\underline{X}]$ *which are in* $RED_{U_1}[\underline{R}/\underline{X}], .., RED_{U_n}[\underline{R}/\underline{X}]$ *then* $t[\underline{V}/\underline{X}][\underline{u}/\underline{x}] \in RED_T[\underline{R}/\underline{X}]$.

Proof: The proof is similar to 6.3.3 in [2]. New cases are handled by lemmas 4.3 and 4.4. □

Once we have proved the generalized reducibility theorem, we can take all reducibility candidates of types $\underline{U}$ to be $\underline{SN}$, where SN_i is the set of strongly normalizing terms of type X_i.

Thus we can prove that all terms of F are reducible, and further, strongly normalizable.

Chapter 5

LP^{θ} language

5.1 Declaration

The results reported in this chapter are a joint work with my thesis supervisor Dr. Anil Seth.

The overall development of section 5.2 is due to him. In particular, notion of SIRC (definition 5.5) is his. Formulation of lemma 5.7, 5.8, 5.9 are due to him. Lemma 5.11 and its proof is by him. Definition 5.12 and formulation of Theorem 5.13 are due to him.

I have worked out various cases in the proof of Theorem 5.13 as well as parts of proofs of some of the lemma above.

5.2 Language Definition

We extend the language L^{θ} in chapter 3 by second order polymorphic types. So our language LP^{θ} language has recursive types, general recursion and also includes system F. Syntax of the language is given in fig. 5.1.

We use $\alpha, \gamma, \eta...$ for type variables, letters $A, B, C...$ for types.We define various syntactic categories following chapter 3. New term forms include $\Lambda\alpha.a$ and aA representing type abstraction and application respectively. $\Lambda\alpha.a$ is a value (irrespective of its body a). The new type $\forall\alpha.A$ for polymorphic terms has been added. Also, a

new syntactic category of 'typing context' is added. *Unit* and *Sum* types have been removed as these can be derived from System F. Apart from these, the definition of our language is same as in chapter 3.

Types	$A, B ::=$	$\alpha \mid A^{\theta} \to B \mid \forall\alpha.A \mid A@\theta \mid \mu\alpha.A$
Terms	$a, b ::=$	$x \mid rec\, f\, x.a \mid ab \mid \Lambda\alpha.a \mid aA \mid roll\, a \mid unroll\, a$
		$\mid box\, a \mid unbox\, x = a \text{ in } b$
Language		
Classifiers	$\theta ::=$	$L \mid P$
Typing Context	$\Delta ::=$	$\cdot \mid \Delta, \alpha : \star$
Term Context	$\Gamma ::=$	$\cdot \mid \Gamma, x :^{\theta} A$
Values	$v ::=$	$x \mid rec\, f\, x.a \mid \Lambda\alpha.a \mid box\, v \mid roll\, v$

Syntactic abbreviation : $\lambda x.a \triangleq rec\, f\, x.a$ when $f \notin FV(a)$

Figure 5.1: LP^{θ} Syntax

$$\frac{\alpha : \star \in \Delta}{\Delta \vdash \alpha : \star} \qquad \frac{\Delta \vdash A : \star \quad \Delta \vdash B : \star}{\Delta \vdash A^{\theta} \to B : \star}$$

$$\frac{\Delta \vdash A : \star}{\Delta \vdash A@\theta : \star} \qquad \frac{\Delta, \alpha : \star \vdash A : \star}{\Delta \vdash \mu\alpha.A : \star} \qquad \frac{\Delta, \alpha : \star \vdash A : \star}{\Delta \vdash \forall\alpha.A : \star}$$

Figure 5.2: LP^{θ} Type formation rules

Term forming judgments, $\Delta, \Gamma \vdash^{\theta} x : A$ are defined in fig. 5.4.

5.2.1 Operational Semantics

The definition of evaluation contexts given in figure 5.3 is the same as in chapter 3 except for an addition of a new evaluation context $[\cdot]A$.

A new reduction $(\Lambda\gamma.a)A \rightsquigarrow [A/\gamma]a$ is added to the reduction rules '$\rightsquigarrow$' of chapter 3.

Evaluation Contexts $\mathscr{E} ::= [\cdot] \mid [\cdot]b \mid [\cdot]A \mid v[\cdot] \mid box[\cdot] \mid unbox\, x = [\cdot]\, in\, b \mid roll[\cdot] \mid unroll[\cdot]$

$$\frac{}{(rec\, f\, x.a)\, v \rightsquigarrow [v/x][rec\, f\, x.a/f]a}\ (\text{SBETA}) \qquad \frac{}{unbox\, x = box\, v\, in\, b \rightsquigarrow [v/x]b}(\text{SUNBOX})$$

$$\frac{a \rightsquigarrow b}{\mathscr{E}[a] \rightsquigarrow \mathscr{E}[b]}\ (\text{SCTX}) \quad \frac{}{unroll(roll\, v) \rightsquigarrow v}\ (\text{SUNROLL}) \quad \frac{}{(\Lambda\gamma.a)A \rightsquigarrow [A/\gamma]a}\ (\text{STYPBETA})$$

$$\frac{}{a \rightsquigarrow^0 a}\ (\text{MSREFL}) \qquad \frac{a \rightsquigarrow^k b \quad b \rightsquigarrow b'}{a \rightsquigarrow^{k+1} b'}\ (\text{MSSTEP}) \qquad \frac{a \rightsquigarrow^k b}{a \rightsquigarrow^* b'}\ (\text{ASANY})$$

Figure 5.3: LP^{θ} Operational Semantics

5.2.2 Type Safety

Lemma 5.1 (Canonical forms) *: Let $\Delta \vdash^{\theta} a : A$, such that a is a value. Then the following holds*

1. *If $A = B^{\theta'} \to C$ then a is $rec\, f\, x.b$.*

Further, if $\theta = L$ then f is not free in b and $\Delta, x :^{\theta'} B \vdash^{L} b : C$.

If $\theta = P$ then $\Delta, x :^{\theta'} B, f :^{P} B^{\theta'} \to C \vdash^{P} b : C$.

2. *If $A = \forall\gamma.B$ then a is $\Lambda\gamma.b$ and $\Delta, \gamma : \star \vdash^{\theta} b : B$.*

3. *If $A = B@\theta'$ then a is $box\, v$ and $\Delta \vdash^{\theta'} v : B$.*

4. *If $A = \mu\gamma.B$ then a is $roll\, b$, $\theta = P$ and $\Delta \vdash^{P} b : [\mu\gamma.A/\gamma]A$.*

Proof: By induction on derivation of the judgment $\Delta \vdash^{\theta} a : A$. We inspect various typing rules, where the type matches the one required. As Γ is empty in all forms, we don't consider **START**. We consider cases 1 and 2, with empty Γ.

1. Given $A = B^{\theta'} \to C$, we have 2 typing rules assigning this type, TLAM and TREC. They differ in their consistency classifier.

$$\frac{\Delta \vdash A : \star}{\Delta, x :^{\theta} A \vdash^{\theta} x : A}\ (\text{START}) \qquad \frac{\Delta, \Gamma \vdash^{\theta} t : A \quad \Delta \vdash B : \star}{\Delta, \Gamma, y :^{\theta'} B \vdash^{\theta} t : A}\ y \notin dom(\Gamma)\ (\text{WEAKEN})$$

$$\frac{\Delta, \Gamma, x :^{\theta} A \vdash^{L} t : B}{\Delta, \Gamma \vdash^{L} \lambda x.t : A^{\theta} \to B}\ (\text{TLAM}) \qquad \frac{\Delta, \Gamma, y :^{\theta} A, f :^{P} A^{\theta} \to B \vdash^{P} a : B}{\Delta, \Gamma \vdash^{P} rec\ f\ y.a : A^{\theta} \to B}\ (\text{TREC})$$

$$\frac{\Delta, \gamma : \star, \Gamma \vdash^{\theta} t : A}{\Delta, \Gamma \vdash^{\theta} \Lambda\gamma.t : \forall\gamma.A},\ \gamma \notin FV(\Gamma)\ (\text{TYPABS}) \qquad \frac{\Delta, \Gamma \vdash^{\theta} a : A}{\Delta, \Gamma \vdash^{P} box\ a : A@\theta}\ (\text{TBOXP})$$

$$\frac{\Delta, \Gamma \vdash^{\theta} t : \forall\gamma.A \quad \Delta \vdash C : \star}{\Delta, \Gamma \vdash^{\theta} tC : [C/\gamma]A}\ (\text{TYPAPP}) \qquad \frac{\Delta, \Gamma \vdash^{L} a : A}{\Delta, \Gamma \vdash^{L} box\ a : A@\theta}\ (\text{TBOXL})$$

$$\frac{\Delta, \Gamma \vdash^{\theta} a : A@\theta' \quad \Delta, \Gamma, x :^{\theta'} A \vdash^{\theta} b : B}{\Delta, \Gamma \vdash^{P} unbox\ x = a\ in\ b : B}\ (\text{TUNBOX}) \qquad \frac{\Delta, \Gamma \vdash^{P} v : A}{\Delta, \Gamma \vdash^{L} box\ v : A@P}\ (\text{TBOXLV})$$

$$\frac{\Delta, \Gamma \vdash^{\theta} a : A^{\theta'} \to B \quad \Delta, \Gamma \vdash^{\theta} box\ b : A@\theta'}{\Delta, \Gamma \vdash^{\theta} ab : B}\ (\text{TAPP})$$

$$\frac{\Delta, \Gamma \vdash^{L} a : A}{\Delta, \Gamma \vdash^{P} a : A}\ (\text{TSUB}) \qquad \frac{\Delta, \Gamma \vdash^{P} v : A \quad FO(A)}{\Delta, \Gamma \vdash^{L} v : A}\ (\text{TFOVAL})$$

$$\frac{\Delta, \Gamma \vdash^{P} a : [\mu\alpha.A/\alpha]A}{\Delta, \Gamma \vdash^{P} roll\ a : \mu\alpha.A}\ (\text{TROLL}) \qquad \frac{\Delta, \Gamma \vdash^{P} a : \mu\alpha.A}{\Delta, \Gamma \vdash^{P} unroll\ a : [\mu\alpha.A/\alpha]A}\ (\text{TUNROLL})$$

$$\frac{}{FO(A@\theta)}\ (\text{FOAT})$$

Figure 5.4: LP^{θ} Term formation rules

($\theta = L$) In this case, the TLAM rule applies. Instantiating with empty typing context Γ and by inversion, f is not free in b and $\Delta, x :^{\theta'} B \vdash^{L} b : C$.

($\theta = P$) In this case, the TREC rule applies. Instantiating with empty typing context Γ and by inversion, $\Delta, x :^{\theta'} B, f :^{P} B^{\theta'} \to C \vdash^{P} b : C$

2. Given $A = \forall\gamma.B$, only TYPABS applies for the required type. According to this rule, the term of this type is $\Lambda\gamma.b$. Instantiating with empty typing context Γ, and from inversion, we get $\Delta, \gamma : \star \vdash^{\theta} b : B$.

□

Theorem 5.2 (Progress) *: If $\Delta \vdash a : A$ then either a is a value or there is an a' s.t. $a \rightsquigarrow a'$.*

Proof: The proof is by induction on derivation of judgment $\Delta \vdash a : A$. We consider various cases on the last rule applied in the derivation of $\Delta \vdash a : A$. New cases to be considered are rules TYPABS and TYPAPP. If the last ruled applied is TYPABS then a is a value.

If the last ruled applied is TYPAPP then the derivation ends with rule $\dfrac{\Delta, \Gamma \vdash^{\theta} b : \forall\gamma.C \quad \Delta \vdash B : \star}{\Delta, \Gamma \vdash^{\theta} bB : [B/\gamma]C}$ and $a = bB$. By I.H. on $\Delta, \Gamma \vdash^{\theta} b : \forall\gamma.C$, either b is a value or there is a b' s.t. $b \rightsquigarrow b'$. We consider these two cases below.

1. If b is a value then by canonical forms lemma above, $b = \Lambda\gamma.c$.

Now $bB = (\Lambda\gamma.c)B \rightsquigarrow [B/\gamma]c$. We take a' as $[B/\gamma]c$.

2. If $b \rightsquigarrow b'$ then $bB \rightsquigarrow b'B$. We take a' as $b'B$.

□

Lemma 5.3 (Substitution) *:*

(i) *If $\Delta_1, \alpha, \Delta_2 \vdash A : \star$ and $\Delta_1, \Delta_2 \vdash B : \star$ then*

$\Delta_1, \Delta_2 \vdash [B/\alpha]A : \star$

(ii) *If $\Delta_1, \alpha, \Delta_2, \Gamma \vdash^{\theta} a : A$ and $\Delta_1, \Delta_2 \vdash B : \star$ then*

$\Delta_1, \Delta_2, [B/\alpha]\Gamma \vdash^{\theta} [B/\alpha]a : [B/\alpha]A$

(iii) *$\Delta, \Gamma_1, x :^{\theta'} B, \Gamma_2 \vdash^{\theta} a : A$ and $\Delta, \Gamma_1, \Gamma_2 \vdash^{\theta'} v : B$ then*

$\Delta, \Gamma_1, \Gamma_2 \vdash^{\theta} [v/x]a : A$

Proof:

(i) is proved by induction on derivation of $\Delta_1, \alpha, \Delta_2 \vdash A : \star$. We consider the derivation of the $\forall\zeta.A$ type.

Let $\Delta_1, \alpha, \Delta_2 \vdash \forall\zeta.A : \star$ and $\Delta_1, \Delta_2 \vdash B : \star$.

Then the last rule applied is $\dfrac{\Delta_1, \alpha, \Delta_2, \zeta : \star \vdash A : \star}{\Delta_1, \alpha, \Delta_2 \vdash \forall \zeta.A : \star}$

By inversion and induction hypothesis, we have $\Delta_1, \Delta_2, \zeta : \star \vdash [B/\alpha]A : \star$

and hence applying the same rule again gives us

$\Delta_1, \Delta_2 \vdash \forall \zeta.[B/\alpha]A : \star$

$\Rightarrow \Delta_1, \Delta_2 \vdash [B/\alpha]\forall \zeta.A : \star$

(ii) is proved by induction on derivation of $\Delta_1, \alpha, \Delta_2, \Gamma \vdash^{\theta} a : A$. We consider the typing derivation of *roll a* term.

Let $\Delta_1, \alpha, \Delta_2, \Gamma \vdash^{\theta} roll\ a : \mu\zeta.A$ and $\Delta_1, \Delta_2 \vdash B : \star$.

If the last rule applied was TROLL then the derivation ends with

$\dfrac{\Delta_1, \alpha, \Delta_2, \Gamma \vdash^{P} a : [\mu\zeta.A/\zeta]A}{\Delta_1, \alpha, \Delta_2, \Gamma \vdash^{P} roll\ a : \mu\zeta.A}$ rule.

By inversion and induction hypothesis, we have

$\Delta_1, \Delta_2, [B/\alpha]\Gamma \vdash^{P} [B/\alpha]a : [B/\alpha][\mu\zeta.A/\zeta]A$

and hence applying TROLL gives us

$\Delta_1, \Delta_2, [B/\alpha]\Gamma \vdash^{\theta} roll\ [B/\alpha]a : \mu\zeta.[B/\alpha]A.$

As $roll\ ([B/\alpha]a)$=$[B/\alpha](roll\ a)$ and $\mu\zeta.[B/\alpha]A$ =$[B/\alpha]\mu\zeta.A$, we get

$\Delta_1, \Delta_2, [B/\alpha]\Gamma \vdash^{P} [B/\alpha](roll\ a) : [B/\alpha]\mu\zeta.A.$

(iii) is proved by induction on derivation of $\Delta, \Gamma_1, x :^{\theta'} B, \Gamma_2 \vdash^{\theta} a : A$. We consider the typing derivation of $\lambda y.t$ term.

Let $\Delta, \Gamma_1, x :^{\theta'} B, \Gamma_2 \vdash^{L} \lambda y.t : C^{\theta} \rightarrow D$ and $\Delta, \Gamma_1, \Gamma_2 \vdash^{\theta'} v : B$.

If the last rule applied was TLAM then the derivation ends with

$\dfrac{\Delta, \Gamma_1, x :^{\theta'} B, \Gamma_2, y :^{\theta} C \vdash^{L} t : D}{\Delta, \Gamma_1, x :^{\theta'} B, \Gamma_2 \vdash^{L} \lambda y.t : C^{\theta} \rightarrow D}$ By inversion and induction hypothesis, we have $\Delta, \Gamma_1, \Gamma_2, y :^{\theta} C \vdash^{L} [v/x]t : D$, and hence applying TLAM gives us

$\Delta, \Gamma_1, \Gamma_2 \vdash^{L} \lambda y.[v/x]t : C^{\theta} \rightarrow D$

$\Rightarrow \Delta, \Gamma_1, \Gamma_2 \vdash^{L} [v/x]\lambda y.t : C^{\theta} \rightarrow D$

□

Theorem 5.4 (Preservation) *: If* $\Delta, \Gamma \vdash^{\theta} a : A$ *and* $a \rightsquigarrow a'$ *then* $\Delta, \Gamma \vdash^{\theta} a' : A$.

Proof: The proof is by induction on the definition of '$\rightsquigarrow$' relation. There are two new cases to be considered, SCTX and STYPBETA. We also consider MSREFL, MSSTEP and ASANY.

SCTX Instance of SCTX for new context $[\cdot]A$,

$$\frac{b \rightsquigarrow b'}{bB \rightsquigarrow b'B}$$

In this case $a = bB$.

By inversion on typing judgments we have that

$$\frac{\Delta, \Gamma \vdash^{\theta} b : \forall\gamma.C \quad \Delta \vdash B : \star}{\Delta, \Gamma \vdash^{\theta} bB : [B/\gamma]C}$$

Therefore $A = [B/\gamma]C$.

By applying I.H. on $b \rightsquigarrow b'$ we have $\Delta, \Gamma \vdash^{\theta} b' : \forall\gamma.C$.

The derivation $\dfrac{\Delta, \Gamma \vdash^{\theta} b' : \forall\gamma.C \quad \Delta \vdash B : \star}{\Delta, \Gamma \vdash^{\theta} b'B : [B/\gamma]C}$ shows that $a' = b'B$ has the desired type A.

STYPBETA $a = (\Lambda\gamma.c)B \rightsquigarrow [B/\gamma]c = a'$.

As in the case **1,** by inversion on typing judgment of a, we have that $\dfrac{\Delta, \Gamma \vdash^{\theta} \Lambda\gamma.c : \forall\gamma.C \quad \Delta \vdash B : \star}{\Delta, \Gamma \vdash^{\theta} (\Lambda\gamma.c)B : [B/\gamma]C}$. Therefore $A = [B/\gamma]C$.

Once again, by inversion on premise $\Delta, \Gamma \vdash^{\theta} (\Lambda\gamma.c) : \forall\gamma.C$, we get $\Delta, \gamma : \star, \Gamma \vdash^{\theta} c : C$.

Using substitution lemma, part (ii) on $\Delta, \gamma : \star, \Gamma \vdash^{\theta} c : C$ and $\Delta \vdash B : \star$,

we get $\Delta, [B/\gamma]\Gamma \vdash^{\theta} [B/\gamma]c : [B/\gamma]C$.

Note that $[B/\gamma]\Gamma = \Gamma$ as γ is not free in Γ using the provision in rule $\dfrac{\Delta, \gamma : \star, \Gamma \vdash^{\theta} c : C}{\Delta, \Gamma \vdash^{\theta} \Lambda\gamma.c : \forall\gamma.C}$.

Therefore we have $\Delta, \Gamma \vdash^{\theta} [B/\gamma]c : [B/\gamma]C$ as desired.

MSREFL $\dfrac{}{a \rightsquigarrow^{0} a}$

Here, $\Delta, \Gamma \vdash^{\theta} a : A$ and $a' = a$, hence, by unicity of typing, we have

$\Delta, \Gamma \vdash^{\theta} a' : A$.

MSSTEP $\dfrac{a \rightsquigarrow^{k} b \quad b \rightsquigarrow b'}{a \rightsquigarrow^{k+1} b'}$

To prove: if $\Delta, \Gamma \vdash^{\theta} a : A$ and $a \rightsquigarrow^{k+1} b'$ then $\Delta, \Gamma \vdash^{\theta} b' : A$

By induction hypothesis, if $\Delta, \Gamma \vdash^{\theta} a : A$ and $a \rightsquigarrow^{k} b$ then $\Delta, \Gamma \vdash^{\theta} b : A$.

Similarly, if $b \rightsquigarrow b'$ then by induction hypothesis, $\Delta, \Gamma \vdash^{\theta} b' : A$.

MSANY $\dfrac{a \rightsquigarrow^{k} b}{a \rightsquigarrow^{*} b'}$

By repeatedly applying MSSTEP on the premise, we go on decreasing k until we reach $a \rightsquigarrow^{0} b'$. Then we apply MSREFL.

After that we repeatedly apply MSSTEP, and prove preservation for any number of finite steps.

□

5.3 Normalization for Logical Fragment

It may be noted that the logical fragment for our system LP^{θ} includes System F under call by value semantics. So proving normalization for it will prove normalization for call by value version of System F. We combine the techniques of [2] for proving (strong) normalization of System F and the techniques of chapter 3 to prove the normalization of our type system in L fragment.

Definition 5.5 *A step indexed reducibility candidate R of type B is $R = (R^{i,\theta})_{i \in N, \theta \in L,P}$, where each $R^{i,\theta}$ is a set of closed values of type B and for all $i \in N, \theta \in L, P$, following holds.*

1. $R^{i+1,\theta} \subseteq R^{i,\theta}$

2. $R^{i,L} \subseteq R^{i,P}$

5.3.1 Parameterized step indexed interpretation

We use letters $R, S, T...$ for (step indexed) reducibility candidates. For a sequence $\underline{a}$, $|\underline{a}|$ denotes length of $\underline{a}$. If $\underline{\alpha}$ is $\alpha_1, \ldots, \alpha_m$ then $\underline{\alpha} : \star$ stands for $\alpha_1 : \star, \ldots, \alpha_m : \star$. If

$\underline{B}$ is a sequence of types then a sequence $\underline{R}$ of step indexed reducibility candidates is of type $\underline{B}$ if $|\underline{B} \models |\underline{R}|$ and for all $i \in \{1, \ldots, |B|\}$, R_i is a step indexed reducibility candidate of type B_i.

Let $A[\underline{\alpha}]$ be a type, where all its free type variables are contained in $\underline{\alpha}$. Let $\underline{B}$ be types and $\underline{R}$ be candidates of reducibility for $\underline{B}$. Let free type variables of types in $\underline{B}$ be contained in $\underline{\eta}$. We define $RED_A^{\mathscr{V},k,\theta}[\underline{R}/\underline{\alpha}]$ and $RED_A^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}]$ as shown in next page.

Here $\underline{\alpha}$ is $\alpha_1, \ldots, \alpha_n$ for some n. $\underline{\eta} : \star$ is an abbreviation for $\eta_1 : \star, \ldots, \eta_m : \star$ if $\underline{\eta}$ is $\eta_1, \ldots, \eta_m$. The definition above, as in chapter 3, is by induction on lexicographic ordering on triple $(k, A, \mathscr{I})$, where k, A are an index and a type respectively and $\mathscr{I}$ is $\mathscr{V}$ or $\mathscr{C}$ with $\mathscr{V} < \mathscr{C}$ in the ordering. In the defn of $RED_{\mu\gamma.A_1}^{\mathscr{V},k,\theta}[\underline{R}/\underline{\alpha}]$ index decreases and in all other cases, type on the rightside is simpler.

5.3.2 Properties of Interpretation

Lemma 5.6 (Downward Closure) *Let $k \in N$ and $\theta \in \{L, P\}$. If $RED_A^{\mathscr{V},k+1,\theta}[\underline{R}/\underline{\alpha}] \subseteq RED_A^{\mathscr{V},k,\theta}[\underline{R}/\underline{\alpha}]$ then $RED_A^{\mathscr{C},k+1,\theta}[\underline{R}/\underline{\alpha}] \subseteq RED_A^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}]$.*

Proof: Follows easily from the definition of $RED_A^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}]$.

Case θ is L : Let $a \in RED_A^{\mathscr{C},k+1,L}[\underline{R}/\underline{\alpha}]$.

Then by definition of $RED_A^{\mathscr{C},k+1,L}[\underline{R}/\underline{\alpha}]$, $\underline{\eta} : \star \vdash^L a : [\underline{B}/\underline{\alpha}]A$ and $a \rightsquigarrow^* v \in RED_A^{\mathscr{V},k+1,L}[\underline{R}/\underline{\alpha}]$.

Given$RED_A^{\mathscr{V},k+1,}[\underline{R}/\underline{\alpha}] \subseteq RED_A^{\mathscr{V},k,L}[\underline{R}/\underline{\alpha}]$, we have $v \in RED_A^{\mathscr{V},k,L}[\underline{R}/\underline{\alpha}]$ as well.

Putting this back in the definition of $RED_A^{\mathscr{C},k+1,L}[\underline{R}/\underline{\alpha}]$, we get

$\underline{\eta} : \star \vdash^L a : [\underline{B}/\underline{\alpha}]A$ and $a \rightsquigarrow^* v \in RED_A^{\mathscr{V},k,L}[\underline{R}/\underline{\alpha}]$.

This is the definition of $RED_A^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$, which a fulfills. As a was an arbitrary term in $RED_A^{\mathscr{C},k+1,L}[\underline{R}/\underline{\alpha}]$, we have $RED_A^{\mathscr{C},k+1,}[\underline{R}/\underline{\alpha}] \subseteq RED_A^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$

Case θ is P : Similar to L case.

□

$$
\begin{aligned}
RED^{\mathscr{V},k,\theta}_{\alpha_i}[\underline{R}/\underline{\alpha}] &= R^{k,\theta}_i \\
RED^{\mathscr{V},k,\theta}_{A_1@\theta'}[\underline{R}/\underline{\alpha}] &= \left\{box\ v \mid v \in RED^{\mathscr{V},k,\theta'}_{A_1}[\underline{R}/\underline{\alpha}]\right\}
\end{aligned}
$$

$$
\begin{aligned}
RED^{\mathscr{V},k,L}_{A_1^\theta \to A_2}[\underline{R}/\underline{\alpha}] &= \{rec\ f\ x.a \mid \underline{\eta} : \star \vdash^L rec\ f\ x.a\ : [\underline{B}/\underline{\alpha}]A_1^\theta \to [\underline{B}/\underline{\alpha}]A_2 \text{ and} \\
&\qquad \forall j \leq k(v \in RED^{\mathscr{V},j,\theta}_{A_1}[\underline{R}/\underline{\alpha}] \Rightarrow [v/x]a \in RED^{\mathscr{C},j,L}_{A_2}[\underline{R}/\underline{\alpha}])\}
\end{aligned}
$$

$$
\begin{aligned}
RED^{\mathscr{V},k,P}_{A_1^\theta \to A_2}[\underline{R}/\underline{\alpha}] &= \{rec\ f\ x.a \mid \underline{\eta} : \star \vdash^P rec\ f\ x.a\ [\underline{B}/\underline{\alpha}]A_1^\theta \to [\underline{B}/\underline{\alpha}]A_2 \text{ and} \\
&\qquad \forall j < k(v \in RED^{\mathscr{V},j,\theta}_{A_1}[\underline{R}/\underline{\alpha}] \Rightarrow \\
&\qquad [v/x][rec\ f\ x.a/f]a \in RED^{\mathscr{C},j,P}_{A_2}[\underline{R}/\underline{\alpha}])\} \\
RED^{\mathscr{V},k,L}_{\mu\gamma.A_1}[\underline{R}/\underline{\alpha}] &= \emptyset \\
RED^{\mathscr{V},k,P}_{\mu\gamma.A_1}[\underline{R}/\underline{\alpha}] &= \{roll\ v \mid \underline{\eta} : \star \vdash^P roll\ v : \mu\gamma.[\underline{B}/\underline{\alpha}]A_1 \text{ and} \\
&\qquad \forall j < k(v \in RED^{\mathscr{V},j,P}_{[\mu\gamma.A_1/\gamma]A_1}[\underline{R}/\underline{\alpha}])\}
\end{aligned}
$$

$$
\begin{aligned}
RED^{\mathscr{V},k,L}_{\forall\gamma.A_1}[\underline{R}/\underline{\alpha}] &= \{\Lambda\gamma.t \mid \underline{\eta} : \star \vdash^L \Lambda\gamma.t : \forall\gamma.[\underline{B}/\underline{\alpha}]A_1 \text{ and for all types } C \\
&\qquad \text{and reducibility candidates } S \text{ of type } C, \\
&\qquad [C/\gamma]t \in RED^{\mathscr{C},k,L}_{A_1}[\underline{R}, S/\underline{\alpha}, \gamma]\}
\end{aligned}
$$

$$
\begin{aligned}
RED^{\mathscr{V},k,P}_{\forall\gamma.A_1}[\underline{R}/\underline{\alpha}] &= \{\Lambda\gamma.t \mid \underline{\eta} : \star \vdash^P \Lambda\gamma.t : \forall\gamma.[\underline{B}/\underline{\alpha}]A_1 \text{ and for all types } C \\
&\qquad \text{and reducibility candidates } S \text{ of type } C, \\
&\qquad [C/\gamma]t \in RED^{\mathscr{C},k-1,P}_{A_1}[\underline{R}, S/\underline{\alpha}, \gamma]\}
\end{aligned}
$$

(Counting type application step in P fragment)

$$
RED^{\mathscr{C},k,L}_{A}[\underline{R}/\underline{\alpha}] = \left\{a \mid \underline{\eta} : \star \vdash^L a : [\underline{B}/\underline{\alpha}]A \text{ and } a \rightsquigarrow^* v \in RED^{\mathscr{V},k,L}_{A}[\underline{R}/\underline{\alpha}]\right\}
$$

$$
\begin{aligned}
RED^{\mathscr{C},k,P}_{A}[\underline{R}/\underline{\alpha}] &= \{a \mid \underline{\eta} : \star \vdash^P a : [\underline{B}/\underline{\alpha}]A \text{ and} \\
&\qquad \forall j \leq k(a \rightsquigarrow_j v \Rightarrow v \in RED^{\mathscr{V},k-j,P}_{A}[\underline{R}/\underline{\alpha}])\}
\end{aligned}
$$

Figure 5.5: LP^θ Definition of Syntactic Categories

Lemma 5.7 (FO values same in L and P) *The programmatic interpretation of first order types is the same as logical interpretation.*

Proof: From inspection of figure 5.4, we get that $A@\theta$ is the only first order type. From figure 5.5, we get the interpretation of this type as

$RED^{\mathscr{V},k,\theta}_{A_1@\theta'}[\underline{R}/\underline{\alpha}] = \left\{box\ v \mid v \in RED^{\mathscr{V},k,\theta'}_{A_1}[\underline{R}/\underline{\alpha}]\right\}$

Now, the definition of this interpretation is for any θ and the elements of this set are independent of θ, and thus for any value of θ, the interpretation remains the same. □

Lemma 5.8 (Subsumption) *Let $k \in N$. If $RED^{\mathscr{V},k,L}_A[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{V},k,P}_A[\underline{R}/\underline{\alpha}]$ then $RED^{\mathscr{C},k,L}_A[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{C},k,P}_A[\underline{R}/\underline{\alpha}]$.*

Proof: Follows easily from the definition of $RED^{\mathscr{C},k,P}_A[\underline{R}/\underline{\alpha}]$. Let $a \in RED^{\mathscr{C},k,L}_A[\underline{R}/\underline{\alpha}]$. Then by definition of $RED^{\mathscr{C},k,L}_A[\underline{R}/\underline{\alpha}]$, $\underline{\eta} : \star \vdash^L a : [\underline{B}/\underline{\alpha}]A$ and $a \rightsquigarrow^* v \in RED^{\mathscr{V},k,L}_A[\underline{R}/\underline{\alpha}]$. Given $RED^{\mathscr{V},k,L}_A[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{V},k,P}_A[\underline{R}/\underline{\alpha}]$, we have $v \in RED^{\mathscr{V},k,P}_A[\underline{R}/\underline{\alpha}]$ as well. Putting this back in the definition of $RED^{\mathscr{C},k,L}_A[\underline{R}/\underline{\alpha}]$, we get

$\underline{\eta} : \star \vdash^L a : [\underline{B}/\underline{\alpha}]A$ and $a \rightsquigarrow^* v \in RED^{\mathscr{V},k,P}_A[\underline{R}/\underline{\alpha}]$.

Restricting number of reduction steps of a to atmost k steps, and counting those steps, we can re-write the above statement as :

$\underline{\eta} : \star \vdash^L a : [\underline{B}/\underline{\alpha}]A$ and $\forall j \leq k(a \rightsquigarrow^j v \Rightarrow v \in RED^{\mathscr{V},k-j,P}_A[\underline{R}/\underline{\alpha}])$.

The above statement is the definition of $RED^{\mathscr{C},k,P}_A[\underline{R}/\underline{\alpha}]$. Since a was an arbitrary element of $RED^{\mathscr{C},k,L}_A[\underline{R}/\underline{\alpha}]$, we have $RED^{\mathscr{C},k,L}_A[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{C},k,P}_A[\underline{R}/\underline{\alpha}]$. □

Lemma 5.9 (RED is SIRC) *$(RED^{\mathscr{V},k,\theta}_A[\underline{R}/\underline{\alpha}])_{k\in N,\theta\in\{L,P\}}$ is a step indexed reducibility candidate of type A.*

Proof: We need to show that, for all $k \in N$,

(i) $RED^{\mathscr{V},k+1,\theta}_A[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{V},k,\theta}_A[\underline{R}/\underline{\alpha}]$

and

(ii) $RED^{\mathscr{V},k,L}_A[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{V},k,P}_A[\underline{R}/\underline{\alpha}]$

This is proved by induction on (k, A) using Lemma 5.6 to deal with cases of $\mathscr{C}$ instead of $\mathscr{V}$ in the induction step. The proof starts by considering various cases for A.

Case A is α_i

(i) $RED^{\mathscr{V},k+1,\theta}_{\alpha_i}[\underline{R}/\underline{\alpha}] = R^{k+1,\theta}_i \subseteq R^{k,\theta}_i = RED^{\mathscr{V},k,\theta}_{\alpha_i}[\underline{R}/\underline{\alpha}]$

by definition of $R^{k,\theta}_i$from 5.5.

(ii) $RED^{\mathscr{V},k,L}_{\alpha_i}[\underline{R}/\underline{\alpha}] = R^{k,L}_i \subseteq R^{k,P}_i = RED^{\mathscr{V},k,P}_{\alpha_i}[\underline{R}/\underline{\alpha}]$

by definition of $R^{k,\theta}_i$from 5.5.

Case A is $A_1^{\theta_1} \to A_2$

(i) Subcase: $\theta = L$

$$\text{LHS} = RED^{\mathscr{V},k+1,L}_{A_1^{\theta_1}\to A_2}[\underline{R}/\underline{\alpha}]$$
$$= \{rec\ f\ x.a \mid \underline{\eta} : \star \vdash^L rec\ f\ x.a\ : ([\underline{B}/\underline{\alpha}]A_1)^{\theta_1} \to [\underline{B}/\underline{\alpha}]A_2,$$
$$\text{and } \forall j \leq k+1 (v \in RED^{\mathscr{V},j,\theta_1}_{A_1}[\underline{R}/\underline{\alpha}]$$
$$\Rightarrow [v/x]a \in RED^{\mathscr{C},j,L}_{A_2}[\underline{R}/\underline{\alpha}])\}$$

Using induction hypothesis on A_1 and A_2, we select $i < j$, such that we get $RED^{\mathscr{V},j,\theta_1}_{A_1}[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{V},i,\theta_1}_{A_1}[\underline{R}/\underline{\alpha}]$, and $RED^{\mathscr{V},j,L}_{A_2}[\underline{R}/\underline{\alpha}]) \subseteq RED^{\mathscr{V},i,L}_{A_2}[\underline{R}/\underline{\alpha}])$ on which we apply lemma 5.6 to get

$RED^{\mathscr{C},j,L}_{A_2}[\underline{R}/\underline{\alpha}]) \subseteq RED^{\mathscr{C},i,L}_{A_2}[\underline{R}/\underline{\alpha}])$

Thus we can rewrite LHS as

$$\{rec\ f\ x.a \mid \underline{\eta} : \star \vdash^L rec\ f\ x.a\ : ([\underline{B}/\underline{\alpha}]A_1)^{\theta_1} \to [\underline{B}/\underline{\alpha}]A_2,$$
$$\text{and } \forall i \leq k (v \in RED^{\mathscr{V},i,\theta_1}_{A_1}[\underline{R}/\underline{\alpha}]$$
$$\Rightarrow [v/x]a \in RED^{\mathscr{C},i,L}_{A_2}[\underline{R}/\underline{\alpha}])\}$$
$$= RED^{\mathscr{V},k,L}_{A_1^{\theta_1}\to A_2}[\underline{R}/\underline{\alpha}] = \text{RHS}$$

As $rec\ f\ x.a$ is an arbitrary term in $RED^{\mathscr{V},k+1,L}_{A_1^{\theta_1}\to A_2}[\underline{R}/\underline{\alpha}]$, we have $RED^{\mathscr{V},k+1,L}_{A_1^{\theta_1}\to A_2}[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{V},k,L}_{A_1^{\theta_1}\to A_2}[\underline{R}/\underline{\alpha}]$

Subcase: $\theta = P$

Similar to the previous case.

(ii) $\text{LHS} = RED^{\mathscr{V},k,L}_{A_1^{\theta_1}\to A_2}[\underline{R}/\underline{\alpha}]$

$$= \{rec\ f\ x.a \mid \underline{\eta} : \star \vdash^L rec\ f\ x.a\ : ([\underline{B}/\underline{\alpha}]A_1)^{\theta_1} \to [\underline{B}/\underline{\alpha}]A_2,$$

and $\forall j \leq k(v \in RED^{\mathscr{V},j,\theta_1}_{A_1}[\underline{R}/\underline{\alpha}]$

$\Rightarrow [v/x]a \in RED^{\mathscr{C},j,L}_{A_2}[\underline{R}/\underline{\alpha}])\}$

Using induction hypothesis on A_2,

$RED^{\mathscr{V},j,L}_{A_2}[\underline{R}/\underline{\alpha}]) \subseteq RED^{\mathscr{V},j,P}_{A_2}[\underline{R}/\underline{\alpha}])$ on which we apply lemma 5.8 to get

$RED^{\mathscr{C},j,L}_{A_2}[\underline{R}/\underline{\alpha}]) \subseteq RED^{\mathscr{C},j,P}_{A_2}[\underline{R}/\underline{\alpha}])$

Thus we can rewrite LHS as

$\{rec\ f\ x.a \mid \underline{\eta} : \star \vdash^L rec\ f\ x.a\ : ([\underline{B}/\underline{\alpha}]A_1)^{\theta_1} \to [\underline{B}/\underline{\alpha}]A_2,$

and $\forall j \leq k(v \in RED^{\mathscr{V},j,\theta_1}_{A_1}[\underline{R}/\underline{\alpha}]$

$\Rightarrow [v/x]a \in RED^{\mathscr{C},j,P}_{A_2}[\underline{R}/\underline{\alpha}])\} = RED^{\mathscr{V},k,P}_{A_1^{\theta_1} \to A_2}[\underline{R}/\underline{\alpha}] =$ RHS

As $rec\ f\ x.a$ is an arbitrary term in $RED^{\mathscr{V},k,L}_{A_1^{\theta_1} \to A_2}[\underline{R}/\underline{\alpha}]$, we have

$RED^{\mathscr{V},k,L}_{A_1^{\theta_1} \to A_2}[\underline{R}/\underline{\alpha}] \in RED^{\mathscr{V},k,P}_{A_1^{\theta_1} \to A_2}[\underline{R}/\underline{\alpha}]$

Case A is $\forall\zeta.A_1$

(i) Subcase: $\theta = L$

LHS $= RED^{\mathscr{V},k+1,L}_{\forall\zeta.A_1}[\underline{R}/\underline{\alpha}]$

$= \{\Lambda\zeta.t \mid \underline{\eta} : \star \vdash \Lambda\zeta.t : \forall\zeta.[\underline{B}/\underline{\alpha}]A_1$

and for all types D and reducibility candidates T of type D,

$[D/\zeta]t \in RED^{\mathscr{C},k+1,L}_{A_1}[\underline{R}, T/\underline{\alpha}, \zeta]\}$

By induction hypothesis on A_1, we can rewrite the above equation as

$\{\Lambda\zeta.t \mid \underline{\eta} : \star \vdash \Lambda\zeta.t : \forall\zeta.[\underline{B}/\underline{\alpha}]A_1$

and for all types D and reducibility candidates T of type D,

$[D/\zeta]t \in RED^{\mathscr{C},k,L}_{A_1}[\underline{R}, T/\underline{\alpha}, \zeta]\}$

$= RED^{\mathscr{V},k,L}_{\forall\zeta.A_1}[\underline{R}/\underline{\alpha}]=$ RHS.

As $\Lambda\zeta.t$ is an arbitrary element in $RED^{\mathscr{V},k+1,L}_{\forall\zeta.A_1}[\underline{R}/\underline{\alpha}]$, we have

$RED^{\mathscr{V},k+1,L}_{\forall\zeta.A_1}[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{V},k,L}_{\forall\zeta.A_1}[\underline{R}/\underline{\alpha}]$

Subcase: $\theta = P$

This is similar to the previous subcase.

(ii) LHS $= RED^{\mathscr{V},k,L}_{\forall\zeta.A_1}[\underline{R}/\underline{\alpha}]$

$= \{\Lambda\zeta.t \mid \underline{\eta} : \star \vdash \Lambda\zeta.t : \forall\zeta.[\underline{B}/\underline{\alpha}]A_1$

and for all types D and reducibility candidates T of type D,

$$[D/\zeta]t \in RED^{\mathscr{C},k,L}_{A_1}[\underline{R}, T/\underline{\alpha}, \zeta]\}$$

By induction hypothesis on A_1, and applying 5.6, we can rewrite the above equation as

$\{\Lambda\zeta.t \mid \underline{\eta} : \star \vdash \Lambda\zeta.t : \forall\zeta.[\underline{B}/\underline{\alpha}]A_1$

and for all types D and reducibility candidates T of type D,

$$[D/\zeta]t \in RED^{\mathscr{C},k-1,P}_{A_1}[\underline{R}, T/\underline{\alpha}, \zeta]\}$$

$= RED^{\mathscr{V},k,P}_{\forall\zeta.A_1}[\underline{R}/\underline{\alpha}]$ = RHS.

As $\Lambda\zeta.t$ is an arbitrary term in $RED^{\mathscr{V},k,L}_{\forall\zeta.A_1}[\underline{R}/\underline{\alpha}]$, we have

$RED^{\mathscr{V},k,L}_{\forall\zeta.A_1}[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{V},k,P}_{\forall\zeta.A_1}[\underline{R}/\underline{\alpha}]$

Case A is $\mu\zeta.A_1$

(i) Subcase: $\theta = L$

both LHS and RHS are $\emptyset$.

Subcase: $\theta = P$

LHS $= RED^{\mathscr{V},k+1,P}_{\mu\zeta.A_1}[\underline{R}/\underline{\alpha}]$

$= \{roll\ v \mid \underline{\eta} : \star \vdash^P roll\ v : \mu\zeta.[\underline{B}/\underline{\alpha}]A_1$ and $\forall j < k+1 (v \in RED^{\mathscr{V},j,P}_{[\mu\zeta.A_1/\zeta]A_1}[\underline{R}/\underline{\alpha}])\}$

By induction on $(j, [\mu\zeta.A_1/\zeta]A_1)$, for any $i < j$, above equation becomes

$\{roll\ v \mid \underline{\eta} : \star \vdash^P roll\ v : \mu\zeta.[\underline{B}/\underline{\alpha}]A_1$ and $\forall i < k (v \in RED^{\mathscr{V},i,P}_{[\mu\zeta.A_1/\zeta]A_1}[\underline{R}/\underline{\alpha}])\}$

$= RED^{\mathscr{V},k,P}_{\mu\zeta.A_1}[\underline{R}/\underline{\alpha}]$ = RHS.

As $roll\ v$ was any arbitrary term in $RED^{\mathscr{V},k+1,P}_{\mu\zeta.A_1}[\underline{R}/\underline{\alpha}]$, we have

$RED^{\mathscr{V},k+1,P}_{\mu\zeta.A_1}[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{V},k,P}_{\mu\zeta.A_1}[\underline{R}/\underline{\alpha}]$

(ii) LHS $= RED^{\mathscr{V},k,L}_{\mu\zeta.A_1}[\underline{R}/\underline{\alpha}]$

$= \emptyset \subseteq RED^{\mathscr{V},k,P}_{\mu\zeta.A_1}[\underline{R}/\underline{\alpha}]$

holds true vacuously.

□

Lemma 5.10 (Forward/Backward inclusion) *Let* $a \rightsquigarrow^* b$ *then*

(i) $a \in RED_A^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}] \Leftrightarrow b \in RED_A^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$

(ii) *If* $a \rightsquigarrow^i b$ *then* $a \in RED_A^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}] \Rightarrow b \in RED_A^{\mathscr{C},k-i,P}[\underline{R}/\underline{\alpha}] \wedge b \in RED_A^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}] \Rightarrow a \in RED_A^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$

Proof:

(i) $\Rightarrow$**direction :** From definition of $RED_A^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$, we have

$\underline{\eta} : \star \vdash^L a : [\underline{B}/\underline{\alpha}]A$ and $a \rightsquigarrow^* v \in RED_A^{\mathscr{V},k,L}[\underline{R}/\underline{\alpha}]$.

By preservation, we have $\underline{\eta} : \star \vdash^L b : [\underline{B}/\underline{\alpha}]A$

As we have a call by value language, and given $a \rightsquigarrow^* b$ and from definition, $a \rightsquigarrow^* v$, we must have $b \rightsquigarrow^* v$.

Putting them both together, we have, $\underline{\eta} : \star \vdash^L b : [\underline{B}/\underline{\alpha}]A$ and $b \rightsquigarrow^* v \in RED_A^{\mathscr{V},k,L}[\underline{R}/\underline{\alpha}]$.

Hence $b \in RED_A^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$.

$\Leftarrow$**direction :** From definition of $RED_A^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$, we have

$\underline{\eta} : \star \vdash^L b : [\underline{B}/\underline{\alpha}]A$ and $b \rightsquigarrow^* v \in RED_A^{\mathscr{V},k,L}[\underline{R}/\underline{\alpha}]$.

a can only have type $[\underline{B}/\underline{\alpha}]A$. To see why, consider the type of a to be τ. Then as $a \rightsquigarrow^* b$, we have $b : \tau$. But given $\underline{\eta} : \star \vdash^L b : [\underline{B}/\underline{\alpha}]A$, we get $\tau = [\underline{B}/\underline{\alpha}]A$.

As we have a call by value language, and given $a \rightsquigarrow^* b$ and from definition, $b \rightsquigarrow^* v$, we must have $a \rightsquigarrow^* v$.

Putting them both together, we have, $\underline{\eta} : \star \vdash^L a : [\underline{B}/\underline{\alpha}]A$ and $a \rightsquigarrow^* v \in RED_A^{\mathscr{V},k,L}[\underline{R}/\underline{\alpha}]$.

Hence $a \in RED_A^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$.

(ii) $\Rightarrow$**direction :** From definition of $RED_A^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$, we have

$\underline{\eta} : \star \vdash^P a : [\underline{B}/\underline{\alpha}]A$ and $\forall j \leq k(a \rightsquigarrow^j v \Rightarrow v \in RED_A^{\mathscr{V},k-j,P}[\underline{R}/\underline{\alpha}])$.

By preservation, we have $\underline{\eta} : \star \vdash^P b : [\underline{B}/\underline{\alpha}]A$

Consider for $i \leq j \leq k, a \rightsquigarrow^i b$ and let $j' = j - i$ and $k' = k - i$. Then, if $a \rightsquigarrow^j v$, we have $b \rightsquigarrow^{j'} v$.

Thus, we have, $\underline{\eta} : \star \vdash^P b : [\underline{B}/\underline{\alpha}]A$ and $\forall j' \leq k'(b \rightsquigarrow^{j'} v \Rightarrow v \in RED_A^{\mathscr{V},k'-j',P}[\underline{R}/\underline{\alpha}])$.

Hence $b \in RED_A^{\mathscr{C},k',P}[\underline{R}/\underline{\alpha}] = RED_A^{\mathscr{C},k-i,P}[\underline{R}/\underline{\alpha}]$.

$\Leftarrow$**direction :** From definition of $RED_A^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$, we have

$\underline{\eta} : \star \vdash^P b : [\underline{B}/\underline{\alpha}]A$ and $\forall j \leq k(b \rightsquigarrow^j v \Rightarrow v \in RED_A^{\mathscr{V},k-j,P}[\underline{R}/\underline{\alpha}])$.

As a reduces to b, and our language has deterministic operational semantics, we have, by unicity of typing, $\underline{\eta} : \star \vdash^P a : [\underline{B}/\underline{\alpha}]A$

Consider for some i, $a \rightsquigarrow^i b$ and let $j' = j + i$ and $k' = k + i$. Then, if $b \rightsquigarrow^j v$, we have $a \rightsquigarrow^{j'} v$.

Thus, we have, $\underline{\eta} : \star \vdash^P a : [\underline{B}/\underline{\alpha}]A$ and $\forall j' \leq k'(a \rightsquigarrow^{j'} v \Rightarrow v \in RED_A^{\mathscr{V},k'-j',P}[\underline{R}/\underline{\alpha}])$.

Hence $a \in RED_A^{\mathscr{C},k',P}[\underline{R}/\underline{\alpha}] = RED_A^{\mathscr{C},k+i,P}[\underline{R}/\underline{\alpha}] \subseteq RED_A^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$.

□

Lemma 5.11 (Type Substitution in Parametric SIRC) *Let A be a type with its free type variables contained in $\underline{\alpha} = \alpha_1, \ldots, \alpha_n$ and γ. Let $\underline{B} = B_1, \ldots, B_n$ be any types and let $\underline{R} = R_1, \ldots, R_n$ be step indexed reducibility candidates of types $B_1, \ldots, B_n$ respectively. Let C be another type with its free type variables contained in $\underline{\alpha}$ and $S = (RED_C^{\mathscr{V},i,\theta}[\underline{R}/\underline{\alpha}])_{i \in N, \theta \in \{L,P\}}$. Then for any $k \in N$ and $\theta \in \{L, P\}$,*

1. $RED_{[C/\gamma]A}^{\mathscr{V},k,\theta}[\underline{R}/\underline{\alpha}] = RED_A^{\mathscr{V},k,\theta}[\underline{R}, S/\underline{\alpha}, \gamma]$

2. $RED_{[C/\gamma]A}^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}] = RED_A^{\mathscr{C},k,\theta}[\underline{R}, S/\underline{\alpha}, \gamma]$

Proof: We prove 1. and 2. simultaneously by induction on $(k, A, \mathscr{I})$, where k is the index, A is the type and $\mathscr{I}$ is one of $\mathscr{V}$ or $\mathscr{C}$ with $\mathscr{V} < \mathscr{C}$. The induction step is proved by considering various cases for A as follows.

Case A is α_i

$$\text{LHS} = RED_{[C/\gamma]\alpha_i}^{\mathscr{V},k,\theta}[\underline{R}/\underline{\alpha}] = RED_{\alpha_i}^{\mathscr{V},k,\theta}[\underline{R}/\underline{\alpha}] = R_i^{k,\theta}$$

$$\text{RHS} = RED^{\mathscr{V},k,\theta}_{\alpha_i}[\underline{R},\ S/\underline{\alpha},\gamma] = R^{k,\theta}_i$$

Case A is γ

$$\text{LHS} = RED^{\mathscr{V},k,\theta}_{[C/\gamma]\gamma}[\underline{R}/\underline{\alpha}] = RED^{\mathscr{V},k,\theta}_{C}[\underline{R}/\underline{\alpha}]$$

$$\text{RHS} = RED^{\mathscr{V},k,\theta}_{\gamma}[\underline{R},\ S/\underline{\alpha},\gamma] = S^{k,\theta} = RED^{\mathscr{V},k,\theta}_{C}[\underline{R}/\underline{\alpha}]$$

Case A is $A_1^{\theta_1} \to A_2$

Subcase: $\theta = L$

$$\text{LHS} = RED^{\mathscr{V},k,L}_{[C/\gamma]A_1^{\theta_1}\to[C/\gamma]A_2}[\underline{R}/\underline{\alpha}]$$

$$= \{rec\ f\ x.a \mid \underline{\eta} : \star \vdash^L rec\ f\ x.a\ : ([\underline{B}/\underline{\alpha}][C/\gamma]A_1)^{\theta_1} \to [\underline{B}/\underline{\alpha}][C/\gamma]A_2,$$

$$\text{and } \forall j \le k(v \in RED^{\mathscr{V},j,\theta_1}_{[C/\gamma]A_1}[\underline{R}/\underline{\alpha}]$$

$$\Rightarrow [v/x]a \in RED^{\mathscr{C},j,L}_{[C/\gamma]A_2}[\underline{R}/\underline{\alpha}])\}$$

Using induction hypothesis on A_1 and A_2

$$= \{rec\ f\ x.a \mid \underline{\eta} : \star \vdash^L rec\ f\ x.a\ : ([\underline{B}, C[\underline{B}/\underline{\alpha}]/\underline{\alpha},\gamma]A_1)^{\theta_1} \to [\underline{B}, C[\underline{B}/\underline{\alpha}]/\underline{\alpha},\gamma]A_2),$$

$$\text{and } \forall j \le k(v \in RED^{\mathscr{V},j,\theta_1}_{A_1}[\underline{R}, S/\underline{\alpha},\gamma]$$

$$\Rightarrow [v/x]a \in RED^{\mathscr{C},j,L}_{A_2}[\underline{R}, S/\underline{\alpha},\gamma])\}$$

$$= RED^{\mathscr{V},k,L}_{A_1^{\theta_1}\to A_2}[\underline{R}, S/\underline{\alpha},\gamma].$$

Subcase: $\theta = P$

$$\text{LHS} = RED^{\mathscr{V},k,P}_{[C/\gamma]A_1^{\theta_1}\to[C/\gamma]A_2}[\underline{R}/\underline{\alpha}]$$

$$= \{rec\ f\ x.a \mid \underline{\eta} : \star \vdash^P rec\ f\ x.a\ : ([\underline{B}/\underline{\alpha}][C/\gamma]A_1)^{\theta_1} \to [\underline{B}/\underline{\alpha}]([C/\gamma]A_2),$$

$$\text{and } \forall j < k(v \in RED^{\mathscr{V},j,\theta_1}_{[C/\gamma]A_1}[\underline{R}/\underline{\alpha}]$$

$$\Rightarrow [v/x][rec\ f\ x.a/f]a \in RED^{\mathscr{C},j,P}_{[C/\gamma]A_2}[\underline{R}/\underline{\alpha}])\}$$

Using induction hypothesis on A_1 and A_2

$$= \{rec\ f\ x.a \mid \underline{\eta} : \star \vdash^P rec\ f\ x.a\ : ([\underline{B}, C[\underline{B}/\underline{\alpha}]/\underline{\alpha},\gamma]A_1)^{\theta_1} \to [\underline{B}, C[\underline{B}/\underline{\alpha}]/\underline{\alpha},\gamma]A_2),$$

$$\text{and } \forall j < k(v \in RED^{\mathscr{V},j,\theta_1}_{A_1}[\underline{R}, S/\underline{\alpha},\gamma]$$

$$\Rightarrow [v/x][rec\ f\ x.a/f]a \in RED^{\mathscr{C},j,P}_{A_2}[\underline{R}, S/\underline{\alpha},\gamma])\}$$

$$= RED^{\mathscr{V},k,P}_{A_1^{\theta_1}\to A_2}[\underline{R}, S/\underline{\alpha},\gamma].$$

Case A is $\forall\zeta.A_1$

Subcase: $\theta = L$

LHS $= RED^{\mathscr{V},k,L}_{[C/\gamma]\forall\zeta.A_1}[\underline{R}/\underline{\alpha}]$

$= RED^{\mathscr{V},k,L}_{\forall\zeta.[C/\gamma]A_1}[\underline{R}/\underline{\alpha}]$

$= \{\Lambda\zeta.t \mid \underline{\eta} : \star \vdash \Lambda\zeta.t : \forall\zeta.[\underline{B}/\underline{\alpha}][C/\gamma]A_1$

and for all types D and reducibility candidates T of type D,

$$[D/\zeta]t \in RED^{\mathscr{C},k,L}_{[C/\gamma]A_1}[\underline{R}, T/\underline{\alpha}, \zeta]\}$$

By induction hypothesis on A_1, we get

$= \{\Lambda\zeta.t \mid \underline{\eta} : \star \vdash \Lambda\zeta.t : \forall\zeta.[\underline{B}, C[\underline{B}/\underline{\alpha}]/\underline{\alpha}, \gamma]A_1$

and for all types D and reducibility candidates T of type D,

$$[D/\zeta]t \in RED^{\mathscr{C},k,L}_{A_1}[\underline{R}, T, S/\underline{\alpha}, \zeta, \gamma]\}$$

$[= RED^{\mathscr{V},k,L}_{\forall\zeta.A_1}[\underline{R}, S/\underline{\alpha}, \gamma]]$

Subcase: $\theta = P$

This is similar to the previous case.

Case A is $\mu\zeta.A_1$

When θ is L, both LHS and RHS are $\emptyset$.

So we consider $\theta = P$.

LHS $= RED^{\mathscr{V},k,P}_{[C/\gamma]\mu\zeta.A_1}[\underline{R}/\underline{\alpha}]$

$= RED^{\mathscr{V},k,P}_{\mu\zeta.[C/\gamma]A_1}[\underline{R}/\underline{\alpha}]$

$= \{roll\ v \mid \underline{\eta} : \star \vdash^P roll\ v : \mu\zeta.[\underline{B}/\underline{\alpha}][C/\gamma]A_1$

and $\forall j < k(v \in RED^{\mathscr{V},j,P}_{[\mu\zeta.[C/\gamma]A_1/\zeta][C/\gamma]A_1}[\underline{R}/\underline{\alpha}])\}$

As ζ is not free in C, we have $[\mu\zeta.[C/\gamma]A_1/\zeta][C/\gamma]A_1 = [C/\gamma][\mu\zeta.A_1/\zeta]A_1$

$= \{roll\ v \mid \underline{\eta} : \star \vdash^P roll\ v : \mu\zeta.[\underline{B}, C[\underline{B}/\underline{\alpha}]/\underline{\alpha}, \gamma]A_1$ and $\forall j < k(v \in RED^{\mathscr{V},j,P}_{[C/\gamma][\mu\zeta.A_1/\zeta]A_1}[\underline{R}/\underline{\alpha}])\}$

By induction hypothesis on $(j, [\mu\zeta.A/\zeta]A_1, P)$, we get

$= \{roll\ v \mid \underline{\eta} : \star \vdash^P roll\ v : \mu\zeta.[\underline{B}, C[\underline{B}/\underline{\alpha}]/\underline{\alpha}, \gamma]A_1$ and $\forall j < k(v \in RED^{\mathscr{V},j,P}_{[\mu\zeta.A_1/\zeta]A_1}[\underline{R}, S/\underline{\alpha}, \gamma])\}$

$= RED^{\mathscr{V},k,P}_{\mu\zeta.A_1}[\underline{R}, S/\underline{\alpha}, \gamma]$ = RHS

We now prove 2.

Case $\theta = L$,

Let $a \in RED^{\mathscr{C},k,L}_{[C/\gamma]A}[\underline{R}/\underline{\alpha}]$

$\Rightarrow \underline{\eta} : \star \vdash^L a : [\underline{B}/\underline{\alpha}][C/\gamma]A$ and $a \rightsquigarrow^* v \in RED^{\mathscr{V},k,L}_{[C/\gamma]A}[\underline{R}/\underline{\alpha}]$

By I.H. on $(k, A, \mathscr{V})$,

$\Rightarrow \underline{\eta} : \star \vdash^L a : [\underline{B}, C[\underline{B}]/\underline{\alpha}, \gamma]A$ and $a \rightsquigarrow^* v \in RED^{\mathscr{V},k,L}_{A}[\underline{R}, S/\underline{\alpha}, \gamma]$

$\Rightarrow a \in RED^{\mathscr{C},k,L}_{A}[\underline{R}, S/\underline{\alpha}, \gamma]$

Case $\theta = P$

Let $a \in RED^{\mathscr{C},k,P}_{[C/\gamma]A}[\underline{R}/\underline{\alpha}]$

$\Rightarrow \underline{\eta} : \star \vdash^P a : [\underline{B}/\underline{\alpha}][C/\gamma]A$

and $\forall j \leq k(a \rightsquigarrow_j v \Rightarrow v \in RED^{\mathscr{V},k-j,P}_{[C/\gamma]A}[\underline{R}/\underline{\alpha}])$

By I.H. on $(k - j, A, \mathscr{V})$,

$\Rightarrow \underline{\eta} : \star \vdash^P a : [\underline{B}, C[\underline{B}]/\underline{\alpha}, \gamma]A$

and $\forall j \leq k(a \rightsquigarrow_j v \Rightarrow v \in RED^{\mathscr{V},k-j,P}_{A}[\underline{R}, S/\underline{\alpha}, \gamma])$

$\Rightarrow a \in RED^{\mathscr{C},k,P}_{A}[\underline{R}, S/\underline{\alpha}, \gamma]$

□

Definition 5.12 (Well formed substitution) *Let* Δ *be* $\underline{\alpha} : \star$ *and* Γ *be* $x_1 :^{\theta_1} A_1(\underline{\alpha}), \ldots, x_n :^{\theta_n} A_n(\underline{\alpha})$*, where* $\underline{\alpha} = \alpha_1, \ldots, \alpha_m$*. Let* ρ *be a function from* $dom(\Delta) \cup dom(\Gamma)$ *s.t. for* $u \in dom(\Delta)$, $\rho(u) \in Types$ *and for* $u \in dom(\Gamma)$, $\rho(u) \in Terms$*. Let* $\underline{R}$ *be step indexed reducibility candidates for* $\underline{\rho(\alpha_i)}$*. We say* $\Delta, \Gamma \models_{k,\underline{R}} \rho$ *if* $\rho(x_j) \in RED^{\mathscr{V},k,\theta_j}_{A_j}[\underline{R}/\underline{\alpha}]$ *for* $j \in \{1, \ldots, n\}$*.*

For ρ *as above, we can homomorphically extend it to set Terms. In other words,* $\rho(t)$ *is defined as the term obtained by substituting in* t *free variables by their* ρ *values.*

Theorem 5.13 (Soundness) *Let* $\Delta, \Gamma \vdash^{\theta} t : A$. *Let* $\underline{R}$ *be step indexed reducibility candidates of types* $\rho(dom(\Delta))$ *and let* ρ *be s.t.* $\Delta, \Gamma \models_{k,\underline{R}} \rho$. *Then* $\rho(t) \in RED_A^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}]$.

Proof: By induction on the derivation $\Delta, \Gamma \vdash t : A$. We consider cases based on the last step in the typing derivation.

START $\dfrac{\Delta \vdash A : \star}{\Delta, x :^{\theta} A \vdash^{\theta} x : A}$

To prove : $\rho'(x) \in RED_A^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}]$

Given $\Delta, x :^{\theta} A \vDash_{k,\underline{R}} \rho'$ s.t. $\rho' = \rho[x \to v]$ and $v \in RED_A^{\mathscr{V},k,\theta}[\underline{R}/\underline{\alpha}]$.

Thus $\rho'(x) : \rho'(A)$ and $\rho'(x) \rightsquigarrow v \in RED_A^{\mathscr{V},k,\theta}[\underline{R}/\underline{\alpha}]$ (definition of $RED_A^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}]$)

Hence $\rho'(x) \in RED_A^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}]$ $\square$

WEAKEN $\dfrac{\Delta, \Gamma \vdash^{\theta} t : A \quad \Delta \vdash B : \star}{\Delta, \Gamma, y :^{\theta'} B \vdash^{\theta} t : A}\ y \notin dom(\Gamma)$

To prove : $\rho'(t) \in RED_A^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}]$ where

By induction hypothesis, $\Delta\Gamma \vDash_{k,\underline{R}} \rho$ and $\rho(t) \in RED_A^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}]$

Given $\rho' = \rho[y \to v]$ s.t. $v \in RED_B^{\mathscr{V},k,\theta'}[\underline{R}/\underline{\alpha}]$

As $y \notin dom(\Gamma)$, so, $y \notin freevars(t)$.

Hence $\rho'(t) = \rho[y \to v](t) = [y \to v](\rho(t)) = \rho(t) \in RED_A^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}]$ $\square$

TLAM $\dfrac{\Delta, \Gamma, x :^{\theta} A_1 \vdash^{L} a : A_2}{\Delta, \Gamma \vdash^{L} rec\ fx.a : A_1^{\theta} \to A_2}$

To prove: $\rho(rec\ fx.a) = rec\ fx.\rho(a) \in RED_{A_1^{\theta} \to A_2}^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$

By definition of $RED_{A_1^{\theta} \to A_2}^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$, we need to prove

(i) $\underline{\eta} : \star \vdash^{L} rec\ fx.\rho(a) : \rho(A_1^{\theta} \to A_2)$ (by appeal to the substitution lemma 5.3) $\square$

(ii) Being a value, $rec\ fx.\rho(a) \in RED_{A_1^{\theta} \to A_2}^{\mathscr{V},k,L}$, for which we now need to prove

$\forall j \leq k (v \in RED_{A_1}^{\mathscr{V},j,\theta}[\underline{R}/\underline{\alpha}] \Rightarrow [v/x]\rho(a) \in RED_{A_2}^{\mathscr{C},j,L}[\underline{R}/\underline{\alpha}])$

By induction hypothesis, $\Delta, \Gamma, x :^{\theta} A_1 \vDash_{j,\underline{R}} \rho'$

where $j \leq k,\ \rho' = \rho[x \to v]$ s.t. $v \in RED_{A_1}^{\mathscr{V},j,\theta}[\underline{R}/\underline{\alpha}]$

and $\rho'(a) \in RED^{\mathscr{C},j,L}_{A_2}[\underline{R}/\underline{\alpha}]$.

$\Rightarrow [x \to v]\rho(a) \in RED^{\mathscr{C},j,L}_{A_2}[\underline{R}/\underline{\alpha}]$

$\Rightarrow \forall j \leq k(v \in RED^{\mathscr{V},j,\theta}_{A_1}[\underline{R}/\underline{\alpha}] \Rightarrow [v/x]\rho(a) \in RED^{\mathscr{C},j,L}_{A_2}[\underline{R}/\underline{\alpha}])$ □

TREC $\dfrac{\Delta, \Gamma, y :^{\theta} A_1, f :^{P} A_1^{\theta} \to A_2 \vdash^{P} a : A_2}{\Delta, \Gamma \vdash^{P} rec\ f\ y.a : A_1^{\theta} \to A_2}$

To prove: $\rho(rec\ fy.a) = rec\ fy.\rho(a) \in RED^{\mathscr{C},k,P}_{A_1^{\theta} \to A_2}[\underline{R}/\underline{\alpha}]$

By definition of $RED^{\mathscr{C},k,P}_{A_1^{\theta} \to A_2}[\underline{R}/\underline{\alpha}]$, we need to prove

(i) $\underline{\eta} : \star \vdash^{P} rec\ fx.\rho(a) : \rho(A_1^{\theta} \to A_2)$ (by appeal to the substitution lemma 5.3) □

(ii) Being a value, $rec\ fx.\rho(a) \in RED^{\mathscr{V},k,P}_{A_1^{\theta} \to A_2}$, for which we now need to prove:

$\forall j < k(v \in RED^{\mathscr{V},j,\theta}_{A_1}[\underline{R}/\underline{\alpha}] \Rightarrow [v/x][rec\ f\ x.a/f]\rho(a) \in RED^{\mathscr{C},j,P}_{A_2}[\underline{R}/\underline{\alpha}])$

By induction hypothesis, $\Delta, \Gamma, y :^{\theta} A_1, f :^{P} A_1^{\theta} \to A_2 \vDash_{j,\underline{R}} \rho'$

where $j \leq k,\ \rho' = \rho[y \to v][f \to rec\ fy.a]$ s.t. $v \in RED^{\mathscr{V},j,\theta}_{A_1}[\underline{R}/\underline{\alpha}]$

and $rec\ fy.a \in RED^{\mathscr{V},j,P}_{A_1^{\theta} \to A_2}[\underline{R}/\underline{\alpha}]$ and $\rho'(a) \in RED^{\mathscr{C},j,P}_{A_2}[\underline{R}/\underline{\alpha}]$.

$\Rightarrow [x \to v][f \to rec\ fy.a]\rho(a) \in RED^{\mathscr{C},j,P}_{A_2}[\underline{R}/\underline{\alpha}]$

$\Rightarrow \forall j \leq k(v \in RED^{\mathscr{V},j,\theta}_{A_1}[\underline{R}/\underline{\alpha}] \Rightarrow ([v/x][f \to rec\, fy.a]\rho(a) \in RED^{\mathscr{C},j,P}_{A_2}[\underline{R}/\underline{\alpha}])$ □

TYPABS - L fragment $\dfrac{\Delta, \gamma : \star, \Gamma \vdash^{L} t : A}{\Delta, \Gamma \vdash^{L} \Lambda\gamma.t : \forall\gamma.A},\ \gamma \notin FV(\Gamma)$

To prove: $\rho(\Lambda\gamma.t) = \Lambda\gamma.\rho(t) \in RED^{\mathscr{C},k,L}_{\forall\gamma.A}[\underline{R}/\underline{\alpha}]$

By definition of $RED^{\mathscr{C},k,L}_{\forall\gamma.A}[\underline{R}/\underline{\alpha}]$, we need to prove

(i) $\underline{\eta} : \star \vdash^{L} \Lambda\gamma.\rho(t) : \forall\gamma.\rho(A)$ (proved by appeal to the substitution lemma 5.3) □

(ii) Being a value, $\Lambda\gamma.\rho(t) \in RED^{\mathscr{V},k,L}_{\forall\gamma.A}[\underline{R}/\underline{\alpha}]$, for which our proof obligation becomes:

$\forall$ types C and reducibility candidates S of type C,$[C/\gamma]\rho(t) \in RED^{\mathscr{C},k,L}_{A}[\underline{R}, S/\underline{\alpha}, \gamma]$.

By induction hypothesis, $\Delta, \gamma : \star, \Gamma \vDash_{j,\underline{R}} \rho'$ where $j \leq k,\ \rho' = \rho[\gamma \to C]$ and $\rho'(t) \in RED_A^{\mathscr{C},j,L}[\underline{R}, S/\underline{\alpha}, \gamma]$ for a valid type C and its reducibility candidate S.

Instantiating j with k, we have

$\Rightarrow [C/\gamma]\rho(t) \in RED_A^{\mathscr{C},k,L}[\underline{R}, S/\underline{\alpha}, \gamma]$ $\square$

TYPABS - P fragment $\dfrac{\Delta, \gamma : \star, \Gamma \vdash^P t : A}{\Delta, \Gamma \vdash^P \Lambda\gamma.t : \forall\gamma.A},\ \gamma \notin FV(\Gamma)$

To prove: $\rho(\Lambda\gamma.t) = \Lambda\gamma.\rho(t) \in RED_{\forall\gamma.A}^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$

By definition of $RED_{\forall\gamma.A}^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$, we need to prove

(i) $\underline{\eta} : \star \vdash^P \Lambda\gamma.\rho(t) : \forall\gamma.\rho(A)$ (proved by appeal to the substitution lemma 5.3) $\square$

(ii) Being a value, $\Lambda\gamma.\rho(t) \in RED_{\forall\gamma.A}^{\mathscr{V},k,P}[\underline{R}/\underline{\alpha}]$, for which our proof obligation becomes:

$\forall$ types C and reducibility candidates S of type C,$[C/\gamma]\rho(t) \in RED_A^{\mathscr{C},k-1,P}[\underline{R}, S/\underline{\alpha}, \gamma]$.

By induction hypothesis, $\Delta, \gamma : \star, \Gamma \vDash_{j,\underline{R}} \rho'$ where $j \leq k,\ \rho' = \rho[\gamma \to C]$ and $\rho'(t) \in RED_A^{\mathscr{C},j,P}[\underline{R}, S/\underline{\alpha}, \gamma]$ for a valid type C and its reducibility candidate S.

Instantiating j with $k-1$,

$\Rightarrow [C/\gamma]\rho(t) \in RED_A^{\mathscr{C},k-1,P}[\underline{R}, S/\underline{\alpha}, \gamma]\square$

TFOVAL $\dfrac{\Delta, \Gamma \vdash^P v : A \quad FO(A)}{\Delta, \Gamma \vdash^L v : A}$

To prove : $\rho(v) \in RED_A^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$

By induction hypothesis, we have $\rho(v) \in RED_A^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$.

As programmatic interpretation of first order types is the same as logical

interpretation (5.7) and A is a first order type (from the premise), hence

$RED_A^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}] = RED_A^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$

Therefore $\rho(v) \in RED_A^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$ $\square$

TROLL $\dfrac{\Delta, \Gamma \vdash^P a : [\mu\alpha.A/\alpha]A}{\Delta, \Gamma \vdash^P roll\ a : \mu\alpha.A}$

To prove : $\rho(roll\ a) = roll\ \rho(a) \in RED_{\mu\alpha.A}^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$

By definition of $RED_{\mu\alpha.A}^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$, we need to prove:

(i) $\underline{\eta} : \star \vdash^L roll\ \rho(a) : \mu\alpha.\rho(A)$ (by appeal to the substitution lemma 5.3) $\square$

(ii) if $roll\ \rho(a) \rightsquigarrow^j v$ s.t. v is irreducible, then $v \in RED_{\mu\alpha.A}^{\mathscr{V},k-j,P}[\underline{R}/\underline{\alpha}]$

As our language has deterministic operational semantics, $\rho(a)$ must have evaluated down to some irreducible term v_1 in j_1 steps s.t. $roll\ v_1 = v \wedge j_1 \leq j$. Our proof obligation now is to prove $v_1 \in RED_{[\mu\alpha.A/\alpha]A}^{\mathscr{V},k-j_1,P}[\underline{R}/\underline{\alpha}]$

By induction hypothesis, we have

$\rho(a) \in RED_{[\mu\alpha.A/\alpha]A}^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$.

Now $\rho(a) \rightsquigarrow^{j_1} v_1$.

Then, by definition of $RED_{[\mu\alpha.A/\alpha]A}^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$, $v_1 \in RED_{[\mu\alpha.A/\alpha]A}^{\mathscr{V},k-j_1,P}[\underline{R}/\underline{\alpha}]$ $\square$

TUNROLL $\dfrac{\Delta, \Gamma \vdash^P a : \mu\alpha.A}{\Delta, \Gamma \vdash^P unroll\ a : [\mu\alpha.A/\alpha]A}$

To prove: $\rho(unroll\ a) = unroll\ \rho(a) \in RED_{[\mu\alpha.A/\alpha]A}^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$

By definition of $RED_{[\mu\alpha.A/\alpha]A}^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$, we need to prove:

(i) $\underline{\eta} : \star \vdash^P unroll\ \rho(a) : [\mu\alpha.A/\alpha]\rho(A)$ (by appeal to the substitution lemma 5.3) $\square$

(ii) if $unroll\ \rho(a) \rightsquigarrow^j v$ s.t. v is irreducible, then $v \in RED_{[\mu\alpha.A/\alpha]A}^{\mathscr{V},k-j,P}[\underline{R}/\underline{\alpha}]$

As our language has deterministic operational semantics, $\rho(a)$ must have evaluated down to some irreducible term $unroll\ v_1$ in j_1 steps s.t. $unroll\ v_1 = v \wedge j_1 \leq j$. Our proof obligation now is to prove $v_1 \in RED_{\mu\alpha.A}^{\mathscr{V},k-j_1,P}[\underline{R}/\underline{\alpha}]$

By induction hypothesis, we have

$\rho(a) \in RED^{\mathscr{C},k,P}_{\mu\alpha.A}[\underline{R}/\underline{\alpha}]$.

Now $\rho(a) \rightsquigarrow^{j_1} v$. Then, by definition of $RED^{\mathscr{C},k,P}_{\mu\alpha.A}[\underline{R}/\underline{\alpha}]$, $v_1 \in RED^{\mathscr{V},k-j_1,P}_{\mu\alpha.A}[\underline{R}/\underline{\alpha}]$ $\square$

TSUB $\dfrac{\Delta, \Gamma \vdash^L a : A}{\Delta, \Gamma \vdash^P a : A}$

To prove : $\rho(a) \in RED^{\mathscr{C},k,P}_{A}[\underline{R}/\underline{\alpha}]$

As $RED^{\mathscr{V},k,L}_{A}[\underline{R}/\underline{\alpha}]$ is SIRC, we have

$RED^{\mathscr{V},k,L}_{A}[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{V},k,P}_{A}[\underline{R}/\underline{\alpha}]$

Applying lemma 5.8, we get

$RED^{\mathscr{C},k,L}_{A}[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{C},k,P}_{A}[\underline{R}/\underline{\alpha}]$

By induction hypothesis, $\rho(a) \in RED^{\mathscr{C},k,L}_{A}[\underline{R}/\underline{\alpha}] \in RED^{\mathscr{C},k,P}_{A}[\underline{R}/\underline{\alpha}]$ $\square$.

TBOXL $\dfrac{\Delta, \Gamma \vdash^L a : A}{\Delta, \Gamma \vdash^L box\ a : A@\theta}$

To prove : $\rho(box\ a) = box\ \rho(a) \in RED^{\mathscr{C},k,L}_{A@\theta}[\underline{R}/\underline{\alpha}]$

From definition of $RED^{\mathscr{C},k,L}_{A@\theta}[\underline{R}/\underline{\alpha}]$, we need to prove:

(i) $\underline{\eta} : \star \vdash^L box\ \rho(a) : \rho(A@\theta)$ (proved by appeal to the substitution lemma 5.3) $\square$

(ii) $box\ \rho(a) \rightsquigarrow^* v \in RED^{\mathscr{V},k,L}_{A@\theta}[\underline{R}/\underline{\alpha}]$

s.t. v is irreducible. If that is true then $\rho(a)$ must have also reduced to a term v_1 s.t. $box\ v_1 = v$. So, our proof obligation is now to show that $v_1 \in RED^{\mathscr{V},k,L}_{A}[\underline{R}/\underline{\alpha}]$

By induction hypothesis, we have $\rho(a) \in RED^{\mathscr{C},k,L}_{A}[\underline{R}/\underline{\alpha}]$.

Instantiating with v_1, we immediately get $v_1 \in RED^{\mathscr{V},k,L}_{A}[\underline{R}/\underline{\alpha}]$ $\square$

TBOXLV $\dfrac{\Delta, \Gamma \vdash^P v : A}{\Delta, \Gamma \vdash^L box\ v : A@P}$

To prove: $\rho(box\ v) = box\ \rho(v) \in RED^{\mathscr{C},k,L}_{A@P}[\underline{R}/\underline{\alpha}]$

From definition of $RED^{\mathscr{C},k,L}_{A@P}[\underline{R}/\underline{\alpha}]$, we need to prove:

(i) $\underline{\eta} : \star \vdash^L box\ \rho(v) : \rho(A@P)$ (proved by appeal to the substitution lemma 5.3) $\square$

(ii) Being a value, $box\ \rho(v) \in RED^{\mathscr{V},k,L}_{A@P}[\underline{R}/\underline{\alpha}]$

So, our proof obligation is now to show that

$\rho(v) \in RED^{\mathscr{V},k,P}_{A}[\underline{R}/\underline{\alpha}]$

By induction hypothesis, $\rho(v) \in RED^{\mathscr{C},k,P}_{A}[\underline{R}/\underline{\alpha}]$.

$\Rightarrow \rho(v) \in RED^{\mathscr{V},k,P}_{A}[\underline{R}/\underline{\alpha}]$ (as $\rho(v)$ is value) $\square$

TBOXP $\dfrac{\Delta, \Gamma \vdash^\theta a : A}{\Delta, \Gamma \vdash^P box\ a : A@\theta}$

To prove : $\underline{\eta} : \star \vdash^P \rho(box\ a) = box\ \rho(a) \in RED^{\mathscr{C},k,P}_{A@\theta}[\underline{R}/\underline{\alpha}]$

From definition of $RED^{\mathscr{C},k,P}_{A@\theta}[\underline{R}/\underline{\alpha}]$, we need to prove:

(i) $box\ \rho(a) : \rho(A@\theta)$ (proved by appeal to the substitution lemma 5.3) $\square$

(ii) $box\ \rho(a) \rightsquigarrow^j v \in RED^{\mathscr{V},k,P}_{A@\theta}[\underline{R}/\underline{\alpha}]$

s.t. v is irreducible. If that is true then $\rho(a)$ must have also reduced to a term v_1 in j_1 steps s.t. $box\ v_1 = v \wedge j_1 \leq j$. So, our proof obligation is now to show that

$v_1 \in RED^{\mathscr{V},k-j_1,P}_{A}[\underline{R}/\underline{\alpha}]$

By induction hypothesis, we have $\rho(a) \in RED^{\mathscr{C},k,\theta}_{A}[\underline{R}/\underline{\alpha}]$.

Subcase $\theta = P$

Instantiating with v_1, j_1, we immediately get $v_1 \in RED^{\mathscr{V},k-j_1,P}_{A}[\underline{R}/\underline{\alpha}]$

Subcase $\theta = L$

Instantiating with v_1, we get $v_1 \in RED^{\mathscr{V},k,L}_{A}[\underline{R}/\underline{\alpha}]$. As $RED^{\mathscr{V},k,L}_{A}[\underline{R}/\underline{\alpha}]$ is SIRC, we have by definition 5.5:

$v_1 \in RED^{\mathscr{V},k,L}_{A}[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{V},k,P}_{A}[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{V},k-j_1,P}_{A}[\underline{R}/\underline{\alpha}]$ $\square$

TUNBOX $\dfrac{\Delta, \Gamma \vdash^{\theta} a : A@\theta' \quad \Delta, \Gamma, x :^{\theta'} A \vdash^{\theta} b : B}{\Delta, \Gamma \vdash^{P} unbox\ x = a\ in\ b : B}$

To prove : $\rho(unbox\ x = a\ in\ b) \in RED_B^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$

From definition of $RED_B^{\mathscr{C},k,P}[\underline{R}/\underline{\alpha}]$, we need to prove:

(i) $\underline{\eta} : \star \vdash^{P} \rho(unbox\ x = a\ in\ b) : \rho(B)$ (proved by appeal to the substitution lemma 5.3) $\square$

(ii) If for some $j \leq k, \rho(unbox\ x = a\ in\ b) \rightsquigarrow^{j} v$ and v is irreducible, then $v \in RED_B^{\mathscr{V},k-j,P}[\underline{R}/\underline{\alpha}]$

As v is irreducible, $\rho(a)$ must have also reduced down to some irreducible term v' in j' steps, s.t. $unbox\ x = v'\ in\ \rho(v)$. Our proof obligation is now to show that $v' \in RED_A^{\mathscr{V},k-j',\theta'}[\underline{R}/\underline{\alpha}]$

By induction hypothesis, $\rho(a) \in RED_{A@\theta'}^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}]$ and

$\rho[x \to v'](b) \in RED_B^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}]$ s.t. $v' \in RED_{A@\theta'}^{\mathscr{V},k,\theta}[\underline{R}/\underline{\alpha}]$.

Now $\rho(a)$ reduces to v' in j' steps. Hence $v' \in RED_{A@\theta'}^{\mathscr{V},k-j',\theta}[\underline{R}/\underline{\alpha}]$.

TAPP $\dfrac{\Delta, \Gamma \vdash^{\theta} a : A^{\theta'} \to B \quad \Delta, \Gamma \vdash^{\theta} box\ b : A@\theta'}{\Delta, \Gamma \vdash^{\theta} ab : B}$

To prove : $\rho(ab) \in RED_B^{\mathscr{C},k,\theta}[\underline{R}/\underline{\alpha}]$

Case $(\theta = L)$ By induction hypothesis, we have $\rho(a) \in RED_{A^{\theta'} \to B}^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$ and $\rho(b) \in RED_{A@\theta'}^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$.

Let $\rho(a) \rightsquigarrow^{*} a' \wedge Val(a')$, then $a' \in RED_{(A^{\theta'} \to B)}^{\mathscr{V},k,L}[\underline{R}/\underline{\alpha}]$.

Similarly, let $\rho(b) \rightsquigarrow^{*} b' \wedge Val(b')$, then $(box\ b') \in RED_{A@\theta'}^{\mathscr{V},k,L}[\underline{R}/\underline{\alpha}]$.

From definition of $RED_{A^{\theta'} \to B}^{\mathscr{V},k,L}[\underline{R}/\underline{\alpha}]$, we have

$a'b' \in RED_B^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$.

Hence, from 5.10, $\rho(ab) \in RED_B^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$.

Case $(\theta = P)$ We need to prove that $\rho(ab) \in RED_B^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$.

If $\rho(a) \rightsquigarrow^{j} a' \wedge Val(a')$ and $\rho(b) \rightsquigarrow^{j_1} b' \wedge Val(b')$ then

$\rho(ab) \rightsquigarrow^{j+j_1} a'b'$, and from 5.10, our proof obligation now is to show that $a'b' \in RED_B^{\mathscr{C},k-j-j_1,P}[\underline{R}/\underline{\alpha}]$.

By induction hypothesis, we have $\rho(a) \in RED^{\mathscr{C},k,P}_{A^{\theta'} \to B}[\underline{R}/\underline{\alpha}]$ and $\rho(b) \in RED^{\mathscr{C},k,P}_{A@\theta'}[\underline{R}/\underline{\alpha}]$.

As $\rho(a) \rightsquigarrow^j a' \wedge Val(a')$, then $a' \in RED^{\mathscr{V},k-j,P}_{(A^{\theta'} \to B)}[\underline{R}/\underline{\alpha}]$.

Similarly, as $\rho(b) \rightsquigarrow^{j_1} b' \wedge Val(b')$, then $\rho(box\ b') \in RED^{\mathscr{V},k-j_1,P}_{A@\theta'}[\underline{R}/\underline{\alpha}]$.

From definition of $RED^{\mathscr{V},k-j,P}_{A^{\theta'} \to B}[\underline{R}/\underline{\alpha}]$, if $k - j_1 < k - j$ then we have $a'b' \in RED^{\mathscr{C},k-j_1,P}_{B}[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{C},k-j-j_1,P}_{B}[\underline{R}/\underline{\alpha}]$.

if $k - j_1 \geq k - j > k - j - j_1$, then as $RED^{\mathscr{V},k-j_1,P}_{A@\theta'}[\underline{R}/\underline{\alpha}]$ is SIRC (from lemma 5.9), we know that $RED^{\mathscr{V},k-j_1,P}_{A@\theta'}[\underline{R}/\underline{\alpha}] \subseteq RED^{\mathscr{V},k-j-j_1,P}_{A@\theta'}[\underline{R}/\underline{\alpha}]$.

Then as $\rho(box\ b') \in RED^{\mathscr{V},k-j-j_1,P}_{A@\theta'}[\underline{R}/\underline{\alpha}]$ and $k - j - j_1 < k - j$, so, $a'b' \in RED^{\mathscr{C},k-j-j_1,P}_{B}[\underline{R}/\underline{\alpha}]$.

TYPAPP $\dfrac{\Delta, \Gamma \vdash^{\theta} t : \forall\gamma.A \quad \Delta \vdash C : \star}{\Delta, \Gamma \vdash^{\theta} tC : [C/\gamma]A}$

Case $(\theta = L)$ To prove: $\rho(tC) \in RED^{\mathscr{C},k,L}_{[\rho(C)/\gamma]A}[\underline{R}/\underline{\alpha}]$.

By induction hypothesis, we have $\rho(t) \in RED^{\mathscr{C},k,L}_{\forall\gamma.A}[\underline{R}/\underline{\alpha}]$.

Let $\rho(t) \rightsquigarrow^* \Lambda\gamma.t' \wedge Val(\Lambda\gamma.t')$. Then $\Lambda\gamma.t' \in RED^{\mathscr{V},k,L}_{\forall\gamma.A}[\underline{R}/\underline{\alpha}]$. Instantiating the definition of $RED^{\mathscr{V},k,L}_{\forall\gamma.A}[\underline{R}/\underline{\alpha}]$, with type $\rho(C)$ and S (SIRC of the type $\rho(C)$), we get $[\rho(C)/\gamma]t' \in RED^{\mathscr{C},k,L}_{A}[\underline{R}, S/\underline{\alpha}, \gamma] = RED^{\mathscr{C},k,L}_{[\rho C/\gamma]A}[\underline{R}/\underline{\alpha}]$ using lemma 5.11.

As $\rho(tC) \rightsquigarrow^* [\rho(C)/\gamma]t'$, from lemma 5.10, we have $\rho(tC) \in RED^{\mathscr{C},k,L}_{[\rho(C)/\gamma]A}[\underline{R}/\underline{\alpha}]$.

Case $(\theta = P)$ To prove: $\rho(tC) \in RED^{\mathscr{C},k,P}_{[\rho(C)/\gamma]A}[\underline{R}/\underline{\alpha}]$. Or, if for some j, $\rho(t) \rightsquigarrow^j \Lambda\gamma.t'$ s.t. $Val(\Lambda\gamma.t')$ then $[\rho(C)/\gamma]t' \in RED^{\mathscr{C},k-j-1,P}_{[\rho(C)/\gamma]A}[\underline{R}/\underline{\alpha}]$.

By induction hypothesis, we have $\rho(t) \in RED^{\mathscr{C},k,P}_{\forall\gamma.A}[\underline{R}/\underline{\alpha}]$.

Let $\rho(t) \rightsquigarrow^j \Lambda\gamma.t' \wedge Val(\Lambda\gamma.t')$. Then $\Lambda\gamma.t' \in RED^{\mathscr{V},k-j,P}_{\forall\gamma.A}[\underline{R}/\underline{\alpha}]$.

Instantiating the definition of $RED^{\mathscr{V},k-j,P}_{\forall\gamma.A}[\underline{R}/\underline{\alpha}]$ with type $\rho(C)$ and its SIRC S, we get $[\rho(C)/\gamma]t' \in RED^{\mathscr{C},k-j-1,P}_{A}[\underline{R}, S/\underline{\alpha}, \gamma] = RED^{\mathscr{C},k-j-1,P}_{[\rho(C)/\gamma]A}[\underline{R}/\underline{\alpha}]$ using lemma 5.11.

□

Theorem 5.14 (Normalization) *If* $\bullet \vdash^L t : A$ *then* $\exists v(Val(v) \wedge t \rightsquigarrow^* v)$.

Proof: Initializing theorem 5.13 with empty context and L fragment, we have $t \in RED_A^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$ for any natural number k. Hence from the definition of $RED_A^{\mathscr{C},k,L}[\underline{R}/\underline{\alpha}]$, $\bullet \vdash^L t : A \wedge a \rightsquigarrow^* v \in RED_A^{\mathscr{V},k,L}[\underline{R}/\underline{\alpha}]$ □

References

[1] Chris Casinghino, Vilhelm Sjöberg, and Stephanie Weirich. "Step-Indexed Normalization for a Language with General Recursion". In: *Proceedings Fourth Workshop on Mathematically Structured Functional Programming, MSFP@ETAPS 2012, Tallinn, Estonia, 25 March 2012*. Ed. by James Chapman and Paul Blain Levy. Vol. 76. EPTCS. 2012, pp. 25–39. DOI: 10.4204/EPTCS.76.4. URL: http://dx.doi.org/10.4204/EPTCS.76.4.

[2] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types*. New York, NY, USA: Cambridge University Press, 1989. ISBN: 0-521-37181-3.

[3] Chris Casinghino, Vilhelm Sjöberg, and Stephanie Weirich. "Combining proofs and programs in a dependently typed language". In: *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*. Ed. by Suresh Jagannathan and Peter Sewell. ACM, 2014, pp. 33–46. DOI: 10.1145/2535838.2535883. URL: http://doi.acm.org/10.1145/2535838.2535883.

[4] Limin Jia et al. "Dependent Types and Program Equivalence". In: *SIGPLAN Not.* 45.1 (Jan. 2010), pp. 275–286. ISSN: 0362-1340. DOI: 10.1145/1707801.1706333. URL: http://doi.acm.org/10.1145/1707801.1706333.

[5] Andrew W. Appel and David McAllester. "An Indexed Model of Recursive Types for Foundational Proof-carrying Code". In: *ACM Trans. Program. Lang. Syst.* 23.5 (Sept. 2001), pp. 657–683. ISSN: 0164-0925. DOI: 10.1145/504709.504712. URL: http://doi.acm.org/10.1145/504709.504712.

[6] Henk Barendregt. "Lambda Calculi with Types". In: *HANDBOOK OF LOGIC IN COMPUTER SCIENCE*. Oxford University Press, 1992, pp. 117–309.

[7] Benjamin C. Pierce. *Advanced Topics in Types and Programming Languages*. The MIT Press, 2004. ISBN: 0262162288.

[8] Adam Chlipala. *Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant*. The MIT Press, 2013. ISBN: 0262026651, 9780262026659.

[9] A.K. Wright and M. Felleisen. "A Syntactic Approach to Type Soundness". In: *Inf. Comput.* 115.1 (Nov. 1994), pp. 38–94. ISSN: 0890-5401. DOI: 10.1006/inco.1994.1093. URL: http://dx.doi.org/10.1006/inco.1994.1093.

[10] Amal Ahmed. *Logical Relations*. 2015. URL: https://www.cs.uoregon.edu/research/summerschool/summer15/index.php.