Abstract

Proof search has been used to specify a wide range of computation systems. In order to build a framework for reasoning about such specifications, we make use of a sequent calculus involving induction and co-induction. These proof principles are based on a proof theoretic (rather than set-theoretic) notion of definition \[10, 11, 47, 25\]. Definitions are akin to logic programs, where the left and right rules for defined atoms allow one to view theories as “closed” or defining fixed points. The use of definitions and free equality makes it possible to reason intentionally about syntax. We add in a consistent way rules for pre and post fixed points, thus allowing the user to reason inductively and co-inductively about properties of computational systems making full use of higher-order abstract syntax. Consistency is guaranteed via cut-elimination, where we give the first, to our knowledge, cut-elimination procedure in the presence of general inductive and co-inductive definitions.

Key words: logical frameworks, (co)-induction, higher-order abstract syntax, cut-elimination, parametric reducibility.

1. Introduction

A common approach to specifying computation systems is via deductive systems. Those are used to specify and reason about various logics, as well as aspects of programming languages such as operational semantics, type theories, abstract machines etc. Such specifications can be represented as logical theories in a suitably expressive formal logic where proof-search can then be used to model the computation. A logic used as a specification language is known as a logical frameworks [38], which comes equipped with a representation methodology. The encoding of the syntax of deductive systems inside formal logic can benefit from the use of higher-order abstract syntax (HOAS) a high-level and declarative treatment of object-level bound variables and substitution. At the same time, we want to use such a logic to reason over the meta-theoretical properties of object languages, for example type preservation in operational semantics [20], soundness and completeness of compilation [32] or congruence of bisimulation in transition systems [27]. Typically this involves reasoning by (structural) induction and, when dealing with infinite behavior, co-induction [23].

The need to support both inductive and co-inductive reasoning and some form of HOAS requires some careful design decisions, since the two are prima facie notoriously incompatible. While any meta-language based on a λ-calculus can be used to specify and animate HOAS encodings, meta-reasoning has traditionally involved (co)inductive specifications both at the level of the syntax and of the judgements — which are
of course unified at the type-theoretic level. The first provides crucial freeness properties for datatypes constructors, while the second offers principles of case analysis and (co)induction. This is well-known to be problematic, since HOAS specifications may lead to non-monotone (co)inductive operators, which by cardinality and consistency reasons are not permitted in inductive logical frameworks. Moreover, even when HOAS is weakened so as to be made compatible with standard proof assistants \cite{10} such as HOL or Coq, the latter suffer the fate of allowing the existence of too many functions and yielding the so-called exotic terms. Those are canonical terms in the signature of an HOAS encoding that do not correspond to any term in the deductive system under study. This causes a loss of adequacy in HOAS specifications, which is one of the pillar of formal verification, and it undermines the trust in formal derivations. On the other hand, logics such as LF \cite{20} that are weak by design in order to support this style of syntax are not directly endowed with (co)induction principles.

The contribution of this paper lies in the design of a new logic, called \textit{Linc} \textsuperscript{−} (for a logic with \(\lambda\)-terms, induction and co-induction) \cite{3} which carefully adds principles of induction and co-induction to a higher-order intuitionistic logic based on a proof theoretic notion of \textit{definition}, following on work (among others) by Lars Hallnäs \cite{19}, Eriksson \cite{11}, Schroeder-Heister \cite{47} and McDowell and Miller \cite{25}. Definitions are akin to logic programs, but allow us to view theories as “closed” or defining fixed points. This alone permits to perform case analysis independently from induction principles. Our approach to formalizing induction and co-induction is via the least and greatest solutions of the fixed point equations specified by the definitions. The proof rules for induction and co-induction make use of the notion of \textit{pre-fixed points} and \textit{post-fixed points} respectively. In the inductive case, this corresponds to the induction invariant, while in the co-inductive one to the so-called simulation. Judgements are encoded as definitions accordingly to their informal semantics, either inductive or co-inductive.

The simply typed language and the notion of free equality underlying \textit{Linc} \textsuperscript{−}, enforced via (higher-order) unification in an inference rule, make it possible to reason \textit{intensionally} about syntax. In fact, we can support HOAS encodings of constants and we can \textit{prove} the freeness properties of those constants, namely injectivity, distinctness and case exhaustion, although they cannot be the constructors of a (recursive) datatype. \textit{Linc} \textsuperscript{−} can be proved to be a conservative extension of \textit{FO} \(\lambda\Delta \mathbb{N}\) \cite{25} and a generalization (with a term language based on simply typed \(\lambda\)-calculus) of Martin-Löf first-order theory of iterated inductive definitions \cite{24}. Moreover, to the best of our knowledge, it is the first sequent calculus with a syntactical cut-elimination theorem for co-inductive definitions. In recent years, several logical systems have been designed that build on the core features of \textit{Linc} \textsuperscript{−}. In particular, one interesting, and orthogonal, extension is the addition of the \(\nabla\)-quantifier \cite{31,52,53,14}, which allows one to reason about the intentional aspects of \textit{names and bindings} in object syntax specifications (see, e.g., \cite{15, 54}). The cut elimination proof presented in this paper can be used as a springboard towards cut elimination procedures for more expressive (conservative) extensions of \textit{Linc} \textsuperscript{−}.

In fact, the possibility of adapting the cut elimination proof for \textit{Linc} \textsuperscript{−} to various extensions of \textit{Linc} \textsuperscript{−} with \(\nabla\) is one of the main reasons to introduce a \textit{direct} syntactic cut elimination proof. We note that there are at least a couple of indirect methods to prove cut elimination in a logic with inductive and/or co-inductive definitions. The first of such methods relies on encodings of inductive and co-inductive definitions as second-order (or higher-order) formulæ. This approach is followed in a recent work by Baelde and Miller \cite{6} where a logic similar to \textit{Linc} \textsuperscript{−} is considered. Cut elimination in their work is proved indirectly via an encoding into higher-order linear logic. However, in the presence of \(\nabla\), the existence of such an encoding is presently unknown. The second approach is via semantical methods. This approach is taken in a recent work by Brotherston and Simpson \cite{8}, which provide a model for a classical first-order logic with inductive definitions, hence, cut elimination follows by the semantical completeness of the cut free fragment. It is not obvious how such semantical methods can be adapted to prove cut elimination for extensions of \textit{Linc} \textsuperscript{−} with \(\nabla\). This is because the semantics of \(\nabla\) itself is not yet very well understood, although there have been some recent attempts, see \cite{20,46,12}.

The present paper is an extended and revised version of \cite{33}. In the conference paper, the co-inductive rule had a technical side condition that is restrictive and unnatural. The restriction was essentially imposed by the particular cut elimination proof technique outlined in that paper. This restriction has been removed.

\footnote{The “minus” in the terminology refers to the lack of the \(\nabla\)-quantifier w.r.t. the eponymous logic in Tiu’s thesis \cite{52}.}
in the present version, and the (co-)induction rules have been generalized. For the latter, the formulation of the rules is inspired by a second-order encoding of least and greatest fixed points. Consequently, we now develop a new cut elimination proof, which is radically different from the previous proof, using a reducibility-candidate technique, which is influenced by Girard’s strong normalisation proof for System F [18]. This paper is concerned only with the cut elimination proof of Linc\(^{−}\). For examples and applications of Linc\(^{−}\) and its extensions with \(\nabla\), we refer the interested reader to [52, 5, 14, 13, 15, 54].

The rest of the paper is organized as follows. Section 2 introduces the sequent calculus for the logic. Section 3 presents two transformations of derivations that are essential to the cut reduction rules and the cut elimination proof in subsequent sections. Section 4 is the heart of the paper: we first (Subsection 4.1) give a (sub)set of reduction rules that transform a derivation ending with a cut rule to another derivation. The complete set of reduction can be found in Appendix A. We then introduce the crucial notions of normalizability (Subsection 4.2) and of parametric reducibility after Girard (Subsection 4.3). Detailed proofs of the main lemma related to reducibility candidates are in Appendix B. The central result of this paper, i.e., cut elimination, is proved in details in Subsection 4.4. Section 5 surveys the related work and concludes the paper.

2. The Logic Linc\(^{−}\)

The logic Linc\(^{−}\) shares the core fragment of FO\(\lambda\Delta^N\), which is an intuitionistic version of Church’s Simple Theory of Types. We shall assume that the reader is familiar with Church’s simply typed \(\lambda\)-calculus (with both \(\beta\) and \(\eta\) rules), so we shall recall only the basic syntax of the calculus here. A simple type is either a base type or a compound type formed using the function-type constructor \(\to\). Types are ranged over by \(\alpha\), \(\beta\) and \(\tau\). We assume an infinite set of typed variables, written \(x_\alpha\), \(y_\beta\), etc. The syntax of \(\lambda\)-terms is given by the following grammar:

\[
s, t ::= x_\tau \mid (\lambda x_\tau, t) \mid (s\ t)
\]

To simplify presentation, in the following we shall often omit the type index in variables and \(\lambda\)-abstraction. The notion of free and bound variables are defined as usual.

Following Church, we distinguish a base type \(o\) to denote formulae, and we shall represent formulæ as simply typed \(\lambda\)-terms of type \(o\). We assume a set of typed constants that correspond to logical connectives. The constants \(\top\) and \(\bot\) denote ‘true’ and ‘false’, respectively. Propositional binary connectives, i.e., \(\land\), \(\lor\), and \(\exists\), are assigned the type \(o \to o \to o\). Quantifiers are represented by indexed families of constants: \(\forall_\tau\) and \(\exists_\tau\), both are of type \((\tau \to o) \to o\). We also assume a family of typed equality symbols \(\equiv_\tau\) \(\tau \to \tau \to o\). Although we adopt a representation of formulæ as \(\lambda\)-terms, we shall use a more traditional notation when writing down formulæ. For example, instead of writing \((\forall A\ B)\), we shall use an infix notation \((A \land B)\). Similarly, we shall write \(\forall_\alpha x\ P\) instead of \(\forall_\alpha (\lambda x_\alpha\ P)\). Again, we shall omit the type annotation when it can be inferred from the context of the discussion.

The type \(\tau\) in quantifiers and the equality predicate are restricted to those simple types that do not contain occurrences of \(o\). Hence our logic is essentially first-order, since we do not allow quantification over predicates. As we shall often refer to this kind of restriction to types, we give the following definition:

**Definition 1.** A simple type \(\tau\) is essentially first-order (efo) if it is generated by the following grammar:

\[
\tau ::= k \mid \tau \to \tau
\]

where \(k\) is a base type other than \(o\).

For technical reasons (for presenting (co-)inductive proof rules), we introduce a notion of parameter into the syntax of formulæ. Intuitively, they play the role of eigenvariables ranging over the recursive call in a fixed point expression. More precisely, to each predicate symbol \(p\), we associate a countably infinite set \(\mathcal{F}_p\), called the parameter set for \(p\). Elements of \(\mathcal{F}_p\) are ranged over by \(X^p\), \(Y^p\), \(Z^p\), etc, and have the same type as \(p\). When we refer to formulæ of Linc\(^{−}\), we have in mind simply-typed \(\lambda\)-terms of type \(o\) in \(\beta\eta\)-long normal form. Thus formulæ of the logic Linc\(^{−}\) can be equivalently defined via the following grammar:

\[
F ::= X^p t^p s =_\tau t \mid p t^p \mid \top \mid F \land F \mid F \lor F \mid F \supset T \mid \forall_\tau x.F \mid \exists_\tau x.F.
\]
\[
\frac{C \rightarrow C}{\text{init}} \quad \frac{B, B, \Gamma \rightarrow C}{B, \Gamma \rightarrow C} \quad \frac{\Gamma \rightarrow C}{wL}
\]

\[
\frac{\Delta_1 \rightarrow B_1 \quad \ldots \quad \Delta_n \rightarrow B_n \quad B_1, \ldots, B_n, \Gamma \rightarrow C}{\Delta_1, \ldots, \Delta_n, \Gamma \rightarrow C}
\]

\[\text{mc, where } n > 0\]

\[
\frac{\bot, \Gamma \rightarrow B}{sL} \quad \frac{\Gamma \rightarrow \top}{\top R}
\]

\[
\frac{B_i, \Gamma \rightarrow D}{B_1 \land B_2, \Gamma \rightarrow D} \quad \frac{B, \Gamma \rightarrow D}{\land L, i \in \{1, 2\}} \quad \frac{B, \Gamma \rightarrow D}{\land R, i \in \{1, 2\}}
\]

\[
\frac{B \lor C, \Gamma \rightarrow D}{B \lor C, \Gamma \rightarrow D} \quad \frac{\Gamma \rightarrow B}{\lor L} \quad \frac{\Gamma \rightarrow B}{\lor R}
\]

\[
\frac{\Gamma \rightarrow B}{B \supset C, \Gamma \rightarrow D} \quad \frac{\Gamma \rightarrow B}{\supset L} \quad \frac{\Gamma \rightarrow B, y}{\supset R, i \in \{1, 2\}}
\]

\[
\frac{\forall x. B x, \Gamma \rightarrow C}{\exists x. B x, \Gamma \rightarrow C} \quad \frac{\exists x. B x, \Gamma \rightarrow C}{\exists R}
\]

\[
\begin{align*}
\{ \Gamma \rho \rightarrow C \rho, \mid s \rho = \beta \eta t \rho \} & \quad \text{eqL} \\
\frac{s = t, \Gamma \rightarrow C}{t = t} & \quad \text{eqR}
\end{align*}
\]

\[
\begin{align*}
\frac{B S \vec{y} \rightarrow S \vec{y}, \Gamma, S \vec{t} \rightarrow C}{\Gamma, p \vec{t} \rightarrow C} & \quad \text{IL, } p \vec{x} \equiv B p \vec{x} \\
\frac{\Gamma \rightarrow B X p \vec{t}}{\Gamma \rightarrow p \vec{t}} & \quad \text{IR, } p \vec{x} \equiv B p \vec{x}
\end{align*}
\]

\[
\begin{align*}
\frac{B X p \vec{t}, \Gamma \rightarrow C}{\Gamma \rightarrow S \vec{t}} & \quad \text{CLR, } p \vec{x} \equiv B p \vec{x} \\
\frac{B X p \vec{t}, \Gamma \rightarrow C}{X p \vec{t}, \Gamma \rightarrow C} & \quad \text{CLR}_p, p \vec{x} \equiv B p \vec{x}
\end{align*}
\]

\[
\begin{align*}
\frac{\Gamma \rightarrow S \vec{t}}{p \vec{t}, \Gamma \rightarrow C} & \quad \text{CLR, } p \vec{x} \equiv B p \vec{x} \\
\frac{\Gamma \rightarrow S \vec{t}}{S \vec{y}, \Gamma \rightarrow B S \vec{y}} & \quad \text{CLR, } p \vec{x} \equiv B p \vec{x}
\end{align*}
\]

\[\text{Figure 1: The inference rules of } \mathbb{L}_n\]
where $\tau$ is an efo-type. We shall omit the type annotation in $s =_\tau t$ when it is not important to the discussion.

A substitution is a type-preserving mapping from variables to terms. We assume the usual notion of capture-avoiding substitutions. Substitutions are ranged over by lower-case Greek letters, e.g., $\theta$, $\rho$ and $\sigma$. Application of substitution is written in postfix notation, e.g., $t\theta$ denotes the term resulting from an application of substitution $\theta$ to $t$. Composition of substitutions, denoted by $\circ$, is defined as $(t(\theta \circ \rho)) = (t\theta)\rho$.

The whole logic is presented in the sequent calculus in Figure 1, including rules for equality and fixed points, as we discuss in Section 2.1 and 2.2. A sequent is denoted by $\Gamma \rightarrow C$ where $C$ is a formula in $\beta\eta$-long normal form and $\Gamma$ is a multiset of formulae, also in $\beta\eta$-$\beta\eta$-long normal form. Notice that in the presentation of the rule schemes, we make use of HOAS, e.g., in the application $Bx$ it is implicit that $B$ has no free occurrence of $x$. Similarly for the (co)induction rules. We work modulo $\alpha$-conversion without further notice. In the $\forall R$ and $\exists L$ rules, $y$ is an eigenvariable that is not free in the lower sequent of the rule. The $mc$ rule is a generalization of the cut rule that simplifies the presentation of the cut-elimination proof.

Whenever we write a sequent, it is assumed implicitly that the formulae are well-typed: the type context, i.e., the types of the constants and the eigenvariables used in the sequent, is left implicit as well as they can be inferred from the type annotations of the (eigen)variables.

In some inference rules, reading them bottom up, new eigenvariables and parameters may be introduced in the premises of the rules, for instance, in $\exists L$ and $\forall R$, as typical in sequent calculus. However, unusually, we shall also allow $\exists R$, $\forall L$ and $mc$ to possibly introduce new eigenvariables (and new parameters, in the case of $mc$), again reading the rules bottom-up. Thus the term $t$ in the premise of the $\exists R$-rule may contain a free occurrence of an eigenvariable not already occurring in the conclusion of the rule. The implication of this is that $\exists x. \bot$ is provable for any type $\tau$; in other words, there is an implicit assumption that all types are non-empty. Hence the quantifiers in our setting behave more classically than intuitionistically. The reason for this rather awkward treatment of quantifiers is merely a technical convenience. We could forgo the non-emptiness assumption on types by augmenting sequents with an explicit signature acting as a typing environment, and insisting that the term $t$ in $\exists R$ to be well-formed under the typing environment of the conclusion of the rule. However, adding explicit typing contexts into sequents introduces another layer of bureaucracy in the proof of cut elimination, which is not especially illuminating. And since our primary goal is to show the central arguments in cut elimination involving (co-)induction, we opt to present a slightly simplified version of the logic so that the main technical arguments (which are already quite complicated) in the cut elimination proof, related to (co-)induction rules, can be seen more clearly. The cut elimination proof presented in the paper can be adapted to a different presentation of $\text{Linc}^-$ with explicit typing contexts; see [52, 53] for an idea of how such an adaptation may be done.

We extend the logical fragment with a proof theoretic notion of equality and fixed points.

### 2.1. Equality

The right introduction rule for equality is reflexivity, that is, it recognizes that two terms are syntactically equal. The left introduction rule is more interesting. The substitution $\rho$ in $eq\mathcal{L}$ is a unifier of $s$ and $t$. Note that we specify the premise of $eq\mathcal{L}$ as a set, with the intention that every sequent in the set is a premise of the rule. This set is of course infinite; every unifier of $(s, t)$ can be extended to another one (e.g., by adding substitution pairs for variables not in the terms). However, in many cases, it is sufficient to consider a particular set of unifiers, which is often called a complete set of unifiers (CSU) $\mathcal{U}$, from which any unifier can be obtained by composing a member of the CSU set with a substitution. In the case where the terms are first-order terms, or higher-order terms with the pattern restriction [30], the set CSU is a singleton, i.e., there exists a most general unifier (MGU) for the terms.

Our rules for equality actually encompasses the notion of free equality as commonly found in logic programming, in the form of Clark’s equality theory [3]: injectivity of function symbols, inequality between distinct function symbols, and the “occur-check” follow from rule $eq\mathcal{L}$-rule. For instance, given a base type $nt$ (for natural numbers) and the constants $z : nt$ (zero) and $s : nt \to nt$ (successor), we can derive $\forall x. z = (s\ x) \supset \bot$ as follows:

\[
\begin{align*}
\frac{z = (s\ y) \supset \bot}{eq\mathcal{L}} \\
\frac{\supset z = (s\ y) \supset \bot}{\forall R} \\
\frac{\supset \forall x. z = (s\ x) \supset \bot}{\forall R}
\end{align*}
\]
Since $z$ and $s \overline{y}$ are not unifiable, the eq$L$ rule above has empty premise, thus concluding the derivation. A similar derivation establishes the occur-check property, e.g., $\forall x. x = (s x) \supset \bot$. We can also prove the injectivity of the successor function, i.e. $\forall x\forall y. (s x) = (s y) \supset x = y$.

This proof theoretic notion of equality has been considered in several previous work e.g. by Schroeder-Heister [47], and McDowell and Miller [25].

2.2. Induction and co-induction

One way of adding induction and co-induction to a logic is to introduce fixed point expressions and their associated introduction and elimination rules, i.e. using the $\mu$ and $\nu$ operators of the (first-order) $\mu$-calculus. This is essentially what we shall follow here, but with a different notation. Instead of using a "nameless" notation with $\mu$ and $\nu$ to express fixed points, we associate a fixed point equation with an atomic formula. That is, we associate certain designated predicates with a definition. This notation is clearer and more convenient as far as our applications are concerned. For a proof system using nameless notation for (co)inductive predicates, the interested reader is referred to recent work by Baelde and Miller [6].

**Definition 2.** An *inductive definition clause* is written $\forall \overline{x}. p \overline{x} \overset{\mu}{=} B \overline{x}$, where $p$ is a predicate constant. The atomic formula $p \overline{x}$ is called the *head* of the clause, and the formula $B \overline{x}$, where $B$ is a closed term containing no occurrences of parameters, is called the *body*. Similarly, a *co-inductive definition clause* is written $\forall \overline{x}. p \overline{x} \overset{\nu}{=} B \overline{x}$. The symbols $\overset{\mu}{=}$ and $\overset{\nu}{=}$ are used simply to indicate a definition clause: they are not a logical connective. We shall write $\forall \overline{x}. p \overline{x} \overset{\triangle}{=} B \overline{x}$ to denote a definition clause generally, i.e., when we are not interested in the details of whether it is an inductive or a co-inductive definition. A *definition* is a finite set of definition clauses. A predicate may occur only at most once in the heads of the clauses of a definition. We shall restrict to *non-mutually recursive* definitions. That is, given two clauses $\forall \overline{x}. p \overline{x} \overset{\triangle}{=} B \overline{x}$ and $\forall \overline{y}. q \overline{y} \overset{\triangle}{=} C \overline{y}$ in a definition, where $p \neq q$, if $p$ occurs in $C$ then $q$ does not occur in $B$, and vice versa.

Note that the above restriction to non-mutual recursion is immaterial, since in the first-order case it is well known how one can easily encode mutually recursive predicates as a single predicate with an extra argument. The rationale behind that restriction is merely to simplify the presentation of inference rules and the cut elimination proof. Were we to allow mutually recursive definitions, the introduction rules $\text{IC}$ and $\text{CIR}$ for a predicate $p$ would have possibly more than two premises, depending on the number of predicates which are mutually dependent on $p$ (see [4] for a presentation of introduction rules for mutually dependent definitions).

For technical convenience we also bundle up all the definitional clause for a given predicate in a single clause, following the same principles of the iff-completion in logic programming. Further, in order to simplify the presentation of rules that involve predicate substitutions, we denote a definition using an abstraction over predicates, that is

$$\forall \overline{x}. p \overline{x} \overset{\triangle}{=} B p \overline{x}$$

where $B$ is an abstraction with no free occurrence of predicate symbol $p$ and variables $\overline{x}$. Substitution of $p$ in the body of the clause with a formula $S$ can then be written simply as $B S \overline{x}$. When writing definition clauses, we often omit the outermost universal quantifiers, with the assumption that free variables in a clause are universally quantified. For example even numbers are defined as follows:

$$ev \ x \overset{\lambda}{=} (x = z) \lor (\exists y. x = (s (s y)) \land ev \ y)$$

where in this case $B$ is of the form $\lambda p w. (w = z) \lor (\exists y. w = (s (s y)) \land p y)$.

The left and right rules for (co-)inductively defined atoms are given at the bottom of Figure 1. In rules $\text{IC}$ and $\text{CIR}$, the abstraction $S$ is an invariant of the (co-)induction rule. The variables $\overline{y}$ are new eigenvariables and $Xp$ is a new parameter not already occurring in the lower sequent. For the induction rule $\text{IC}$, $S$ denotes a pre-fixed point of the underlying fixed point operator. Similarly, for the co-induction rule $\text{CIR}$, $S$ can be seen as denoting a post-fixed point of the same operator. Here, we use a characterization of induction and co-induction proof rules as, respectively, the least and the greatest solutions to a fixed point equation.

Notice that the right-introduction rules for inductive predicates and parameters (dually, the left-introduction rules for co-inductive predicates and parameters) are slightly different from the corresponding rules in Lin-like logics (see Remark [4]). These rules can be better understood by the usual interpretation of (co-)inductive
definitions in second-order logic \cite{39,37} (to simplify presentation, we show only the propositional case here):

\[
\begin{align*}
\mu p \equiv B p & \Rightarrow \forall p. (B p \supset p) \supset p \\
\nu p \equiv B p & \Rightarrow \exists p. p \land (p \supset B p).
\end{align*}
\]

Then the right-introduction rule for inductively defined predicate will involve an implicit universal quantification over predicates. As standard in sequent calculus, such a universal quantified predicate will be replaced by a new eigenvariable (in this case, a new parameter), reading the rule bottom up. Note that if we were to follow the above second-order interpretation literally, an alternative rule for inductive predicates could be:

\[
\frac{B X p \supset X p, \Gamma \rightarrow X p}{\Gamma \rightarrow p} \text{ IR}_{\mu} \quad \mu X \equiv B X.
\]

Then there would be no need to add the IR_{\mu}-rule since it would be derivable, using the clause \( B X p \supset X p \) in the left hand side of the sequent. (This, of course, is true only when such an IR_{\mu} instance appears above an IR instance for \( p \).) Our presentation has the advantage that it simplifies the cut-elimination arguments in the subsequent sections. The left-introduction rule for co-inductively defined predicate can be explained dually.

A similar encoding of (co-)inductive definitions as second-order formulae is used in \cite{38}, where cut-elimination is indirectly proved by appealing to a focused proof system for higher-order linear logic. A similar approach can be followed for Linc, but we prefer to develop a direct cut-elimination proof, since such a proof can serve as the basis of cut-elimination for extensions of Linc, for example, with the \( \nabla \)-quantifier \cite{31,14}.

**Remark 1 (Fixed point unfolding).** A commonly used form of introduction rules for definitions, or fixed points, uses an unfolding of the definitions. This form of rules is followed in several related logics, e.g., \( FOL^{\Delta IN} \) \cite{25}, Linc \cite{33,52} and \( \mu \)-MALL \cite{3}. The right-introduction rule for inductive definitions, for instance, takes the form:

\[
\frac{\Gamma \rightarrow B p \vec{t}}{\Gamma \rightarrow p \vec{t}} \text{ IR}', \quad \mu \equiv B p \vec{x}.
\]

That is, in the premise, the predicate \( p \) is replaced with the body of the definition. The logic Linc, like \( FOL^{\Delta IN} \), imposes a stratification on definitions, which amounts to a strict positivity condition: the head of a definition can only appear in a strictly positive position in the body, i.e., it never appears to the left of an implication. Let us call such a definition a stratified definition. For stratified definitions, the rule IR' can be derived as follows:

\[
\begin{align*}
&\frac{B X p \vec{x} \rightarrow X p \vec{u} \text{ init}}{\frac{X p \vec{u} \rightarrow X p \vec{u} \text{ init}}{B X p \vec{x} \rightarrow X p \vec{u} \text{ init}}}
\end{align*}
\]

\[
\frac{B p \vec{t} \rightarrow B X p \vec{t}}{\frac{B p \vec{t} \rightarrow p \vec{t} \text{ init}}{B p \vec{t} \rightarrow p \vec{t} \text{ init}}} \text{ IR} \quad \mu \equiv B p \vec{x}.
\]

where the ‘dots’ are a derivation composed using left and right introduction rules for logical connectives in \( B \). Notice that all leaves of the form \( p \vec{u} \rightarrow X p \vec{u} \) can be proved by using the LC rule, with \( X p \) as the inductive invariant. Conversely, given a stratified definition, any proof in Linc using that definition can be transformed into a proof of Linc simply by replacing \( X p \) with \( p \). Note that once IR' is shown admissible, one can also prove admissibility of unfolding of inductive definitions on the left of a sequent; see \cite{52} for a proof.

Since a defined atomic formula can be unfolded via its introduction rules, the notion of size of a formula as simply the number of connectives in it would not take into account this possible unfolding. We shall define a more general notion assigning a positive integer to each predicate symbol, which we refer to as its level. A similar notion of level of a predicate was introduced for \( FOL^{\Delta IN} \) \cite{25}. However, in \( FOL^{\Delta IN} \), the level of a predicate is only used to guarantee monotonicity of definitions.
Definition 3 (Size of formulae). To each predicate $p$ we associate a natural number $\text{lvl}(p)$, the level of $p$. Given a formula $B$, its size $|B|$ is defined as follows:

1. $|X^p \bar{t}| = 1$, for any $X^p$ and any $\bar{t}$.
2. $|p \bar{t}| = \text{lvl}(p)$.
3. $|\bot| = |\top| = |(s = t)| = 1$.
4. $|B \land C| = |B \lor C| = |B \supset C| = |B| + |C| + 1$.
5. $|\forall x. B x| = |\exists x. B x| = |B x| + 1$.

Note that in this definition, we do not specify precisely any particular level assignment to predicates. We show next that there is a level assignment that has a property that will be useful later in proving cut elimination.

Lemma 1 (Level assignment). Given any definition $D$, there is a level assignment to every predicate $p$ occurring in $D$ such that if $\forall \bar{x}. p \bar{x} \triangleq B p \bar{x}$ is in $D$, then $|p \bar{x}| > |B X^p \bar{x}|$ for every parameter $X^p \in \mathcal{P}_p$.

Proof. Let $\prec$ be a binary relation on predicate symbols defined as follows: $q \prec p$ iff $q$ occurs in the body of the definition clause for $p$. Let $\prec^*$ be the reflexive-transitive closure of $\prec$. Since we restrict to non-mutually recursive definitions and there are only finitely many definition clauses (Definition 2), it follows that $\prec^*$ is a well-founded partial order. We now compute a level assignment to predicate symbols by induction on $\prec^*$. This is simply done by letting $\text{lvl}(p) = 1$, if $p$ is undefined, and $\text{lvl}(p) = |B X^p \bar{x}| + 1$, for some parameter $X^p$, if $\forall \bar{x}. p \bar{x} \triangleq B p \bar{x}$. Note that in the latter case, by induction hypothesis, every predicate symbol $q$, other than $p$, in $B$ has already been assigned a level, so $|B X^p \bar{x}|$ is already defined at this stage. Note also that it does not matter which $X^p$ we choose since all parameters have the same size. 

We shall assume from now on that predicates are assigned levels satisfying the condition of Lemma 1 so whenever we have a definition clause of the form $\forall \bar{x}. p \bar{x} \triangleq B p \bar{x}$, we shall implicitly assume that $|p \bar{x}| > |B X^p \bar{x}|$ for every parameter $X^p \in \mathcal{P}_p$.

Remark 2 (Non-monotonicity). In FOL$\Delta$N, a notion of stratification is used to rule out non-monotone (or in Halnäs’ terminology partial [19]) definitions, such as, $p \triangleq p \supset \bot$, for which cut-elimination is problematic. In fact, from the above definition both $p$ and $p \supset \bot$ are provable, but there is no direct proof of $\bot$. This can be traced back to the fact that unfolding of definitions in Lincl and FOL$\Delta$N is allowed on both the left and the right hand side of sequent. In Lincl$^-$, inconsistency does not even allow a non-monotone definition as above, due to the fact that arbitrary unfolding of fixed points is not permitted. Instead, only a limited form of unfolding is allowed, i.e., in the form of unfolding of inductive parameters on the right, and co-inductive parameters on the left. As a consequence of this restrictive unfolding, in Lincl$^-$ one cannot reason about some well-founded inductive definitions which are not stratified. For example, consider the non-stratified definition:

$$\forall x. ev \ x \triangleq (x = z) \lor (\exists y. x = (s \ y) \land (ev \ y \supset \bot))$$

If this definition were to be interpreted as a logic program (with negation-as-failure), for example, then its least fixed point is exactly the set of even natural numbers. However, the above encoding in Lincl$^-$ is incomplete with respect to this interpretation, since not all even natural numbers can be derived using the above definition. For example, it is easy to see that $ev \ (s \ (s \ z))$ is not derivable, since this would require a derivation of $X^{ev} \ (s \ z) \rightarrow \bot$, for some inductive parameter $X^{ev}$, which is impossible because no unfolding of inductive parameter is allowed on the left of a sequent. The same idea prevents the derivability of $\rightarrow p$ given the definition $p \triangleq p \supset \bot$. So while inconsistency in the presence of non-monotone definitions is avoided in Lincl$^-$, its reasoning power does not extend that of Lincl significantly.

\footnote{Other ways beyond stratification of recovering cut-elimination are disallowing contraction or restricting to an \textit{init} rule for undefined atoms.}
3. Eigenvariables and parameters instantiations

We now discuss some properties of derivations in $\text{Linc}^-$ which involve instantiations of eigenvariables and parameters. These properties will be used in the cut-elimination proof in subsequent sections.

Before we proceed, it will be useful to introduce the following derived rule in $\text{Linc}^-$:

$$ \frac{\Gamma \theta \rightarrow C\theta}{\Gamma \rightarrow C} \text{ subst.} $$

This rule is just a ‘macro’ for the following derivation:

$$ \frac{\rightarrow t = t}{t \rightarrow t} \text{ eqR} \frac{\lbrace \Gamma \theta \rightarrow C\theta \rbrace_{\theta}}{t = t, \Gamma \rightarrow C} \text{ eqL} \frac{\Gamma \rightarrow C}{mc} $$

where $t$ is some arbitrary term. The motivation behind the rule $\text{subst}$ is purely technical; it allows us to prove that a derivation transformation (i.e., substitutions of eigenvariables in derivations in Section 3.1) commutes with cut reduction (see Lemma 9). Since the rule $\text{subst}$ hides a simple form of cut, to prove cut-elimination of $\text{Linc}^-$, we have to show that $\text{subst}$, in addition to $mc$, is admissible. In the following, $\epsilon$ denotes the identity substitution, i.e., $\epsilon(x) = x$ for every variable $x$.

**Lemma 2 (\text{subst}-elimination).** For every $\Gamma$ and $C$, if the sequent $\Gamma \rightarrow C$ is (cut-free) derivable in $\text{Linc}^-$ with $\text{subst}$ then it is (cut-free) derivable in $\text{Linc}^-$ without $\text{subst}$.

**Proof.** Given a derivation $\Pi$ of $\Gamma \rightarrow C$ with occurrences of $\text{subst}$, obtain a $\text{subst}$-free derivation by simply replacing any subderivation in $\Pi$ of the form:

$$ \frac{\lbrace \Pi \theta \rightarrow B\theta \rbrace_{\theta}}{\Delta \rightarrow B} \text{ subst} $$

with its premise $\Pi'$.

Following [23], we define a measure which corresponds to the height of a derivation:

**Definition 4.** Given a derivation $\Pi$ with premise derivations $\lbrace \Pi_i \rbrace_{i \in I}$, for some index set $I$, the measure $\text{ht}(\Pi)$ is the least upper bound $\text{lub}(\lbrace \text{ht}(\Pi_i) \rbrace_{i \in I}) + 1$.

Note that given the possible infinite branching of $\text{eqL}$ rule, these measures can in general be (countable) ordinals. Therefore proofs and definitions on those measures require transfinite induction and recursion. However, in most of the proofs to follow, we do case analysis on the last rule of a derivation. In such a situation, the inductive cases for both successor and limit ordinals are basically covered by the case analysis on the inference figures involved, and we shall not make explicit use of transfinite principles.

With respect to the use of eigenvariables and parameters in a derivation, there may be occurrences of the formers that are not free in the end sequent. We refer to these variables and parameters as the internal variables and parameters, respectively. We view the choices of those variables and parameters as arbitrary and therefore identify derivations which differ on the choice of internal variables and parameters. In other terms, we quotient derivations modulo injective renaming of internal eigenvariables and parameters.

### 3.1. Instantiating eigenvariables

The following definition extends eigenvariable substitutions to apply to derivations. Since we identify derivations that differ only in the choice of internal eigenvariables, we will assume that such variables are chosen to be distinct from the variables in the domain of the substitution and from the free variables of the range of the substitution. Thus applying a substitution to a derivation will only affect the variables free in the end-sequent.

**Definition 5.** If $\Pi$ is a derivation of $\Gamma \rightarrow C$ and $\theta$ is a substitution, then we define the derivation $\Pi \theta$ of $\Gamma \theta \rightarrow C\theta$ as follows:
1. Suppose \( \Pi \) ends with the eq\( \mathcal{L} \) rule

\[
\frac{\Gamma \rho \to C \rho}{s \to t, \Gamma \to C} \quad \text{eq}\( \mathcal{L} \)
\]

where each \( \rho \) satisfies \( s\rho = \beta\eta t\rho \). Observe that any unifier for the pair \( (s\theta, t\theta) \) can be transformed to another unifier for \( (s, t) \), by composing the unifier with \( \theta \). Thus \( \Pi \theta \) is

\[
\frac{\Gamma' \theta \rho' \to C \theta \rho'}{s\theta = t\theta, \Gamma' \to C} \quad \text{eq}\( \mathcal{L} \),
\]

where \( s\theta\rho' = \beta\eta t\theta\rho' \).

2. If \( \Pi \) ends with \text{subst} with premise derivations \( \{\Pi^\rho\}_\rho \), then \( \Pi \theta \) also ends with the same rule and has premise derivations \( \{\Pi^{\theta\rho'}\}_\rho' \).

3. If \( \Pi \) ends with any other rule and has premise derivations \( \Pi_1, \ldots, \Pi_n \), then \( \Pi \theta \) also ends with the same rule and has premise derivations \( \Pi_1 \theta, \ldots, \Pi_n \theta \).

Among the premises of the inference rules of \( \text{Lin}^- \) (with the exception of CI\( \mathcal{R} \)), certain premises share the same right-hand side formula with the sequent in the conclusion. We refer to such premises as major premises.

**Definition 6.** Given an inference rule \( R \) with one or more premise sequents, we define its major premise sequents as follows.

1. If \( R \) is either \( \supset \mathcal{L} \), mc or \( \mathcal{L} \), then its rightmost premise is the major premise.
2. If \( R \) is CI\( \mathcal{R} \) then its left premise is the major premise.
3. Otherwise, all the premises of \( R \) are major premises.

A minor premise of a rule \( R \) is a premise of \( R \) which is not a major premise. The definition extends to derivations by replacing premise sequents with premise derivations.

The proofs of the following two lemma are straightforward from Definition 5 and induction on the height of derivations.

**Lemma 3.** For any substitution \( \theta \) and derivation \( \Pi \) of \( \Gamma \to C \), \( \Pi \theta \) is a derivation of \( \Gamma \theta \to C\theta \).

**Lemma 4.** For any derivation \( \Pi \) and substitution \( \theta \), \( \text{ht}(\Pi) \geq \text{ht}(\Pi \theta) \).

**Lemma 5.** For any derivation \( \Pi \) and substitutions \( \theta \) and \( \rho \), the derivations \( (\Pi \theta) \rho \) and \( \Pi(\theta \circ \rho) \) are the same derivation.

### 3.2. Instantiating parameters

**Definition 7.** A parameter substitution \( \Theta \) is a partial map from parameters to pairs of proofs and closed terms such that whenever

\[ \Theta(Xp) = (\Pi_S, S) \]

then \( S \) has the same type as \( p \) and either one of the following holds:

- \( p \bar{x} \vdash B p \bar{x} \), for some \( B \) and \( \bar{x} \), and \( \Pi_S \) is a derivation of \( BS \bar{x} \to S \bar{x} \), or
- \( p \bar{x} \vdash B p \bar{x} \), for some \( B \) and \( \bar{x} \), and \( \Pi_S \) is a derivation of \( S \bar{x} \to BS \bar{x} \).

The support of \( \Theta \) is the set

\[ \text{supp}(\Theta) = \{Xp \mid \Theta(Xp) \text{ is defined}\} \]

We consider only parameter substitutions with finite support.

We say that \( Xp \) is fresh for \( \Theta \), written \( Xp \# \Theta \), if for each \( Y^q \in \text{supp}(\Theta) \), \( Xp \neq Y^q \) and \( Xp \) does not occur in \( S \) whenever \( \Theta(Y^q) = (\Pi_S, S) \).
We shall often enumerate a parameter substitution using a similar notation to (eigenvariables) substitution, e.g.,

\[ [(\Pi_1, S_1)/X^{p_1}, \ldots, (\Pi_n, S_n)/X^{p_n}] \]

denotes a parameter substitution \( \Theta \) with support \( \{X^{p_1}, \ldots, X^{p_n}\} \) and \( \Theta(X^{p_i}) = (\Pi_i, S_i) \).

Given a formula \( C \) and a parameter substitution \( \Theta \) as above, we write \( C\Theta \) to denote the formula

\[ C[S_1/X^{p_1}, \ldots, S_n/X^{p_n}] \]

**Definition 8.** Let \( \Pi \) be a derivation of \( \Gamma \rightarrow C \) and let \( \Theta \) be a parameter substitution. Define the derivation \( \Pi\Theta \) of \( \Gamma \Theta \rightarrow \Theta \) by induction on the height of \( \Pi \) as follows:

- Suppose \( C = X^p \vec{t} \) for some \( X^p \) such that \( \Theta(X^p) = (\Pi_S, S) \) and \( \Pi \) ends with \( IR_p \), as shown below left.

\[
\begin{array}{c}
\Pi' \\
\Gamma \rightarrow B X^p \vec{t} \\
\Gamma \rightarrow X^p \vec{t}
\end{array}
\]

\[
\begin{array}{c}
\Pi\Theta \\
\Gamma \Theta \rightarrow B S \vec{t} \\
\Gamma \Theta \rightarrow S \vec{t}
\end{array}
\]

\[
\begin{array}{c}
\Pi'[\vec{t}/\vec{x}] \\
\Gamma'[\vec{t}/\vec{x}] \rightarrow C \Theta \\
\Gamma'[\vec{t}/\vec{x}] \rightarrow C \Theta
\end{array}
\]

- Similarly, suppose \( \Pi \) ends with \( CL\vec{t}_p \) on \( X^p \vec{t} \) and \( X^p \in \text{supp}(\Theta) \):

\[
\begin{array}{c}
\Pi' \\
B X^p \vec{t}, \Gamma' \rightarrow C \Theta
\end{array}
\]

where \( p \vec{x} \overset{\nu}{=} B p \vec{x} \) and \( \Theta(X^p) = (\Pi_S, S) \). Then \( \Pi\Theta \) is

\[
\begin{array}{c}
S \vec{t} \rightarrow S \vec{t} \overset{\text{init}}{\rightarrow} S \vec{t} \rightarrow B S \vec{t} \\
S \vec{t} \rightarrow B S \vec{t} \overset{mc}{\rightarrow} B S \vec{t}, \Gamma' \Theta \rightarrow C \Theta \rightarrow C \Theta
\end{array}
\]

- In all other cases, suppose \( \Pi \) ends with a rule \( R \) with premise derivations \( \{\Pi_i\}_{i \in I} \) for some index set \( I \). Since we identify derivations up to renaming of internal parameters, we assume without loss of generality that the internal eigenvariables in the premises of \( R \) (if any) do not appear in \( \Theta \). Then \( \Pi\Theta \) ends with the same rule, with premise derivations \( \{\Pi_i\Theta\}_{i \in I} \).

**Remark 3.** Notice that the definition of application of parameter substitution in derivations in Definition 8 is asymmetric in the treatment of inductive and co-inductive parameters, i.e., in the cases where \( \Pi \) ends with \( IR_p \) and \( CL\vec{t}_p \). In the latter case, the substituted derivation uses a seemingly unnecessary cut

\[
\begin{array}{c}
S \vec{t} \rightarrow S \vec{t} \overset{\text{init}}{\rightarrow} S \vec{t} \rightarrow B S \vec{t} \\
S \vec{t} \rightarrow B S \vec{t} \overset{mc}{\rightarrow} B S \vec{t}, \Gamma' \Theta \rightarrow C \Theta \rightarrow C \Theta
\end{array}
\]

The reason behind this is rather technical; in our main cut elimination proof, we need to establish that \( \Pi_S[\vec{t}/\vec{x}] \) is “reducible” (i.e., all the cuts in it can be eventually eliminated), given that the above cut is reducible. In a typical cut elimination procedure, say Gentzen’s proof for LK, one would have expected that the above cut reduces to \( \Pi_S[\vec{t}/\vec{x}] \), hence reducibility of \( \Pi_S \) would follow from reducibility of the above cut. However, according to our cut reduction rules (see Section 4.1), the above cut does not necessarily reduce to \( \Pi_S[\vec{t}/\vec{x}] \). However, if the instance of \( \text{init} \) appears instead on the right premise of the cut, e.g., as in

\[
\begin{array}{c}
\Pi_S[\vec{t}/\vec{x}] \\
B S \vec{t} \rightarrow S \vec{t} \overset{\text{init}}{\rightarrow} S \vec{t} \rightarrow S \vec{t} \\
B S \vec{t} \rightarrow S \vec{t} \overset{mc}{\rightarrow} S \vec{t}
\end{array}
\]
the cut elimination procedure does reduce this to \( \Pi_S[\overrightarrow{t}/\overrightarrow{x}] \), so it is not necessary to introduce explicitly this cut instance in the case involving inductive parameters. It is possible to define a symmetric notion of parameter substitution, but that would require different cut reduction rules than the ones we proposed in this paper. Another possibility would be to push the asymmetry to the definition of reducibility (see Section 4). We have explored these alternative options, but for the purpose of proving cut elimination, we found that the current definition yields a simpler proof.\(^4\)

The following lemma states that the derivation \( \Pi\Theta \) is well-formed.

**Lemma 6.** Let \( \Theta \) be a parameter substitution and \( \Pi \) a derivation of \( \Gamma \rightarrow C \). Then \( \Pi\Theta \) is a derivation of \( \Gamma\Theta \rightarrow C\Theta \).

Note that since parameter substitutions replace parameters with closed terms, they commute with (eigen-variable) substitutions.

**Lemma 7.** For every derivation \( \Pi \), substitution \( \delta \), parameter substitution \( \Theta \), the derivation \( (\Pi\Theta)\delta \) is the same as the derivation \( (\Pi\delta)\Theta \).

In the following, we denote with \([\Theta,(\Pi_S,S)/X_p]\), where \( X_p \# \Theta \), a parameter substitution obtained by extending \( \Theta \) with the map \( X_p \mapsto (\Pi_S,S) \).

**Lemma 8.** Let \( \Pi \) be a derivation of \( \Gamma \rightarrow C \), \( \Theta \) a parameter substitution and \( X_p \) a parameter such that \( X_p \notin \text{supp}(\Theta) \) and \( X_p \) does not occur in \( \Gamma \rightarrow C \). Then \( \Pi[\Theta,(\Pi_S,S)/X_p] = \Pi\Theta \) for every \( \Pi_S \) and \( S \).

4. Cut elimination for \( Linc^- \)

The central result of our work is cut-elimination, from which consistency of the logic follows. Gentzen’s classic proof of cut-elimination for first-order logic uses an induction on the size of the cut formula. The cut-elimination procedure consists of a set of reduction rules that reduces a cut of a compound formula to cuts on its sub-formulae of smaller size. In the case of \( Linc^- \), the use of induction/co-induction complicates the reduction of cuts. Consider for example a cut involving the induction rules:

\[
\frac{\begin{array}{l}
\Delta \rightarrow B X_p \overrightarrow{t} \\
\Pi_1 \\
\Pi_2 \\
\Delta \rightarrow \overrightarrow{p} t, \Gamma \rightarrow C \\
\end{array}}{\Delta, \Gamma \rightarrow C} \quad \text{IR}
\]

\[
\frac{\begin{array}{l}
\Pi_B \\
\Pi \\
\Delta, \Gamma \rightarrow C \\
\end{array}}{\Delta \rightarrow B S \overrightarrow{y} \rightarrow S \overrightarrow{y} \overrightarrow{t}, \Gamma \rightarrow C} \quad \text{mc}
\]

There are at least two problems in reducing this cut. First, any permutation upwards of the cut will necessarily involve a cut with \( S \) that can be of larger size than \( p \), and hence a simple induction on the size of the cut formula will not work. Second, the invariant \( S \) does not appear in the conclusion of the left premise of the cut. The latter means that we need to transform the left premise so that its end sequent will agree with the right premise. Any such transformation will most likely be global, and hence simple induction on the height of derivations will not work either.

We shall use the reducibility technique to prove cut elimination. More specifically, we shall build on the notion of reducibility introduced by Martin-Löf to prove normalization of an intuitionistic logic with iterative inductive definition\(^{24}\). Martin-Löf’s proof has been adapted to sequent calculus by McDowell and Miller\(^{25}\), but in a restricted setting where only natural number induction is allowed. Since our logic involves arbitrary stratified inductive definitions, which also includes iterative inductive definitions, we shall need different, and more general, cut reductions. But the real difficulty in our case is in establishing cut elimination in the presence of co-inductive definitions, for which there is no known direct cut elimination proof (prior to our work\(^{33}\) on which this article is based on), at the best of our knowledge, as far as the sequent calculus is concerned.

\(^4\) But we conjecture that in the classical case a fully symmetric definition of parameter substitution and cut reduction would be needed. But this is outside the scope of the current paper.
4.1. Cut reduction

The main part of the reducibility technique is a definition of the family of reducible sets of derivations. In Martin-Löf’s theory of iterative inductive definition, this family of sets is defined inductively by the “type” of the derivations they contain, i.e., the formula in the right-hand side of the end sequent in a derivation. Extending this definition of reducibility to Linc− is not obvious. In particular, in establishing the reducibility of a derivation of type \( p \overline{I} \) ending with a CIR rule one must first establish the reducibility of its premise derivations, which may have larger types, since \( S \overline{I} \) could be any formula. Therefore a simple inductive definition based on types of derivations would not be well-founded.

The key to properly “stratify” the definition of reducibility is to consider reducibility under parameter substitutions. This notion of reducibility, called \textit{parametric reducibility}, was originally developed by Girard to prove strong normalisation of System F, i.e., in the interpretation of universal types. As with strong normalisation of System F, (co-)inductive parameters are substituted with some “reducibility candidates”, which in our case are certain sets of derivations satisfying closure conditions similar to those for System F, but which additionally satisfy certain closure conditions related to (co-)inductive definitions.

The remainder of this section is structured as follows. In Section 4.1 we define a set of cut reduction rules that are used to elimination the applications of the cut rule. For the cases involving logical operators, the cut-reduction rules used to prove the cut-elimination for Linc− are the same as those of \( F\text{O} \Delta N \) \cite{22}. The crucial differences are, of course, in the reduction rules involving induction and co-induction rules, where we use the transformation described in Definition 7. We then proceed to define two notions essential to our cut elimination proof: \textit{normalizability} (Section 4.2) and \textit{parametric reducibility} (Section 4.3). These can be seen as counterparts for Martin-Löf’s notions of normalizability and \textit{computability} \cite{24}, respectively. Normalizability of a derivation implies that all the cuts in it can be eventually eliminated (via the cut reduction rules defined earlier). Reducibility is a stronger notion, in that it implies normalizability. The main part of the cut elimination proof is presented in Section 4.4 where we show that every derivation is reducible, hence it can be turned into a cut-free derivation.

4.1. Cut reduction

We now define a reduction relation on derivations ending with \( mc \). This reduction relation is an extension of the similar cut reduction relation used in McDowell and Miller’s cut elimination proof \cite{22}. In particular, the reduction rules involving introduction rules for logical connectives are the same. The main differences are, of course, in the reduction rules involving induction and co-induction rules. There is also slight difference in one reduction rule involving equality, which in our case utilises the derived rule \( \text{subst} \). Therefore in the following definition, we shall highlight only those reductions that involve (co-)induction and equality rules. The complete list of reduction rules can be found in Appendix A.

To ease presentation, we shall use the following notations to denote certain forms of derivations. The derivation

\[
\Delta_1 \to B_1 \quad \cdots \quad \Delta_n \to B_n \quad \Gamma \to C \quad mc
\]

is abbreviated as \( mc(\Pi_1, \ldots, \Pi_n, \Pi) \). Whenever we write \( mc(\Pi_1, \ldots, \Pi_n, \Pi) \) we assume implicitly that the derivation is well-formed, i.e., \( \Pi \) is a derivation ending with some sequent \( \Gamma \to C \) and the right-hand side of the end sequent of each \( \Pi_i \) is a formula \( F \in \Gamma \). Similarly, we abbreviated as \( \text{Id}_B \) the derivation

\[
\overline{B} \to \overline{B} \quad \text{init}
\]

and \( \text{subst}(\{\Pi^\theta\}_{\theta}) \) denotes a derivation ending with the rule \( \text{subst} \) with premise derivations \( \{\Pi^\theta\}_{\theta} \).

**Definition 9.** We define a \textit{reduction} relation between derivations. The redex is always a derivation \( \Xi \) ending with the multicut rule

\[
\Delta_1 \to B_1 \quad \cdots \quad \Delta_n \to B_n \quad B_1, \ldots, B_n, \Gamma \to C \quad mc
\]

We refer to the formulas \( B_1, \ldots, B_n \) produced by the \( mc \) as \textit{cut formulas}.

If \( n = 0 \), \( \Xi \) reduces to the premise derivation \( \Pi \). For \( n > 0 \) we specify the reduction relation based on the last rule of the premise derivations. If the rightmost premise derivation \( \Pi \) ends with a left rule acting
on a cut formula $B_i$, then the last rule of $\Pi_i$ and the last rule of $\Pi$ together determine the reduction rules that apply. Following McDowell and Miller [25], we classify these rules according to the following criteria: we call the rule an essential case when $\Pi_i$ ends with a right rule; if it ends with a left rule or $\text{subst}$, it is a left-commutative case; if $\Pi_i$ ends with the init rule, then we have an axiom case; a multicut case arises when it ends with the $\text{mc}$ rule. When $\Pi$ does not end with a left rule acting on a cut formula, then its last rule is alone sufficient to determine the reduction rules that apply. If $\Pi$ ends with $\text{subst}$ or a rule acting on a formula other than a cut formula, then we call this a right-commutative case. A structural case results when $\Pi$ ends with a contraction or weakening on a cut formula. If $\Pi$ ends with the init rule, this is also an axiom case; similarly a multicut case arises if $\Pi$ ends in the $\text{mc}$ rule. For simplicity of presentation, we always show $i = 1$.

We show here the cases involving (co-)induction rules.

**Essential cases.**

- **$\text{eqL}/\text{eqR}$** Suppose $\Pi_1$ and $\Pi$ are

$$
\Delta_1 \rightarrow s = t \quad \text{eqR}
$$

$$
\frac{B_2, \ldots, B_n, \Gamma \rightarrow C}{s = t, B_2, \ldots, B_n, \Gamma \rightarrow C} \quad \text{eqL}
$$

Note that in this case, $\rho$ in $\Pi$ ranges over all substitution, as any substitution is a unifier of $s$ and $t$. Let $\Xi$ be the derivation $\text{mc}(\Pi_2, \ldots, \Pi_n, \text{subst}((\Pi')_{\rho})$. In this case, $\Xi$ reduces to

$$
\frac{\Delta_2, \ldots, \Delta_n, \Gamma \rightarrow C}{\Delta_1, \Delta_2, \ldots, \Delta_n, \Gamma \rightarrow C} \quad wL
$$

We use the double horizontal lines to indicate that the relevant inference rule (in this case, $wL$) may need to be applied zero or more times.

- **$\text{1R}/\text{1L}$** Suppose $\Pi_1$ and $\Pi$ are, respectively,

$$
\begin{align*}
\Delta_1 \rightarrow D X^p \overrightarrow{t} & \quad \text{1R} \\
\Delta_1 \rightarrow p \overrightarrow{t} & \quad \text{1L}
\end{align*}
$$

$$
\begin{align*}
\Pi'_1 \rightarrow S \overrightarrow{t} & \quad \text{C1R} \\
\Pi_1' \rightarrow S \overrightarrow{y} & \quad \text{C1L}
\end{align*}
$$

where $p \overrightarrow{t} \equiv D p \overrightarrow{x}$ and $X^p$ is a new parameter. Then $\Xi$ reduces to

$$
\text{mc}(mc(\Pi'_1[p(\Pi_1, S)/X^p], \Pi_1[\overrightarrow{t}/\overrightarrow{y}]), \Pi_2, \ldots, \Pi_n, \Pi').
$$

- **$\text{C1R}/\text{C1L}$** Suppose $\Pi_1$ and $\Pi$ are

$$
\begin{align*}
\Delta_1 \rightarrow S \overrightarrow{t} & \quad \text{C1R} \\
\Delta_1 \rightarrow p \overrightarrow{t} & \quad \text{C1L}
\end{align*}
$$

where $p \overrightarrow{t} \equiv D p \overrightarrow{x}$ and $X^p$ is a new parameter. Then $\Xi$ reduces to

$$
\text{mc}(mc(\Pi'_1[\Pi_1, S[\overrightarrow{t}/\overrightarrow{y}]), \Pi_2, \ldots, \Pi_n, \Pi'[(\Pi_1, S)/X^p]).
$$

**Left-commutative cases.** In the following, we suppose that $\Pi$ ends with a left rule, other than $\{cL, wL\}$, acting on $B_1$.

- **$\text{1C}/\circ \text{L}$** Suppose $\Pi_1$ is

$$
\begin{align*}
\Pi_1 \rightarrow S \overrightarrow{y} & \quad \text{1C} \\
\Pi'_1 \rightarrow p \overrightarrow{t}, \Delta' \rightarrow B_1 & \quad \text{1L}
\end{align*}
$$

where $p \overrightarrow{t} \equiv D p \overrightarrow{x}$ and $X^p$ is a new parameter. Then $\Xi$ reduces to

$$
\text{mc}(mc(\Pi'_1[p(\Pi_1, S)/X^p], \Pi_1[\overrightarrow{t}/\overrightarrow{y}]), \Pi_2, \ldots, \Pi_n, \Pi'[(\Pi_1, S)/X^p]).
$$
where \( p \overline{x} \equiv D p \overline{x} \). Let \( \Xi_1 = mc(\Pi'_1, \Pi_2, \ldots, \Pi_n, \Pi) \). Then \( \Xi \) reduces to

\[
\frac{\Pi_S}{D S \overline{y} \rightarrow S \overline{y}} S \overline{t}, \Delta'_1, \ldots, \Delta_n, \Gamma \rightarrow C \quad \text{I}\mathcal{L}
\]

\[
\frac{\Xi_1}{p \overline{t}, \Delta'_1, \ldots, \Delta_n \rightarrow C} \quad \text{I}\mathcal{L}
\]

**Right-commutative cases:**

\(-/\text{I}\mathcal{L}\) Suppose \( \Pi \) is

\[
\frac{\Pi_S}{D S \overline{y} \rightarrow S \overline{y}} B_1, \ldots, B_n, S \overline{t}, \Gamma' \rightarrow C \quad \text{I}\mathcal{L}
\]

\[
\frac{B_1, \ldots, B_n, p \overline{t}, \Gamma' \rightarrow C}{\Delta'_1, \ldots, \Delta_n, p \overline{t}, \Gamma' \rightarrow C} \quad \text{I}\mathcal{L}
\]

where \( p \overline{x} \equiv D p \overline{x} \). Let \( \Xi_1 = mc(\Pi_1, \ldots, \Pi_n, \Pi') \). Then \( \Xi \) reduces to

\[
\frac{\Pi_S}{D S \overline{y} \rightarrow S \overline{y}} \Delta_1, \ldots, \Delta_n, S \overline{t}, \Gamma' \rightarrow C \quad \text{I}\mathcal{L}
\]

\[
\frac{\Xi_1}{\Delta'_1, \ldots, \Delta_n \rightarrow C} \quad \text{I}\mathcal{L}
\]

\(-/\text{CIR}\) Suppose \( \Pi \) is

\[
\frac{\Pi'}{B_1, \ldots, B_n, \Gamma \rightarrow S \overline{t}} \quad \text{CIR}
\]

\[
\frac{S \overline{y} \rightarrow D S \overline{y}}{p \overline{t}} \quad \text{CIR}
\]

\[
\frac{B_1, \ldots, B_n, p \overline{t}, \Gamma' \rightarrow C}{\Delta'_1, \ldots, \Delta_n, p \overline{t}, \Gamma' \rightarrow C} \quad \text{CIR}
\]

where \( p \overline{x} \equiv D p \overline{x} \). Let \( \Xi_1 = mc(\Pi_1, \ldots, \Pi_n, \Pi') \). Then \( \Xi \) reduces to

\[
\frac{\Xi_1}{\Delta_1, \ldots, \Delta_n, \Gamma \rightarrow S \overline{t}} \quad \text{CIR}
\]

\[
\frac{S \overline{y} \rightarrow D S \overline{y}}{p \overline{t}} \quad \text{CIR}
\]

\[
\frac{\Delta'_1, \ldots, \Delta_n \rightarrow C}{\Delta_1, \ldots, \Delta_n \rightarrow C} \quad \text{CIR}
\]

It is clear from an inspection of the inference rules in Figure 1 and the definition of cut reduction (see Appendix A) that every derivation ending with a multicut has a reduct. Note that since the left-hand side of a sequent is a multiset, the same formula may occur more than once in the multiset. In the cut reduction rules, we should view these occurrences as distinct so that no ambiguity arises as to which occurrence of a formula is subject to the \( \text{mc} \) rule.

The following lemma shows that the reduction relation is preserved by eigenvariable substitution. The proof is given in Appendix B.

**Lemma 9.** Let \( \Pi \) be a derivation ending with a \( \text{mc} \) and let \( \theta \) be a substitution. If \( \Pi \theta \) reduces to \( \Xi \) then there exists a derivation \( \Pi' \) such that \( \Xi = \Pi' \theta \) and \( \Pi \) reduces to \( \Pi' \).

**4.2. Normalizability**

**Definition 10.** We define the set of normalizable derivations to be the smallest set that satisfies the following conditions:

1. If a derivation \( \Pi \) ends with a multicut, then it is normalizable if every reduct of \( \Pi \) is normalizable.
2. If a derivation \( \Pi \) ends with any rule other than a multicut, then it is normalizable if the premise derivations are normalizable.

The set of all normalizable derivations is denoted by \( \text{NM} \).

Each clause in the definition of normalizability asserts that a derivation is normalizable if certain (possibly infinitely many) other derivations are normalizable. We call the latter the predecessors of the former. Thus a derivation is normalizable if the tree of its successive predecessors is well-founded. We refer to this well-founded tree as its normalization. Since a normalization is well-founded, it has an associated induction principle: for any property \( P \) of derivations, if for every derivation \( \Pi \) in the normalization, \( P \) holds for every predecessor of \( \Pi \) implies that \( P \) holds for \( \Pi \), then \( P \) holds for every derivation in the normalization. We shall define explicitly a measure on a normalizable derivation based on its normalization tree.
**Definition 11 (Normalization Degree).** Let \( \Pi \) be a normalizable derivation. The *normalization degree* of \( \Pi \), denoted by \( \text{nd}(\Pi) \), is defined by induction on the normalization of \( \Pi \) as follows:

\[
\text{nd}(\Pi) = 1 + \text{ lub}\{\text{nd}(\Pi') \mid \text{\Pi' is a predecessor of } \Pi\}
\]

The normalization degree of \( \Pi \) is basically the height of its normalization tree. Note that \( \text{nd}(\Pi) \) can be an ordinal in general, due to the possibly infinite-branching rule eq\( \mathcal{L} \).

**Lemma 10.** If there is a normalizable derivation of a sequent, then there is a cut-free derivation of the sequent.

**Proof.** Similarly to [25]. \( \Box \)

In the proof of the main lemma for cut elimination (Lemma 21) we shall use induction on the normalization degree, instead of using directly the normalization ordering. The reason is that in some inductive cases in the proof, we need to compare a (normalizable) derivation with its instances, but the normalization ordering does not necessarily relate the two, e.g., \( \Pi \) and \( \Pi\theta \) may not be related by the normalization ordering, although their normalization degrees are (see Lemma 12). Later, we shall define a stronger ordering called reducibility, which implies normalizability. In the cut elimination proof for FO\( \Delta \mathcal{IN} \) [25], in one of the inductive cases, an implicit reducibility ordering is assumed to hold between derivation \( \Pi \) and its instance \( \Pi\theta \). As the reducibility ordering in their setting is a subset of the normalizability ordering, this assumption may not hold in all cases, and as a consequence there is a gap in the proof in [25].

The next lemma states that normalization is closed under substitutions.

**Lemma 11.** If \( \Pi \) is a normalizable derivation, then for any substitution \( \theta \), \( \Pi\theta \) is normalizable.

**Proof.** By induction on \( \text{nd}(\Pi) \).

1. If \( \Pi \) ends with a multicut, then \( \Pi\theta \) also ends with a multicut. By Lemma 9 every reduct of \( \Pi\theta \) corresponds to a reduct of \( \Pi \), therefore by induction hypothesis every reduct of \( \Pi\theta \) is normalizable, and hence \( \Pi\theta \) is normalizable.

2. Suppose \( \Pi \) ends with a rule other than multicut and has premise derivations \( \{\Pi_i\} \). By Definition 5, each premise derivation in \( \Pi\theta \) is either \( \Pi_i \) or \( \Pi_i\theta \). Since \( \Pi \) is normalizable, \( \Pi_i \) is normalizable, and so by the induction hypothesis \( \Pi_i\theta \) is also normalizable. Thus \( \Pi\theta \) is normalizable. \( \Box \)

The normalization degree is non-increasing under eigenvariable substitution.

**Lemma 12.** Let \( \Pi \) be a normalizable derivation. Then \( \text{nd}(\Pi) \geq \text{nd}(\Pi\theta) \) for every substitution \( \theta \).

**Proof.** By induction on \( \text{nd}(\Pi) \) using Definition 5 and Lemma 9. Note that \( \text{nd}(\Pi\theta) \) can be smaller than \( \text{nd}(\Pi) \) because substitution may reduces the number of premises in eq\( \mathcal{L} \), i.e., if \( \Pi \) ends with an eq\( \mathcal{L} \) acting on, say \( x = y \) (which are unifiable), and \( \theta \) is a substitution that maps \( x \) and \( y \) to distinct constants then \( \Pi\theta \) ends with eq\( \mathcal{L} \) with empty premise. \( \Box \)

### 4.3. Parametric reducibility

In the following, we shall use the term “type” in two different settings: in categorizing terms and in categorizing derivations. To avoid confusion, we shall refer to the types of terms as syntactic types, and the term “type” is reserved for types of derivations.

Our notion of a type of a set of derivations may abstract from particular first-order terms in a formula. This is because our definition of reducibility (candidates) will have to be closed under eigenvariable substitutions, which is in turn imposed by the fact that our proof rules allow instantiation of eigenvariables in the derivations (i.e., the eq\( \mathcal{L} \) and the subst rules).

---

5 This gap was fixed in [52] by strengthening the main lemma for cut elimination. Recently, Andrew Gacek and Gopalan Nadathur proposed another fix by assigning an explicit ordinal to each reducible derivation, and using the ordering on ordinals to replace the reducibility ordering in the lemma. A discussion of these fixes can be found in the errata page of the paper [25]: [http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/tcs00.errata.html](http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/tcs00.errata.html). We essentially follow Gacek and Nadathur’s approach here, although we assign ordinals to normalizable derivations rather than to reducible derivations.
Definition 12 (Types of derivations). We say that a derivation \( \Pi \) has type \( C \) if the end sequent of \( \Pi \) is of the form \( \Gamma \to \to C \) for some \( \Gamma \). Let \( F \) be a term with syntactic type \( \alpha_1 \to \cdots \to \alpha_n \to o \), where each \( \alpha_i \) is a syntactic efo-type. A set of derivations \( S \) is said to be of type \( F \) if every derivation in \( S \) has type \( F \cdot u_1 \cdots u_n \) for some terms \( u_1, \ldots, u_n \). Given a list of terms \( \bar{u} = u_1 : \alpha_1, \ldots, u_n : \alpha_n \) and a set of derivations \( S \) of type \( F : \alpha_1 \to \cdots \to \alpha_n \to o \), we denote with \( S \bar{u} \) the set

\[
S \bar{u} = \{ \Pi \in S \mid \Pi \text{ has type } F \bar{u} \}
\]

Definition 13 (Reducibility candidate). Let \( F \) be a closed term having the syntactic type \( \alpha_1 \to \cdots \to \alpha_n \to o \). A set of derivations \( R \) of type \( F \) is said to be a reducibility candidate of type \( F \) if the following hold:

CR0 If \( \Pi \in R \) then \( \Pi \theta \in R \), for every \( \theta \).
CR1 If \( \Pi \in R \) then \( \Pi \) is normalizable.
CR2 If \( \Pi \in R \) and \( \Pi \) reduces to \( \Pi' \) then \( \Pi' \in R \).
CR3 If \( \Pi \) ends with \( mc \) and all its reducts are in \( R \), then \( \Pi \in R \).
CR4 If \( \Pi \) ends with \( init \), then \( \Pi \in R \).
CR5 If \( \Pi \) ends with a left-rule or \( subst \), then all its minor premise derivations are normalizable, and all its major premise derivations are in \( R \), then \( \Pi \in R \).

We shall write \( R : F \) to denote a reducibility candidate \( R \) of type \( F \).

The conditions CR1 and CR2 are similar to the eponymous conditions in Girard’s definition of reducibility candidates in his strong normalisation proof for System F (see [18], Chapter 14). Girard’s CR3 is expanded in our definition to CR3, CR4 and CR5. These conditions deal with what Girard refers to as “neutral” proof term (or, in our setting, derivations). Neutrality corresponds to derivations ending in \( mc \), \( init \), \( subst \), or a left rule.

The condition CR0 is needed because our cut reduction rules involve substitution of eigenvariables in some cases (i.e., those that involve permutation of \( eq \mathcal{L} \) and \( subst \) in the left/right commutative cases), and consequently, the notion of reducibility (candidate) needs to be preserved under eigenvariable substitution.

Let \( S \) be a set of derivations of type \( B \) and let \( T \) be a set of derivations of type \( C \). Then \( S \Rightarrow T \) denotes the set of derivations such that \( \Pi \in S \Rightarrow \Pi' \in T \) if and only if \( \Pi \) ends with a sequent \( \Gamma \to \to C \) such that \( B \in \Gamma \) and for every \( \Xi \in S \), we have \( mc(\Xi, \Pi) \in T \).

Let \( S \) be a closed term. Define \( NM_S \) to be the set

\[
NM_S = \{ \Pi \mid \Pi \in NM \text{ and is of type } S \bar{u} \text{ for some } \bar{u} \}
\]

It can be shown that \( NM_S \) is a reducibility candidate of type \( S \).

Lemma 13. Let \( S \) be a term of syntactic type \( \alpha_1 \to \cdots \to \alpha_n \to o \). Then the set \( NM_S \) is a reducibility candidate of type \( S \).

Proof. CR0 follows from Lemma [11], CR1 follows from the definition of \( NM_S \), and the rest follow from Definition [10]. \( \square \)

Definition 14 (Candidate substitution). A candidate substitution \( \Omega \) is a partial map from parameters to triples of reducibility candidates, derivations and closed terms such that whenever \( \Omega(X^p) = (R, \Pi, S) \), we have

- \( S \) has the same syntactic type as \( p \),

\[\text{From now on, we shall assume that the } \alpha_i \text{ are always efo-types.}\]
• $\mathcal{R}$ is a reducibility candidate of type $S$, and
• either one of the following holds:
  - $p \bar{\vec{x}} \models B p \bar{\vec{x}}$ and $\Pi$ is a normalizable derivation of $B S \bar{\vec{y}} \rightarrow S \bar{\vec{y}}$, or
  - $p \bar{\vec{x}} \models B p \bar{\vec{x}}$ and $\Pi$ is a normalizable derivation of $S \bar{\vec{y}} \rightarrow B S \bar{\vec{y}}$.

We denote with $\text{supp}(\Omega)$ the support of $\Omega$, i.e., the set of parameters on which $\Omega$ is defined. Each candidate substitution $\Omega$ determines a unique parameter substitution $\Theta$, given by:

$$\Theta(X^p) = (\Pi, S) \text{ iff } \Omega(X^p) = (R, \Pi, S) \text{ for some } R.$$

We denote with $\text{Sub}(\Omega)$ the parameter substitution $\Theta$ obtained this way. We say that a parameter $X^p$ is fresh for $\Omega$, written $X^p \not\in \text{Sub}(\Omega)$.

Notation. Since every candidate substitution has a corresponding parameter substitution, we shall often treat a candidate substitution as a parameter substitution. In particular, we shall write $C_{\Omega}$ to denote $C(\text{Sub}(\Omega))$ and $\Pi_{\Omega}$ to denote $\Pi(\text{Sub}(\Omega))$.

We are now ready to define the notion of parametric reducibility. We follow a similar approach for $FO\lambda^{\Delta}IN[25]$, where families of reducibility sets are defined by the level of derivations, i.e. the size of the types of derivations. In defining a family (or families) of sets of derivations at level $k$, we assume that reducibility sets at level $j < k$ are already defined. The main difference with the notion of reducibility for $FO\lambda^{\Delta}IN$, aside from the use of parameters in the clause for (co)induction rules (which do not exist in $FO\lambda^{\Delta}IN$), is in the treatment of the induction rules.

**Definition 15 (Parametric reducibility).** Let $F_k$ be the set of all formula of size $k$, i.e. $\{F \mid |F| = k\}$.

The family of parametric reducibility sets $\text{RED}_C[\Omega]$, where $C$ is a formula and $\Omega$ is a candidate substitution, is defined by induction on the size of $C$ as follows. For each $k$,

$$\{\text{RED}_C[\Omega]\}_{C \in F_k}$$

is the smallest family of sets satisfying, for each $C \in F_k$:

**P1** Suppose $C = X^p \bar{\vec{u}}$ for some $\bar{\vec{u}}$ and some parameter $X^p$. If $X^p \in \text{supp}(\Omega)$ then $\text{RED}_C[\Omega] = R \bar{\vec{u}}$, where $\Omega(X^p) = (R, \Pi, S)$. Otherwise, $\text{RED}_C[\Omega] = \text{NM}_X^p \bar{\vec{u}}$.

Otherwise, $C \neq X^p \bar{\vec{u}}$, for any $\bar{\vec{u}}$ and $X^p$. Then a derivation $\Pi$ of type $C_{\Omega}$ is in $\text{RED}_C[\Omega]$ if it is normalizable and one of the following holds:

**P2** $\Pi$ ends with $mc$, and all its reducts are in $\text{RED}_C[\Omega]$.

**P3** $\Pi$ ends with $\supset R$, i.e., $C = B \supset D$ and $\Pi$ is of the form:

$$\begin{align*}
  \Gamma, B\Omega & \rightarrow D\Omega \\
  \Gamma & \rightarrow B\Omega \supset D\Omega \supset R
\end{align*}$$

and for every substitution $\rho$, $\Pi' \rho \in (\text{RED}_{B\rho}[\Omega] \Rightarrow \text{RED}_{D\rho}[\Omega])$.

**P4** $\Pi$ ends with $IR$, i.e.,

$$\begin{align*}
  \Gamma & \rightarrow B X^p \bar{\vec{t}} \\
  \Gamma & \rightarrow p \bar{\vec{t}} \quad IR, \text{ where } p \bar{\vec{x}} \models B p \bar{\vec{x}}
\end{align*}$$

without loss of generality, assume that $X^p \# \Omega$: for every reducibility candidate $(S : I)$, where $I$ is a closed term of the same syntactic type as $p$, for every normalizable derivation $\Pi_I$ of $B I \bar{\vec{y}} \rightarrow I \bar{\vec{y}}$, if for every $\bar{\vec{u}}$ the following holds:

$$\Pi_I[\bar{\vec{u}}/\bar{\vec{y}}] \in (\text{RED}_{B X^p \bar{\vec{u}}}[\Omega], (S, \Pi_I, I)/X^p) \Rightarrow S \bar{\vec{u}})$$

then

$$mc(\Pi'[\Pi_I, I]/X^p, \Pi_I[\bar{\vec{t}}/\bar{\vec{y}}]) \in S \bar{\vec{t}}$$
P5 Π ends with CIR, i.e.,

\[
\frac{\Pi' \quad \Pi_I}{\Gamma \xrightarrow{I \bar{u}/\bar{y}} B \bar{I} \bar{y} \xrightarrow{\Pi_I} \Pi} \text{CIR, where } p \bar{x} \equiv B p \bar{x}
\]

and there exist a parameter \(X^p\) such that \(X^p \# \Omega\) and a reducibility candidate \((S : I)\) such that \(\Pi' \in S\) and

\[
\Pi_I[\bar{u}/\bar{y}] \in (S \bar{u} \Rightarrow \text{RED}_{B X^p \bar{u}}[\Omega, (S, \Pi_I, I)/X^p]) \text{ for every } \bar{u}.
\]

P6 Π ends with any other rule and its major premise derivations are in the parametric reducibility sets of the appropriate types.

We shall write \(\text{RED}_C\), instead of \(\text{RED}_C[\Omega]\), when the \(\text{supp}(\Omega)\) of a candidate substitution is the empty set.

A derivation Π of type \(C\) is reducible if Π \(\in \text{RED}_C\).

Some comments and comparison with Girard’s definition of parametric reducibility for System F \[18\] are in order, although our technical setting is somewhat different from that of Girard:

- Condition P3 quantifies over \(\rho\). This is needed to show that reducibility is closed under substitution (see Lemma 15). A similar quantification is used in the definition of reducibility for \(F\alpha\delta_{\text{IN}}\) \[22\] for the same purpose. In the same clause, we also quantify over derivations in \(\text{RED}_{B\rho}[\Omega]\), but since \(B\rho\) has smaller size than \(B \supset D\), this quantification is legitimate and the definition is well-founded. Note also the similar quantification in P4 and P5, where the parametric reducibility set \(\text{RED}_{B\rho}[\Omega]\) is defined in terms of \(\text{RED}_{B X^p \bar{u}}[\Omega]\). By Lemma 1 \(p \bar{u} > [B X^p \bar{u}]\) so in both cases the set \(\text{RED}_{B X^p \bar{u}}[\Omega]\) is already defined by induction. It is clear by inspection of the clauses that the definition of parametric reducibility is well-founded.

- Clauses P2 and P6 are needed to show that the notion of parametric reducibility is closed under left-rules, \(id\) and \(mc\), i.e., condition CR3 – CR5. This is also a point where our definition of parametric reducibility diverges from a typical definition of reducibility in natural deduction (e.g., \[18\]), where closure under reduction for “neutral” terms is a derived property.

- P4 (and dually P5) can be intuitively explained in terms of the second-order encoding of inductive definitions. To simplify presentation, we restrict to the propositional case, so, P4 can be simplified as follows:

Suppose Π ends with IR, i.e.,

\[
\frac{\Pi' \quad \Pi_I}{\Gamma \xrightarrow{B X^p} \Pi} \text{IR, where } p \mu \equiv B p
\]

without loss of generality, assume that \(X^p \# \Omega\), for every reducibility candidate \((S : I)\), where \(I\) is a closed term of the same syntactic type as \(p\), for every normalizable derivation \(\Pi_I\) of \(B I \rightarrow I\), if \(\Pi_I \in (\text{RED}_{B X^p}[\Omega, (S, \Pi_I, I)/X^p] \Rightarrow S)\), then \(mc(\Pi'[\Pi_I, I)/X^p], \Pi_I) \in S\).

Note that in the propositional \(\text{Linc}^-\), the set

\[
\text{RED}_{B X^p}[\Omega, (S, \Pi_I, I)/X^p] \Rightarrow S
\]

is equivalent to \(\text{RED}_{B X^p \supset X^p}[\Omega, (S, \Pi_I, I)/X^p]\), i.e., a set of reducible derivations of type \(B I \supset I\). So, intuitively, \(\Pi'\) can be seen as a higher-order function that takes any function of type \(B I \supset I\) (i.e., the derivation \(\Pi_I\)), and turns it into a derivation of type \(I\) (i.e., the derivation \(mc(\Pi'[\Pi_I, I)/X^p], \Pi_I)\)), for all candidate \((S : I)\). This intuitive reading matches the second-order interpretation of \(p\), i.e., \(\forall I.(B I \supset I) \supset I\), where the universal quantification is interpreted as the universal type constructor and \(\supset\) is interpreted as the function type constructor in System F.
We shall now establish a list of properties of the parametric reducibility sets that will be used in the main cut elimination proof. The main property that we are after is one which shows that a certain set of derivations formed using a family of parametric reducibility sets actually forms a reducibility candidate. This will be important later in constructing a reducibility candidate which acts as a co-inductive “witness” in the main cut elimination proof. The proofs of the following lemmas are mostly routine and rather tedious; so we omit them here, but they can be found in Appendix B.

Lemma 14. If $\Pi \in \text{RED}_C[\Omega]$ then $\Pi$ is normalizable.

Since every $\Pi \in \text{RED}_C[\Omega]$ is normalizable, $\text{nd}(\Pi)$ is defined. This fact will be used implicitly in subsequent proofs, i.e., we shall do induction on $\text{nd}(\Pi)$ to prove properties of $\text{RED}_C[\Omega].$

Lemma 15. If $\Pi \in \text{RED}_C[\Omega]$ then for every substitution $\rho$, $\Pi \rho \in \text{RED}_{C\rho}[\Omega].$

Lemma 16. Let $\Omega = [\Omega', (R, \Pi_S, S) / X^p]$. Let $C$ be a formula such that $X^p \# C$. Then for every $\Pi, \Pi \in \text{RED}_C[\Omega]$ if and only if $\Pi \in \text{RED}_{C[S/X^p]}[\Omega].$

Lemma 17. Let $\Omega$ be a candidate substitution and $F$ be a closed term of syntactic type $\alpha_1 \to \cdots \to \alpha_n \to o$. Then the set

$$\mathcal{R} = \{ \Pi \mid \Pi \in \text{RED}_F[\bar{u}] for some \bar{u} \}$$

is a reducibility candidate of type $F_{\Omega}$.

Lemma 18. Let $\Omega$ be a candidate substitution and let $X^p$ be a parameter such that $X^p \# \Omega$. Let $S$ be a closed term of the same type as $p$ and let

$$\mathcal{R} = \{ \Pi \mid \Pi \in \text{RED}_{S\bar{u}}[\Omega] for some \bar{u} \}.$$

Suppose $[\Omega, (R, \Psi, S\bar{u}) / X^p]$ is a candidate substitution, for some $\Psi$. Then

$$\text{RED}_{C[S/X^p]}[\Omega] = \text{RED}_{C[S/X^p]}[\Omega].$$

4.4. Cut elimination

We shall now show that every derivation is reducible, hence every derivation can be normalized to a cut-free derivation. But in order to prove this, we need a slightly more general lemma, which states that every derivation is in $\text{RED}_C[\Omega]$ for a certain kind of candidate substitution $\Omega$. The precise definition is given below.

Definition 16 (Definitional closure). A candidate substitution $\Omega$ is definitively closed if for every $X^p \in \text{supp}(\Omega)$, if $\Omega(X^p) = (R, \Pi_S, S)$ then either one of the following holds:

- $p \bar{x} \equiv B p \bar{x}$, for some $B$ and for every $\bar{u}$ of the appropriate syntactic types:
  $$\Pi_S[\bar{u}/\bar{x}] \in \text{RED}_{B X^p \bar{u}}[\Omega] \Rightarrow \mathcal{R} \bar{u}.$$

- $p \bar{x} \equiv B p \bar{x}$, for some $B$ and for every $\bar{u}$ of the appropriate syntactic types:
  $$\Pi_S[\bar{u}/\bar{x}] \in \mathcal{R} \bar{u} \Rightarrow \text{RED}_{B X^p \bar{u}}[\Omega].$$

The next two lemmas show that definitively closed substitutions can be extended in a way that preserves definitional closure.

Lemma 19. Let $\Omega = [\Omega', (R, \Pi_S, S) / X^p]$ be a candidate substitution such that $p \bar{x} \equiv B p \bar{x}$, $\Omega'$ is definitively closed, and for every $\bar{u}$ of the same types as $\bar{x}$,

$$\Pi_S[\bar{u}/\bar{x}] \in \text{RED}_{B X^p \bar{u}}[\Omega] \Rightarrow \mathcal{R} \bar{u}.$$

Then $\Omega$ is definitively closed.
Proof. Let \( Y^q \in \text{supp}(\Omega) \). Suppose \( \Omega(Y^q) = (S, \Pi_I, I) \). We need to show that

\[
\Pi_I[\vec{t}/\vec{x}] \in \text{RED}_{B Y^q \vec{t}}[\Omega] \Rightarrow S \vec{t}
\]

for every \( \vec{t} \) of the same types as \( \vec{x} \). If \( Y^q = X^p \) then this follows from the assumption of the lemma. Otherwise, \( Y^q \in \text{supp}(\Omega') \), and by the definitional closure assumption on \( \Omega' \), we have

\[
\Pi_I[\vec{t}/\vec{x}] \in \text{RED}_{B Y^q \vec{t}}[\Omega'] \Rightarrow S \vec{t}
\]

for every \( \vec{t} \). Since \( X^p \#(B Y^q \vec{t}) \) (recall that definition clauses cannot contain occurrences of parameters), by Lemma 16 we have \( \text{RED}_{B Y^q \vec{t}}[\Omega'] = \text{RED}_{B Y^q \vec{t}}[\Omega] \), and therefore the result. \( \square \)

Lemma 20. Let \( \Omega = [\Omega', (\mathcal{R}, \Pi_S, S)/X^p] \) be a candidate substitution such that \( p \vec{x} \equiv B p \vec{x}, \Omega' \) is definitionally closed, and for every \( \vec{u} \) of the same types as \( \vec{x} \),

\[
\Pi_S[\vec{u}/\vec{x}] \in \mathcal{R} \vec{u} \Rightarrow \text{RED}_{B X^p \vec{u}}[\Omega]
\]

Then \( \Omega \) is definitionally closed.

Proof. Analogous to the proof of Lemma 19. \( \square \)

We are now ready to state the main lemma for cut elimination.

Lemma 21. Let \( \Omega \) be a definitionally closed candidate substitution. Let \( \Pi \) be a derivation of \( B_1, \ldots, B_n, \Gamma \rightarrow C \), and let

\[
\begin{align*}
\Delta_1 & \rightarrow B_1 \Omega, \\
\Pi_1 & \\
\vdots & \\
\Pi_n & \rightarrow B_n \Omega
\end{align*}
\]

where \( n \geq 0 \), be derivations in, respectively, \( \text{RED}_{B_1}[\Omega], \ldots, \text{RED}_{B_n}[\Omega] \). Then the derivation \( \Xi \)

\[
\begin{array}{ccc}
\Delta_1, \ldots, \Delta_n & \longrightarrow & C \Omega \\
\Pi_1 & \\
\vdots & \\
\Pi_n & \rightarrow B_n \Omega & B_1 \Omega, \ldots, B_n \Omega, \Gamma \Omega & \longrightarrow C \Omega
\end{array}
\]

\( \text{mc} \)

is in \( \text{RED}_C[\Omega] \).

Proof. The proof is by induction on

\[
\mathcal{M}(\Xi) = \langle \text{ht}(\Pi), \sum_{i=1}^{n} |B_i|, ND(\Xi) \rangle
\]

where \( ND(\Xi) \) is the multiset \( \{nd(\Pi_1), \ldots, nd(\Pi_n)\} \) of normalization degrees of \( \Pi_1 \) to \( \Pi_n \). Note that the measure \( \mathcal{M} \) can be well-ordered using the lexicographical ordering. We shall refer to this ordering as simply \( < \). Note also that \( \mathcal{M} \) is insensitive to the order in which \( \Pi_i \) is given, thus when we need to distinguish one of the \( \Pi_i \), we shall refer to it as \( \Pi_1 \) without loss of generality. The derivation \( \Xi \) is in \( \text{RED}_C[\Omega] \) if all its reducts are in \( \text{RED}_C[\Omega] \).

CASE I: \( n = 0 \). In this case, \( \Xi \) reduces to \( \Pi \Omega \), thus it is enough to show that that \( \Pi \Omega \in \text{RED}_C[\Omega] \). This is proved by case analysis on \( C \) and on the last rule of \( \Pi \).

I.1. Suppose \( \Xi = [Y^q \vec{t}, \Gamma] \rightarrow C \). If \( X^p \notin \text{supp}(\Omega) \) then we need only to show that \( \Pi \Omega \) is normalizable. This follows mostly straightforwardly from the induction hypothesis and Lemma 19. The only interesting case is when \( \Xi \) ends with \( \text{CL}_p \) on some \( Y^q \vec{u} \) such that \( Y^q \in \text{supp}(\Omega) \), i.e., \( \Pi \) takes the form

\[
\begin{array}{c}
DY^q \vec{u}, \Gamma \rightarrow C \\
Y^q \vec{u}, \Gamma \rightarrow C \end{array}
\]

\( \text{CL}_p \).
Suppose $\Omega(Y^\nu) = (\mathcal{R}, \Pi_S, S)$. Then $\Pi\Omega = mc(mc(Id_{\mathcal{S}}, \Pi_S[^u/x]), \Pi\Omega)$. By \textbf{CR4} we have that $Id_{\mathcal{S}} \in \mathcal{R}$, so by the definitional closure of $\Omega$ and \textbf{CR3}, we have $mc(Id_{\mathcal{S}}[\bar{u}/\bar{x}], \Pi\Omega) \in \text{RED}_{\mathcal{D} \in \mathcal{S}}[\Omega]$. Since $ht(\Pi') < ht(\Pi)$, and since $\Pi\Omega = mc(mc(Id_{\mathcal{S}}[\bar{u}/\bar{x}], \Pi\Omega)), \Pi\Omega)$, by the induction hypothesis, we have $\Pi\Omega \in \text{RED}_{\mathcal{C}_1'[\Omega]}$, and therefore, by Lemma 14, $\Pi\Omega$ is normalizable. Note that this case is actually independent of the form of $C$.

Otherwise, suppose $X^p \in \text{supp}(\Omega)$, and $\Omega(X^p) = (\mathcal{R}, \Pi_S, S)$. Then there are several cases to consider, based on the last rule of $\Pi$. In all cases, we need to show that $\Pi\Omega \in \mathcal{R}$. Note that since $\Pi\Omega$ is of type $S'\ell$, $\Pi\Omega \in \mathcal{R}$ implies that $\Pi\Omega \in \mathcal{R}$. So in the following in some cases we need to show only that $\Pi\Omega \in \mathcal{R}$.

- $\Pi$ ends with \textit{init}: then $\Pi\Omega$ also ends with \textit{init} and by \textbf{CR4}, $\Pi\Omega \in \mathcal{R}$.
- $\Pi$ ends with $mc$: This follows from the induction hypothesis and Lemma 14.
- $\Pi$ ends with $\text{CLC}_p$: Suppose $\Pi$ ends with $\text{CLC}_p$ acting on a formula $Y^q \bar{u}$. If $Y^q \notin \text{supp}(\Omega)$, then this follows immediately from the induction hypothesis and \textbf{CR5}. If $Y^q \in \text{supp}(\Omega)$, then we use the same arguments as shown above.
- $\Pi$ ends with \textit{subst} or a left-rule other than $\text{CLC}_p$: Suppose the premise derivations of the rule are
  $$\left\{ \Psi_i \rightarrow C_i \right\}_{i \in I}$$
for some index set $I$. Then $\Pi\Omega$ ends with the same left rule and has premise derivations $\{\Psi_i \Omega\}_{i \in I}$.

By the induction hypothesis, $\Psi_i \in \text{RED}_{\mathcal{C}_1'[\Omega]}$ for every $i \in I$ and by Lemma 14, each $\Psi_i$ is also normalizable. The latter implies that $\Pi\Omega$ is normalizable. Note that if $\Psi_i$ is a major premise derivation, then $C_i = X^p \bar{u}$ for some $\bar{u}$, and we have $\Psi_i \Omega \in \mathcal{R}$. Therefore, by \textbf{CR5}, we have that $\Pi\Omega \in \mathcal{R}$.

- Suppose $\Pi$ ends with $\text{IR}_p$:
  $$\frac{\Pi'}{\Gamma \rightarrow D X^p \bar{t}' \text{IR}_p}$$
  where $p \bar{x} \overline{\mu} = D p \bar{x}$. Then $\Pi\Omega = mc(\Pi\Omega, \Pi_S[^\bar{t}/\bar{x}])$. From the induction hypothesis, we have that $\Pi\Omega \in \text{RED}_{D X^p \bar{t}_1' \bar{t}_2}'[\Omega]$. This, together with the definitional closure of $\Omega$, implies that $\Pi\Omega$ is indeed in $\mathcal{R}$.

1.2. Suppose $C \neq X^p \bar{t}'$ for any parameter $X^p$ and any terms $\bar{t}$.

Most subcases follow easily from the induction hypothesis, Lemma 14 and Definition 15. The subcases where $\Pi$ ends with a left rule follow the same lines of arguments as in Case I.1 above. We show here the non-trivial subcases involving right-introduction rules:

-1.2.a. Suppose $\Pi$ ends with $\supset \mathcal{R}$, as shown below left. Then $\Pi\Omega$ is as shown below right.

  $$\begin{align*}
  &\Gamma \rightarrow \Pi' C_1 \supset C_2, \\
  &\Gamma \rightarrow C_1 \supset C_2, \\
  &\Gamma \rightarrow \Pi' \Omega, C_1 \Omega \rightarrow C_2 \Omega
  \end{align*}$$

To show $\Pi\Omega \in \text{RED}_{\mathcal{C}_1'[\Omega]}$, we need to show that $\Pi\Omega$ is normalizable and that

$$\Pi' \Omega \in \text{RED}_{\mathcal{C}_1'[\Omega]} \Rightarrow \text{RED}_{\mathcal{C}_2'[\Omega]}$$

(1)

for every $\theta$. Since $ht(\Pi') < ht(\Pi)$, by the induction hypothesis, $\Pi\Omega \in \text{RED}_{\mathcal{C}_2'[\Omega]}$. Normalizability of $\Pi\Omega$ then follows immediately from this and Lemma 14. It remains to show that Statement 1 holds:

Let $\Psi$ be a derivation in $\text{RED}_{\mathcal{C}_1'[\Omega]}$. Let $\Xi_1 = mc(\Psi, \Pi\Omega \theta)$. Note that since parameter substitution commutes with eigenvariable substitution, $\Pi'\theta \otimes \Pi\Omega \theta = \Pi'\theta \Pi\Omega \theta$. Since $ht(\Pi' \theta) \leq ht(\Pi') < ht(\Pi)$ (Lemma 14), by induction hypothesis, we have $\Xi_1 \in \text{RED}_{\mathcal{C}_2'[\Omega]}$. In other words, Statement 1 holds for arbitrary $\theta$, and therefore by Definition 15, $\Pi\Omega \in \text{RED}_{\mathcal{C}_1'[\Omega]}$.  

22
I.2.b. Suppose Π ends with 1R, as shown below left, where \( p \bar{x} \overset{\text{c}}{=} D p \bar{x} \). We can assume w.l.o.g. that \( X^p \not\in \Omega \).
Then \( \Pi \Omega \) is as shown below right.

\[
\begin{array}{c}
\Pi' \\
\end{array}
\quad
\begin{array}{c}
\Gamma \overset{\Pi'}{\rightarrow} D X^p \bar{t} \\
\Gamma \rightarrow p \bar{t}
\end{array}
\quad
\begin{array}{c}
\Pi' \Omega \\
\end{array}
\quad
\begin{array}{c}
\Gamma \overset{\Pi' \Omega}{\rightarrow} D X^p \bar{t} \\
\Gamma \rightarrow p \bar{t}
\end{array}
\quad
\begin{array}{c}
\text{1R} \\
\text{1R}
\end{array}
\]

To show that \( \Pi \Omega \in \text{RED}_{C}[\Omega] \), we need to show that \( \Pi \Omega \) is normalizable (as before this easily follows from the induction hypothesis and Lemma 14) and that

\[
mc((\Pi' \Omega) [(\Pi_S, S)/X^p], (\Pi_S [\bar{u}/\bar{x}]) \in \mathcal{R} \bar{t} 
\]  
(2)

for every candidate \((\mathcal{R} : S)\) and every \( \Pi_S \) that satisfies:

\[
\Pi_S [\bar{u}/\bar{x}] \in \text{RED}_{D X^p \bar{u}} [\Omega, (\mathcal{R}, \Pi_S, S)/X^p] \Rightarrow \mathcal{R} \bar{u} \text{ for every } \bar{u}
\]

Let \( \Omega' = [\Omega, (\mathcal{R}, \Pi_S, S)/X^p] \). Note that since \( X^p \not\in \Omega \), we have \( \Pi' \Omega [(\Pi_S, S)/X^p] = \Pi' \Omega' \). So Statement 2 above can be rewritten to

\[
mc(\Pi' \Omega', (\Pi_S [\bar{u}/\bar{x}]) \in \mathcal{R} \bar{t}
\]

By Lemma 19 we have that \( \Omega' \) is definitionally closed. Therefore we can apply the induction hypothesis to \( \Pi' \) and \( \Omega' \), obtaining \( \Pi' \Omega' \in \text{RED}_{D X^p \bar{t}} [\Omega] \). This, together with definitional closure of \( \Omega' \), immediately implies Statement 2 above, hence \( \Pi \Omega \) is indeed in \( \text{RED}_{C}[\Omega] \).

I.2.c. Suppose Π ends with CITR, as shown below left, where \( p \bar{y} \overset{\text{c}}{=} D p \bar{y} \). Let \( S' = S \Omega \). Then \( \Pi \Omega \) is as shown below right.

\[
\begin{array}{c}
\Pi' \\
\end{array}
\quad
\begin{array}{c}
\Gamma \overset{\Pi'}{\rightarrow} S \bar{t} \quad S \bar{x} \rightarrow D S \bar{x} \\
\Gamma \rightarrow p \bar{t}
\end{array}
\quad
\begin{array}{c}
\Pi' \Omega \\
\end{array}
\quad
\begin{array}{c}
\Gamma \overset{\Pi' \Omega}{\rightarrow} S' \bar{t} \quad S' \bar{x} \rightarrow D S' \bar{x} \\
\Gamma \rightarrow p \bar{t}
\end{array}
\quad
\begin{array}{c}
\text{CITR} \\
\text{CITR}
\end{array}
\]

Note that \( \Pi \Omega \) is normalizable, by the induction hypothesis and Lemma 12. To show that \( \Pi \Omega \in \text{RED}_{C}[\Omega] \) it remains to show that there exists a reducibility candidate \((\mathcal{R} : S')\) such that

(a) \( \Pi' \Omega \in \mathcal{R} \), and

(b) \( \Pi_S \Omega [\bar{u}/\bar{x}] \in \mathcal{R} \bar{u} \Rightarrow \text{RED}_{D X^p \bar{u}} [\Omega, (\mathcal{R}, \Pi_S \Omega, S')/X^p] \) for a new \( X^p \not\in \Omega \).

Let \( \mathcal{R} = \{ \Psi \mid \Psi \in \text{RED}_{S \bar{u}} [\Omega] \} \). By Lemma 17 \( \mathcal{R} \) is a reducibility candidate of type \( S' \). By the induction hypothesis, we have \( \Pi' \Omega \in \mathcal{R} \), so \( \mathcal{R} \) satisfies (a). Since substitution does not increase the height of derivations, we have that \( ht(\Pi_S [\bar{u}/\bar{x}]) \leq ht(\Pi_S) \), and therefore, by applying the induction hypothesis to \( \Pi_S [\bar{x}/\bar{u}] \), we have \( mc(\Psi, \Pi_S \Omega [\bar{u}/\bar{x}]) \in \text{RED}_{D S \bar{u}} [\Omega] \) for every \( \Psi \in \text{RED}_{S \bar{u}} [\Omega] \). In other words,

\[
\Pi_S \Omega [\bar{u}/\bar{x}] \in \text{RED}_{S \bar{u}} [\Omega] \Rightarrow \text{RED}_{D S \bar{u}} [\Omega]
\]

Notice that \( \text{RED}_{S \bar{u}} [\Omega] \) is exactly \( \mathcal{R} \bar{u} \). So the above statement can be rewritten to

\[
\Pi_S \Omega [\bar{u}/\bar{x}] \in \mathcal{R} \bar{u} \Rightarrow \text{RED}_{D S \bar{u}} [\Omega]
\]

By Lemma 18 \( \text{RED}_{D S \bar{u}} [\Omega] = \text{RED}_{D X^p \bar{u}} [\Omega, (\mathcal{R}, \Pi_S \Omega, S')/X^p] \), which means that \( \mathcal{R} \) indeed satisfies condition (b) above, and therefore \( \Pi \Omega \in \text{RED}_{C}[\Omega] \).

CASE II: \( n > 0 \). To show that \( \Xi \in \text{RED}_{C}[\Omega] \) in this case, we need to show that all its reducts are in \( \text{RED}_{C}[\Omega] \) and that \( \Xi \) is normalizable. The latter follows from the former by Lemma 14 and Definition 10, so in the following we need only to show the former.

Note that in this case, we do not need to distinguish cases based on whether \( C \) is headed by a parameter or not. To see why, suppose \( C = X^p \bar{t} \) for some parameter \( X^p \). If \( X^p \not\in \text{supp}(\Omega) \) then to show \( \Xi \in \text{RED}_{C}[\Omega] \) we need to show that it is normalizable, which means that we need to show that all its reducts are normalizable. But since all reducts of \( \Xi \) has the same type \( X^p \bar{t} \), showing their normalizability amounts to the same thing
as showing that they are in \( \text{RED}_C[\Omega] \). If \( X^p \in \text{supp}(\Omega) \), then to show \( \Xi \in \text{RED}_C[\Omega] \) we need to show that \( \Xi \in \mathcal{R} \). Then by \textbf{CR3}, it is enough to show that all reducts of \( \Xi \) are in \( \mathcal{R} \), which is the same as showing that all reducts of \( \Xi \) are in \( \text{RED}_C[\Omega] \).

Since the applicable reduction rules to \( \Xi \) are driven by the shape of \( \Pi\Omega \), and since \( \Pi\Omega \) is determined by \( \Pi \), we shall perform case analysis on \( \Pi \) in order to determine the possible reduction rules that apply to \( \Xi \), and show in each case that the reduct of \( \Xi \) is in the same parametric reducibility set. There are several main cases depending on whether \( \Pi \) ends with a rule acting on a cut formula \( B_i \) or not. Again, when we refer to \( B_i \), without loss of generality, we assume \( i = 1 \).

In the following, we say that an instance of CI\( L_p \) is \textit{trivial} if it applies to a formula \( Y^q \bar{u} \) for some \( \bar{u} \), but \( Y^q \notin \text{supp}(\Omega) \). Otherwise, we say that it is non-trivial.

### II.1

Suppose \( \Pi \) ends with a left rule, other than \( cL \), \( wL \) and a non-trivial CI\( L_p \), on \( B_1 \) and \( \Pi_1 \) ends with a right-introduction rule. There are several subcases depending on the logical rules that are applied to \( B_1 \). We show here the non-trivial cases:

\[
\begin{align*}
\supseteq L / \supseteq R & \quad \text{Suppose } \Pi_1 \text{ and } \Pi \text{ are} \\
\Delta_1, B'_1\Pi_1 & \quad \Delta_1, B'_1\Pi \supseteq R \\
\Pi' & \quad B_{2,\ldots,\gamma}, B'_1, B_{2,\ldots,\gamma} \rightarrow C \\
\Pi'' & \quad B'_1 \supseteq B'_1, B_{2,\ldots,\gamma}, \gamma \rightarrow C \quad \supseteq L.
\end{align*}
\]

Let \( \Xi_1 = mc(\Pi_2, \ldots, \Pi_n, \Pi\Omega) \). Then \( \Xi_1 \in \text{RED}_{B'_1}[\Omega] \) by induction hypothesis since \( \text{ht}(\Pi') < \text{ht}(\Pi) \) and therefore \( \mathcal{M}(\Xi_1) < \mathcal{M}(\Xi) \). Since \( \Pi_1 \in \text{RED}_{B'_1}[\Omega] \), by Definition 15 we have

\[
\Pi'_1 \in \text{RED}_{B'_1}[\Omega] \Rightarrow \text{RED}_{B'_1}[\Omega]
\]

and therefore the derivation \( \Xi_2 = mc(\Xi_1, \Pi'_1) \) with end sequent \( \Delta_1, \ldots, \Delta_n, \Gamma\Omega \rightarrow B'_1\Omega \) is in \( \text{RED}_{B'_1}[\Omega] \).

Let \( \Xi_3 = mc(\Xi_3, \Pi_2, \ldots, \Pi_n, \Pi''\Omega) \).

The reduct of \( \Xi \) in this case is the derivation \( \Xi' \):

\[
\begin{align*}
\Xi' & \quad \Delta_1, \ldots, \Delta_n, \Gamma\Omega, \Delta_2, \ldots, \Delta_n, \Gamma\Omega \rightarrow C\Omega \\
\Delta_1, \ldots, \Delta_n, \Gamma\Omega & \quad \supseteq L
\end{align*}
\]

By the induction hypothesis, we have \( \Xi_1 \in \text{RED}_C[\Omega] \), and therefore, by Lemma 13 it is normalizable. By Definition 10 this means that \( \Xi' \) is normalizable and by Definition 15, \( \Xi' \in \text{RED}_C[\Omega] \).

\[
\forall L / \forall R \quad \text{Suppose } \Pi_1 \text{ and } \Pi \text{ are} \\
\Delta_1 & \quad \Pi'_1 \rightarrow B'_1\Pi \[y/x] \\
\Delta_1 & \quad \forall x. B'_1\Pi \forall R \\
\Pi' & \quad B'_1[\{t/x\}, B_{2,\ldots,\gamma}, \gamma \rightarrow C \forall L
\]

The reduct of \( \Xi \) in this case is

\[
\Xi' = mc(\Pi'_1[\{t/y\}], \Pi_2, \ldots, \Pi_n, \Pi''\Omega).
\]

Since \( \Pi'_1 \in \text{RED}_{B'_1[\{t/y\}]}[\Omega] \), by Lemma 13 we have

\[
\Pi'_1[\{t/y\}] \in \text{RED}_{B'_1[\{t/x\}]}[\Omega]
\]

Note that \( \text{ht}(\Pi') < \text{ht}(\Pi) \), so we can apply the induction hypothesis to obtain \( \Xi' \in \text{RED}_C[\Omega] \).

\[
\text{eqR / eqL} \quad \text{Suppose } \Pi_1 \text{ and } \Pi \text{ are} \\
\Delta_1 & \quad s = \text{eqR} \\
\Pi^p & \quad \{B_{2\rho,\ldots,\gamma}, \gamma \rightarrow C\rho\}_\rho \text{ eqL}
\]

24
Suppose $\Pi'$ be the derivation:

$$
\begin{array}{c}
\frac{B_2, B_n, \Gamma \rightarrow C}{\Pi'_{\rho}}
\end{array}
$$

and let $\Xi_1 = mc(\Pi_2, \ldots , \Pi_n, \Pi' \Omega)$. Since $ht(\Pi') = ht(\Pi)$ and since the total size of cut formulas in $\Xi_1$ is smaller than in $\Xi$, by the induction hypothesis, we have $\Xi_1 \in \text{RED}_{C'}[\Omega]$. Then the reduct of $\Xi$ in this case is the derivation $\Xi'$:

$$
\Xi_1 \quad \Xi' = \frac{\Delta_2, \ldots , \Delta_n, \Gamma \rightarrow C}{\Delta_1, \Delta_2, \ldots , \Delta_n, \Gamma \rightarrow C} \quad \text{wL}
$$

which is also in $\text{RED}_{C'}[\Omega]$, by the definition of parametric reducibility.

**IR/IL** Suppose $\Pi_1$ and $\Pi$ are the derivations

$$
\begin{array}{c}
\frac{\Pi_1'}{\Pi_1'} \rightarrow \Pi_1' \frac{\Pi_S}{\Pi_S} \rightarrow \Pi_S' \frac{\Pi'}{\Pi'} \rightarrow \Pi'
\end{array}
$$

where $p \bar{y} \equiv D p \bar{y}$ and $X^p$ is a new parameter not occurring in the end sequent of $\Pi_1$ (we can assume w.l.o.g. that $X^p \# \Omega$ and that it does not occur either in the end sequent of $\Pi$). Then $\Pi'$ is the derivation

$$
\frac{\Pi_S \Omega}{\Pi_S} \rightarrow \Pi_S' \frac{\Pi'}{\Pi'} \rightarrow \Pi'
$$

where $S' = S \Omega$. Let $\Xi_1 = mc(\Pi_1'[(\Pi_S \Omega, S')/X^p], \Pi_S \Omega[\bar{u}/\bar{x}])$. Then the reduct of $\Xi$ in this case is the derivation

$$
\Xi' = mc(\Xi_1, \Pi_2, \ldots , \Pi_n, \Pi' \Omega).
$$

Since $ht(\Pi_S[\bar{u}/\bar{x}]) \leq ht(\Pi_S) < ht(\Pi)$ by the induction hypothesis, we have

$$
\Pi_S \Omega[\bar{u}/\bar{x}] \in \text{RED}_{D S \bar{a}}[\Omega] \Rightarrow \text{RED}_{D S \bar{a}}[\Omega]. \quad (5)
$$

Let $R = \{ \Psi \mid \Psi \in \text{RED}_{D S \bar{a}}[\Omega] \text{ for some } \bar{a} \}$. Then by Lemma [17] $R$ is a reducibility candidate of type $S'$. Moreover, by Lemma [18] we have

$$
\text{RED}_{D S \bar{a}}[\Omega] = \text{RED}_{D X^p \bar{a}}[\Omega, (R, \Pi_S \Omega, S')/X^p].
$$

This, together with Statement [5] above, implies that

$$
\Pi_S \Omega[\bar{u}/\bar{x}] \in \text{RED}_{D X^p \bar{a}}[\Omega, (R, \Pi_S \Omega, S')/X^p] \Rightarrow R \bar{u} \quad (6)
$$

for every $\bar{u}$.

Since $\Pi_1 \in \text{RED}_{p \bar{t} '[\Omega]}$, it follows from Definition [18] that for every reducibility candidate $(S : I)$ and $\Pi_I$ such that

$$
\Pi_I[\bar{u}/\bar{x}] \in \text{RED}_{D X^p \bar{a}}[\Omega, (S, \Pi_I, I)/X^p] \Rightarrow S \bar{u} \text{ for every } \bar{u},
$$

we have

$$
mc(\Pi_1'[(\Pi_I, I)/X^p], \Pi_I[\bar{t}/\bar{x}]) \in S \bar{t}.
$$

Substituting $R$ for $S$, $\Pi_S \Omega$ for $\Pi_I$ and $S'$ for $I$, and using Statement [5] above, we obtain:

$$
\Xi_1 = mc(\Pi_1'[(\Pi_S \Omega, S')/X^p], \Pi_S \Omega[\bar{t}/\bar{x}]) \in R \bar{t} = \text{RED}_{S \bar{t} [\Omega]}.
$$

Since $ht(\Pi') < ht(\Pi)$, we can then apply the induction hypothesis to conclude that $\Xi' \in \text{RED}_{C'}[\Omega]$.

25
\[ \Pi' \]
\[ \Delta_1 \rightarrow S \vec{t} \quad S \vec{x} \rightarrow DS \vec{x} \]
\[ \Delta_1 \rightarrow p \vec{t} \] \hspace{1cm} (CIR)
\[ D X^p \vec{t}, B_2, \ldots, \Gamma \rightarrow C \]
\[ p \vec{t}, B_2, \ldots, \Gamma \rightarrow C \] \hspace{1cm} (CLC)

where \( p \vec{y} \equiv DP \vec{y} \) and \( X^p \) is a parameter not already occurring in the end sequent of \( \Pi \) (and w.l.o.g. assume also \( X^p \# \Pi \) and \( X^p \) not occurring in \( \Delta_i \) or \( B_i \)). Then \( \Pi \Omega \) is

\[ \Pi \Omega \]
\[ D X^p \vec{t}, B_2 \Omega, \ldots, \Gamma \Omega \rightarrow C \Omega \]
\[ p \vec{t}, B_2 \Omega, \ldots, \Gamma \Omega \rightarrow C \Omega \] \hspace{1cm} (CLC).

Since \( \Pi_1 \in \text{RED}_{p \vec{t}}[\Omega] \), by Definition [15] there exists a reducibility candidate \((R : S)\) such that \( \Pi_1' \in R \) and such that for every \( \vec{u} \),

\[ \Pi_S[\vec{u}/\vec{x}] \in R \vec{u} \Rightarrow \text{RED}_{D X^p \vec{u}}[\Omega, (R, \Pi_S, S)/X^p)]. \]

Let \( \Omega' = [\Omega, (R, \Pi_S, S)/X^p] \). Then by Lemma [20] \( \Omega' \) is definitionally closed.

Let \( \Xi_1 = mc(\Pi_1', \Pi_S[\vec{u}/\vec{x}]) \). By the definitional closure of \( \Omega' \), we have that \( \Xi_1 \in \text{RED}_{D X^p \vec{t}}[\Omega'] \).

The reduct of \( \Xi \) in this case is the derivation

\[ \Xi' = mc(\Xi_1, \Pi_2, \ldots, \Pi_n, \Pi' \Omega') . \]

Note that since \( X^p \) does not occur in \( \Delta_i \) or \( B_i \), by Lemma [16] we have that

\[ \Pi_i \in \text{RED}_{B_i}[\Omega] = \text{RED}_{B_i}[\Omega'] \]

for every \( i \in \{2, \ldots, n\} \). Therefore, by induction hypothesis, we have that

\[ \Xi' \in \text{RED}_{C}[\Omega'] . \]

But since \( X^p \) is also new for \( C \), we have \( \text{RED}_{C}[\Omega'] = \text{RED}_{C}[\Omega] \), and therefore

\[ \Xi' \in \text{RED}_{C}[\Omega] . \]

**II.2.** \( \Pi \) ends with a left rule, other than \( cL \), \( wL \) and a non-trivial instance of \( \text{CLC}_p \), acting on \( B_1 \), and \( \Pi_1 \) ends with a left-rule or \( \text{subst} \).

Note that in these cases, the reducts always end with a left-rule. The proof for the following cases abide to the same pattern: we first establish that the premise derivations of the reduct are either normalizable or in certain reducibility sets. We then proceed to show that the reduct itself is reducible by applying to the closure conditions of reducibility under applications of left-rules. For the latter, we need to distinguish three cases depending on \( C \): If \( C = X^p \vec{t} \) for some \( X^p \in \text{supp}(\Omega) \), then closure under left-rules is guaranteed by \( \text{C5} \); if \( X^p \notin \text{supp}(\Omega) \) then we need to show that the reduct is normalizable, and the closure condition under left-rules is guaranteed by the definition of normalizability. Otherwise, \( C \) is not headed by any parameter, and in this case, the closure condition follows from \( \text{P6} \). We shall explicitly do these case analysis in one of the subcases below, but will otherwise leave them implicit. We show the non-trivial subcases only; other cases can be proved by straightforward applications of the induction hypothesis.

\[ \supseteq \text{L} \wedge \text{L} \]

Suppose \( \Pi_1 \) is

\[ \Delta_1 \rightarrow D_1 \quad D_2, \Delta_1' \rightarrow B_1 \Omega \]
\[ D_1 \supseteq D_2, \Delta_1' \rightarrow B_1 \Omega \] \hspace{1cm} (L)

Since \( \Pi_1 \in \text{RED}_{B_1}[\Omega] \), it follows from Definition [15] that \( \Pi_1' \) is normalizable and \( \Pi_1'' \in \text{RED}_{B_1}[\Omega] \).

Let \( \Xi_1 = mc(\Pi_1'', \Pi_2, \ldots, \Pi_n, \Pi \Omega) \). Since \( nd(\Pi_1') < nd(\Pi_1) \), by induction hypothesis, \( \Xi_1 \in \text{RED}_{C}[\Omega] \).
The reduct of $\Xi$ in this case is the derivation $\Xi'$:

$$\begin{array}{c}
\Pi'_1 \\
\Delta'_1 \rightarrow D_1 \\
\Delta'_1, \ldots, \Gamma \Omega \rightarrow D_1 \quad w\mathcal{L} \\
D_1 \supset D_2, \Delta'_1, \Delta_2, \ldots, \Gamma \Omega \rightarrow C \Omega
\end{array} \quad \Xi_1 \quad \supset \mathcal{L}$$

Since $\Pi'_1$ is normalizable, by Definition 10 the left premise derivation of $\Xi'$ is normalizable, and since reducibility implies normalizability (Lemma 11), the right premise is also normalizable, hence $\Xi'$ is normalizable.

Now to show $\Xi' \in \text{RED}_C[\Omega]$, we need to distinguish three cases based on $C$:

- Suppose $C = X^p \bar{t}$ for some $X^p \in \text{supp}(\Omega)$ and $\Omega(X^p) = (R, \Pi_\delta, S)$. Then we need to show that $\Xi' \in R \bar{t}$. This follows from Definition 13 more specifically, from CR5 and the fact that $\Xi_1 \in \text{RED}_C[\Omega] = R \bar{t}$.

- Suppose $C = X^p \bar{t}$ but $X^p \notin \text{supp}(\Omega)$. Then we need to show that $\Xi'$ is normalizable. But this follows immediately from the normalizability of both of its premise derivations.

- Suppose $C \neq X^p \bar{t}$ for any parameter $X^p$ and any terms $\bar{t}$. Since $\Xi_1 \in \text{RED}_C[\Omega]$, by Definition 13 we have $\Xi' \in \text{RED}_C[\Omega]$.

\[ \text{eq}\mathcal{L}/ \circ \mathcal{L} \] Suppose $\Pi_1$ is as shown below left. Then the reduct of $\Xi$ in this case is shown below right, where $\Xi^p = mc(\Pi_1, \Pi_{2\rho}, \ldots, \Pi_n, \Pi_\rho, \Pi_{\rho}).$

$$\begin{array}{c}
\Pi_1 \\
\Delta'_1 \rightarrow B_1 \Omega \\
\Delta'_1, \ldots, \Gamma \Omega \rightarrow C \Omega
\end{array} \quad \Xi^p \quad \supset \mathcal{L}$$

$\Xi^p \in \text{RED}_{C_p}[\Omega]$ by the induction hypothesis (since $\text{nd}(\Pi'_1) < \text{nd}(\Pi_1)$ and the other measures are non-increasing). Hence, the reduct of $\Xi$ is in $\text{RED}_C[\Omega]$ by the definition of parametric reducibility.

\[ \text{IL}/ \circ \mathcal{L} \] Suppose $\Pi_1$ is

$$\begin{array}{c}
\Pi_S \\
D \bar{S} \bar{x} \rightarrow S \bar{x} \\
\Pi'_1 \\
S \bar{x}, \Delta'_1 \rightarrow B_1 \Omega
\end{array} \quad \Xi_1 \quad \supset \mathcal{L}$$

Since $\Pi_1 \in \text{RED}_{B_1}[\Omega]$, we have that $\Pi_S$ is normalizable and $\Pi'_1 \in \text{RED}_{B_1}[\Omega]$. Let $\Xi_1$ be the derivation $mc(\Pi'_1, \Pi_{2\rho}, \ldots, \Pi_n, \Pi_{\rho}).$

Then $\Xi_1 \in \text{RED}_{B_1}[\Omega]$ by the induction hypothesis, since $\text{nd}(\Pi'_1) < \text{nd}(\Pi_1)$. Therefore the reduct of $\Xi$

$$\begin{array}{c}
\Pi_S \\
D \bar{S} \bar{x} \rightarrow S \bar{x} \\
\Pi'_1 \\
S \bar{u}, \Delta'_1, \ldots, \Delta_n, \Gamma \Omega \rightarrow C \Omega
\end{array} \quad \Xi_1 \quad \supset \mathcal{L}$$

is also in $\text{RED}_C[\Omega]$.

II.3. II ends with a left rule, other than $c\mathcal{L}$, $w\mathcal{L}$ and a non-trivial instance of $\text{CL}_p$, acting on $B_1$, and $\Pi$ ends with $mc$ or $\text{init}$: These cases follow straightforwardly from the induction hypothesis.

II.4. Suppose $\Pi$ ends with a non-trivial application of $\text{CL}_p$ on $B_1$. That is, $B_1 = X^p \bar{t}$, for some $X^p \in \text{supp}(\Omega)$ and some $\bar{t}$, and $\Pi$ is

$$\begin{array}{c}
\Pi' \\
D X^p \bar{t}, B_2, \ldots, B_n, \Gamma \rightarrow C \\
X^p \bar{t}, B_2, \ldots, B_n, \Gamma \rightarrow C
\end{array} \quad \text{CL}_p$$

27
where \( p \bar{x} \not\equiv D p \bar{x} \). Suppose \( \Omega(X^p) = (\mathcal{R}, \Pi_S, S) \). Then \( \Pi\Omega \) is \( mc(mc(Id_{S\bar{\bar{t}}}, \Pi_S[\bar{\bar{t}}/\bar{x}]), \Pi'\Omega) \). Let \( \Xi_1 = mc(\Pi_1, mc(Id_{S\bar{\bar{t}}}, \Pi_S[\bar{\bar{t}}/\bar{x}])) \). Note that \( \Xi_1 \) has exactly one reduct, that is,
\[
\Xi_2 = mc(mc(\Pi_1, Id_{S\bar{\bar{t}}}), \Pi_S[\bar{\bar{t}}/\bar{x}]).
\]
Note that \( mc(\Pi_1, Id_{S\bar{\bar{t}}}) \) also has exactly one reduct, namely, \( \Pi_1 \). Since \( \Pi_1 \in \mathcal{RED}_{X^p\bar{\bar{t}}}[\Omega] = \mathcal{R}\bar{\bar{t}} \), this means, by \( \text{CR3} \), that \( mc(\Pi_1, Id_{S\bar{\bar{t}}}) \) is in \( \mathcal{R}\bar{\bar{t}} \). Since \( \Omega \) is definitionally closed, we have that \( \Xi_2 \in \mathcal{RED}_{D_{X^p\bar{\bar{t}}}}[\Omega] \). And since \( \Xi_2 \) is the only reduct of \( \Xi_1 \), this also means that, by Definition \( \text{II} \), \( \Xi_1 \in \mathcal{RED}_{D_{X^p\bar{\bar{t}}}}[\Omega] \).

The reduct of \( \Xi \), i.e. the derivation \( mc(\Xi_1, \Pi_2, \ldots, \Pi_n, \Pi'\Omega) \) is therefore in \( \mathcal{RED}_C[\Omega] \) by the induction hypothesis.

**II.5.** Suppose \( \Pi \) ends with \( w\mathcal{L} \) or \( c\mathcal{L} \) acting on \( B_1 \), or \( \text{init} \). Then \( \Pi\Omega \) also ends with the same rule. The cut reduction rule that applies in this case is either \(-/w\mathcal{L} \), \(-/c\mathcal{L} \) or \(-/\text{init} \). In these cases, parametric reducibility of the reducts follow straightforwardly from the assumption (in case of \( \text{init} \)), the induction hypothesis and Definition \( \text{II} \).

**II.6.** Suppose \( \Pi \) ends with \( mc \). Then \( \Pi\Omega \) also ends with \( mc \). The reduction rule that applies in this case is the reduction \(-/mc \). Parametric reducibility of the reduct in this case follows straightforwardly from the induction hypothesis and Definition \( \text{II} \).

**II.7.** Suppose \( \Pi \) ends with \( \text{subst} \) or a rule acting on a formula other than a cut formula. Most cases follow straightforwardly from the induction hypothesis, Lemmas \( \text{I} \) and Lemma \( \text{II} \) (which is needed in the reduction case \(-/\mathcal{L} \) and \(-/\text{subst} \)). We show the interesting subcases here:

\([-/\mathcal{R}_p\] \)

Suppose \( \Pi \) ends with a non-trivial \( \mathcal{R}_p \), i.e., \( \Pi \) is
\[
\begin{array}{c}
\Pi' \\
B_1, \ldots, B_n, \Gamma \rightarrow D X^p \bar{\bar{t}} \\
\hline
B_1, \ldots, B_n, \Gamma \rightarrow X^p \bar{\bar{t}}
\end{array}
\]
\( \mathcal{R}_p \)

where \( p \bar{x} \not\equiv D p \bar{x} \) and \( X^p \in \text{supp}(\Omega) \). Suppose \( \Omega(X^p) = (\mathcal{R}, \Pi_S, S) \). Then \( \Pi\Omega \) is the derivation \( mc(\Pi'\Omega, \Pi_S[\bar{\bar{t}}/\bar{x}]) \).

The reduct of \( \Xi \) in this case is the derivation
\[
\Xi' = mc(mc(\Pi_1, \ldots, \Pi_n, \Pi'\Omega), \Pi_S[\bar{\bar{t}}/\bar{x}]).
\]

By the induction hypothesis, we have \( mc(\Pi_1, \ldots, \Pi_n, \Pi'\Omega) \in \mathcal{RED}_{D_{X^p\bar{\bar{t}}}}[\Omega] \). This, together with the definitional closure of \( \Omega \), implies that \( \Xi' \in \mathcal{R}\bar{\bar{t}} = \mathcal{RED}_{D_{X^p\bar{\bar{t}}}}[\Omega] \).

\([-/\mathcal{R} \] \)

Suppose \( \Pi \) is
\[
\begin{array}{c}
\Pi' \\
B_1, \ldots, B_n, \Gamma \rightarrow D X^p \bar{\bar{t}} \\
\hline
B_1, \ldots, B_n, \Gamma \rightarrow p \bar{\bar{t}}
\end{array}
\]
\( \mathcal{R} \)

where \( p \bar{y} \not\equiv D p \bar{y} \). Without loss of generality we can assume that \( X^p \) is chosen to be sufficiently fresh (e.g., not occurring in \( \Omega, \Delta_1, B_1 \), etc.). Let \( \Xi_1 = mc(\Pi_1, \ldots, \Pi_n, \Pi'\Omega) \). Then the reduct of \( \Xi \) is the derivation \( \Xi' \)
\[
\begin{array}{c}
\Xi_1 \\
\Delta_1, \ldots, \Delta_n, \Gamma \Omega \rightarrow D X^p \bar{\bar{t}} \\
\hline
\Delta_1, \ldots, \Delta_n, \Gamma \Omega \rightarrow p \bar{\bar{t}}
\end{array}
\]
\( \mathcal{R} \).

To show that \( \Xi' \in \mathcal{RED}_C[\Omega] \), we first need to show that it is normalizable. This follows straightforwardly from the induction hypothesis (which shows that \( \Xi_1 \in \mathcal{RED}_{D_{X^p\bar{\bar{t}}}}[\Omega] \)) and Lemma \( \text{II} \). It then remains to show that
\[
\Xi_2 = mc(\Xi_1[(\Pi_S, S)/X^p], \Pi_S[\bar{\bar{t}}/\bar{x}]) \in \mathcal{R}\bar{\bar{t}}
\]
for every reducibility candidate \( (\mathcal{R} : S) \) and every \( \Pi_S \) such that
\[
\Pi_S[\bar{u}/\bar{x}] \in \mathcal{RED}_{D_{X^p\bar{u}}}[\Omega, (\mathcal{R}, \Pi_S, S)/X^p] \Rightarrow \mathcal{R}\bar{u}, \text{ for every } \bar{u}.
\]
So suppose \((\mathcal{R}, \Pi_S, S)\) satisfies Statement 7 above. Let \(\Omega' = [\Omega, (\mathcal{R}, \Pi_S, S)/X^p]\). By Lemma 19 \(\Omega'\) is definitional closed. Note that since we assume that \(X^p\) is a fresh parameter not occurring in \(B_i\), we have \(\text{RED}_{B_i}[\Omega] = \text{RED}_{B_i}[\Omega']\) by Lemma 10 and \(\Pi'[(\Pi_S, S)/X^p] = \Pi \in \text{RED}_{B_i}[\Omega']\) by Lemma 18 for every \(i \in \{1, \ldots, n\}\). Therefore, by the induction hypothesis we have:

\[
\Xi_1[(\Pi_S, S)/X^p] = mc(\Pi_1, \ldots, \Pi_n, \Pi'\Omega') \in \text{RED}_{D, X^p, \Gamma}[\Omega'].
\]

This, together with the definitional closure of \(\Omega'\), implies that \(\Xi_2 \in \mathcal{R} \tilde{\gamma}\).

\(-/\text{CIL}_p\) Suppose \(\Pi\) ends with a non-trivial CIL\(_p\), i.e., \(\Pi\) is

\[
\frac{\Pi'}{B_1, \ldots, B_n, D X^p \tilde{\gamma}, \Gamma' \rightarrow C} \frac{B_1, \ldots, B_n, X^p \tilde{\gamma}, \Gamma' \rightarrow C} \text{CIL}_p
\]

where \(p \equiv D p \tilde{\gamma}\) and \(X^p \in \text{supp}(\Omega)\). Suppose \(\Omega(X^p) = (\mathcal{R}, \Pi_S, S)\). Then \(\Pi\Omega\) is

\[
mc(mc(\text{Id}_{\tilde{\gamma}, \Pi_S[i/\tilde{x}]}, \Pi')\Omega).
\]

Let \(\Xi_1 = mc(\text{Id}_{\tilde{\gamma}, \Pi_S[i/\tilde{x}]})\). By CR4, \(\text{Id}_{\tilde{\gamma}, \Pi_S[i/\tilde{x}] \in \mathcal{R} \tilde{\gamma}}\), and therefore, by definitional closure of \(\Omega\), we have \(\Xi_1 \in \text{RED}_{D, X^p, \Gamma}[\Omega]\). The reduct of \(\Xi\) in this case is

\[
mc(\Xi_1, \Pi_1, \ldots, \Pi_n, \Pi'\Omega)
\]

which is in \(\text{RED}_C[\Omega]\) by the induction hypothesis.

\(-/\text{CIR}\) Suppose \(\Pi\) is

\[
\frac{\Pi'}{B_1, \ldots, B_n, \Gamma \rightarrow S \tilde{\gamma}} \frac{\Pi_S}{S \tilde{\gamma} \rightarrow D S \tilde{\gamma}} \text{CIR}
\]

where \(p \equiv D p \tilde{\gamma}\). Let \(S' = S\Omega\). The derivation \(\Pi\Omega\) in this case is

\[
\frac{\Pi'\Omega}{B_1, \ldots, B_n, \Gamma \rightarrow S' \tilde{\gamma}} \frac{\Pi_S\Omega}{S' \tilde{\gamma} \rightarrow D S' \tilde{\gamma}} \text{CIR}
\]

Let \(\Xi_1\) be the derivation \(mc(\Pi_1, \ldots, \Pi_n, \Pi'\Omega)\). By the induction hypothesis, \(\Xi_1 \in \text{RED}_{S, \tilde{\gamma}}[\Omega]\) and \(\Pi_S\Omega \in \text{RED}_{D, S, \tilde{\gamma}}[\Omega]\), hence both \(\Xi_1\) and \(\Pi_S\Omega\) are also normalizable by Lemma 14. The reduct of \(\Xi\) is the derivation \(\Xi'\)

\[
\Xi_1
\]

\[
\frac{\Xi_1}{\Delta_1, \ldots, \Delta_n, \Gamma \Omega \rightarrow S' \tilde{\gamma}} \frac{\Pi_S\Omega}{S' \tilde{\gamma} \rightarrow D S' \tilde{\gamma}} \text{CIR}
\]

Let \(X^p\) be a parameter fresh for \(\Omega, \Gamma, \Delta_i\) and \(B_i\).

To show that \(\Xi' \in \text{RED}_C[\Omega]\) we must first show that it is normalizable. This follows from immediately from normalizability of \(\Xi_1\) and \(\Pi_S\Omega\). Then we need to find a reducibility candidate \((\mathcal{R} : S')\) such that

(a) \(\Xi_1 \in \mathcal{R}\), and

(b) \(\Pi_S\Omega[i/\tilde{x}] \in \mathcal{R} \tilde{u} \Rightarrow \text{RED}_{D, X^p, \tilde{u}}[\Omega, (\mathcal{R}, \Pi_S, S)/X^p]\).

Let \(\mathcal{R} = \{\Psi \mid \Psi \in \text{RED}_{S, \tilde{u}}[\Omega]\}\). As in case 1.2.c., we show, using Lemma 17 that \(\mathcal{R}\) is a reducibility candidate of type \(S'\). By the induction hypothesis, we have \(\Xi_1 \in \mathcal{R}\), so \(\mathcal{R}\) satisfies (a). Using the same argument as in case 1.2.c. we can show that \(\mathcal{R}\) also satisfies (b), i.e. by appealing to the induction hypothesis, applied to \(\Pi_S\).

□
Corollary 22. Every derivation is reducible.

Proof. The proof follows from Lemma 21 by setting \( n = 0 \) and \( \Omega \) to the empty candidate substitution. \( \Box \)

Since reducibility implies cut-elimination and since every cut-free derivation can be turned into a subst-free derivation (Lemma 2), it follows that every proof can be transformed into a cut-free and subst-free derivation.

Corollary 23. Given a fixed definition, a sequent has a derivation in Linc− if and only if it has a cut-free and subst-free derivation.

The consistency of Linc− is an immediate consequence of cut-elimination. By consistency we mean the following: given a fixed definition and an arbitrary formula \( C \), it is not the case that both \( C \) and \( C \supset \bot \) are provable.

Corollary 24. The logic Linc− is consistent.

5. Related work and conclusions

Of course, there is a long association between mathematical logic and inductive definitions [1] and in particular with proof-theory, starting with the Takeuti’s conjecture, the earliest relevant entry for our purposes being Martin-Löf’s original formulation of the theory of iterated inductive definitions [24]. From the representation of algebraic types [7] and the introduction of (co)inductive types in system F [28, 10], (co)induction/recursion became mainstream and made it into type-theoretic proof assistants such as Coq [30], first via a primitive recursive operator, but eventually in the let-rec style of functional programming languages, as in Gimenez’s Calculus of Infinite Constructions [17]. Unlike works in these type-theoretic settings, we put less emphasis on proof terms and strong normalization; in fact, our cut elimination procedure is actually a form of weak normalization, in the sense that our procedure only guarantees termination with respect to a particular strategy, i.e., by reducing the lowest cuts in a derivation tree. Our notion of equality, which internalizes unification in its left introduction rule, departs from the more traditional notion of equality. As a consequence of these differences, it is not at all obvious that strong normalization proofs for term calculi with (co-)inductive types can be adapted straightforwardly to our setting.

Baelde and Miller have recently introduced an extension of multiplicative-additive linear logic with least and greatest fixed points [6], called \( \mu \text{MALL} \). In that work, cut elimination is proved indirectly via a second-order encoding of the least and the greatest fixed point operators into higher-order linear logic and via an appeal to completeness of focused proofs for higher-order linear logic. Such an encoding can also be used for proving cut elimination for Linc−, but as we noted earlier, our main concern here is to provide a basis for cut elimination for (orthogonal) extensions of Linc− with the \( \nabla \)-quantifier, for which there are currently no known encodings into higher-order (linear) logic. Baelde has also given a direct cut-elimination proof for \( \mu \text{MALL} \) [4]. The proof uses a notion of orthogonality in the definition of reducibility, defined via classical negation, so it is not clear if it can be adapted straightforwardly to the intuitionistic setting like ours.

Circular proofs are also connected with the proof-theory of fixed point logics and process calculi [45, 51], as well as in traditional sequent calculi such as in [8]. The issue is the equivalence between systems with local vs. global induction, that is, between fixed point rules vs. well-founded and guarded induction (i.e. circular proofs). In the traditional sequent calculus, it is unknown whether every global inductive proof can be translated into a local one.

In higher order logic (co)inductive definitions are usually obtained via the Tarski set-theoretic fixed point construction, as realized for example in Isabelle/HOL [37]. As we mentioned before, those approaches are at odd with HOAS even at the level of the syntax. This issue has originated a research field in its own and we only mention the main contenders: in the Twelf system [49] the LF type theory is used to encode deductive systems as judgments and to specify meta-theorems as relations (type families) among them; a logic programming-like interpretation provides an operational semantics to those relations, so that an external check for totality (incorporating termination, well-modedness and coverage [50, 40]) verifies that the given relation is indeed a realizer for that theorem. Coinduction is still unaccounted for and may require a switch to a different operational semantics for LF. There exists a second approach to reasoning in LF that is built on the idea of devising an explicit (meta-)meta-logic (\( M_{\omega} \)) for reasoning (inductively) about the framework [48].
It can be seen as a constructive first-order inductive type theory, whose quantifiers range over possibly open LF objects. In this calculus it is possible to express and inductively prove meta-logical properties of an object level system. $\mathcal{M}_\omega$ can be also seen as a dependently-typed functional programming language, and as such it has been refined into the Delphin programming language [41]. In a similar vein Beluga [41] is based on context modal logic [34], which provides a basis for a different foundation for programming with HOAS and dependent types. Because all of these systems are programming languages, we refrain from a deeper discussion. We only note that systems like Delphin or Beluga separate data from computations. This means they are always based on eager evaluation, whereas co-recursive functions should be interpreted lazily. Using standard techniques such as thunks to simulate lazy evaluation in such a context seems problematic (Pientka, personal communication).

Weak higher-order abstract syntax [10] is an approach that strives to co-exist with an inductive setting. The problem of negative occurrences in datatypes is handled by replacing them with a new type. Similarly for hypothetical judgments, although axioms are needed to reason about them, to mimic what is inferred by the cut rule in our architecture. Miculan et al.’s framework [21] embraces this axiomatic approach extending Coq with the “theory of contexts” (ToC). The theory includes axioms for the the reification of key properties of names akin to freshness. Furthermore, higher-order induction and recursion schemata on expressions are also assumed. Hybrid [2, 22] is a $\lambda$-calculus on top of Isabelle/HOL which provides the user with a Full HOAS syntax, compatible with a classical (co)-inductive setting. Linc$^-$ improves on the latter on several counts. First it disposes of Hybrid notion of abstraction, which is used to carve out the “parametric” function space from the full HOl function space. Moreover it is not restricted to second-order abstract syntax, as the current Hybrid version is (and as ToC cannot escape from being). Finally, at higher types, reasoning via $eqL$ and fixed points is more powerful than inversion, which does not exploit higher-order unification.

Nominal logic gives a different foundation to programming and reasoning with names. It can be presented as a first-order theory [42], which includes primitives for variable renaming and freshness, and a (derived) “new” freshness quantifier. It is endowed of natural principles of structural induction and recursion over syntax [43]. Urban et al. have engineered a nominal datatype package inside Isabelle/HOL [35] analogous to the standard datatype package but defining equivalence classes of term constructors. Co-induction/recursion on nominal datatypes is not available, but to be fair it is also currently absent from Isabelle/HOL.

We have presented a proof theoretical treatment of both induction and co-induction in a sequent calculus compatible with HOAS encodings. The proof principle underlying the explicit proof rules is basically fixed point (co)induction. However, the formulation of the rules is inspired by a second-order encoding of least and greatest fixed points. We have developed a new cut elimination proof, radically different from previous proofs [25, 52], using a reducibility-candidate technique à la Girard.

Consistency of the logic is an easy consequence of cut-elimination. Our proof system is, as far as we know, the first which incorporates a co-induction proof rule with a direct cut elimination proof. This schema can be used as a springboard towards cut elimination procedures for more expressive (conservative) extensions of Linc$^-$, for example in the direction of $FO\Lambda$ [31], or more recently, the logic $LG\omega$ [53] by Tiu and the logic $G$ by Gacek et al. [14].

An interesting problem is the connection with circular proofs, which is particularly attractive from the viewpoint of proof search, both inductively and co-inductively. This could be realized by directly proving a cut-elimination result for a logic where circular proofs, under termination and guardedness conditions completely replace (co)inductive rules. Indeed, the question whether “global” proofs are equivalent to “local” proofs [8] is still unsettled.

Acknowledgements The Linc$^-$ logic was developed in collaboration with Dale Miller. We thank David Baelde for his comments to a draft of this paper.

References

[1] Peter Aczel. An introduction to inductive definitions. In J. Barwise, editor, Handbook of Mathematical Logic, volume 90 of Studies in Logic and the Foundations of Mathematics, chapter C.7, pages 739–782. North-Holland, Amsterdam, 1977.

[2] Simon Ambler, Roy Crole, and Alberto Momigliano. Combining higher order abstract syntax with tactical theorem proving and (co)induction. In V. A. Carreño, editor, Proceedings of the 15th International
[3] Franz Baader and Wayne Snyder. Unification theory. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 445–532. Elsevier and MIT Press, 2001.

[4] David Baelde. Least and greatest fixed points in linear logic. *CoRR*, abs/0910.3383, 2009.

[5] David Baelde, Andrew Gacek, Dale Miller, Gopalan Nadathur, and Alwen Tiu. The Bedwyr system for model checking over syntactic expressions. In Frank Pfenning, editor, *CADE*, volume 4603 of *Lecture Notes in Computer Science*, pages 391–397. Springer, 2007.

[6] David Baelde and Dale Miller. Least and greatest fixed points in linear logic. In *LPAR*, Lecture Notes in Computer Science, pages 92–106. Springer, 2007.

[7] C. Bohm and A. Berarducci. Automatic synthesis of typed λ-programs on term algebras. *Theoretical Computer Science*, 39(2-3):135–153, August 1985.

[8] James Brotherston and Alex Simpson. Complete sequent calculi for induction and infinite descent. In *LICS*, pages 51–62. IEEE Computer Society, 2007.

[9] K. L. Clark. Negation as failure. In J. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[10] Joelle Despeyroux and Andre Hirschowitz. Higher-order abstract syntax with induction in Coq. In *Fifth International Conference on Logic Programming and Automated Reasoning*, pages 159–173, June 1994.

[11] Lars-Henrik Eriksson. A finitary version of the calculus of partial inductive definitions. In L.-H. Eriksson, L. Hallnäs, and P. Schroeder-Heister, editors, *Proceedings of the Second International Workshop on Extensions to Logic Programming*, volume 596 of *Lecture Notes in Artificial Intelligence*, pages 89–134. Springer-Verlag, 1991.

[12] Murdoch J. Gabbay. Fresh logic: proof-theory and semantics for FM and nominal techniques. *Journal of Applied Logic*, 5(2):356–387, 2007.

[13] Andrew Gacek. The Abella interactive theorem prover (system description). In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *IJCAR*, volume 5195 of *Lecture Notes in Computer Science*, pages 154–161. Springer, 2008.

[14] Andrew Gacek, Dale Miller, and Gopalan Nadathur. Combining generic judgments with recursive definitions. In *LICS*, pages 33–44. IEEE Computer Society, 2008.

[15] Andrew Gacek, Dale Miller, and Gopalan Nadathur. Reasoning in Abella about structural operational semantics specifications. *Electr. Notes Theor. Comput. Sci.*, 228:85–100, 2009.

[16] Herman Geuvers. Inductive and coinductive types with iteration and recursion. In B. Nordström, K. Pettersson, and G. Plotkin, editors, *Informal Proceedings Workshop on Types for Proofs and Programs, Båstad, Sweden, 8–12 June 1992*, pages 193–217. Dept. of Computing Science, Chalmers Univ. of Technology and Göteborg Univ., 1992.

[17] Eduardo Giménez. *Un Calcul de Constructions Infinies et son Application a la Verification des Systemes Communicants*. PhD thesis PhD 96-11, Laboratoire de l’Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, December 1996.

[18] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types*. Cambridge University Press, 1989.

[19] Lars Hallnäs. Partial inductive definitions. *Theor. Comput. Sci.*, 87(1):115–142, 1991.

[20] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the ACM*, 40(1):143–184, 1993.
[21] Furio Honsell, Marino Miculan, and Ivan Scagnetto. An axiomatic approach to metareasoning on nominal algebras in HOAS. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, ICALP, volume 2076 of Lecture Notes in Computer Science, pages 963–978. Springer, 2001.

[22] Hybrid Group. Hybrid: A package for higher-order syntax in Isabelle/HOL and Coq. hybrid.dsi.unimi.it 2008, Accessed Wed Feb 24 2010.

[23] Bart Jacobs and Jan Rutten. A tutorial on (co)algebras and (co)induction. Bulletin of the European Association for Theoretical Computer Science, 62:222–259, June 1997. Surveys and Tutorials.

[24] Per Martin-Löf. Hauptsatz for the intuitionistic theory of iterated inductive definitions. In J. E. Fenstad, editor, Proceedings of the Second Scandinavian Logic Symposium, volume 63 of Studies in Logic and the Foundations of Mathematics, pages 179–216. North-Holland, 1971.

[25] Raymond McDowell and Dale Miller. Cut-elimination for a logic with definitions and induction. Theoretical Computer Science, 232:91–119, 2000.

[26] Raymond McDowell and Dale Miller. Reasoning with higher-order abstract syntax in a logical framework. ACM Transactions on Computational Logic, 3(1):80–136, January 2002.

[27] Raymond McDowell, Dale Miller, and Catuscia Palamidessi. Encoding transition systems in sequent calculus. TCS, 294(3):411–437, 2003.

[28] N. P. Mendler. Recursive types and type constraints in second-order lambda calculus. In LICS, pages 30–36. IEEE Computer Society, 1987.

[29] Marino Miculan and Kidane Yemane. A unifying model of variables and names. In Vladimiro Sassone, editor, FoSSaCS, volume 3441 of Lecture Notes in Computer Science, pages 170–186. Springer, 2005.

[30] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. In Peter Schroeder-Heister, editor, Extensions of Logic Programming: International Workshop, Tübingen, volume 475 of LNAI, pages 253–281. Springer-Verlag, 1991.

[31] Dale Miller and Alwen Tiu. A proof theory for generic judgments. ACM Trans. Comput. Logic, 6(4):749–783, 2005.

[32] Alberto Momigliano and Simon Ambler. Multi-level meta-reasoning with higher order abstract syntax. In A. Gordon, editor, FOSSACS’03, volume 2620 of LNCS, pages 375–392. Springer Verlag, 2003.

[33] Alberto Momigliano and Alwen Tiu. Induction and co-induction in sequent calculus. In Stefano Berardi, Mario Coppe, and Ferruccio Damiani, editors, TYPES, volume 3085 of Lecture Notes in Computer Science, pages 293–308. Springer, 2003.

[34] Aleksandar Nanevski, Frank Pfenning, and Brigitte Pientka. Contextual modal type theory. ACM Trans. Comput. Log., 9(3), 2008.

[35] Nominal Methods Group. Nominal Isabelle. isabelle.in.tum.de/nominal 2008, Accessed Sun Feb 14 2010.

[36] Christine Paulin-Mohring. Inductive definitions in the system Coq: Rules and properties. In M. Bezem and J. F. Groote, editors, Proceedings of the International Conference on Typed Lambda Calculi and Applications, pages 328–345, Utrecht, The Netherlands, March 1993. Springer-Verlag LNCS 664.

[37] Lawrence C. Paulson. Mechanizing coinduction and corecursion in higher-order logic. Journal of Logic and Computation, 7(2):175–204, March 1997.

[38] Frank Pfenning. Logical frameworks. In Alan Robinson and Andrei Voronkov, editors, Handbook of Automated Reasoning, chapter 17, pages 1063–1147. Elsevier Science Publisher and MIT Press, 2001.
[39] Frank Pfenning and Christine Paulin-Mohring. Inductively defined types in the Calculus of Constructions. In M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, *Proceedings of the Fifth Conference on the Mathematical Foundations of Programming Semantics, Tulane University, New Orleans, Louisiana*, pages 209–228. Springer-Verlag LNCS 442, March 1989.

[40] Brigitte Pientka. Verifying termination and reduction properties about higher-order logic programs. *J. Autom. Reasoning*, 34(2):179–207, 2005.

[41] Brigitte Pientka. A type-theoretic foundation for programming with higher-order abstract syntax and first-class substitutions. In George C. Necula and Philip Wadler, editors, *POPL*, pages 371–382. ACM, 2008.

[42] Andrew M. Pitts. Nominal logic, A first order theory of names and binding. *Information and Computation*, 186(2):165–193, 2003.

[43] Andrew M. Pitts. Alpha-structural recursion and induction. *J. ACM*, 53(3):459–506, 2006.

[44] Adam Poswolsky and Carsten Schürmann. Practical programming with higher-order encodings and dependent types. In Sophia Drossopoulou, editor, *ESOP*, volume 4960 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2008.

[45] Luigi Santocanale. A calculus of circular proofs and its categorical semantics. In Mogens Nielsen and Uffe Engberg, editors, *FoSSaCS*, volume 2303 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 2002.

[46] Ulrich Schöpp. Modelling generic judgements. *Electr. Notes Theor. Comput. Sci.*, 174(5):19–35, 2007.

[47] Peter Schroeder-Heister. Rules of definitional reflection. In M. Vardi, editor, *Eighth Annual Symposium on Logic in Computer Science*, pages 222–232. IEEE Computer Society Press, IEEE, June 1993.

[48] Carsten Schürmann. *Automating the Meta-Theory of Deductive Systems*. PhD thesis, Carnegie-Mellon University, 2000. CMU-CS-00-146.

[49] Carsten Schürmann. The Twelf proof assistant. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *TPHOLs*, volume 5674 of *Lecture Notes in Computer Science*, pages 79–83. Springer, 2009.

[50] Carsten Schürmann and Frank Pfenning. A coverage checking algorithm for LF. In David A. Basin and Burkhard Wolff, editors, *TPHOLs*, volume 2758 of *Lecture Notes in Computer Science*, pages 120–135. Springer, 2003.

[51] C. Spenger and M. Dams. On the structure of inductive reasoning: Circular and tree-shaped proofs in the $\mu$-calculus. In A. Gordon, editor, *FOSSACS*’03, volume 2620 of *LNCS*, pages 425–440. Springer Verlag, 2003.

[52] Alwen Tiu. *A Logical Framework for Reasoning about Logical Specifications*. PhD thesis, Pennsylvania State University, May 2004.

[53] Alwen Tiu. A logic for reasoning about generic judgments. *Electr. Notes Theor. Comput. Sci.*, 174(5):3–18, 2007.

[54] Alwen Tiu and Dale Miller. Proof search specifications of bisimulation and modal logics for the $\pi$-calculus. *ACM Trans. Comput. Logic*, 11(2):1–35, 2010.
A. The complete set of cut reduction rules

**Essential cases:**

1. **∧R / ∧L** If $\Pi_1$ and $\Pi$ are

   \[
   \begin{align*}
   \Pi'_1 & \Delta_1 \rightarrow B'_1 \Delta_1 \rightarrow B''_1 \land R \\
   \Pi''_1 & \Delta_1 \rightarrow B'_1 \land B''_1 \land & \Pi' & B'_1, B_2, \ldots, B_n, \Gamma \rightarrow C \\
   & B_1' \land B''_1, B_2, \ldots, B_n, \Gamma \rightarrow C & & \land L
   \end{align*}
   \]

   then $\Xi$ reduces to $mc(\Pi'_1, \Pi'', \Pi')$. The case for the other $\land L$ rule is symmetric.

2. **∨R / ∨L** Suppose $\Pi_1$ and $\Pi$ are

   \[
   \begin{align*}
   \Pi'_1 & \Delta_1 \rightarrow B'_1 \lor B''_1 \lor R \\
   & B_1' \lor B''_1, B_2, \ldots, B_n, \Gamma \rightarrow C & & \lor L
   \end{align*}
   \]

   Then $\Xi$ reduces to $mc(\Pi'_1, \Pi'', \Pi')$. The case for the other $\lor R$ rule is symmetric.

3. **⊃ R / ⊃ L** Suppose $\Pi_1$ and $\Pi$ are

   \[
   \begin{align*}
   \Pi'_1 & \Delta_1 \rightarrow B'_1 \lor B''_1 \lor R \\
   & B_1' \lor B''_1, B_2, \ldots, B_n, \Gamma \rightarrow C & & \lor L
   \end{align*}
   \]

   Let $\Xi_1 = mc(me(\Pi_2, \ldots, \Pi_n, \Pi'_1), \Pi'_1)$. Then $\Xi$ reduces to

   \[
   \Xi_1 \rightarrow B''_1 \left\{ \Delta, \Pi, \Pi'' \right\}_{i \in \{2, \ldots, n\}} \left( \Delta_1', \ldots, \Delta_n', \Gamma, \Delta_2', \ldots, \Delta_n', \Gamma \rightarrow C \right) mc \left( \Delta_1', \ldots, \Delta_n', \Gamma \rightarrow C \right)
   \]

4. **∀R / ∀L** If $\Pi_1$ and $\Pi$ are

   \[
   \begin{align*}
   \Pi'_1 & \Delta_1 \rightarrow B'_1[y/x] \forall R \\
   & \Delta_1 \rightarrow \forall x.B'_1 & & \forall L
   \end{align*}
   \]

   then $\Xi$ reduces to $mc(\Pi'_1[t/y], \Pi_2, \ldots, \Pi_n, \Pi')$.

5. **∃R / ∃L** If $\Pi_1$ and $\Pi$ are

   \[
   \begin{align*}
   \Pi'_1 & \Delta_1 \rightarrow B'_1[t/x] \exists R \\
   & \Delta_1 \rightarrow \exists x.B'_1 & & \exists L
   \end{align*}
   \]

   then $\Xi$ reduces to $mc(\Pi'_1, \Pi_2, \ldots, \Pi'[t/y])$.

6. **IR / IL** Suppose $\Pi_1$ and $\Pi$ are, respectively,

   \[
   \begin{align*}
   \Pi'_1 & \Delta_1 \rightarrow D Xp \tilde{t} IR \\
   & \Delta_1 \rightarrow p \tilde{t} & & IL
   \end{align*}
   \]

   where $p \bar{x} \equiv D p x$ and $Xp$ is a new parameter. Then $\Xi$ reduces to

   \[
   mc(mc(\Pi'_1, (\Pi_S, S)/(Xp)), (\Pi_S, (\tilde{t}/\bar{y})), \Pi_2, \ldots, \Pi_n, \Pi').
   \]
Suppose $\Pi_1$ and $\Pi$ are

$\Pi_1' \quad \Pi_S \quad \Pi''$

$\Delta_1 \rightarrow S \vec{t} \quad S \vec{y} \rightarrow D S \vec{y} \quad \Delta_1 \rightarrow p \vec{t}$

\[ \text{CIR} \quad \Delta_1 \rightarrow p \vec{t} \]

$D X^p \vec{t}, \ldots, \Gamma \rightarrow C \quad \text{CL}$

where $p \vec{y} \equiv D p \vec{y}$ and $X^p$ is a new parameter. Then $\Xi$ reduces to

\[ mc(mc(\Pi_1', \Pi_S[\vec{t}/\vec{y}]), \Pi_2, \ldots, \Pi_n, \Pi''[(\Pi_S, S)/X^p]). \]

**eqR/eqL** Suppose $\Pi_1$ and $\Pi$ are

$\Delta_1 \rightarrow s = t \quad \text{eqR}$

\[ \begin{align*}
\Delta_1 \rightarrow s = t & \quad \text{eqR} \\
\Pi & \rightarrow \Pi' \quad \text{eqL}
\end{align*} \]

Note that in this case, $\rho$ in $\Pi$ ranges over all substitution, as any substitution is a unifier of $s$ and $t$. Let $\Xi_1$ be the derivation $mc(\Pi_2, \ldots, \Pi_n, subst(\{\Pi'\}_\rho))$. Then $\Xi$ reduces to

\[ \Xi_1 \]

$\Delta_2, \ldots, \Delta_n, \Gamma \rightarrow C \quad \text{wL}$

**Left-commutative cases.** In the following cases, we suppose that $\Pi$ ends with a left rule, other than $\{cL, wL\}$, acting on $B_1$.

**•L/ ◦ L** Suppose $\Pi_1$ is as below left, where $\bullet L$ is any left rule except $\supset L$, eqL, or I$L$. Let $\Xi' = mc(\Pi_1', \Pi_2, \ldots, \Pi_n, \Pi)$. Then $\Xi$ reduces to the derivation given below right.

\[ \begin{align*}
\Delta_1 \rightarrow B_1 & \quad \bullet L \\
\Pi' & \rightarrow \Pi'' \quad \Xi' \quad \text{eqL}
\end{align*} \]

$\Delta_1', \Delta_2, \ldots, \Delta_n, \Gamma \rightarrow C$

**⊃L/ ◦ L** Suppose $\Pi_1$ is

$\Delta_1' \rightarrow D_1' \quad D_1', \Delta_1' \rightarrow B_1 \quad \supset L$

Let $\Xi_1 = mc(\Pi_1', \Pi_2, \ldots, \Pi_n, \Pi)$. Then $\Xi$ reduces to

\[ \begin{align*}
\Pi_1' & \rightarrow \Pi'' \quad \Xi_1 \quad \text{wL} \\
\Delta_1', \Delta_2, \ldots, \Delta_n, \Gamma \rightarrow D_1' \quad D_1', \Delta_1', \Delta_2, \ldots, \Delta_n, \Gamma \rightarrow C
\end{align*} \]

**I$L/ L** Suppose $\Pi_1$ is

$DS \vec{y} \rightarrow S \vec{y} \quad S \vec{t}, \Delta_1' \rightarrow B_1$

\[ \text{I$L} \]

where $p \vec{y} \not\equiv D p \vec{y}$. Let $\Xi_1 = mc(\Pi'_1, \Pi_2, \ldots, \Pi_n, \Pi)$. Then $\Xi$ reduces to

\[ \begin{align*}
\Pi_S & \rightarrow \Pi' \quad \Xi_1 \quad \text{I$L} \\
DS \vec{y} \rightarrow S \vec{y} \quad S \vec{t}, \Delta_1', \ldots, \Delta_n, \Gamma \rightarrow C
\end{align*} \]
Suppose \( \Pi \) is as below left. Let \( \Xi^\rho = mc(\Pi_1^\rho, \Pi_2^\rho, \ldots, \Pi_n^\rho, \Pi)^\rho \). Then \( \Xi \) reduces to the derivation given below right.

\[
\begin{align*}
\begin{array}{c}
\{ \Delta_1^\rho \rightarrow B_1^\rho \} \\
\{ \Delta_1^\rho, \Delta_2^\rho, \ldots, \Delta_n^\rho, \Gamma \rightarrow C \rho \}
\end{array}
\end{align*}
\]

Suppose \( \Pi_1 \) is as shown below left. Let \( \Xi^i = mc(\Pi_1^i, \Pi_2^i, \ldots, \Pi_n^i, \Pi)^i \). Then \( \Xi \) reduces to

\[
\begin{align*}
\begin{array}{c}
\{ \Delta_1^i, \Delta_2^i, \ldots, \Delta_n^i, \Gamma \rightarrow C \}
\end{array}
\end{align*}
\]

Right-commutative cases:

Suppose \( \Pi \) is as given below left, where where \( \circ \mathcal{L} \) is any left rule other than \( \mathcal{C} \mathcal{L} \), eq\( \mathcal{L} \), or I\( \mathcal{L} \) acting on a formula other than \( B_1, \ldots, B_n \). Let \( \Xi^i = mc(\Pi_1^i, \Pi_2^i, \ldots, \Pi_n^i, \Pi) \). Then \( \Xi \) reduces to the derivation given below right.

\[
\begin{align*}
\begin{array}{c}
\{ \Delta_1^i, \Delta_2^i, \ldots, \Delta_n^i, \Gamma \rightarrow C \}
\end{array}
\end{align*}
\]

Suppose \( \Pi \) is

\[
\begin{align*}
\begin{array}{c}
\{ \Delta_1^i, \Delta_2^i, \ldots, \Delta_n^i, \Gamma' \rightarrow D' \}
\end{array}
\end{align*}
\]

Let \( \Xi_1 = mc(\Pi_1, \Pi_2, \ldots, \Pi_n^i, \Pi) \) and let \( \Xi_2 = mc(\Pi_1, \Pi_2, \ldots, \Pi_n^i, \Pi') \). Then \( \Xi \) reduces to

\[
\begin{align*}
\begin{array}{c}
\Delta_1^i, \Delta_2^i, \ldots, \Delta_n^i, \Gamma' \rightarrow D' \\
\Delta_1^i, \Delta_2^i, \ldots, \Delta_n^i, \Gamma' \rightarrow C
\end{array}
\end{align*}
\]

Suppose \( \Pi \) is

\[
\begin{align*}
\begin{array}{c}
\{ \Delta_1^i, \Delta_2^i, \ldots, \Delta_n^i, \Gamma' \rightarrow D' \}
\end{array}
\end{align*}
\]

where \( p \bar{y} \neq D p \bar{y} \). Let \( \Xi_1 = mc(\Pi_1, \Pi_2, \Pi') \). Then \( \Xi \) reduces to

\[
\begin{align*}
\begin{array}{c}
\Delta_1^i, \Delta_2^i, \ldots, \Delta_n^i, \Gamma' \rightarrow C
\end{array}
\end{align*}
\]

Suppose \( \Pi \) is as shown below left. Let \( \Xi^\rho = mc(\Pi_1^\rho, \ldots, \Pi_n^\rho, \Pi)^\rho \). Then \( \Xi \) reduces to the derivation below right.

\[
\begin{align*}
\begin{array}{c}
\{ \Delta_1^\rho, \Delta_2^\rho, \ldots, \Delta_n^\rho, \Gamma \rightarrow C \rho \}
\end{array}
\end{align*}
\]

If \( \Pi = subst(\{\Pi_1^\rho\}) \) then \( \Xi \) reduces to \( subst(\{mc(\Pi_1^\rho, \ldots, \Pi_n^\rho, \Pi)^\rho\}) \).

If \( \Pi \) is as below left, where where \( \circ \mathcal{R} \) is any right rule except CI\( \mathcal{R} \), then \( \Xi \) reduces to the derivation below right, where \( \Xi^i = mc(\Pi_1^i, \Pi_2^i, \ldots, \Pi_n^i, \Pi) \).

\[
\begin{align*}
\begin{array}{c}
\{ \Delta_1^i, \Delta_2^i, \ldots, \Delta_n^i, \Gamma \rightarrow C \}
\end{array}
\end{align*}
\]
\(-/\text{CTR}\) Suppose \(\Pi\) is
\[
\begin{array}{c}
\Pi' \\
B_1, \ldots, B_n, \Gamma \rightarrow S \bar{t}' \quad S \bar{y} \rightarrow DS \bar{y} \\
\Pi_S \\
B_1, \ldots, B_n, \Gamma \rightarrow p \bar{t}'
\end{array}
\]
CTR,
\]
where \(p \bar{y} \equiv D p \bar{y}\). Let \(\Xi_1 = mc(\Pi_1, \ldots, \Pi_n, \Pi')\). Then \(\Xi\) reduces to
\[
\begin{array}{c}
\Xi_1 \\
\Delta_1, \ldots, \Delta_n, \Gamma \rightarrow S \bar{t}' \quad S \bar{y} \rightarrow DS \bar{y} \\
\Pi_S \\
\Delta_1, \ldots, \Delta_n, \Gamma \rightarrow p \bar{t}'
\end{array}
\]
CTR

\section*{Multicut cases:}
\(mc/\circ L\) If \(\Pi\) ends with a left rule, other than \(cL\) and \(wL\), acting on \(B_1\) and \(\Pi_1\) ends with a multicut and reduces to \(\Pi'_1\), then \(\Xi\) reduces to \(mc(\Pi'_1, \Pi_2, \ldots, \Pi_n, \Pi)\).

\(-/mc\) Suppose \(\Pi\) is
\[
\begin{array}{c}
\left\{ \left\{ \left\{ B_i \right\}_{i \in I^j} \right\}_{j \in \{1, \ldots, m\}} \right\} \left\{ D^j \right\}_{j \in \{1, \ldots, m\}} \left\{ B_i \right\}_{i \in I'} \Gamma \rightarrow C \\
\Pi_j^I \\
\Pi_j^I \Gamma \rightarrow C
\end{array}
\]
mc

\[
\begin{array}{c}
\left\{ \left\{ \left\{ B_i \right\}_{i \in I^j} \right\}_{j \in \{1, \ldots, m\}} \right\} \left\{ D^j \right\}_{j \in \{1, \ldots, m\}} \\
\Pi_j^I \Gamma \rightarrow C
\end{array}
\]
mc

Then \(\Xi\) reduces to
\[
\begin{array}{c}
\left\{ \Xi_j \right\}_{j \in \{1, \ldots, m\}} \\
\Delta_1, \ldots, \Delta_n, \Gamma^1, \ldots, \Gamma^m, \Gamma' \rightarrow C
\end{array}
\]
mc

\section*{Structural cases:}
\(-/cL\) If \(\Pi\) is as shown below left, then \(\Xi\) reduces to the derivation shown below right, where \(\Xi_1 = mc(\Pi_1, \Pi_2, \ldots, \Pi_n, \Pi')\).
\[
\begin{array}{c}
\Pi' \\
B_1, B_2, \ldots, B_n, \Gamma \rightarrow C \\
\Pi
\end{array}
\]
cL

\[
\begin{array}{c}
\Xi_1 \\
\Delta_1, \Delta_2, \ldots, \Delta_n, \Gamma \rightarrow C
\end{array}
\]
cL

\(-/wL\) If \(\Pi\) is as shown below left, then \(\Xi\) reduces to the derivation shown below right, where \(\Xi_1 = mc(\Pi_2, \ldots, \Pi_n, \Pi')\).
\[
\begin{array}{c}
\Pi' \\
B_2, \ldots, B_n, \Gamma \rightarrow C \\
\Pi
\end{array}
\]
wL

\[
\begin{array}{c}
\Xi_1 \\
\Delta_1, \Delta_2, \ldots, \Delta_n, \Gamma \rightarrow C
\end{array}
\]
wL

\section*{Axiom cases:}
\(\text{init}/\circ L\) Suppose \(\Pi\) ends with a left-rule acting on \(B_1\) and \(\Pi_1\) ends with the init rule. Then it must be the case that \(\Delta_1 = \{B_1\}\) and \(\Xi\) reduces to \(mc(\Pi_2, \ldots, \Pi_n, \Pi)\).

\(-/\text{init}\) If \(\Pi\) ends with the init rule, then \(n = 1, \Gamma\) is the empty multiset, and \(C\) must be a cut formula, i.e., \(C = B_1\). Therefore \(\Xi\) reduces to \(\Pi_1\).
B. Proofs for Section 4.1 and Section 4.3

Lemma 9. Let $\Pi$ be a derivation ending with a $mc$ and let $\theta$ be a substitution. If $\Pi\theta$ reduces to $\Xi$ then there exists a derivation $\Pi'$ such that $\Xi = \Pi'\theta$ and $\Pi$ reduces to $\Pi'$.

Proof. Observe that the redexes of a derivation are not affected by eigenvariable substitution, since the cut reduction rules are determined by the last rules of the premise derivations, which are not changed by substitution. Therefore, any cut reduction rule that is applied to $\Pi\theta$ to get $\Xi$ can also be applied to $\Pi$. Suppose that $\Pi'$ is the reduct of $\Pi$ obtained this way. In all cases, except for the cases where the reduction rule applied is either $IR/IC$, $CL/CI$, or those involving $eqL$, it is a matter of routine to check that $\Pi'\theta = \Xi$.

For the reduction rules $IR/IC$ and $CL/CI$, we need Lemma 9 which shows that eigenvariable substitution commutes with parameter substitution. We show here the case involving $eqL$. The only interesting case is the reduction $eqL/eqR$. For simplicity, we show the case where $\Pi$ ends with $mc$ with three premises; it is straightforward to adapt the following analysis to the more general case. So suppose $\Pi$ is the derivation:

$$
\begin{align*}
\Delta_1 & \to t = t & \text{eqR} \\
\Pi_2 & \Delta_2 \to B & \{ B\rho, \Gamma\rho \to C\rho \}^\rho \text{ eqL} \\
\Delta_1, \Delta_2, \Gamma & \to C & mc
\end{align*}
$$

According to Definition 5 the derivation $\Pi\theta$ is:

$$
\begin{align*}
\Delta_1 & \to t\theta = t\theta & \text{eqR} \\
\Pi_2 & \Delta_2 \to B\theta & \{ B\theta\rho', \Gamma\theta\rho' \to C\theta\rho' \}^\rho' \text{ eqL} \\
\Delta_1, \Delta_2, \Gamma & \to C\theta & mc
\end{align*}
$$

Let $\Psi = mc(\Pi_2\theta, \text{subst}(\{\Pi^{(\theta\rho')}\}_\rho))$. The reduct of $\Pi\theta$ in this case (modulo the different order in which the weakening steps are applied) is:

$$
\begin{align*}
\Delta_2 & \to C\theta & \text{wL}
\end{align*}
$$

Let us call this derivation $\Xi$.

Let $\Psi' = mc(\Pi_2, \text{subst}(\{\Pi^\rho\}_\rho))$. The above reduct can be matched by the following reduct of $\Pi$ (using the same order of applications of the weakening steps):

$$
\begin{align*}
\Delta_2 & \to C & \text{wL}
\end{align*}
$$

Let us call this derivation $\Pi'$. By Definition 5 we have $\Psi' = \Psi\theta$, and obviously, also $\Xi = \Pi'\theta$. □

Lemma 14. If $\Pi \in \text{RED}_{\text{C}}[\Omega]$ then $\Pi$ is normalizable.

Proof. By case analysis on $C$. If $C = X^p\bar{u}$ for some $\bar{u}$ and $X^p \in \text{supp}(\Omega)$ then $\Pi \in R$, where $\Omega(X^p) = (R, \Pi_S, S)$, hence it is normalizable by Definition 13 (specifically, condition CR1). Otherwise, $\Pi$ is normalizable by Definition 15. □

Lemma 15. If $\Pi \in \text{RED}_{\text{C}}[\Omega]$ then for every substitution $\rho$, $\Pi\rho \in \text{RED}_{\text{C}_{\rho}}[\Omega]$.

Proof. By induction on $|C|$ with sub-induction on $\text{nd}(\Pi)$.

Suppose $C = X^q\bar{u}$, for some $\bar{u}$ and some $X^q \in \text{supp}(\Omega)$, and suppose $\Omega(X^q) = (R, \Pi_S, S)$. Then $\Pi \in R$ by Definition 15. By Definition 13 (CR0) we also have $\Pi\rho \in R$. Otherwise, suppose $X^q \notin \text{supp}(\Omega)$. Then $\Pi \in \text{NM}_{X^q}$ by Definition 15. By Lemma 11 we have $\Pi\rho \in \text{NM}_{X^q}$, therefore $\Pi\rho \in \text{RED}_{\text{C}_{\rho}}[\Omega]$.

Otherwise, $C \neq X^q\bar{u}$ for any $\bar{u}$ and any parameter $X^q$. In this case, to apply the inner induction hypothesis, we need to show that $\Pi\rho$ is normalizable, which follows immediately from Lemma 14 and Lemma 11.

We distinguish several cases based on the last rule of $\Pi$:
• Suppose \( \Pi \) ends with \( mc \), i.e., \( \Pi = mc(\Pi_1, \ldots, \Pi_n, \Pi') \) for some \( \Pi_1, \ldots, \Pi_n \) and \( \Pi' \). By Lemma 9, every reduct of \( \Pi_\rho \), say \( \Xi \), is the result of applying \( \rho \) to a reduct of \( \Pi \). By the inner induction hypothesis (on the normalization degree), every reduct of \( \Pi_\rho \) is in \( \text{RED}_{C_\rho}(\Omega) \), and therefore \( \Pi_\rho \) is also in \( \text{RED}_{C_\rho}(\Omega) \) by Definition 15 (P2).

• Suppose \( \Pi \) ends with \( \supseteq R \), with the premise derivation \( \Pi' \). In this case, \( C = B \supseteq D \) for some \( B \) and \( D \). Since \( \Pi \in \text{RED}_{C}(\Omega) \), we have that (P3)

\[
\Pi' \theta \in \left( \text{RED}_{B\theta}(\Omega) \Rightarrow \text{RED}_{D\theta}(\Omega) \right)
\]

for every \( \theta \). We need to show that \( \Pi' \rho \delta \in \left( \text{RED}_{B\rho\delta}(\Omega) \Rightarrow \text{RED}_{D\rho\delta}(\Omega) \right) \) for every \( \delta \). Note that by Lemma 5, \( \Pi' \rho \delta = \Pi'(\rho \circ \delta) \), so this is just an instance of Statement 5 above.

• \( \Pi \) ends with \( IR \) or \( CR \): This follows from Definition 13 and the fact that reducibility candidates are closed under substitution (condition CR0 in Definition 13). In the case where \( \Pi \) ends with \( IR \), we also need the fact that eigenvariable substitution commutes with parameter substitution (Lemma 7). In the case where \( \Pi \) ends with \( CR \), to establish \( \Pi_\rho \in \text{RED}_{C_\rho}(\Omega) \), we can use the same reducibility candidate which is used to establish \( \Pi \in \text{RED}_{C}(\Omega) \).

• \( \Pi \) ends with a rule other than \( mc, \supseteq R, IR \) or \( CR \): This case follows straightforwardly from the induction hypothesis.

\[
\square
\]

**Lemma 16**. Let \( \Omega = [\Omega', (R, \Pi_S, S)/X^p] \). Let \( C \) be a formula such that \( X^p \# C \). Then for every \( \Pi, \Pi \in \text{RED}_{C}(\Omega) \) if and only if \( \Pi \in \text{RED}_{C}(\Omega') \).

**Proof.** By induction on \( |C| \) with sub-induction on \( nd(\Pi) \).

Suppose \( C = Y^q \bar{u} \) for some \( Y^q \in \text{supp}(\Omega) \) and suppose \( \Omega(Y^q) = (\mathcal{R}', \Pi_I, I) \). Since \( X^p \# C \), this means that \( Y^q \in \text{supp}(\Omega') \) and \( \Omega'(Y^q) = \Omega(Y^q) \). Then obviously, \( \Pi \in \text{RED}_{C}(\Omega) \) if \( \Pi \in \text{RED}_{C}(\Omega') \). If \( Y^q \notin \text{supp}(\Omega) \), then obviously \( \text{RED}_{C}(\Omega) = \text{NM}_{Y^q} \bar{u} = \text{RED}_{C}(\Omega') \).

Otherwise, suppose \( C \neq Y^q \bar{u} \), and \( \Pi \in \text{RED}_{C}(\Omega) \). The latter implies that \( \Pi \) is normalizable. We show, by induction on \( nd(\Pi) \) that \( \Pi \in \text{RED}_{C}(\Omega') \). In most cases, this follows straightforwardly from the induction hypothesis. We show the interesting cases here:

• Suppose \( \Pi \) ends with \( \supseteq R \), i.e., \( C = B \supseteq D \) for some \( B \) and \( D \) and \( \Pi \) is of the form:

\[
\begin{array}{c}
\Pi' \\
\Gamma, B\Omega \rightarrow D\Omega \\
\Gamma \rightarrow B\Omega \supseteq D\Omega \supseteq R
\end{array}
\]

Note that since \( X^p \# C \), we have that \( B\Omega = B\Omega' \) and \( D\Omega = D\Omega' \). Since \( \Pi \in \text{RED}_{C}(\Omega) \), we have

\[
\Pi' \rho \in \left( \text{RED}_{B\rho}(\Omega) \Rightarrow \text{RED}_{D\rho}(\Omega) \right)
\]

for every \( \rho \). Since \( |B| < |C| \) and \( |D| < |D'| \), by the (outer) induction hypothesis, we have \( \text{RED}_{B\rho}(\Omega) = \text{RED}_{B\rho}(\Omega') \) and \( \text{RED}_{D\rho}(\Omega) = \text{RED}_{D\rho}(\Omega') \). Therefore, we also have that

\[
\Pi' \rho \in \left( \text{RED}_{B\rho}(\Omega') \Rightarrow \text{RED}_{D\rho}(\Omega') \right)
\]

for every \( \rho \). This means, by Definition 15 that \( \Pi \in \text{RED}_{C}(\Omega') \).

• Suppose \( \Pi \) ends with \( IR \):

\[
\begin{array}{c}
\Pi' \\
\Gamma \rightarrow D Y^q \bar{u} \\
\Gamma \rightarrow q \ell \supseteq R
\end{array}
\]

where \( q \bar{x} \overset{P3}{=} D q \bar{x} \) and \( Y^q \) is a new parameter. Since we identify derivations which differ only in the choice of internal variables and parameters, we can assume without loss of generality that \( Y^q \# \Omega \). Note
Lemma 17. Let \( \Omega \) be a candidate substitution and \( F \) a closed term of type \( \alpha_1 \to \cdots \to \alpha_n \to \alpha \). Then the set \( \mathcal{R} = \{ \Pi \mid \Pi \in \text{RED}_{F \theta}[\Omega] \text{ for some } \theta \} \) is a reducibility candidate of type \( F \Omega \).

Proof. Suppose \( F = X^p \) for some \( X^p \in \text{supp}(\Omega) \) and suppose \( \Omega(X^p) = (S, \Pi, F) \). Then in this case, we have \( \mathcal{R} = \mathcal{S} \), so \( \mathcal{R} \) is a reducibility candidate of type \( F \) by assumption. If \( F = X^p \) but \( X^p \not\in \text{supp}(\Omega) \) then in this case \( \mathcal{R} = \text{NM}_{X^p} \), and by Lemma 14 \( \mathcal{R} \) is also a reducibility candidate.

Otherwise, \( F \not= X^p \) for any parameter \( X^p \). We need to show that \( \mathcal{R} \) satisfies CR0 - CR5. CR0 follows from Lemma 14 CR1 follows from Lemma 13 and the rest follow from Definition 15. \( \square \)

that since the body of a definition cannot contain occurrences of parameters, we also have \( X^p \not\in \text{supp}(\Omega) \). Suppose \( S \) is a reducibility candidate of type \( I \), for some closed term \( I \) of the same syntactic type as \( q \), and suppose \( \Pi_I \) is a normalizable derivation of \( D_1 I \bar{y} \to I \bar{y} \).

\[
\Pi_I[\bar{u}\mathbin{\bar{y}}] \in (\text{RED}_{(D_1 Y^q \bar{u})}[\Omega], (S, \Pi_I, I)/Y^q) \Rightarrow S \bar{u})
\]

for every \( \bar{u} \) of the appropriate types. To show that \( \Pi \in (\text{RED}_{C}[\Omega]) \) we need to show that

\[
\text{mc}(\Pi'|[\Pi_I, I)/Y^q], \Pi_I[\bar{u}/\bar{y}]) \in \mathcal{S} \bar{I}
\]

Since \( |(D_1 Y^q \bar{u})| < |p \bar{t}| \) by Lemma 14 we have, by the outer induction hypothesis,

\[
\text{RED}_{(D_1 Y^q \bar{u})}[\Omega], (S, \Pi_I, I)/Y^q] = \text{RED}_{(D_1 Y^q \bar{u})}[\Omega], (S, \Pi_I, I)/Y^q]
\]

Hence, by Statement 9 we also have

\[
\Pi_I[\bar{u}\mathbin{\bar{y}}] \in (\text{RED}_{(D_1 Y^q \bar{u})}[\Omega], (S, \Pi_I, I)/Y^q) \Rightarrow S \bar{u})
\]

for arbitrary \( \bar{u} \). Now since \( \Pi \in (\text{RED}_{C}[\Omega]) \) (from the assumption), this means that

\[
\text{mc}(\Pi'|[\Pi_I, I)/Y^q], \Pi_I[\bar{u}/\bar{y}]) \in \mathcal{S} \bar{I}
\]

and therefore \( \Pi \) is indeed in \( \text{RED}_{C}[\Omega] \).

• Suppose \( \Pi \) ends with \( \text{CIR} \):

\[
\begin{array}{c}
\Pi' \\
\Gamma \to I \bar{t} \quad \Pi_I \quad I \bar{y} \to B \quad I \bar{y} \\
\Gamma \to q \bar{I} \\
\text{CIR}
\end{array}
\]

where \( q \bar{y} \mathbin{\bar{u}} = B q \bar{x} \). Since \( \Pi \in \text{RED}_{C}[\Omega] \), by Definition 14 (P4), there exist a parameter \( Y^q \) such that \( Y^q \not\in \Omega \) and a reducibility candidate \( (S : I) \) such that \( \Pi' \in S \) and

\[
\Pi'[\bar{u}\mathbin{\bar{y}}] \in (S \bar{u} \Rightarrow \text{RED}_{B Y^q \bar{u}}[\Omega], (S, \Pi_I, I)/Y^q])
\]

for every \( \bar{u} \). To show \( \Pi \in \text{RED}_{C}[\Omega'] \) we need to find a reducibility candidate satisfying P4. We simply use \( S \) as that candidate. It remains to show that

\[
\Pi'[\bar{u}\mathbin{\bar{y}}] \in (S \bar{u} \Rightarrow \text{RED}_{B Y^q \bar{u}}[\Omega], (S, \Pi_I, I)/Y^q])
\]

This follows from Statement 10 above and the outer induction hypothesis, since

\[
\text{RED}_{B Y^q \bar{u}}[\Omega], (S, \Pi_I, I)/Y^q] = \text{RED}_{B Y^q \bar{u}}[\Omega', (S, \Pi_I, I)/Y^q]
\]

The converse, i.e., \( \Pi \in \text{RED}_{C}[\Omega] \) implies \( \Pi \in \text{RED}_{C}[\Omega] \), can be proved analogously. In particular, in the case where \( \Pi \) ends with \( \text{CIR} \), we rely on the fact that the choice of the new parameter \( Y^q \) is immaterial, as long as it is new, so we can assume without loss of generality that \( Y^q \not= X^p \). \( \square \)

Lemma 17. Let \( \Omega \) be a candidate substitution and \( F \) a closed term of type \( \alpha_1 \to \cdots \to \alpha_n \to \alpha \). Then the set \( \mathcal{R} = \{ \Pi \mid \Pi \in \text{RED}_{F \theta}[\Omega] \text{ for some } \theta \} \) is a reducibility candidate of type \( F \Omega \).
Lemma 18. Let \( \Omega \) be a candidate substitution and let \( X^p \) be a parameter such that \( X^p \# \Omega \). Let \( S \) be a closed term of the same type as \( p \) and let

\[
\mathcal{R} = \{ \Pi \mid \Pi \in \text{RED}_{S, \bar{u}}[\Omega] \text{ for some } \bar{u} \}.
\]

Suppose \([\Omega, (\mathcal{R}, \Psi, S\Omega)/X^p] \) is a candidate substitution, for some \( \Psi \). Then

\[
\text{RED}_{C[S/X^p]}[\Omega] = \text{RED}_{C[\Omega, (\mathcal{R}, \Psi, S\Omega)/X^p]}.
\]

Proof. By induction on \(|C|\). If \( C = X^p \bar{u} \), then

\[
\text{RED}_{C[\Omega, (\mathcal{R}, \Psi, S\Omega)/X^p]} = \mathcal{R} \bar{u} = \text{RED}_{S, \bar{u}}[\Omega]
\]

by assumption. The other cases where \( C \) is \( Y^q \bar{u} \) for some parameter \( Y^q \neq X^p \) are straightforward. So suppose \( C \neq Y^q \bar{u} \) for any \( \bar{u} \) and any parameter \( Y^q \). We show that for every \( \Pi, \Pi \in \text{RED}_{C[S/X^p]}[\Omega] \) iff \( \Pi \in \text{RED}_{C[\Omega, (\mathcal{R}, \Psi, S\Omega)/X^p]} \). Note that if \( X^p \) does not occur in \( C \) then \( C[S/X^p] = C \), and by Lemma 16 we have

\[
\text{RED}_{C[S/X^p]}[\Omega] = \text{RED}_{C[\Omega]} = \text{RED}_{C[\Omega, (\mathcal{R}, \Psi, S\Omega)/X^p]}.
\]

So assume that \( X^p \) is not vacuous in \( C \). Let \( \Omega' = [\Omega, (\mathcal{R}, \Psi, S\Omega)/X^p] \).

- Suppose \( \Pi \in \text{RED}_{C[S/X^p]}[\Omega] \). Then \( \Pi \) is normalizable. We show, by induction on \( nd(\Pi) \), that \( \Pi \in \text{RED}_{C[\Omega']} \). Most cases follow immediately from the induction hypothesis. The only interesting case is when \( \Pi \) ends with \( \supset \mathcal{R} \), where \( C = B \supset D \), for some \( B \) and \( D \), and \( \Pi \) takes the form:

\[
\Pi' \Gamma, B[S/X^p] \Omega \longrightarrow D[S/X^p] \Omega \supset \mathcal{R}
\]

Since \( \Pi \in \text{RED}_{C[S/X^p]}[\Omega] \), we have that

\[
\Pi' \rho \in (\text{RED}_{B[S/X^p] \rho}[\Omega] \Rightarrow \text{RED}_{D[S/X^p] \rho}[\Omega])
\]

for every \( \rho \). By the outer induction hypothesis (on the size of \( C \)), we have

\[
\Pi' \rho \in (\text{RED}_{B \rho}[\Omega'] \Rightarrow \text{RED}_{D \rho}[\Omega'])
\]

hence \( \Pi \in \text{RED}_{C[\Omega']} \).

- The converse, i.e., \( \Pi \in \text{RED}_{C[\Omega']} \) implies \( \Pi \in \text{RED}_{C[S/X^p]}[\Omega] \), can be proved analogously.

\( \square \)