The Pennsylvania State University
The Graduate School
Department of Computer Science and Engineering

A LOGICAL FRAMEWORK FOR REASONING ABOUT LOGICAL SPECIFICATIONS

A Thesis in
Computer Science and Engineering
by
Alwen F. Tiu

© 2004 Alwen F. Tiu

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2004

The thesis of Alwen F. Tiu was reviewed and approved* by the following:

John Hannan
Associate Professor of Computer Science and Engineering
Thesis Adviser
Chair of Committee

Mahmut Kandemir
Assistant Professor of Computer Science and Engineering

Padma Raghavan
Associate Professor of Computer Science and Engineering

Stephen Simpson
Professor of Mathematics

Frank Pfenning
Professor of Computer Science
Carnegie Mellon University
Special member

Raj Acharya
Professor of Computer Science and Engineering
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

We present a new logic, Linc, which is designed to be used as a framework for specifying and reasoning about operational semantics. Linc is an extension of first-order intuitionistic logic with a proof theoretic notion of *definitions*, induction and co-induction, and a new quantifier ∇. Definitions can be seen as expressing fixed point equations, and the least and greatest solutions for the fixed point equations give rise to the induction and co-induction proof principles. The quantifier ∇ focuses on the intensional reading of $\forall$ and is used to reason about encodings of object systems involving abstractions. The logic Linc allows quantification over λ-terms which makes it possible to reason about encodings involving *higher-order abstract syntax*, a clean and declarative treatment of syntax involving abstraction and substitution. All these features of Linc co-exist within the same logic, allowing for expressing proofs involving induction and co-induction on both first-order and higher-order encodings of operational semantics. We prove the cut-elimination and the consistency results for Linc, extending the reducibility technique due to Tait and Martin-Löf. We illustrate the applications of Linc in a number of areas, ranging from data structures, abstract transition systems, object logic and functional programming. The expressive power of the full logic is demonstrated in the encoding of π-calculus, where we show that the notion of *names* in the calculus can naturally be interpreted in the quantification theory of Linc.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I thank Dale Miller, my former adviser, for his encouragement and guidance during the conduct of the research project presented in this thesis. I have benefited from a number discussions with Catuscia Palamidessi on the subject of π-calculus. Alberto Momigliano has contributed to this work through our joint work on induction and co-induction. He has also provided me with many examples and background literatures on induction and co-induction. I have also benefited from the discussions with Gopalan Nadathur on the reducibility technique used in the cut-elimination proof in this thesis. I thank Jamie Gabbay and Alexis Saurin for their interest in my work and for valuable discussions concerning the ∇ quantifier. I thank John Hannan for accepting the responsibility to be my formal adviser. My former PhD committee members: John Hannan, Vijay Saraswat, Stephen Simpson have provided helpful suggestions in the early phase of this work. Alexis, Dale, John and Frank Pfenning have read and reviewed an earlier draft of this thesis, and made useful suggestions to improve it. I thank Kai Brünnler, Paola Bruscoli, Alessio Guglielmi, Charles Stewart and Lutz Strassburger for the interesting discussions during my visit in Dresden. I am grateful for the support and encouragement I received from my family. My stay in State College and Paris would not have been enjoyable without the company of my friends. I especially thank Bina Gubhaju for her constant support and friendship, and also for her help in editing the thesis.

This work has been funded in part by NSF grants CCR-9912387, INT-9815645, and INT-9815731. My stay in France has been kindly supported by LIX, École polytechnique.

Chapter 1

Introduction

The operational semantics of computation systems are commonly specified as *deductive systems*, in the style of Plotkin's structural operational semantics [46] or Kahn's natural semantics [26]. Computation in this setting is represented as *deduction.* In this representation, it is possible to use the structure of deduction to prove properties of the specified computation system, by structural induction for example. A deductive system is usually given as a set of inference rules, which typically admit simple logical reading. Consider for example the evaluation relation in functional programming language, e.g.,

$$\frac{M \Downarrow \lambda x.P \quad P[N/x] \Downarrow V}{(MN) \Downarrow V}$$

where $P[N/x]$ denotes a term obtained from P by substituting all free occurrences of x in P with N (of course, incidental capturing of free variables in N must be avoided). The reading of the rule is usually described at the meta-language (English for example) as something like "*if* M *evaluates* to $\lambda x.P$ *and* $P[N/x]$ evaluates to V, *then* (MN) evaluates to V". We may formalize [1] this informal specification as the logical theory

$$M \Downarrow \lambda x.P \wedge P[N/x] \Downarrow V \supset (MN) \Downarrow V$$

Given this logical specification, evaluation (computation) of an expression M is interpreted as *proof search* for the formula $\exists V.M \Downarrow V$. We thus can equally represent a deductive system (and hence operational semantics) in logic as *logical theories*, based on which proof search is used to model computation. This idea of using logic to represent computation system is in line with the study of *logical frameworks* [44].

Reasoning about the properties of computation in its encoding in logic can benefit from the structural properties of formal (normal) proofs, compositionality of proofs, and other meta-theories of the specification logic. In this thesis, we introduce a new logic, called Linc (for a logic with λ-terms, induction, ∇ and co-induction), to be used as a framework for specifying and reasoning about operational semantics. Linc is an extension of first-order intuitionistic logic presented in sequent calculus [18]. More specifically, Linc can be seen as an extension of the logic $FO\lambda^{\Delta\mathbb{N}}$ [28]. The main novelties of Linc are the formalization of *induction* and *co-induction* [43] proof principles as explicit proof rules, and the introduction of a new quantifier, ∇, to facilitate reasoning about *generic judgments*, i.e., judgments which involve universal quantifications. The induction rules

[1] Of course, to be fully formal, we need to formalize substitution as well. We leave out the explicit representation of substitution to simplify the illustration.

extend $FO\lambda^{\Delta\mathbb{N}}$'s natural number induction to allow induction proofs over *iterative inductive definitions* [27]. Both induction and co-induction proof principles are essential in reasoning about recursively defined objects, such as fixed points construction in functional programming languages or process calculi, while the ∇ quantifier naturally encodes the dynamics of *names* in computation systems. The notion of names abstracts the use of identifiers in performing computation, such as reference to location in memories or networks, pointers, encryption keys, nonces in security protocols, etc. Underlying these uses of names are the notions of scoping and freshness of names and distinction among names. These notions find natural interpretation in the quantification theory of Linc, in which ∇ plays an essential role.

Our formalization of (co-)induction is actually based on a proof theoretic notion of *definition*, following on work by Schroeder-Heister [50], Eriksson [14], Girard [21], Stärk [55] and McDowell and Miller [29]. Definitions are essentially logic programs which are stratified so as to avoid cyclic calling through negations. Deductive systems can alternatively be encoded as definitions. The introduction rules for defined atoms treat the definitions as "closed" or defining fixed points. This alone gives us the ability to perform *case analyses*, which can be used for reasoning about the *must-behaviors* of computation systems, i.e., properties which are true for all computation paths. Our approach to formalizing induction and co-induction is via the least and greatest solutions of the fixed equations specified by the definitions. Such least and greatest solutions are guaranteed to exist by the stratification on definitions (which are basically some monotonicity condition). The proof rules for the induction and co-induction makes use of the notion of *pre-fixed points* and *post-fixed points* [43], respectively. In the inductive case, this is simply the induction invariant, while in the co-inductive one it corresponds to the so-called *simulation*.

Another important feature of Linc is that it allows quantification over λ-terms which makes it possible to support *higher-order abstract syntax* (HOAS). Higher-order abstract syntax is a declarative treatment of syntax involving abstraction and substitution. The idea of HOAS is to use λ-abstraction to encode object-level (computation system) abstraction and to use β-reduction to encode substitution. Linc has a built-in equality predicate whose introduction rules make use of unification and matching, on both first-order and higher-order λ-terms. The full system with ∇ and (co-)induction allows for expressing a rich class of inductive and co-inductive proofs involving higher-order abstract syntax, where the notion of freshness and scoping of variables plays an essential role.

1.1 A logical treatment of names and abstractions

Part of the motivation in having a new quantifier, ∇, in our framework is that the existing ones are not expressive enough for encoding certain uses of abstractions in operational semantics. Consider for example the problem of encoding the π-calculus [40]. The static structure of the abstractions in the π-calculus is encoded in Linc as λ-terms, following the encoding style of higher-order abstract syntax. Name-binding operator in this object system is mapped to function symbol which takes as argument an abstraction.

Consider the following expressions in π-calculus:

$$(x)P \qquad \text{and} \qquad a(x).P,$$

where (x) and $a(x)$ are the binding operators and P is a process expression. The binder (x) is called restriction and $a(x)$ is the input prefix. To represent these terms in Linc, we first introduce the types n and p corresponding to the syntactic categories of names and process expressions in π-calculus. We then define the constants

$$\nu : (n \to p) \to p \qquad \mathit{in} : n \to (n \to p) \to p.$$

The above process expressions are encoded as the meta-level terms $\nu\lambda x.P$ and $\mathit{in}\ a\ \lambda x.P$. The use of higher-order abstract syntax produces a better reading of the inference rules in π-calculus. In particular, complicated side conditions disappear from the description of the rules, as they are now part of the description of meta-level syntax (see Chapter 6 for details).

The input prefix and the restriction operator have different operational meaning. An input-prefixed process $x(y).P$ can evolve into the process $P[w/y]$ for any name w, while the process $(y)P$ can evolve to $P[a/y]$ for some "fresh" name a. These different dynamics of names are captured in Linc by the use of $\forall$ (to encode "for all names") and ∇ (to encode the freshness of names). A common approach to encode fresh-name generation is via the use of $\forall$-quantifier. The use of universal quantification has been considered in the *extended natural semantics* [23]. The operational behavior of this encoding in proof search is given by *eigenvariable*, which provides a fresh, scoped constant in proof search. Eigenvariable has been used in the encoding of restrictions in the π-calculus [34], nonces in security protocols [6], reference locations in imperative programming [8, 35], and constructors hidden within abstract data-types [32].

The $\forall$-encoding is adequate as long as we are concerned only with encoding computations but not with proving properties about the computations. In the latter case, the use of $\forall$ for fresh name generation is problematic. To see why, suppose we are given a computation $P\,x\,y$ involving two fresh names x and y. In the $\forall$-encoding, we would have a proof for $\forall x \forall y.P\,x\,y$. However, since $\forall x \forall y.P\,x\,y \supset \forall z.P\,z\,z$ is a theorem of logic, we would have another computation in which the names x and y are identified. Let us refer to this theorem as the *diagonal property* of $\forall$. Of course, this might not be the intended meaning of the fresh names in the object system. One key idea in designing ∇ is that the diagonal property should fail for ∇, that is, $\nabla x \nabla y.P\,x\,y \supset \nabla z.P\,z\,z$ should not be a theorem in Linc. Thus, ∇, by design, does not entails $\forall$. That is, the extensionality property, $\nabla x.Bx \supset B\,t$, for any term t, does not hold either. This suggests that ∇ is suitable for encoding object systems with fresh name generation, where in the course of computation this fresh name stays unchanged, e.g., the ν-binder in π-calculus and its variants, and is less suitable for encoding abstraction which implies certain extensionality behavior, e.g., the λ-binder in λ-calculus. The latter implies that Linc does not directly capture fully the induction over higher-order abstract syntax, although it can still be done via an indirect approach as it was done in $FO\lambda^{\Delta\mathbb{N}}$.

1.2 Outline of the thesis

The remainder of this thesis is organized as follows. Chapter 2 presents the logic Linc in an incremental fashion, starting from its core fragment, which is basically Church's Simple Theory of Types with the quantifier ∇, followed by the introduction of definitions and induction and co-induction. Chapter 3 presents the meta-theories of the logic, some of which are used in the proof of cut-elimination in Chapter 4. Chapter 4 presents the central result of this thesis, that is, the cut-elimination proof for Linc. The consistency of Linc is shown to be a simple corollary of cut-elimination. Our proof of cut-elimination extends the cut-elimination proof of $FO\lambda^{\Delta\mathbb{N}}$ which makes use of the technique of reducibility due to Tait [56]. Chapter 5 illustrates the applications of Linc in a number of areas, ranging from reasoning about (co-)inductive data structures (natural numbers, finite lists and lazy lists), abstract transition systems, object logic and functional language (the lazy λ-calculus). These examples illustrate the use of both induction and co-induction rules, in the first-order and higher-order encodings. The application of ∇, with induction and co-induction, is demonstrated in the encoding of π-calculus in Chapter 6. In this chapter we show that the quantifiers in Linc capture the treatment of names in π-calculus, as illustrated in the encoding of *strong late bisimulation* and *strong late congruence*. Chapter 7 summarizes the thesis and discusses possible future work.

Chapter 2

The Logic Linc

In this chapter we define formally the sequent calculus of the logic Linc. Linc is an extension of Church's Simple Theory of Types with a new quantifier ∇, a proof theoretic notion of *definitions* and induction and co-induction proof principles. The quantifier ∇ focuses on the intensional reading of $\forall$, i.e., to provide fresh, scoped constants in proof search. To incorporate this new quantifier, it is necessary to consider a richer context in sequents. Section 2.1 shows how this can be done and gives the core rules of Linc. Section 2.2 formalizes the notion of definitions and Section 2.3 shows how to extend the notion of definitions to define the induction and co-induction rules. Section 2.4 shows some derived rules which are useful later in applications. Section 2.5 reviews some related work.

2.1 Intuitionistic logic with generic judgments

The usual intuitionistic sequent is of the form

$$\Sigma\,;\, B_1, \ldots, B_n \longrightarrow B_0$$

where Σ is the (global) signature of the sequent, containing the eigenvariables, and $B_0, \ldots, B_n$ are formulas. We can view sequent of this form as a binding structure, that is, the free variables in the formulas in the sequent are bound by Σ. We consider introducing a new layer of abstraction into the sequent, whose scope is local to each formula. This additional layer of abstraction is motivated by applications in reasoning about object systems involving abstractions [28]. The enriched sequent takes the form

$$\Sigma\,;\, \sigma_1 \triangleright B_1, \ldots, \sigma_n \triangleright B_n \longrightarrow \sigma_0 \triangleright B_0,$$

where each σ_i is a *local signature*, i.e., a list of variables locally scoped over the formula. The dynamic of the local signature is provided by a new quantifier, ∇, in much the same way as the dynamic of global signature (eigenvariables) is provided by $\forall$.

Before we proceed with the inference rules for Linc, let us go through some notational convention. Variables in both global and local signatures are simply typed, following the type system of Church. That is, types are built from *base types* and the binary type constructor $\rightarrow$. The variables can have any types as long as they do not contain the type o, which is reserved for the type of formulas. We refer to types which do not contain o as *object types*, and variables and constants of object types are sometimes referred to as object variables and object constants, respectively. Thus variables in signatures are of object types. Types will be denoted by α, β, and τ, with or without indexes. The typing judgment is of the form $\Sigma \vdash t : \tau$, where Σ is the type context (i.e., set of type variables). Here we use Church's type system for simply typed λ terms. If we need to be explicit about the type of a variable, like when describing signatures, the type information will be written after the variable, separated by colon, e.g., $x : \tau$. Type annotation will often be dropped when it is clear from context which type a variable has, or when the type is irrelevant to the context of discussion.

Given a list of typed variables $\bar{x} = x_1 : \alpha_1, \dots, x_n : \alpha_n$ we sometimes wish to extract just the type information, in which case we would write $\hat{\tau}(\bar{x})$ to denote the list of types $\alpha_1, \dots, \alpha_n$. In addition to signatures (which are basically variables), we shall also assume a set of object-typed constants in proof-construction. Since the set of constants does not grow during proof construction, it will not be written explicitly in the sequent. We assume that the set of variables and constants are always distinct, i.e., there is no accidental use of the same name for denoting both variable and constant.

We shall consider sequents to be binding structures in the sense that the signatures, both the global and local ones, are abstractions over their respective scopes. The variables in Σ and σ will admit α-conversion by systematically changing the names of variables in signatures as well as those in their scope, following the usual convention of the λ-calculus. In general, however, we will assume that the local signatures, σ, contain names different than those in the global signature Σ. The expression $\sigma \triangleright B$ is called a *generic judgment*, or simply judgment. To simplify the presentation, we shall often write a judgment $(x, y, z) \triangleright B$ as $xyz \triangleright B$ where xyz denotes the list x, y, z. Equality between judgments follows from the notion of equality of λ-terms, that is, two judgments $\bar{x} \triangleright B$ and $\bar{y} \triangleright C$ are equal if and only if

$$\lambda \bar{x}.B =_{\beta\eta} \lambda \bar{y}.C.$$

We use script letters $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$ etc. to denote judgments, and the letter $\mathcal{S}$ to denote sequents. We write simply B instead of $\sigma \triangleright B$ if the signature σ is empty.

The core fragment of Linc consists of introduction rules for the logical constants $\wedge, \vee, \supset, \bot, \top, \forall, \exists$ and ∇ and the structural rules, initial rule and the cut rule. These inference rules are given in Figure 2.1. The introduction rules for ∇ affect only the local signatures. The variable y in the ∇ introduction rules follows the usual proviso that it is not free in $\lambda x.B$. The interaction between local and global signatures takes place in $\exists$ and $\forall$ introduction rules. The quantifiers are of type $(\alpha \to o) \to o$, where α is an object type. Thus the logic is in a sense first-order. In the $\forall\mathcal{R}$ (likewise, $\exists\mathcal{L}$), we use raising [33] to denote that the bound variable x can range over the variables in the global signature as well as the local signature σ. Here the variable h must not be free in the lower sequent. In these rules, we use the notation $h\,\sigma$ to denote the successive application of the variable h to the variables in σ. That is, suppose σ is the list $x_1, \dots, x_n$, then $h\,\sigma$ is the term $(h\,x_1 \dots x_n)$. In $\forall\mathcal{L}$ and $\exists\mathcal{R}$, the term t can have free variables from both Σ and σ. The multicut rule *mc* is a generalization of the cut rule due to Slaney [53] to simplify the presentation of cut-elimination proof. Notice that if we remove the ∇ rules and the local signatures from the sequents, then what we have is the usual intuitionistic logic.

Note that the use of raising in $\forall\mathcal{R}$ and $\exists\mathcal{L}$ means that the eigenvariable introduced (reading the rules bottom-up) might not be of the same type as the quantified variable.

$$\frac{}{\Sigma\,;\,\mathcal{C},\Gamma \longrightarrow \mathcal{C}}\;\mathit{init} \qquad \frac{}{\Sigma\,;\,\sigma \rhd \bot,\Gamma \longrightarrow \mathcal{B}}\;\bot\mathcal{L} \qquad \frac{}{\Sigma\,;\,\Gamma \longrightarrow \sigma \rhd \top}\;\top\mathcal{R}$$

$$\frac{\Sigma\,;\,\sigma \rhd B,\Gamma \longrightarrow \mathcal{D}}{\Sigma\,;\,\sigma \rhd B \wedge C,\Gamma \longrightarrow \mathcal{D}}\;\wedge\mathcal{L} \qquad \frac{\Sigma\,;\,\sigma \rhd C,\Gamma \longrightarrow \mathcal{D}}{\Sigma\,;\,\sigma \rhd B \wedge C,\Gamma \longrightarrow \mathcal{D}}\;\wedge\mathcal{L}$$

$$\frac{\Sigma\,;\,\Gamma \longrightarrow \sigma \rhd B \quad \Sigma\,;\,\Gamma \longrightarrow \sigma \rhd C}{\Sigma\,;\,\Gamma \longrightarrow \sigma \rhd B \wedge C}\;\wedge\mathcal{R}$$

$$\frac{\Sigma\,;\,\sigma \rhd B,\Gamma \longrightarrow \mathcal{D} \quad \Sigma\,;\,\sigma \rhd C,\Gamma \longrightarrow \mathcal{D}}{\Sigma\,;\,\sigma \rhd B \vee C,\Gamma \longrightarrow \mathcal{D}}\;\vee\mathcal{L}$$

$$\frac{\Sigma\,;\,\Gamma \longrightarrow \sigma \rhd B}{\Sigma\,;\,\Gamma \longrightarrow \sigma \rhd B \vee C}\;\vee\mathcal{R} \qquad \frac{\Sigma\,;\,\Gamma \longrightarrow \sigma \rhd C}{\Sigma\,;\,\Gamma \longrightarrow \sigma \rhd B \vee C}\;\vee\mathcal{R}$$

$$\frac{\Sigma\,;\,\Gamma \longrightarrow \sigma \rhd B \quad \Sigma\,;\,\sigma \rhd C,\Gamma \longrightarrow \mathcal{D}}{\Sigma\,;\,\sigma \rhd B \supset C,\Gamma \longrightarrow \mathcal{D}}\;\supset\mathcal{L} \qquad \frac{\Sigma\,;\,\sigma \rhd B,\Gamma \longrightarrow \sigma \rhd C}{\Sigma\,;\,\Gamma \longrightarrow \sigma \rhd B \supset C}\;\supset\mathcal{R}$$

$$\frac{\Sigma,\sigma \vdash t : \tau \quad \Sigma\,;\,\sigma \rhd B[t/x],\Gamma \longrightarrow \mathcal{C}}{\Sigma\,;\,\sigma \rhd \forall_\tau x.B,\Gamma \longrightarrow \mathcal{C}}\;\forall\mathcal{L} \qquad \frac{\Sigma,h\,;\,\Gamma \longrightarrow \sigma \rhd B[(h\ \sigma)/x]}{\Sigma\,;\,\Gamma \longrightarrow \sigma \rhd \forall x.B}\;\forall\mathcal{R}$$

$$\frac{\Sigma,h\,;\,\sigma \rhd B[(h\ \sigma)/x],\Gamma \longrightarrow \mathcal{C}}{\Sigma\,;\,\sigma \rhd \exists x.B,\Gamma \longrightarrow \mathcal{C}}\;\exists\mathcal{L} \qquad \frac{\Sigma,\sigma \vdash t : \tau \quad \Sigma\,;\,\Gamma \longrightarrow \sigma \rhd B[t/x]}{\Sigma\,;\,\Gamma \longrightarrow \sigma \rhd \exists_\tau x.B}\;\exists\mathcal{R}$$

$$\frac{\Sigma\,;\,(\sigma,y) \rhd B[y/x],\Gamma \longrightarrow \mathcal{C}}{\Sigma\,;\,\sigma \rhd \nabla x.B,\Gamma \longrightarrow \mathcal{C}}\;\nabla\mathcal{L} \qquad \frac{\Sigma\,;\,\Gamma \longrightarrow (\sigma,y) \rhd B[y/x]}{\Sigma\,;\,\Gamma \longrightarrow \sigma \rhd \nabla x.B}\;\nabla\mathcal{R}$$

$$\frac{\Sigma\,;\,\mathcal{B},\mathcal{B},\Gamma \longrightarrow \mathcal{C}}{\Sigma\,;\,\mathcal{B},\Gamma \longrightarrow \mathcal{C}}\;c\mathcal{L} \qquad \frac{\Sigma\,;\,\Gamma \longrightarrow \mathcal{C}}{\Sigma\,;\,\mathcal{B},\Gamma \longrightarrow \mathcal{C}}\;\mathrm{w}\mathcal{L}$$

$$\frac{\Sigma\,;\,\Delta_1 \longrightarrow \mathcal{B}_1 \quad \cdots \quad \Sigma\,;\,\Delta_n \longrightarrow \mathcal{B}_n \quad \Sigma\,;\,\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{C}}{\Sigma\,;\,\Delta_1,\ldots,\Delta_n,\Gamma \longrightarrow \mathcal{C}}\;mc\text{, where } n \geq 0$$

Fig. 2.1. The core rules of Linc

This is illustrated in the following example.

$$\dfrac{\dfrac{\{x:\alpha, h:\tau\to\gamma\to\beta\}\,;\,\Gamma \longrightarrow (a:\tau, b:\gamma)\rhd B\ (h\,a\,b)\ b}{\{x:\alpha\}\,;\,\Gamma \longrightarrow (a:\tau,b:\gamma)\rhd \forall_{\beta} y.B\ y\ b}\ \forall\mathcal{L}}{\{x:\alpha\}\,;\,\Gamma\longrightarrow (a:\tau)\rhd \nabla_{\gamma} z.\forall_{\beta} y.B\ y\ z}\ \nabla\mathcal{R}$$

Notice that the quantified variable y is of type β while its corresponding eigenvariable h is raised to the type $\tau \to \gamma \to \beta$, taking into account its dependency on $a : \tau$ and $b : \gamma$.

Below are some theorems of Linc involving ∇ which can be proved by inspection on the inference rules in Figure 2.1. In these formulas, we use $\neg C$ to abbreviate $C \supset \bot$ and we write $B \equiv C$ to denote $(B \supset C) \wedge (C \supset B)$.

Proposition 2.1. *The following formulas are provable in the core* Linc*.*

$$\begin{array}{cc}
\nabla x \neg Bx \equiv \neg \nabla x Bx & \nabla x(Bx \wedge Cx) \equiv \nabla x Bx \wedge \nabla x Cx \\
\nabla x(Bx \vee Cx) \equiv \nabla x Bx \vee \nabla x Cx & \nabla x(Bx \supset Cx) \equiv \nabla x Bx \supset \nabla x Cx \\
\nabla x \forall y Bxy \equiv \forall h \nabla x Bx(hx) & \nabla x \exists y Bxy \equiv \exists h \nabla x Bx(hx) \\
\nabla x \forall y Bxy \supset \forall y \nabla x Bxy & \nabla x \top \equiv \top, \quad \nabla x \bot \equiv \bot
\end{array}$$

As a consequence, ∇ can always be given scope over atomic formulas with the cost of raising the types of quantified variables in its scope. Operationally speaking, ∇ provides a declarative way of managing scoping. Below are some non-theorems of Linc involving ∇.

$$\begin{array}{cl}
\nabla x \nabla y Bxy \supset \nabla z Bzz & \nabla x Bx \supset \exists x Bx \\
\nabla z Bzz \supset \nabla x \nabla y Bxy & \nabla x Bx \supset \forall x Bx \\
\forall y \nabla x Bxy \supset \nabla x \forall y Bxy & \exists x Bx \supset \nabla x Bx \\
\nabla x B \equiv B,\ x \text{ not free in } B & \forall x Bx \supset \nabla x Bx \\
\nabla x \nabla y Bxy \equiv \nabla y \nabla x Bxy &
\end{array}$$

A notational convention

In the following sections, we often need to perform substitution of terms or predicates in formulas, e.g., in describing inference rules or other meta-theoretical properties. To simplify the meta-theoretical discussions, we adopt the following convention: when a scheme of term is written in the form

$$M\,x_1\,\ldots\,x_n$$

where $x_1, \ldots, x_n$ are variables, we mean that $x_1, \ldots, x_n$ are not free in any instantiation of M. For example, under this convention the following formulas

$$\forall x.p\,x\,y \qquad (p\,y \wedge q\,y) \qquad (p\,y \supset (r\,z \vee q\,z))$$

are all instances of the scheme $M\,y$. The corresponding instantiations of M are

$$\lambda y.\forall x.p\,x\,y \qquad \lambda y.(p\,y \wedge q\,y) \qquad \lambda y.(p\,y \supset (r\,z \vee q\,z)),$$

respectively. Substitution of a term s for y in this representation is then simply denoted by (the normal form of) $M\,s$.

We still keep the informal (first-order) syntax we used previously, which would allow us to do variable-capture instantiation in certain cases. For example, when we write down the formula scheme

$$\forall x.B$$

we still mean that B can be instantiated with a formula containing x. In such cases, substitution will be written explicitly, as in the inference figure in Figure 2.1 for example. However, in the scheme $\forall x.B\,x$ capture-avoiding instantiation is assumed.

We shall also use the above convention for abstracting predicate symbols in a given formula or judgment whenever it is desired. For example, we would sometimes write a formula scheme of the form $B\,p\,\bar{x}$. Here the scheme variable B is an abstraction with arguments p and $\bar{x}$. With this scheme, instantiation of B must not contain free occurrences of the predicate p or the variables $\bar{x}$. Similar convention applies to judgments. For example, we will use the notation $\mathcal{C}\,p$ to denote a judgment, where $\mathcal{C}$ is now an abstraction over judgments. Using this notation we can express predicate substitution as β-reduction, e.g., $\mathcal{C}\,S$ denotes the judgment $\mathcal{C}\,p$ with all occurrences of p replaced by S.

Notice that this notational convention is rather similar to higher-order abstract syntax (HOAS), in particular, the idea of using β-reduction to express substitution. However, there is one main difference between this syntax and HOAS: using this convention we cannot express the vacuousness of occurrence of variables in a term directly in the syntax. That is, $\forall x.B$ does not indicate that x occurs vacuously in B. In such cases we shall indicate explicitly whether or not a variable occurs vacuously in a term. To avoid confusion with HOAS, it is best to keep in mind that this convention is just a way of avoiding writing substitutions explicitly at the meta-level.

2.2 A proof theoretic notion of definitions

We extend the core logic in Figure 2.1 by allowing the introduction of non-logical constants. An atomic formula, i.e., a formula that contains no occurrences of logical constants, can be defined in terms of other logical or non-logical constants. Its left and right rules are, roughly speaking, carried out by replacing the formula corresponding to its definition with the atom itself. A defined atom can thus be seen as a generalized connective, whose behaviour is determined by its defining clauses.

The syntax of definition clauses used by McDowell and Miller [29] resembles that of logic programs, that is, a definition clause consists of a head and a body, with the usual pattern matching in the head; for example, the predicate *nat* for natural numbers is written

$$\{nat\,z \stackrel{\triangle}{=} \top,\ nat\,s\,x \stackrel{\triangle}{=} nat\,x\}.$$

We adopt a simpler presentation by putting all pattern matching in the body and combining multiple clauses with the same head in one clause with disjunctive body. Of course, this will require us to have explicit equality as part of our syntax. The corresponding *nat* predicate in our syntax will be written

$$nat\ x \stackrel{\triangle}{=} [x = z] \lor \exists y.[x = s\ y] \land nat\ y$$

and corresponds to the notion of *iff-completion* of a logic program [9]. Adopting this form of syntax will help simplifying the meta-theoretical discussions to follow. However, in doing examples and application, we shall use the more familiar syntax of logic program, which we shall show at the end of this chapter to be derivable from the syntax with explicit equations.

Definition 2.2. *A* definition clause *is written* $\forall\bar{x}[p\,\bar{x} \stackrel{\triangle}{=} B\,\bar{x}]$*, where* p *is a predicate constant. The atomic formula* $p\,\bar{x}$ *is called the* head *of the clause, and the formula* $B\,\bar{x}$ *is called the* body*, whose free variables are in* $\bar{x}$*. The symbol* $\stackrel{\triangle}{=}$ *is used simply to indicate a definition clause: it is not a logical connective. A* definition *is a (perhaps infinite) set of definition clauses. A predicate may occur at most once in the heads of the clauses of a definition.*

We will generally omit the outer quantifiers in a definition clause to simplify the presentation.

Not all definition clauses are admitted in our logic, e.g., definitions with circular calling through implications (negations) must be avoided. The reason for this restriction is that without it cut-elimination does not hold [49]. We introduce the notion of *levels* of a formula to define a proper stratification on definitions. To each predicate p we associate a natural number $\mathrm{lvl}(p)$, the level of p. The notion of level is extended to formulas, judgments, sequents and derivations.

Definition 2.3. *Given a formula* B*, its* level $\mathrm{lvl}(B)$ *is defined as follows:*

1. $\mathrm{lvl}(p\,\bar{t}) = \mathrm{lvl}(p)$
2. $\mathrm{lvl}(\bot) = \mathrm{lvl}(\top) = 0$
3. $\mathrm{lvl}(B \land C) = \mathrm{lvl}(B \lor C) = \max(\mathrm{lvl}(B), \mathrm{lvl}(C))$
4. $\mathrm{lvl}(B \supset C) = \max(\mathrm{lvl}(B) + 1, \mathrm{lvl}(C))$
5. $\mathrm{lvl}(\forall x.B) = \mathrm{lvl}(\exists x.B) = \mathrm{lvl}(\nabla x.B) = \mathrm{lvl}(B)$.

The level of a judgment $\sigma \triangleright B$ *is the level of* B*, the level of a sequent* $\Sigma\,;\Gamma \longrightarrow \mathcal{C}$ *is the level of* $\mathcal{C}$ *and the level of a derivation* Π *is the level of its end sequent. A definition clause* $\forall\bar{x}[p\,\bar{x} \stackrel{\triangle}{=} B]$ *is stratified if* $\mathrm{lvl}(B) \leq \mathrm{lvl}(p)$*. A definition is stratified if all its definition clauses are stratified.*

An occurrence of a formula A in a formula C is *strictly positive* if that particular occurrence of A is not to the left of any implication in C. We see that the stratification of definitions above implies that for every definition clause all occurrences of the head in the body are strictly positive.

Given a definition clause $p\,\bar{x} \overset{\triangle}{=} B\,\bar{x}$, the right and left introduction rules for predicate p are

$$\frac{\Sigma\,;\,\sigma \triangleright B\,\bar{t}, \Gamma \longrightarrow \mathcal{C}}{\Sigma\,;\,\sigma \triangleright p\,\bar{t}, \Gamma \longrightarrow \mathcal{C}}\ \mathit{def}\mathcal{L} \qquad \frac{\Sigma\,;\,\Gamma \longrightarrow \sigma \triangleright B\,\bar{t}}{\Sigma\,;\,\Gamma \longrightarrow \sigma \triangleright p\,\bar{t}}\ \mathit{def}\mathcal{R}\ .$$

The rules for equality predicates makes use of substitutions. The notion of substitutions we use here is the usual capture-avoiding substitution. We recall some basic definitions related to substitutions. A *substitution* θ is a mapping (with application written in postfix notation) from variables to terms, such that the set $\{x \mid x\theta \neq x\}$ is finite. Substitutions are denoted by $\theta, \rho, \delta, \gamma$, with or without subscripts. We assume implicitly that substitutions are well-typed. That is, given a substitution θ and $\Sigma \vdash t : \alpha$, the substitution θ is applicable to t only if $\Sigma\theta \vdash t\theta : \alpha$. Here the signature $\Sigma\theta$ denotes the signature that results from removing from Σ the variables in the domain of θ and adding the variables that are free in the range of θ. Of course, any new variables introduced in $\Sigma\theta$ is also assumed to be typed appropriately beforehand.

Substitutions are extended to mappings from terms to terms in the usual fashion. But when we refer to the domain and the range of a substitution, we refer to those sets defined on this most basic function. Composition of substitutions is defined as $x(\theta \circ \rho) = (x\theta)\rho$, for every variable x. Two substitutions θ and ρ are considered equal if for all variables x, $x\rho =_\eta x\theta$ (equal modulo η-conversion). The empty substitution is written as ϵ. The application of a substitution θ to a generic judgment $x_1, \ldots, x_n \triangleright B$, written as $(x_1, \ldots, x_n \triangleright B)\theta$, is $y_1, \ldots, y_n \triangleright B'$, if $(\lambda x_1 \ldots \lambda x_n.B)\theta$ is equal (modulo λ-conversion) to $\lambda y_1 \ldots \lambda y_n.B'$. If Γ is a multiset of generic judgments, then $\Gamma\theta$ is the multiset $\{\mathcal{D}\theta \mid \mathcal{D} \in \Gamma\}$.

The left and right introduction rules for equality are as follows

$$\frac{\{\Sigma\rho\,;\,\Gamma\rho \longrightarrow \mathcal{C}\rho \ \mid\ (\lambda\bar{x}.s)\rho =_{\beta\eta} (\lambda\bar{x}.t)\rho\}}{\Sigma\,;\,\bar{x} \triangleright (s = t), \Gamma \longrightarrow \mathcal{C}}\ \mathrm{eq}\mathcal{L} \qquad \frac{}{\Sigma\,;\,\Gamma \longrightarrow \bar{x} \triangleright t = t}\ \mathrm{eq}\mathcal{R}$$

The substitution ρ in eq$\mathcal{L}$ is called a *unifier* of s and t. In general there can be more than one unifier for a given pair of terms. We therefore specify a set of sequents as the premise of the eq$\mathcal{L}$ rule to mean that each sequent in the set is a premise of the rule. This set can be infinite since there can potentially be infinite numbers of unifiers for a given pair of terms, and hence the eq$\mathcal{L}$ rule is in general not effective. However, in most cases we need only to consider a finite subset of the unifiers in proof search. We shall return to this in Section 2.4. Note that in applying eq$\mathcal{L}$, eigenvariables can be instantiated as a result.

2.3 Induction and co-induction

A definition $p\,x \overset{\triangle}{=} B\,p\,x$ expresses logical equivalence between $p\,x$ and $B\,p\,x$. This means that $p\,t$ is provable for some term t if and only if $B\,p\,t$ is provable. Let us collect all term t such that $p\,t$ is provable, and call it $\mathcal{S}$, and let F be a function from sets of terms to sets of terms:

$$F(\mathcal{T}) = \{s \mid s \in \mathcal{T} \text{ and } B\,p\,s \text{ is provable }\}.$$

Then $\mathcal{S}$ is a fixed point of F, that is, $F(\mathcal{S}) = \mathcal{S}$. To see why this holds, take any $s \in \mathcal{S}$, then by logical equivalence $B\,p\,s$ is provable, so $s \in F(\mathcal{S})$. Conversely, for any $s \in F(\mathcal{S})$, $B\,p\,s$ is provable and again by logical equivalence $p\,s$ is provable and hence is in $\mathcal{S}$. In this sense, the definition clause for p encodes a fixed point equation and provability of $p\,t$ means that t is in some solution of the fixed point equation. If we take into account local signatures, the definition of F would be

$$F(\mathcal{T}) = \{\lambda\bar{x}.s \mid \lambda\bar{x}.s \in \mathcal{T} \text{ and } \bar{x} \triangleright B\,p\,s \text{ is provable }\},$$

and $\mathcal{S}$ would be the set $\{\lambda\bar{x}.s \mid \bar{x} \triangleright p\,s \text{ is provable}\}$. With similar arguments we can show that $\mathcal{S}$ is a fixed point of F. Note that there is no ambiguity with respect to the extra λ's since the type of s is fixed.

The introduction rules for definition in the previous section captures only arbitrary fixed points, not necessarily the least nor the greatest ones. (see e.g., [21] for an example). Therefore we add extra rules that reflect the least and the greatest solutions, and hence capturing induction and co-induction in our system. Note that stratification of definitions guarantee that the associated fixed point operator is monotone, and hence we can use the notion of *pre-fixed point* and the *post-fixed point* to approximate the least and greatest fixed points, respectively. In the following we assume that definitions are not mutual-recursively defined. We could deal with mutual recursion directly but we avoid to do so for clarity of presentation. The results and the technical proofs to follow can be generalized to the system with mutual recursion.

Let $p\,\bar{x} \stackrel{\triangle}{=} B\,p\,\bar{x}$ be a definition clause and let S be a closed term of the same type as p. The induction rules for p are

$$\frac{\bar{x}\,;\, B\,S\,\bar{x} \longrightarrow S\,\bar{x} \quad \Sigma\,;\,\Gamma, \bar{y} \triangleright S\,\bar{t} \longrightarrow \mathcal{C}}{\Sigma\,;\,\Gamma, \bar{y} \triangleright p\,\bar{t} \longrightarrow \mathcal{C}}\ \mu\mathcal{L} \qquad \frac{\Sigma\,;\,\Gamma \longrightarrow \bar{y} \triangleright B\,p\,\bar{t}}{\Sigma\,;\,\Gamma \longrightarrow \bar{y} \triangleright p\,\bar{t}}\ \mu\mathcal{R}$$

The closed term S, which is of the same type as p, is an *invariant* of the induction. The variables $\bar{x}$ are new eigenvariables. We can consider S as encoding a set denoting a pre-fixed point. Notice that the right-rule for induction is *def*$\mathcal{R}$.

The co-induction rules are defined dually.

$$\frac{\Sigma\,;\,\bar{y} \triangleright B\,p\,\bar{t}, \Gamma \longrightarrow \mathcal{C}}{\Sigma\,;\,\bar{y} \triangleright p\,\bar{t}, \Gamma \longrightarrow \mathcal{C}}\ \nu\mathcal{L}$$

$$\frac{\Sigma\,;\,\Gamma \longrightarrow \bar{y} \triangleright S\,\bar{t} \quad \bar{x}\,;\, S\,\bar{x} \longrightarrow (B\,S\,\bar{x})}{\Sigma\,;\,\Gamma \longrightarrow \bar{y} \triangleright p\,\bar{t}}\ \nu\mathcal{R}, \text{ where } \mathrm{lvl}(S) \leq \mathrm{lvl}(p)$$

Dual to the induction rules, S can be seen as denoting a post-fixed point. The $\nu\mathcal{L}$ rule is the *def*$\mathcal{L}$ rule. The reason for the proviso in $\nu\mathcal{R}$ is mainly technical; it allows us to prove cut-elimination. We do not think that the proviso is essential (in the sense that inconsistency might result from its removal), but so far we have not encountered co-inductive proofs which strictly require co-inductive invariants that violate this proviso.

To avoid inconsistency, some care must be taken in applying induction or co-induction in a proof. One obvious pitfall is when the fixed point equation corresponding to a definition clause has different least and greatest solutions. In such case, mixing induction and co-induction on the same definition clause can lead to inconsistency. For example, let $p \stackrel{\triangle}{=} p$ be a definition clause. Given the scheme of rules above without any further restriction, we can construct the following derivation

$$\dfrac{\dfrac{\dfrac{}{\longrightarrow \top}\,\top\mathcal{R} \qquad \dfrac{}{\top \longrightarrow \top}\,\top\mathcal{R}}{\longrightarrow p}\,\nu\mathcal{R} \qquad \dfrac{\dfrac{}{\bot \longrightarrow \bot}\,\bot\mathcal{L} \qquad \dfrac{}{\bot \longrightarrow \bot}\,\bot\mathcal{L}}{p \longrightarrow \bot}\,\mu\mathcal{L}}{\longrightarrow \bot}\,cut$$

In the above derivation we use $\top$ and $\bot$ as the invariants in the instances of $\nu\mathcal{R}$ and $\mu\mathcal{L}$ rules. This example suggests that we have to use a definition clause consistently through out the proof, either inductively or co-inductively, but not both. To avoid this problem, we introduce markings into a definition, whose role is to indicate which introduction rules are applicable to the corresponding defined atoms.

Definition 2.4. *An* extended definition *is a definition* **D** *together with a label on each definition clause in* **D**. *The label on a clause indicates whether the clause is either* inductive*,* co-inductive*, or* regular*. An inductive clause is written as* $p\,\bar{x} \stackrel{\mu}{=} B\,\bar{x}$*, a co-inductive clause is written as* $p\,\bar{x} \stackrel{\nu}{=} B\,\bar{x}$ *and a regular clause is written as* $p\,\bar{x} \stackrel{\triangle}{=} B\,\bar{x}$*. An extended definition is stratified if it is a stratified definition, and it satisfies the following requirement: for every definition clause* $p\,\bar{x} \stackrel{\triangle}{=} B\,\bar{x}$ *(respectively,*$p\,\bar{x} \stackrel{\mu}{=} B\,\bar{x}$*,* $p\,\bar{x} \stackrel{\nu}{=} B\,\bar{x}$*) it holds that for every predicate symbol* $q \neq p$ *in* B*,* $\mathrm{lvl}(q) < \mathrm{lvl}(p)$*.*

The difference between extended definition and definition is the level restriction $\mathrm{lvl}(q) < \mathrm{lvl}(p)$ above, which is just a way of saying that there is no mutual recursion between p and q.

Since we shall only be concerned with extended definition from now on, we shall refer to an extended definition simply as a definition. The induction and co-induction rules need additional provisos. The $\mu\mathcal{L}$ and $\mu\mathcal{R}$ rules can be applied only to an inductively defined atom. Dually, the $\nu\mathcal{L}$ and $\nu\mathcal{R}$ rules can only be applied to a co-inductively defined atom. The *def*$\mathcal{L}$ and *def*$\mathcal{R}$ rules apply only to regular atoms. However, we can show that *def*$\mathcal{L}$ and *def*$\mathcal{R}$ are derived rules for (co-)inductively defined atoms.

Proposition 2.5. *The def*$\mathcal{L}$ *and def*$\mathcal{R}$ *are admissible rules in the core* Linc *system with the induction and/or the co-induction rules.*

Proof We show here how to infer *def*$\mathcal{L}$ using the core rules of Linc and induction rules. The other case with co-induction can be done dually. Let $p\,\bar{x} \stackrel{\triangle}{=} B\,p\,\bar{x}$ be the definition under consideration: *def*$\mathcal{L}$ can be inferred from $\mu\mathcal{L}$ using the body $B\,p$ as the invariant.

$$\dfrac{\overset{\Pi}{\bar{x}\,;\,B\,(B\,p)\,\bar{x} \longrightarrow B\,p\,\bar{x}} \quad \Sigma\,;\,\bar{y} \rhd B\,p\,\bar{t}, \Gamma \longrightarrow \mathcal{C}}{\Sigma\,;\,\bar{y} \rhd p\,\bar{t}, \Gamma \longrightarrow \mathcal{C}}\,\mu\mathcal{L} \quad .$$

We construct the derivation Π by induction on the size of B, i.e., the number of logical constants in B. In the inductive cases, the derivation is constructed by applying the introduction rules for the logical connectives in B, coordinated between left and right introduction rules. For example, if $B\,p\,\bar{x}$ is $B_1\,p\,\bar{x} \wedge B_2\,p\,\bar{x}$ we take Π as

$$\dfrac{\dfrac{\overset{\Pi_1}{\bar{x}\,;\,B_1\,(B\,p)\,\bar{x} \longrightarrow B_1\,p\,\bar{x}}}{\bar{x}\,;\,B_1\,(B\,p)\,\bar{x} \wedge B_2\,(B\,p)\,\bar{x} \longrightarrow B_1\,p\,\bar{x}}\,\wedge\mathcal{L} \quad \dfrac{\overset{\Pi_2}{\bar{y}\,;\,B_2\,(B\,p)\,\bar{x} \longrightarrow B_2\,p\,\bar{x}}}{\bar{y}\,;\,B_2\,(B\,p)\,\bar{x} \wedge B_2\,(B\,p)\,\bar{x} \longrightarrow B_2\,p\,\bar{x}}\,\wedge\mathcal{L}}{\bar{x}\,;\,B_1\,(B\,p)\,\bar{x} \wedge B_2\,(B\,p)\,\bar{x} \longrightarrow B_1\,p\,\bar{x} \wedge B_2\,p\,\bar{x}}\,\wedge\mathcal{R}$$

where Π_1 and Π_2 are obtained by induction hypothesis. Since p occurs strictly positively in $(B\,p)$ by stratification, the only non-trivial base case we need to consider is when we reach the sub-formula $p\,\bar{t}$ of $B\,p\,\bar{x}$ in which case we just apply the $\mu\mathcal{R}$ rule

$$\dfrac{\dfrac{}{\Sigma\,;\,\bar{u} \rhd B\,p\,\bar{t} \longrightarrow \bar{u} \rhd B\,p\,\bar{t}}\,\mathit{init}}{\Sigma\,;\,\bar{u} \rhd B\,p\,\bar{t} \longrightarrow \bar{u} \rhd p\,\bar{t}}\,\mu\mathcal{R}$$

■

The inference rules for equality, definition and (co-)induction are summarized in Figure 2.2. The inference rules of the logic Linc is obtained by augmenting the core inference rules in Figure 2.1 with those in Figure 2.2. Note that in Linc the equality predicate is treated as a logical constant.

2.4 Some derived rules

The description of the rules of Linc given in the previous section is mainly aimed at simplifying the meta-theoretical discussions. In practice, however, when we actually work with a concrete operational semantics, it would help to have more tractable and descriptive rules and definitions. One of the simpler derived rule is a variant of $\wedge\mathcal{L}$.

$$\dfrac{\mathcal{B}, \mathcal{D}, \Gamma \longrightarrow \mathcal{C}}{\mathcal{B} \wedge \mathcal{D}, \Gamma \longrightarrow \mathcal{C}}\,\wedge\mathcal{L}^*$$

It is derived in Linc as follows.

$$\dfrac{\dfrac{\dfrac{\mathcal{B}, \mathcal{D}, \Gamma \longrightarrow \mathcal{C}}{\mathcal{B}, \mathcal{B} \wedge \mathcal{D}, \Gamma \longrightarrow \mathcal{C}}\,\wedge\mathcal{L}}{\mathcal{B} \wedge \mathcal{D}, \mathcal{B} \wedge \mathcal{D}, \Gamma \longrightarrow \mathcal{C}}\,\wedge\mathcal{L}}{\mathcal{B} \wedge \mathcal{D}, \Gamma \longrightarrow \mathcal{C}}\,c\mathcal{L}$$

Equality:

$$\frac{\{\Sigma\rho\,;\,\Gamma\rho \longrightarrow \mathcal{C}\rho \ \mid\ (\lambda\bar{x}.s)\rho =_{\beta\eta} (\lambda\bar{x}.t)\rho\}}{\Sigma\,;\,\bar{x} \triangleright (s = t), \Gamma \longrightarrow \mathcal{C}}\ \mathrm{eq}\mathcal{L} \qquad \frac{}{\Sigma\,;\,\Gamma \longrightarrow \bar{x} \triangleright t = t}\ \mathrm{eq}\mathcal{R}$$

Definition:

$$\frac{\Sigma\,;\,\sigma \triangleright B\,\bar{t}, \Gamma \longrightarrow \mathcal{C}}{\Sigma\,;\,\sigma \triangleright p\,\bar{t}, \Gamma \longrightarrow \mathcal{C}}\ \mathit{def}\mathcal{L} \qquad \frac{\Sigma\,;\,\Gamma \longrightarrow \sigma \triangleright B\,\bar{t}}{\Sigma\,;\,\Gamma \longrightarrow \sigma \triangleright p\,\bar{t}}\ \mathit{def}\mathcal{R}\ ,$$

where $p\,\bar{x} \overset{\triangle}{=} B\,\bar{x}$.

Induction:

$$\frac{\bar{x}\,;\,B\,S\,\bar{x} \longrightarrow S\,\bar{x} \quad \Sigma\,;\,\Gamma, \bar{y} \triangleright S\,\bar{t} \longrightarrow \mathcal{C}}{\Sigma\,;\,\Gamma, \bar{y} \triangleright p\,\bar{t} \longrightarrow \mathcal{C}}\ \mu\mathcal{L} \qquad \frac{\Sigma\,;\,\Gamma \longrightarrow \bar{y} \triangleright B\,p\,\bar{t}}{\Sigma\,;\,\Gamma \longrightarrow \bar{y} \triangleright p\,\bar{t}}\ \mu\mathcal{R}\ ,$$

where $p\,\bar{x} \overset{\mu}{=} B\,p\,\bar{x}$.

Co-induction:

$$\frac{\Sigma\,;\,\bar{y} \triangleright B\,p\,\bar{t}, \Gamma \longrightarrow \mathcal{C}}{\Sigma\,;\,\bar{y} \triangleright p\,\bar{t}, \Gamma \longrightarrow \mathcal{C}}\ \nu\mathcal{L} \qquad \frac{\Sigma\,;\,\Gamma \longrightarrow \bar{y} \triangleright S\,\bar{t} \quad \bar{x}\,;\,S\,\bar{x} \longrightarrow (B\,S\,\bar{x})}{\Sigma\,;\,\Gamma \longrightarrow \bar{y} \triangleright p\,\bar{t}}\ \nu\mathcal{R}$$

where $p\,\bar{x} \overset{\nu}{=} B\,p\,\bar{x}$ and $\mathrm{lvl}(S\,\bar{x}) \leq \mathrm{lvl}(p)$.

Fig. 2.2. Inference rules for equality, definition and (co-)induction.

The more interesting derived rules are those concerning the equality and definitions. The eq$\mathcal{L}$ rule is not effective, since given a pair of terms (s,t), if there is a unifier for the pairs, then there is an infinite number of them. For example we can take a unifier θ of (s,t) and extend it with a substitution pair containing new variables which are not in the domain of θ. We can also vary θ on the choice of new variables introduced. Of course, these differences are not essential, so in practice we need only consider much fewer cases. The notion of *complete set of unifiers* abstracts away from some of these inessential details.

2.4.1 Complete set of unifiers

A *complete set of unifiers* for two terms s and t, written $CSU(s,t)$, is a set of unifiers for s and t with the following property: if θ is a unifier of (s,t), then there is a unifier $\rho \in CSU(s,t)$ such that $\theta = \rho \circ \gamma$ for some substitution γ. In the first-order case, for example, the set which contains only the *most general unifier* is a complete set of unifiers. However, in the higher-order case the complete set of unifiers may contain more than one unifier, sometimes infinitely many of them.

We define an inference rule eq$\mathcal{L}_{csu}$ as follows.

$$\frac{\{\Sigma\rho\,;\,\Gamma\rho \longrightarrow \mathcal{C}\rho \mid \rho \in CSU(\lambda\bar{x}.s, \lambda\bar{x}.t)\}}{\Sigma\,;\,\bar{x} \triangleright s = t, \Gamma \longrightarrow \mathcal{C}}\ \mathrm{eq}\mathcal{L}^{csu}$$

We show that this rule is interadmissible with the eq$\mathcal{L}$ rule. We refer to the logic Linc with eq$\mathcal{L}^{csu}$ but without eq$\mathcal{L}$ as Linc^{csu}.

PROPOSITION 2.6. *The rule* eq$\mathcal{L}$ *and* eq$\mathcal{L}^{csu}$ *are interadmissible, i.e., for every derivation of a sequent in* Linc*, there is a derivation of the same sequent in* Linc^{csu}*, and vice versa.*

Proof Let Π be a derivation in Linc. The proof is by induction on the height of Π. The only non-trivial case is when Π ends with eq$\mathcal{L}$.

$$\frac{\left\{\begin{matrix}\Pi^{\theta} \\ \Sigma\theta\,;\,\Gamma\theta \longrightarrow \mathcal{C}\theta\end{matrix}\right\}_{\theta}}{\Sigma\,;\,\bar{x} \triangleright s = t, \Gamma \longrightarrow \mathcal{C}}\ \mathrm{eq}\mathcal{L}$$

where $(\lambda\bar{x}.s)\theta =_{\beta\eta} (\lambda\bar{x}.t)\theta$. By the definition of CSU, any $\rho \in CSU(\lambda\bar{x}.s, \lambda\bar{x}.t)$ also satisfies the equation $(\lambda\bar{x}.s)\rho =_{\beta\eta} (\lambda\bar{x}.t)\rho$. Therefore we construct a derivation in Linc^{csu} as follows.

$$\frac{\left\{\begin{matrix}\Pi_1^{\rho} \\ \Sigma\rho\,;\,\Gamma\rho \longrightarrow \mathcal{C}\rho\end{matrix}\right\}_{\rho}}{\Sigma\,;\,\bar{x} \triangleright s = t, \Gamma \longrightarrow \mathcal{C}}\ \mathrm{eq}\mathcal{L}^{csu}$$

Here ρ is a unifier in $CSU(\lambda\bar{x}.s, \lambda\bar{x}.t)$ and Π_1^{ρ} is obtained by induction hypothesis on Π^{ρ}.

Conversely, suppose we have a derivation Π in Linc^{csu} ending with eq$\mathcal{L}^{csu}$

$$\frac{\left\{\begin{array}{c}\Pi^{\rho} \\ \Sigma\rho\,;\,\Gamma\rho \longrightarrow \mathcal{C}\rho\end{array}\right\}_{\rho}}{\Sigma\,;\,\bar{x} \triangleright s = t, \Gamma \longrightarrow \mathcal{C}}\ \mathrm{eq}\mathcal{L}^{csu}$$

We need to show that for every unifier θ of $(\lambda\bar{x}.s, \lambda\bar{x}.t)$ there is a derivation Π^{θ} of $\Sigma\rho\,;\,\Gamma\rho \longrightarrow \mathcal{C}\rho$. If $\theta \in CSU(\lambda\bar{x}.s, \lambda\bar{x}.t)$, then by induction hypothesis we have a derivation Π_1^{θ} (in Linc) of the sequent $\Sigma\theta\,;\,\Gamma\theta \longrightarrow \mathcal{C}\theta$. Otherwise, by the definition of CSU, there must be a substitution γ such that $\theta = \rho \circ \gamma$ for some $\rho \in CSU(\lambda\bar{x}.s, \lambda\bar{x}.t)$ and some substitution γ. In this case we first apply the induction hypothesis on Π^{ρ} to get a derivation Π_1^{ρ} of $\Sigma\rho\,;\,\Gamma\rho \longrightarrow \mathcal{C}\rho$, and then we apply Lemma 3.5 (see Chapter 3) with substitution γ to Π_1^{ρ} to get the derivation $\Pi_1^{\rho}\gamma$ of the sequent $\Sigma\theta\,;\,\Gamma\theta \longrightarrow \mathcal{C}\theta$. ∎

2.4.2 Patterned definitions

In most applications, it is often more convenient to work with definition clauses with explicit pattern matching on the head. That is, instead of the definition scheme $p\,\bar{x} \stackrel{\triangle}{=} B\,\bar{x}$ we would have a finite list of clauses

$$p\,\bar{t}_1 \stackrel{\triangle}{=} B_1,\ p\,\bar{t}_2 \stackrel{\triangle}{=} B_2,\ \cdots,\ p\,\bar{t}_n \stackrel{\triangle}{=} B_n$$

We shall refer to these clauses as *patterned clauses.* Formal definition follows.

Definition 2.7. *A* patterned definition clause *is written*

$$\forall\bar{y}.p\,\bar{t} \stackrel{\triangle}{=} B$$

where the free variables in B are in the set of free variables of $\bar{t}$, which in turn is in $\bar{y}$.

A patterned definition is a set of patterned clauses. There can be more than one clause, but finitely many of them, in the definition with the same head. The stratification of the patterned definition is the same as the usual definition and mutual recursion is not allowed. Patterned (co-)inductive clauses and definitions are defined analogously.

The patterned definition serves only as an abbreviation for the usual definition. The corresponding definition of a patterned definition is defined as follows.

Definition 2.8. *Let* **D** *be a patterned definition. The corresponding definition of* **D** *is obtained by replacing the set of definition clauses for p*

$$\begin{array}{c}\forall\bar{y}_1.p\ t_{11}\ \ldots\ t_{1m} \stackrel{\triangle}{=} B_1, \\ \vdots \\ \forall\bar{y}_n.p\ t_{n1}\ \ldots\ t_{nm} \stackrel{\triangle}{=} B_n\end{array}$$

by the definition clause

$$\begin{array}{rcl} p\,x_1 \ldots x_m & \stackrel{\triangle}{=} & \exists \bar{y}_1.(x_1 = t_{11} \wedge \ldots \wedge x_m = t_{1m} \wedge B_1) \vee \\ & & \vdots \\ & & \vee\ \exists \bar{y}_n.(x_1 = t_{n1} \wedge \ldots \wedge x_m = t_{nm} \wedge B_n) \end{array}$$

for every predicate p defined in $\mathbf{D}$.

We wish to derive the rules that apply directly to patterned definition, similar to the definition rules in $FO\lambda^{\Delta\mathbb{N}}$ and $FO\lambda^{\Delta\nabla}$ [38]. As in $FO\lambda^{\Delta\nabla}$, we need to take into account the local signature when applying the rules for patterned definition. This makes the description of the rules slightly complicated. We need a couple more definitions.

Definition 2.9. *Let $\forall_{\tau_1} x_1 \ldots \forall_{\tau_n} x_n.H \stackrel{\triangle}{=} B$ be a patterned definition clause. Let y_1, ..., y_m be a list of variables of types α_1,...,α_m, respectively. A* raised patterned definition clause*, or raised definition clause for short, with respect to the signature $\{y_1 : \alpha_1, \ldots, y_m : \alpha_m\}$ is defined as*

$$\forall h_1 \ldots \forall h_n.\bar{y} \triangleright H\theta \stackrel{\triangle}{=} \bar{y} \triangleright B\theta$$

where θ is the substitution $[(h_1\,\bar{y})/x_1, \ldots, (h_n\,\bar{y})/x_n]$ and h_i, for every $i \in \{1, \ldots, n\}$, is of type $\alpha_1 \to \ldots \to \alpha_m \to \tau_i$.

Definition 2.10. *The four-place relation* $\mathrm{dfn}(\rho, \mathcal{A}, \theta, \mathcal{B})$ *holds for the atomic judgment $\mathcal{A}$, the judgment $\mathcal{B}$, and the substitutions ρ and θ if there is a raised clause $\mathcal{H} \stackrel{\triangle}{=} \mathcal{B}$ in the given definition such that $\mathcal{A}\rho = \mathcal{H}\theta$.*

Observe that in the above definition, for the relation $\mathrm{dfn}(\rho, \mathcal{A}, \theta, \mathcal{B})$ to hold, the judgments $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{H}$ must share the same local signature (up to α-conversion). We are now ready to describe the rules for patterned definitions, which we refer to here as $def\mathcal{L}^{=}$ and $def\mathcal{R}^{=}$.

$$\frac{\Sigma\,;\,\Gamma \longrightarrow \mathcal{B}\theta}{\Sigma\,;\,\Gamma \longrightarrow \mathcal{A}}\ def\mathcal{R}^{=},\ \text{where } \mathrm{dfn}(\epsilon, \mathcal{A}, \theta, \mathcal{B})$$

$$\frac{\{\Sigma\rho\,;\,\mathcal{B}\theta, \Gamma\rho \longrightarrow \mathcal{C}\rho \mid \mathrm{dfn}(\rho, \mathcal{A}, \theta, \mathcal{B})\}}{\Sigma\,;\,\mathcal{A}, \Gamma \longrightarrow \mathcal{C}}\ def\mathcal{L}^{=}$$

Proposition 2.11. *The rules $def\mathcal{L}^{=}$ and $def\mathcal{R}^{=}$ are admissible in* Linc.

Proof Follows immediately from Definition 2.8, Definition 2.9 and Definition 2.10. The $def\mathcal{R}^{=}$ is derived by using $\exists\mathcal{R}$, $\wedge\mathcal{R}$ and eq$\mathcal{R}$ rules, while the $def\mathcal{L}^{=}$ is derived by using $\exists\mathcal{L}$, $c\mathcal{L}$, $\wedge\mathcal{L}$, and eq$\mathcal{L}$ rules. ∎

A variant of $def\mathcal{L}^{=}$ rule is to use the notion of CSU, as the following.

$$\frac{\{\Sigma\rho\,;\,\mathcal{B}\rho, \Gamma\rho \longrightarrow \mathcal{C}\rho \mid\ \mathcal{H} \stackrel{\triangle}{=} \mathcal{B} \text{ is a raised def. clause s.t. } \rho \in CSU(\mathcal{A}, \mathcal{H})\}}{\Sigma\,;\,\mathcal{A}, \Gamma \longrightarrow \mathcal{C}}\ def\mathcal{L}^{csu}$$

where $\mathcal{H} \overset{\triangle}{=} \mathcal{B}$ is a raised definition clause. The admissibility of this rule is immediate from the admissibility of eq$\mathcal{L}^{csu}$.

PROPOSITION 2.12. *The rule def$\mathcal{L}^{csu}$ is admissible in* Linc.

We now illustrate the use of Linc to prove a simple property about natural numbers in the following example. More elaborate examples are discussed in Chapter 5 and Chapter 6.

EXAMPLE 2.13. We can specify the odd and even natural numbers as the following inductive definition clauses.

$$\begin{array}{ll} odd\ N & \overset{\mu}{=} N = (s\ z) \lor \exists M.N = (s\ (s\ M)) \land odd\ M. \\ even\ N & \overset{\mu}{=} N = z \lor \exists M.N = (s\ (s\ M)) \land even\ M. \end{array}$$

Here the constants $z : nt$ and $s : nt \to nt$, where nt is the type representing natural numbers, denote the natural number zero and the successor function, respectively. We prove that the successor of an odd number is an even number. This is stated formally as

$$\forall N odd\ N \supset even\ (s\ N).$$

This formula is proved by structural induction on *odd*, using the invariant

$$\lambda n\ even\ (s\ n).$$

The last three rules of the formal proof is as follows.

$$\cfrac{\cfrac{N\,;\,odd\ N \longrightarrow even\ (s\ N)}{N\,;\,.\longrightarrow odd\ N \supset even\ (s\ N)}\supset\mathcal{R}}{.\,;\,.\longrightarrow \forall N odd\ N \supset even\ (s\ N)}\forall\mathcal{R}$$

Applying the $\mu\mathcal{L}$-rule to the topmost sequent yields two premises

1. $N'\,;\,N' = (s\ z) \lor \exists M.N' = s\ (s\ M) \land even\ (s\ M) \longrightarrow even\ (s\ N')$, and

2. $N\,;\,even\ (s\ N) \longrightarrow even\ (s\ N)$.

Notice that in the first premise, the left-hand side of the sequent is the body of the definition clause of *odd* with the *odd* predicate replaced by the invariant $\lambda n\ even\ (s\ n)$. The second premise is immediately provable by using the *init*-rule. For the first premise, we apply first the $\lor\mathcal{L}$-rule, which gives us the following premises.

1.a. $N'\,;\,N' = (s\ z) \longrightarrow even\ (s\ N')$,

1.b. $N'\,;\,\exists M.N' = s\ (s\ M) \land even\ (s\ M) \longrightarrow even\ (s\ N')$.

The proof for the sequent 1.a. is the following.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\longrightarrow s\ (s\ z) = s\ (s\ z)}\ \text{eq}\mathcal{R} \qquad \dfrac{\dfrac{\dfrac{}{\longrightarrow z = z}\ \text{eq}\mathcal{R}}{\longrightarrow z = z \lor \exists M.z = s\ (s\ M) \land \mathit{even}\ M}\ \lor\mathcal{R}}{\longrightarrow \mathit{even}\ z}\ \mathit{def}\mathcal{R}}{\longrightarrow s\ (s\ z) = s\ (s\ z) \land \mathit{even}\ z}\ \land\mathcal{R}}{\longrightarrow \exists M.s\ (s\ z) = s\ (s\ M) \land \mathit{even}\ M}\ \exists\mathcal{R}}{\longrightarrow (s\ (s\ z)) = z \lor \exists M.s\ (s\ z) = s\ (s\ M) \land \mathit{even}\ M}\ \lor\mathcal{R}}{\longrightarrow \mathit{even}\ (s\ (s\ z))}\ \mathit{def}\mathcal{R}}{N'\,;\, N' = (s\ z) \longrightarrow \mathit{even}\ (s\ N')}\ \text{eq}\mathcal{L}^{csu}$$

Notice that in the application of eq$\mathcal{L}^{csu}$, the eigenvariable N' is instantiated with the term $(s\ z)$, and it is removed from the signature of the premise sequent. The proof for the sequent 1.b. is as follows.

$$\dfrac{\dfrac{\dfrac{M\,;\, \mathit{even}\ (s\ M) \longrightarrow \mathit{even}\ (s\ (s\ (s\ M)))}{N', M\,;\, N' = s\ (s\ M), \mathit{even}\ (s\ M) \longrightarrow \mathit{even}\ (s\ N')}\ \text{eq}\mathcal{L}^{csu}}{N', M\,;\, N' = s\ (s\ M) \land \mathit{even}\ (s\ M) \longrightarrow \mathit{even}\ (s\ N')}\ \land\mathcal{L}^{*}}{N'\,;\, \exists M.N' = s\ (s\ M) \land \mathit{even}\ (s\ M) \longrightarrow \mathit{even}\ (s\ N')}\ \exists\mathcal{L}$$

It is easy to see that the remainder of the proof can be constructed by applying *def*$\mathcal{R}$, $\lor\mathcal{R}$, $\exists\mathcal{R}$ and *init* rules.

2.5 Related work

The notion of definitions as it is in Linc has been previously studied by Hallnäs [22], Schroeder-Heister [50], Eriksson [14], Girard [21], Stärk [55] and McDowell and Miller [29]. We adopt a simpler form of definition clauses by putting all pattern matching in the body of the definition clause, and formulating separate free equality rules. This separation corresponds to *iff-completion* in logic programming [9]. Our formulation of definitions closely resembles that of $FO\lambda^{\Delta\mathbb{N}}$. In fact, Linc can be seen as a consistent extension of $FO\lambda^{\Delta\mathbb{N}}$ with generalized induction and co-induction proof rules and the ∇-quantifier.

It is natural to ask about possible connections between the ∇-quantifier and the new quantifier of Gabbay and Pitts [17, 45]. Both are self dual and both have similar sets of applications in mind. There are significant differences, however: ∇ has a natural proof theory with a cut-elimination theorem but has no set theoretic semantics, while Gabbay and Pitts have a model theory based on set theory but no cut-elimination result. While ∇ neither implies nor is implied by $\forall$ or $\exists$, the quantifier of Gabbay and Pitts is entailed by $\forall$ and entails $\exists$. Gabbay and Pitts quantifier allows quantification on type *name*, which is presupposed to be non-empty, whereas ∇ allows quantification over arbitrary type and does not presuppose any type inhabitant.

There have been several works on formalization of (co)-induction in sequent systems. However, Linc is as far as we know the first sequent system which incorporates both generalized induction and co-induction in the same system and still admits cut-elimination (Chapter 4), from which the consistency of the logic follows. The logic $FO\lambda^{\Delta\mathbb{N}}$ has a formalization of induction, but it is restricted to natural number induction. Eriksson's *calculus of partial inductive definitions* [14] has a formalization of generalized induction but the system has no cut-elimination. The logic $CLP^{\forall}$ [11], has both induction and co-induction rules, in the setting of constraint logic programming. However, this system does not allow mixing of induction and co-induction in the same proof, and cut-rule is not present in the system, thus composition of proofs is not supported in the logic. There are also some recent works in the area of *fixed pointed logics* [48, 52, 54, 10], where (co-)induction is formalized via the notion of *circular proofs*.

Linc can be seen as the meta-theory of the simply typed λ-calculus, in the same sense in which Schürmann's $\mathcal{M}_\omega$ [51] is the meta-theory of LF [24]. The logic $\mathcal{M}_\omega$ is a constructive first-order logic, whose quantifiers range over possibly open LF object over a signature. In the meta-logic it is possible to express and inductively prove meta-logical properties of an object logic. By the adequacy of the encoding, the proof of the existence of the appropriate LF object(s) guarantees the proof of the corresponding object-level property. However, $\mathcal{M}_\omega$ does not support co-induction yet. The new quantifier ∇ also gives Linc more expressive power than $\mathcal{M}_\omega$ in terms of reasoning about object-systems with intrinsic notion of freshness and scoping of names (see Chapter 6).

Chapter 3

Properties of Derivations

We discuss several properties of derivations in Linc. Some of them involve transformations on derivations which will be used extensively in the cut-elimination proof in the chapter to follow. The other properties are concerned with a particular class of definitions used in proof search, namely *Horn* definitions (i.e., definitions whose bodies contain no implications). The interest in this class of definitions is justified by the fact that there are many specifications of operational semantics that fall into this category, as we shall see in the following chapters.

Before we proceed, some remarks on the use of eigenvariables in derivations must be mentioned. In proof search involving $\forall\mathcal{R}$, $\exists\mathcal{L}$ $\mu\mathcal{L}$, $\nu\mathcal{R}$ or eq$\mathcal{L}$, new eigenvariables can be introduced in the premises of the rules. Let us refer to such variables as internal eigenvariables, since they occur only in the premise derivations. We view the choice of such eigenvariables as arbitrary and therefore we identify derivations that differ only in the choice of the eigenvariables introduced by those rules. Another way to look at it is to consider eigenvariables as proof-level binders. Hence when we work with a derivation, we actually work with an equivalence class of derivations modulo renaming of internal eigenvariables.

In this and the following chapters, we shall leave as much information implicit in a sequent or a derivation, as long as no ambiguity arises. In particular, we shall avoid writing the global signature explicitly whenever possible, e.g., when the signature can be inferred from context. For example, in describing transformation on derivations that involves only propositional connectives; often in such cases there is no change in the global signature involved. We shall also omit the local signatures in certain cases. For this to work, we shall allow a slight abuse of notation by writing for example $\mathcal{B} \wedge \mathcal{C}$ instead of $\bar{y} \triangleright B \wedge C$. Whenever we do this, it is assumed implicitly that $\mathcal{B}$ and $\mathcal{C}$ share the same local signature.

3.1 Instantiating derivations

The following definition extends substitutions to apply to derivations. Since we identify derivations that differ only in the choice of variables that are not free in the end-sequent, 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 3.1. *If* Π *is a derivation of* $\Sigma\,;\Gamma \longrightarrow \mathcal{C}$ *and* θ *is a substitution, then we define the derivation* $\Pi\theta$ *of* $\Sigma\theta\,;\Gamma\theta \longrightarrow \mathcal{C}\theta$ *as follows:*

1. *Suppose* Π *ends with the* eq$\mathcal{L}$ *rule*

$$\frac{\left\{ \begin{array}{c} \Pi^\rho \\ \Sigma\rho\,;\,\Gamma'\rho \longrightarrow \mathcal{C}\rho \end{array} \right\}_\rho}{\Sigma\,;\,\bar{x} \triangleright s = t, \Gamma' \longrightarrow \mathcal{C}}\ \text{eq}\mathcal{L}$$

where $(\lambda\bar{x}.s)\rho =_{\beta\eta} (\lambda\bar{x}.t)\rho$. *Observe that any unifier for the pair*

$$((\lambda\bar{x}.s)\theta, (\lambda\bar{x}.t)\theta)$$

can be transformed to another unifier for $(\lambda\bar{x}.s, \lambda\bar{x}.t)$, *by composing the unifier with* θ. *Thus* $\Pi\theta$ *is*

$$\frac{\left\{ \begin{array}{c} \Pi^{\theta\circ\rho'} \\ \Sigma\theta\rho'\,;\,\Gamma'\theta\rho' \longrightarrow \mathcal{C}\theta\rho' \end{array} \right\}_{\rho'}}{\Sigma\theta\,;\,(\bar{x} \triangleright s = t)\theta, \Gamma'\theta \longrightarrow \mathcal{C}\theta}\ \text{eq}\mathcal{L} \quad ,$$

where $(\lambda\bar{x}.s)\theta\rho' =_{\beta\eta} (\lambda\bar{x}.t)\theta\rho'$.

2. *If* Π *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 Linc, certain premises share the same right-hand side judgment as in the sequent in the conclusion. We refer to such premises as major premises. This notion of major premise will be useful in proving cut-elimination, as certain proof transformations involve only major premises.

Definition 3.2. *Given an inference rule* $\bullet$ *with one or more premise sequents, we define its major premise sequents as follows.*

1. *If* $\bullet$ *is either* $\supset \mathcal{L}$, mc *or* $\mu\mathcal{L}$, *then its right premise is the major premise*
2. *If* $\bullet$ *is* $\nu\mathcal{R}$ *then its left premise is the major premise.*
3. *Otherwise, all the premises of* $\bullet$ *are major premises.*

The definition extends to derivations by replacing premise sequents with premise derivations.

The following two measures on derivations will be useful later in proving many properties of the logic. Note that given the possible infinite branching of eq$\mathcal{L}$ rule, these measure in general can be ordinals. Therefore in proofs involving induction on those measures, transfinite induction is needed. However, in most of the inductive proofs to follow, we often do case analyses on the last rule of a derivation. In such a situation, the inductive cases for both successor ordinals and limit ordinals are basically covered by the case analyses on the inference figures involved, and we shall not make explicit use of transfinite induction.

Definition 3.3. *Given a derivation* Π *with premise derivations* $\{\Pi_i\}_i$, *the measure* $\text{ht}(\Pi)$ *is the least upper bound of* $\{\text{ht}(\Pi_i) + 1\}_i$.

Definition 3.4. *Given a derivation* Π *with premise derivations* $\{\Pi_i\}_i$, *the measure* $\mathrm{ind}(\Pi)$ *is defined as follows*

$$\mathrm{ind}(\Pi) = \begin{cases} \mathrm{lub}(\{\mathrm{ind}(\Pi_i)\}_i) + 1, & \textit{if } \Pi \textit{ ends with } \mu\mathcal{L}, \\ \mathrm{lub}(\{\mathrm{ind}(\Pi_i)\}_i), & \textit{otherwise.} \end{cases}$$

Lemma 3.5. *For any substitution* θ *and derivation* Π *of* $\Sigma\,;\,\Gamma \longrightarrow \mathcal{C}$, $\Pi\theta$ *is a derivation of* $\Sigma\theta\,;\,\Gamma\theta \longrightarrow \mathcal{C}\theta$.

Proof This lemma states that Definition 3.1 is well-constructed, and follows by induction on $\mathrm{ht}(\Pi)$. ∎

Lemma 3.6. *For any derivation* Π *and substitution* θ, $\mathrm{ht}(\Pi) \geq \mathrm{ht}(\Pi\theta)$ *and* $\mathrm{ind}(\Pi) \geq \mathrm{ind}(\Pi\theta)$.

Proof By induction on $\mathrm{ht}(\Pi)$. The measures may not be equal because when the derivations end with the eq$\mathcal{L}$ rule, some of the premise derivations of Π may not be needed to construct the premise derivations of $\Pi\theta$. ∎

Lemma 3.7. *For any derivation* Π *and substitutions* θ *and* ρ, *the derivations* $(\Pi\theta)\rho$ *and* $\Pi(\theta \circ \rho)$ *are the same derivation.*

Proof By induction on the measure $\mathrm{ht}(\Pi)$. ∎

3.2 Atomic initial rule

It is a common property of most logics that the initial rule can be restricted to atomic form, that is, the rule

$$\frac{}{\Sigma\,;\,\bar{x} \triangleright p\,\bar{t} \longrightarrow \bar{x} \triangleright p\,\bar{t}}\ \textit{init}$$

where p is a predicate symbol. The more general rule is derived as follows.

Definition 3.8. *We construct a derivation* $\mathrm{Id}_{\mathcal{C}}$ *of the sequent* $\Sigma\,;\,\mathcal{C} \longrightarrow \mathcal{C}$ *inductively as follows. The induction is on the size of the judgment* $\mathcal{C}$. *If* $\mathcal{C}$ *is an atomic judgment we simply apply the atomic initial rule. Otherwise, we apply the left and right introduction rules for the topmost logical constant in* $\mathcal{C}$, *probably with some instances of contraction and weakening rule.*

The proof of the following lemma is straightforward by induction on $\mathrm{ht}(\mathrm{Id}_{\mathcal{C}})$.

Lemma 3.9. *For any judgment* $\mathcal{C}$, *it holds that* $\mathrm{ind}(\mathrm{Id}_{\mathcal{C}}) = 0$.

Restricting the initial rule to atomic form will simplify some technical definitions to follow. We shall use Id instead of $\mathrm{Id}_{\mathcal{C}}$ to denote identity derivations since the judgment $\mathcal{C}$ is always known from context.

3.3 Weakening and scoping of signatures

The following lemmas state that we can extend both the global and local signatures in a provable sequent without affecting its provability. Again as before, when new eigenvariables are introduced in the end sequent of a derivation, it is implicitly assumed that they are different from the eigenvariables occuring in the derivation but not free in the end sequent.

Definition 3.10. *Let* Π *be a derivation of* $\Sigma\,;\,\Gamma \longrightarrow \mathcal{C}$. *Let* Σ' *be a signature different from* Σ. *We construct a derivation* $\Sigma' : \Pi$ *by adding* Σ' *to the signature of every sequent in the derivation tree of* Π, *except in the case* $\mu\mathcal{L}$ *where* Σ' *is added only to the right premise derivation, and in the case* $\nu\mathcal{R}$ *where* Σ' *is added to the left premise derivation.*

Lemma 3.11. *Let* Π *be a derivation of* $\Sigma\,;\,\Gamma \longrightarrow \mathcal{C}$ *and let* Σ' *be a signature distinct from* Σ. *Then* $\Sigma' : \Pi$ *is a derivation of* $\Sigma, \Sigma'\,;\,\Gamma \longrightarrow \mathcal{C}$, *and* $\mathrm{ht}(\Sigma' : \Pi) = \mathrm{ht}(\Pi)$ *and* $\mathrm{ind}(\Sigma' : \Pi) = \mathrm{ind}(\Pi)$.

Proof By induction on $\mathrm{ht}(\Pi)$. ∎

In the following, whenever we write $\Sigma' : \Pi$, it is assumed that the derivation is well-formed. This means implicitly the signature Σ' is distinct from the signature in the end sequent of Π.

Lemma 3.12. Scope weakening. *Let* Π *be a derivation of*

$$\Sigma, x\,;\,\bar{z}\bar{z}_1 \rhd B_1, \dots, \bar{z}\bar{z}_n \rhd B_n \longrightarrow \bar{z}\bar{z}_0 \rhd B_0$$

where x *is not in the lists of variables* $\bar{z}$ *and* $\bar{z}_0, \dots, \bar{z}_n$. *Then there is a derivation* Π' *of*

$$\Sigma\,;\,\bar{z}x\bar{z}_1 \rhd B_1, \dots, \bar{z}x\bar{z}_n \rhd B_n \longrightarrow \bar{z}x\bar{z}_0 \rhd B_0$$

such that $\mathrm{ht}(\Pi') \le \mathrm{ht}(\Pi)$ *and* $\mathrm{ind}(\Pi') \le \mathrm{ind}(\Pi)$.

Proof By induction on $\mathrm{ht}(\Pi)$. We show here the non-trivial cases. In the following we shall denote $\bar{z}\bar{z}_i \rhd B_i$ with $\mathcal{B}_i$ and $\bar{z}x\bar{z}_i \rhd B_i$ with $\mathcal{B}'_i$.

1. Suppose Π ends with $\nabla\mathcal{R}$ (the case with $\nabla\mathcal{L}$ is symmetric).

$$\dfrac{\begin{matrix}\Pi_1\\ \mathcal{B}_1, \dots, \mathcal{B}_n \longrightarrow \bar{z}\bar{z}_0 y \rhd B'\end{matrix}}{\mathcal{B}_1, \dots, \mathcal{B}_n \longrightarrow \bar{z}\bar{z}_0 \rhd \nabla y.B'}\ \nabla\mathcal{R}\ .$$

Then Π' is the derivation

$$\dfrac{\begin{matrix}\Pi'_1\\ \mathcal{B}'_1, \dots, \mathcal{B}'_n \longrightarrow \bar{z}x\bar{z}_0 y \rhd B'\end{matrix}}{\mathcal{B}'_1, \dots, \mathcal{B}'_n \longrightarrow \bar{z}x\bar{z}_0 \rhd \nabla y.B'}\ \nabla\mathcal{R}\ .$$

where Π'_1 is obtained from induction hypothesis.

2. Suppose Π ends with $\forall\mathcal{R}$ (the case with $\exists\mathcal{L}$ is symmetric).

$$\frac{\begin{array}{c}\Pi_1\\ \Sigma, x, h\,;\, \mathcal{B}_1,\dots,\mathcal{B}_n \longrightarrow \bar{z}\bar{z}_0 \rhd B'\,(h\,\bar{z}\bar{z}_0)\end{array}}{\Sigma, x\,;\, \mathcal{B}_1,\dots,\mathcal{B}_n \longrightarrow \bar{z}\bar{z}_0 \rhd \forall y.B'\,y}\ \forall\mathcal{R}$$

We apply the substitution $\theta = [(\lambda\bar{z}\lambda\bar{z}_0.h'\,\bar{z}x\bar{z}_0)/h]$ to Π_1 to get the derivation $\Pi_1\theta$ of

$$\Sigma, x, h'\,;\, \mathcal{B}_1,\dots,\mathcal{B}_n \longrightarrow \bar{z}\bar{z}_0 \rhd B'\,(h'\,\bar{z}x\bar{z}_0).$$

Since $\mathrm{ht}(\Pi_1\theta) \leq \mathrm{ht}(\Pi_1)$ we can apply induction hypothesis to $\Pi_1\theta$ to get the derivation Π_1' of

$$\Sigma, h'\,;\, \mathcal{B}_1',\dots,\mathcal{B}_n' \longrightarrow \bar{z}x\bar{z}_0 \rhd B'\,(h'\,\bar{z}x\bar{z}_0).$$

The derivation Π' is thus

$$\frac{\begin{array}{c}\Pi_1'\\ \Sigma, h'\,;\, \mathcal{B}_1',\dots,\mathcal{B}_n' \longrightarrow \bar{z}x\bar{z}_0 \rhd B'\,(h'\,\bar{z}x\bar{z}_0)\end{array}}{\Sigma\,;\, \mathcal{B}_1',\dots,\mathcal{B}_n' \longrightarrow \bar{z}x\bar{z}_0 \rhd \forall y.B'\,y}\ \forall\mathcal{R}\ .$$

3. Suppose Π ends with eq$\mathcal{L}$

$$\frac{\left\{\begin{array}{c}\Pi^{\rho}\\ (\Sigma, x)\rho\,;\, \mathcal{B}_2\rho,\dots,\mathcal{B}_n\rho \longrightarrow \mathcal{B}_0\rho\end{array}\right\}_{\rho}}{\Sigma, x\,;\, \bar{z}\bar{z}_1 \rhd s = t, \mathcal{B}_2,\dots,\mathcal{B}_n \longrightarrow \mathcal{B}_0}\ \mathrm{eq}\mathcal{L}$$

where $(\lambda\bar{z}\lambda\bar{z}_1.s)\rho =_{\beta\eta} (\lambda\bar{z}\lambda\bar{z}_1.t)\rho$. We need to construct a derivation for the sequent

$$\Sigma\rho'\,;\, \mathcal{B}_2'\rho',\dots,\mathcal{B}_n'\rho' \longrightarrow \mathcal{B}_0'\rho'$$

for each unifier ρ' satisfying

$$(\lambda\bar{z}\lambda x\lambda\bar{z}_1.s)\rho' =_{\beta\eta} (\lambda\bar{z}\lambda x\lambda\bar{z}_1.t)\rho'.$$

If $x \notin \mathrm{dom}(\rho') \cup \mathrm{ran}(\rho')$, it can be shown by induction on the structure of terms that

$$(\lambda\bar{z}\lambda\bar{z}_1.s)\rho' =_{\beta\eta} (\lambda\bar{z}\lambda\bar{z}_1.t)\rho'.$$

Hence in this case we can apply induction hypothesis to $\Pi^{\rho'}$ to get a derivation $\Pi_1^{\rho'}$ of the above sequent. In the case where $x \in \mathrm{dom}(\rho') \cup \mathrm{ran}(\rho')$, notice that ρ' is also a unifier for the pair $(\lambda\bar{z}\lambda w\lambda\bar{z}_1.s[w/x], \lambda\bar{z}\lambda w\lambda\bar{z}_1.t[w/x])$ where w is chosen to be different from any variable in the above sequent. Therefore, the substitution

$\rho'' = [w/x] \circ \rho'$ can be shown to satisfy

$$(\lambda \bar{z} \lambda \bar{z}_1.s)\rho'' =_{\beta\eta} (\lambda \bar{z} \lambda \bar{z}_1.t)\rho''.$$

By the definition of eq$\mathcal{L}$, there is a derivation, that is, $\Pi^{[w/x]\circ\rho'}$ of

$$\Sigma\rho', w\,;\, (\bar{z}\bar{z}_2 \triangleright B_2[w/x])\rho', \ldots, (\bar{z}\bar{z}_n \triangleright B_n[w/x])\rho' \longrightarrow (\bar{z}\bar{z}_0 \triangleright B_0[w/x])\rho'.$$

We can then apply induction hypothesis to get a derivation $\Pi_2^{\rho'}$ of

$$\Sigma\rho'\,;\, (\bar{z}w\bar{z}_2 \triangleright B_2[w/x])\rho', \ldots, (\bar{z}w\bar{z}_n \triangleright B_n[w/x])\rho' \longrightarrow (\bar{z}w\bar{z}_0 \triangleright B_0[w/x])\rho'$$

which is α-equivalent to

$$\Sigma\rho'\,;\, (\bar{z}x\bar{z}_2 \triangleright B_2)\rho', \ldots, (\bar{z}x\bar{z}_n \triangleright B_n)\rho' \longrightarrow (\bar{z}x\bar{z}_0 \triangleright B_0)\rho'.$$

∎

Lemma 3.13. *Let* Π *be a derivation of*

$$\Sigma\,;\, \bar{z}\bar{z}_1 \triangleright B_1\, a, \ldots, \bar{z}\bar{z}_n \triangleright B_n\, a \longrightarrow \bar{z}\bar{z}_0 \triangleright B_0\, a$$

where a *is a global constant. Then there is a derivation* Π' *of*

$$\Sigma\,;\, \bar{z}x\bar{z}_1 \triangleright B_1\, x, \ldots, \bar{z}x\bar{z}_n \triangleright B_n\, x \longrightarrow \bar{z}x\bar{z}_0 \triangleright B_0\, x$$

without the global constant a*, such that* $\mathrm{ht}(\Pi') \leq \mathrm{ht}(\Pi)$ *and* $\mathrm{ind}(\Pi') \leq \mathrm{ind}(\Pi)$.

Proof The proof is by induction on $\mathrm{ht}(\Pi)$. The case analysis is essentially the same as the proof for Lemma 3.12 in most cases. The only difference is in case where Π ends with eq$\mathcal{L}$. In this case we need to show that the set of unifiers for the equation

$$\lambda \bar{z} \lambda x \lambda \bar{z}_i.s\, x = \lambda \bar{z} \lambda x \lambda \bar{z}_i.t\, x$$

is included in the set of unifiers of

$$\lambda \bar{z} \lambda \bar{z}_i.s\, a = \lambda \bar{z} \lambda \bar{z}_i.t\, a$$

where a is treated as a global constant. Let ρ be a unifier for the first equation. Since a is not admitted as a constant in the first equation, it cannot occur free in the range of the substitution ρ. We can assume without loss of generality that the variables in $\bar{z}, x, \bar{z}_i$ are all different from the free variables in ρ (otherwise, apply renaming to both equation so they are different) so we can push the substitution under the binders. Therefore we have $(s\, x)\rho = (s\rho)\, x = (t\rho)\, x = (t\, x)\rho$ and $(s\rho)\, a = (s\rho)\, a$. Hence ρ is also a unifier for the second equation. ∎

Lemma 3.14. Local signature weakening. *Let* Π *be a derivation of*

$$\Sigma\,;\,\sigma_1 \rhd B_1, \ldots, \sigma_n \rhd B_n \longrightarrow \sigma_0 \rhd B_0$$

and let x *be a variable different from* $\Sigma, \sigma_0, \ldots, \sigma_n$*. Then there is a derivation* $l(\Pi, x)$ *of*

$$\Sigma\,;\,x\sigma_1 \rhd B_1, \ldots, x\sigma_n \rhd B_n \longrightarrow x\sigma_0 \rhd B_0$$

such that $\mathrm{ht}(l(\Pi, x)) \leq \mathrm{ht}(\Pi)$ *and* $\mathrm{ind}(l(\Pi, x)) \leq \mathrm{ind}(\Pi)$*..*

Proof Immediate consequence of Lemma 3.11 and Lemma 3.12. ∎

Definition 3.15. Raised sequents. *Let* Σ *be the signature*

$$\{h_1 : \tau_1, \ldots, h_n : \tau_n\}$$

and let $\bar{\alpha}$ *be the list of types* $\alpha_1, \ldots, \alpha_m$*. An* $\bar{\alpha}$-raised signature *of* Σ*, written* $\Sigma^{\bar{\alpha}}$*, is the set*

$$\{h'_1 : \alpha_1 \to \ldots \to \alpha_m \to \tau_1, \ldots, h'_n : \alpha_1 \to \ldots \to \alpha_m \to \tau_n\}$$

where the variables $h'_1, \ldots, h'_n$ *are chosen to be distinct from the ones in* Σ*. Let* $\bar{y}$ *be the list of pairwise distinct variables* $y_1 : \alpha_1, \ldots, y_n : \alpha_m$*, which are not already in* Σ *and* $\Sigma^{\bar{\alpha}}$*, and let* θ *be the substitution*

$$[(h'_1\, y_1 \cdots y_m)/h_1, \ldots, (h'_n\, y_1 \cdots y_m)/h_n].$$

Given a sequent

$$\Sigma\,;\,\sigma_1 \rhd B_1, \ldots, \sigma_n \rhd B_n \longrightarrow \sigma_0 \rhd B_0$$

such that variables in $\sigma_0, \ldots, \sigma_n$ *are all different from* $\Sigma^{\bar{\alpha}}$ *and* $\bar{y}$*, we define the* $\bar{\alpha}$-raised sequent *as the sequent*

$$\Sigma^{\bar{\alpha}}\,;\,\bar{y}\sigma_1 \rhd B_1\theta, \ldots, \bar{y}\sigma_n \rhd B_n\theta \longrightarrow \bar{y}\sigma_0 \rhd B_0\theta$$

Lemma 3.16. *Let* Π *be a proof of a sequent* $\mathcal{S}$*. Then for every list of types* $\bar{\alpha}$*, there is a proof* $\Pi^{\uparrow\bar{\alpha}}$ *of the* $\bar{\alpha}$*-raised sequent* $\mathcal{S}'$ *of* $\mathcal{S}$ *such that* $\mathrm{ht}(\Pi^{\uparrow\bar{\alpha}}) \leq \mathrm{ht}(\Pi)$ *and* $\mathrm{ind}(\Pi^{\uparrow\bar{\alpha}}) \leq \mathrm{ind}(\Pi)$*.*

Proof We apply Lemma 3.11 to introduce the list of new variables $\bar{y}$ of type $\bar{\alpha}$ to the global context, and then apply the substitution θ (as defined in Definition 3.15), followed by weakening of the scope of $\bar{y}$ by Lemma 3.12. Since each step in the construction does not increase the measures of the derivation, the end derivation does not have higher measure than the original one. ∎

Definition 3.17. *A derivation* Π *is a* raised instance *of a derivation* Ξ *if* Π *is obtained from* Ξ *by raising the eigenvariables in the end sequent of* Ξ *(as described in the construction in Lemma 3.16), followed by application of a substitution to the resulting derivation.*

Lemma 3.18. *If* Π *is a raised instance of* Ξ*, then* $\mathrm{ht}(\Pi) \leq \mathrm{ht}(\Xi)$ *and* $\mathrm{ind}(\Pi) \leq \mathrm{ind}(\Xi)$*.*

3.4 Unfolding of derivations

DEFINITION 3.19. Inductive unfolding. *Let* $p\,\bar{x} \overset{\mu}{=} B\,p\,\bar{x}$ *be an inductive definition. Let* Π *be a derivation of* $\Sigma\,;\,\Gamma \longrightarrow \mathcal{C}\,p$ *where* p *occurs strictly positively in* $\mathcal{C}\,p$*. Let* S *be a closed term of the same type as* p *and let* Π_S *be a derivation of the sequent*

$$\bar{x}\,;\,B\,S\,\bar{x} \longrightarrow S\,\bar{x}.$$

We define the derivation $\mu(\Pi, \Pi_S)$ *of* $\Sigma\,;\,\Gamma \longrightarrow \mathcal{C}\,S$ *as follows.*

If $\mathcal{C}\,p = \mathcal{C}\,S$*, then* $\mu(\Pi, \Pi_S) = \Pi$*. Otherwise, we define* $\mu(\Pi, \Pi_S)$ *based on the last rule in* Π*.*

1. *Suppose* Π *ends with init*

$$\dfrac{}{\Sigma\,;\,\bar{y} \rhd p\,\bar{t} \longrightarrow \bar{y} \rhd p\,\bar{t}}\;\mathit{init}.$$

Then $\mu(\Pi, \Pi_S)$ *is the derivation*

$$\dfrac{\overset{\Pi_S}{\bar{x}\,;\,B\,S\,\bar{x} \longrightarrow S\,\bar{x}} \quad \overset{\text{Id}}{\Sigma\,;\,S\,\bar{t} \longrightarrow S\,\bar{t}}}{\Sigma\,;\,\bar{y} \rhd p\,\bar{t} \longrightarrow \bar{y} \rhd S\,\bar{t}}\;\mu\mathcal{L}$$

2. *Suppose* Π *ends with* $\supset \mathcal{L}$

$$\dfrac{\overset{\Pi_1}{\Sigma\,;\,\Gamma' \longrightarrow \bar{y} \rhd D_1} \quad \overset{\Pi_2}{\Sigma\,;\,\bar{y} \rhd D_2, \Gamma' \longrightarrow \mathcal{C}\,p}}{\Sigma\,;\,\bar{y} \rhd D_1 \supset D_2, \Gamma' \longrightarrow \mathcal{C}\,p}\;\supset \mathcal{L}$$

Then $\mu(\Pi, \Pi_S)$ *is the derivation*

$$\dfrac{\overset{\Pi_1}{\Sigma\,;\,\Gamma' \longrightarrow \bar{y} \rhd D_1} \quad \overset{\mu(\Pi_2, \Pi_S)}{\Sigma\,;\,\bar{y} \rhd D_2, \Gamma' \longrightarrow \mathcal{C}\,S}}{\Sigma\,;\,\bar{y} \rhd D_1 \supset D_2, \Gamma' \longrightarrow \mathcal{C}\,S}\;\supset \mathcal{L}$$

3. *Suppose* Π *ends with* $\supset \mathcal{R}$

$$\dfrac{\overset{\Pi'}{\Sigma\,;\,\Gamma, \bar{y} \rhd C_1\,p \longrightarrow \bar{y} \rhd C_2\,p}}{\Sigma\,;\,\Gamma \longrightarrow \bar{y} \rhd C_1\,p \supset C_2\,p}\;\supset \mathcal{R}$$

By the restriction on the occurences of p *in* $\mathcal{C}\,p$*, it must be the case that* $C_1\,p = C_1\,S$*. The derivation* $\mu(\Pi, \Pi_S)$ *is then defined as follows.*

$$\dfrac{\overset{\mu(\Pi', \Pi_S)}{\Sigma\,;\,\Gamma, \bar{y} \rhd C_1\,p \longrightarrow \bar{y} \rhd C_2\,S}}{\Sigma\,;\,\Gamma \longrightarrow \bar{y} \rhd C_1\,p \supset C_2\,S}\;\supset \mathcal{R}$$

4. *Suppose* Π *ends with* mc

$$\dfrac{\overset{\Pi_1}{\Sigma ; \Delta_1 \longrightarrow \mathcal{B}_1} \quad \dots \quad \overset{\Pi_m}{\Sigma ; \Delta_m \longrightarrow \mathcal{B}_m} \quad \overset{\Pi'}{\Sigma ; \mathcal{B}_1, \dots, \mathcal{B}_m, \Gamma' \longrightarrow \mathcal{C}\, p}}{\Sigma ; \Delta_1, \dots, \Delta_m, \Gamma' \longrightarrow \mathcal{C}\, p}\ mc$$

Then $\mu(\Pi, \Pi_S)$ *is*

$$\dfrac{\overset{\Pi_1}{\Sigma ; \Delta_1 \longrightarrow \mathcal{B}_1} \quad \dots \quad \overset{\Pi_m}{\Sigma ; \Delta_m \longrightarrow \mathcal{B}_m} \quad \overset{\mu(\Pi', \Pi_S)}{\Sigma ; \mathcal{B}_1, \dots, \mathcal{B}_m, \Gamma' \longrightarrow \mathcal{C}\, S}}{\Sigma ; \Delta_1, \dots, \Delta_m, \Gamma' \longrightarrow \mathcal{C}\, S}\ mc$$

5. *Suppose* Π *ends with* $\mu\mathcal{L}$ *on some predicate* q *given a definition clause* $q\,\bar{z} \overset{\mu}{=} D\,q\,\bar{z}$.

$$\dfrac{\overset{\Psi}{\bar{z} ; D\,I\,\bar{z} \longrightarrow I\,\bar{z}} \quad \overset{\Pi'}{\Sigma ; \bar{y} \rhd I\,\bar{t}, \Gamma' \longrightarrow \mathcal{C}\, p}}{\Sigma ; \bar{y} \rhd q\,\bar{t}, \Gamma' \longrightarrow \mathcal{C}\, p}\ \mu\mathcal{L}$$

Then $\mu(\Pi, \Pi_S)$ *is the derivation*

$$\dfrac{\overset{\Psi}{\bar{z} ; D\,I\,\bar{z} \longrightarrow I\,\bar{z}} \quad \overset{\mu(\Pi', \Pi_S)}{\Sigma ; \bar{y} \rhd I\,\bar{t}, \Gamma' \longrightarrow C\, S}}{\Sigma ; \bar{y} \rhd q\,\bar{t}, \Gamma' \longrightarrow \mathcal{C}\, S}\ \mu\mathcal{L}$$

6. *Suppose* Π *ends with* $\mu\mathcal{R}$

$$\dfrac{\overset{\Pi'}{\Sigma ; \Gamma \longrightarrow \bar{y} \rhd B\,p\,\bar{t}}}{\Sigma ; \Gamma \longrightarrow \bar{y} \rhd p\,\bar{t}}\ \mu\mathcal{R}$$

where $\bar{t} = t_1, \dots, t_m$. *Then* $\mu(\Pi, \Pi_S)$ *is the derivation*

$$\dfrac{\overset{\mu(\Pi', \Pi_S)}{\Sigma ; \Gamma \longrightarrow \bar{y} \rhd B\,S\,\bar{t}} \quad \overset{\Xi}{\Sigma ; \bar{y} \rhd B\,S\,\bar{t} \longrightarrow \bar{y} \rhd S\,\bar{t}}}{\Sigma ; \Gamma \longrightarrow \bar{y} \rhd S\,\bar{t}}\ mc$$

where the derivation Ξ *is a raised instance of* Π_S.

7. *If* Π *ends with any other rules, and has premise derivations* $\{\Pi_i\}_{i \in \mathcal{I}}$ *for some index set* $\mathcal{I}$, *then* $\mu(\Pi, \Pi_S)$ *also ends with the same rule and has premise derivations* $\{\mu(\Pi_i, \Pi_S)\}_{i \in \mathcal{I}}$.

Definition 3.20. Co-inductive unfolding. *Let* $p\,\bar{x} \overset{\nu}{=} B\,p\,\bar{x}$ *be a co-inductive definition. Let* S *be a closed term of the same type as* p *and let* Π_S *be a derivation of*

$$\bar{x} ; S\,\bar{x} \longrightarrow B\,S\,\bar{x}.$$

Let $\mathcal{C}\,p$ be a judgment such that $\mathrm{lvl}(\mathcal{C}\,p) \leq \mathrm{lvl}(p)$*, and let* Π *be a derivation of* $\Sigma\,;\Gamma \longrightarrow \mathcal{C}\,S$*. We define the derivation* $\nu(\Pi, \Pi_S)$ *of* $\Sigma\,;\Gamma \longrightarrow \mathcal{C}\,p$ *as follows.*

If $\mathcal{C}\,p = \mathcal{C}\,S$*, then* $\nu(\Pi, \Pi_S) = \Pi$*. If* $\mathcal{C}\,p = \bar{y} \triangleright p\,\bar{t}$ *then* $\mathcal{C}\,S = \bar{y} \triangleright S\,\bar{t}$ *and* $\nu(\Pi, \Pi_S)$ *is the derivation*

$$\dfrac{\overset{\Pi}{\Sigma\,;\Gamma \longrightarrow \bar{y} \triangleright S\,\bar{t}} \quad \overset{\Pi_S}{\bar{x}\,; S\,\bar{x} \longrightarrow B\,S\,\bar{x}}}{\Sigma\,;\Gamma \longrightarrow \bar{y} \triangleright p\,\bar{t}}\ \nu\mathcal{R}$$

Otherwise, we define $\nu(\Pi, \Pi_S)$ *based on the last rule in* Π*.*

1. *Suppose* Π *ends with* $\supset\mathcal{L}$

$$\dfrac{\overset{\Pi_1}{\Sigma\,;\Gamma' \longrightarrow \bar{y} \triangleright D_1} \quad \overset{\Pi_2}{\Sigma\,;\bar{y} \triangleright D_2, \Gamma' \longrightarrow \mathcal{C}\,S}}{\Sigma\,;\bar{y} \triangleright D_1 \supset D_2, \Gamma' \longrightarrow \mathcal{C}\,S}\ \supset\mathcal{L}$$

Then $\nu(\Pi, \Pi_S)$ *is the derivation*

$$\dfrac{\overset{\Pi_1}{\Sigma\,;\Gamma' \longrightarrow \bar{y} \triangleright D_1} \quad \overset{\nu(\Pi_2, \Pi_S)}{\Sigma\,;\bar{y} \triangleright D_2, \Gamma' \longrightarrow \mathcal{C}\,p}}{\Sigma\,;\bar{y} \triangleright D_1 \supset D_2, \Gamma' \longrightarrow \mathcal{C}\,p}\ \supset\mathcal{L}$$

2. *Suppose* Π *ends with* $\supset\mathcal{R}$

$$\dfrac{\overset{\Pi'}{\Sigma\,;\Gamma, \bar{y} \triangleright C_1\,S \longrightarrow \bar{y} \triangleright C_2\,S}}{\Sigma\,;\Gamma \longrightarrow \bar{y} \triangleright C_1\,S \supset C_2\,S}\ \supset\mathcal{R}$$

Since $\mathrm{lvl}(\mathcal{C}\,p) \leq \mathrm{lvl}(p)$*, it must be the case that* p *occurs strictly positively in* $\mathcal{C}\,p$*, hence* $C_1\,S = C_1\,p$*. Therefore we construct the derivation* $\nu(\Pi, \Pi_S)$ *as follows.*

$$\dfrac{\overset{\nu(\Pi', \Pi_S)}{\Sigma\,;\Gamma, \bar{y} \triangleright C_1\,S \longrightarrow \bar{y} \triangleright C_2\,p}}{\Sigma\,;\Gamma \longrightarrow \bar{y} \triangleright C_1\,S \supset C_2\,p}\ \supset\mathcal{R}$$

3. *Suppose* Π *ends with mc*

$$\dfrac{\overset{\Pi_1}{\Sigma\,;\Delta_1 \longrightarrow \mathcal{B}_1} \quad \ldots \quad \overset{\Pi_m}{\Sigma\,;\Delta_m \longrightarrow \mathcal{B}_m} \quad \overset{\Pi'}{\Sigma\,;\mathcal{B}_1, \ldots, \mathcal{B}_m, \Gamma' \longrightarrow \mathcal{C}\,S}}{\Sigma\,;\Delta_1, \ldots, \Delta_m, \Gamma' \longrightarrow \mathcal{C}\,S}\ mc$$

Then $\nu(\Pi, \Pi_S)$ *is*

$$\dfrac{\overset{\Pi_1}{\Sigma\,;\Delta_1 \longrightarrow \mathcal{B}_1} \quad \ldots \quad \overset{\Pi_m}{\Sigma\,;\Delta_m \longrightarrow \mathcal{B}_m} \quad \overset{\nu(\Pi', \Pi_S)}{\Sigma\,;\mathcal{B}_1, \ldots, \mathcal{B}_m, \Gamma' \longrightarrow \mathcal{C}\,p}}{\Sigma\,;\Delta_1, \ldots, \Delta_m, \Gamma' \longrightarrow \mathcal{C}\,p}\ mc$$

4. *Suppose* Π *ends with* $\mu\mathcal{L}$ *on a predicate* $q\,\bar t$*, given an inductive definition* $q\,\bar z \overset{\mu}{=} D\,q\,\bar z$*.*

$$\dfrac{\overset{\Psi}{\bar z\,;\,D\,I\,\bar z \longrightarrow I\,\bar z} \qquad \overset{\Pi'}{\Sigma\,;\,\bar y \rhd I\,\bar t, \Gamma' \longrightarrow \mathcal{C}\,S}}{\Sigma\,;\,\bar y \rhd q\,\bar t, \Gamma' \longrightarrow \mathcal{C}\,S}\;\mu\mathcal{L}$$

Then $\nu(\Pi, \Pi_S)$ *is the derivation*

$$\dfrac{\overset{\Psi}{\bar z\,;\,D\,I\,\bar z \longrightarrow I\,\bar z} \qquad \overset{\nu(\Pi', \Pi_S)}{\Sigma\,;\,\bar y \rhd I\,\bar t, \Gamma' \longrightarrow \mathcal{C}\,p}}{\Sigma\,;\,q\,\bar t, \Gamma' \longrightarrow \mathcal{C}\,p}\;\mu\mathcal{L}$$

5. *If* Π *ends with any other rules, and has premise derivations* $\{\Pi_i\}_{i\in\mathcal{I}}$ *for some index set* $\mathcal{I}$*, then* $\nu(\Pi, \Pi_S)$ *also ends with the same rule and has premise derivations* $\{\nu(\Pi_i, \Pi_S)\}_{i\in\mathcal{I}}$*.*

Lemma 3.21. *The derivations* $\mu(\Pi, \Pi_S)\theta$ *and* $\mu(\Pi\theta, \Pi_S)$ *are the same derivation.*

Proof Suppose Π is a derivation of the sequent $\Sigma\,;\,\Gamma \longrightarrow \mathcal{C}p$ and Π_S is a derivation of the sequent $\bar x\,;\,B\,S\,\bar x \longrightarrow S\,\bar x$, given an inductive definition $p\,\bar x \overset{\mu}{=} B\,p\,\bar x$ and an invariant S.

The proof is by induction on $\mathrm{ht}(\Pi)$ and case analysis on $\mathcal{C}$. We make use of the fact that S is a closed term and hence unaffected by the substitution θ, i.e., $S\theta = S$. Most cases follow immediately from inductive hypothesis. We show here the interesting ones. In the following, we omit some information from the sequents when they do not play active role during the transformation steps.

1. Suppose Π ends with $\supset\mathcal{R}$, then $\mathcal{C}\,p = \bar y \rhd C_1\,p \supset C_2\,p$. Since p has only strictly positive occurrences in $\mathcal{C}\,p$, it is the case that $\mathcal{C}\,S = \bar y \rhd C_1\,p \supset C_2\,S$. We show that applying unfolding and substitution in any order produces the same derivation.

$$\begin{tikzcd}
\dfrac{\overset{\Pi'}{\ldots, \bar y \rhd C_1\,p \longrightarrow \bar y \rhd C_2\,p}}{\ldots \longrightarrow \bar y \rhd C_1\,p \supset C_2\,p}\supset\mathcal{R} \arrow[r, Rightarrow, "\mu", "1"'] \arrow[d, Rightarrow, "2\;\theta"] & \dfrac{\overset{\mu(\Pi', \Pi_S)}{\ldots, \bar y \rhd C_1\,p \longrightarrow \bar y \rhd C_2\,S}}{\ldots \longrightarrow \bar y \rhd C_1\,p \supset C_2\,S}\supset\mathcal{R} \arrow[d, Rightarrow, "3\;\theta"] \\
\dfrac{\overset{\Pi'\theta}{\ldots, (\bar y \rhd C_1\,p)\theta \longrightarrow (\bar y \rhd C_2\,p)\theta}}{\ldots \longrightarrow (\bar y \rhd C_1\,p \supset C_2\,p)\theta}\supset\mathcal{R} \arrow[r, Rightarrow, "\mu", "4"'] & \dfrac{\overset{\mu(\Pi'\theta, \Pi_S)}{\ldots, (\bar y \rhd C_1\,p)\theta \longrightarrow (\bar y \rhd C_2\,S)\theta}}{\ldots \longrightarrow (\bar y \rhd C_1\,p \supset C_2\,S)\theta}\supset\mathcal{R}
\end{tikzcd}$$

In Step 3, we apply the inductive hypothesis $\mu(\Pi', \Pi_S)\theta = \mu(\Pi'\theta, \Pi_S)$.

2. Suppose Π ends with eq$\mathcal{L}$.

$$\dfrac{\left\{\begin{array}{c}\Pi^{\rho}\\ \Gamma'\rho \longrightarrow (\mathcal{C}\,p)\rho\end{array}\right\}_{\rho}}{\bar{y} \rhd s = t, \Gamma'\,;\,\mathcal{C}\,p \longrightarrow}\ \mathrm{eq}\mathcal{L} \qquad \overset{\mu}{\Longrightarrow}_1 \qquad \dfrac{\left\{\begin{array}{c}\mu(\Pi^{\rho}, \Pi_S)\\ \Gamma'\rho \longrightarrow (\mathcal{C}\,S)\rho\end{array}\right\}_{\rho}}{\bar{y} \rhd s = t, \Gamma' \longrightarrow \mathcal{C}\,S}\ \mathrm{eq}\mathcal{L}$$

$$\Downarrow_2 \theta \qquad\qquad\qquad\qquad \Downarrow_3 \theta$$

$$\dfrac{\left\{\begin{array}{c}\Pi^{\theta\circ\rho'}\\ \Gamma'\theta\rho' \longrightarrow (\mathcal{C}\,p)\theta\rho'\end{array}\right\}_{\rho'}}{(\bar{y} \rhd s = t)\theta, \Gamma'\theta \longrightarrow (\mathcal{C}\,p)\theta}\ \mathrm{eq}\mathcal{L} \qquad \overset{\mu}{\Longrightarrow}_4 \qquad \dfrac{\left\{\begin{array}{c}\mu(\Pi^{\theta\circ\rho'}, \Pi_S)\\ \Gamma'\theta\rho' \longrightarrow (\mathcal{C}\,S)\theta\rho'\end{array}\right\}_{\rho'}}{(\bar{y} \rhd s = t)\theta, \Gamma'\theta \longrightarrow (\mathcal{C}\,S)\theta}\ \mathrm{eq}\mathcal{L}$$

In the transformation in Step 3, we apply eq$\mathcal{L}$ to $(\bar{y} \rhd s = t)\theta$. Note that, by Definition 3.1, applying substitution to a derivation ending with eq$\mathcal{L}$ does not change or add new premises. Therefore the premise derivations of $\mu(\Pi, \Pi_S)\theta$ are already contained in the premise set of $\mu(\Pi, \Pi_S)$. Hence inductive hypothesis is not needed in this case. And since in Step 2 we apply eq$\mathcal{L}$ to the same judgment $(\bar{y} \rhd s = t)\theta$, the set of unifiers obtained in Step 2 and Step 3 are exactly the same and therefore applying Step 2 and 4 produces the same derivations as applying Step 1 and 3.

3. Suppose Π ends with $\mu\mathcal{R}$ and $\mathcal{C}\,p = p\bar{u}$ for some $\bar{u}$.

$$\dfrac{\begin{array}{c}\Pi'\\ \Gamma \longrightarrow \bar{y} \rhd B\,p\,\bar{u}\end{array}}{\Gamma \longrightarrow \bar{y} \rhd p\,\bar{u}}\ \mu\mathcal{R} \qquad \overset{\mu}{\Longrightarrow}_1 \qquad \dfrac{\begin{array}{c}\mu(\Pi', \Pi_S)\\ \Gamma \longrightarrow \bar{y} \rhd B\,S\,\bar{u}\end{array} \quad \begin{array}{c}\Xi\\ \bar{y} \rhd B\,S\,\bar{u} \longrightarrow \bar{y} \rhd S\,\bar{u}\end{array}}{\Gamma \longrightarrow \bar{y} \rhd S\,\bar{u}}\ mc$$

$$\Downarrow_2 \theta \qquad\qquad\qquad\qquad \Downarrow_3 \theta$$

$$\dfrac{\begin{array}{c}\Pi'\theta\\ \Gamma\theta \longrightarrow (\bar{y} \rhd B\,p\,\bar{u})\theta\end{array}}{\Gamma\theta \longrightarrow (\bar{y} \rhd p\,\bar{u})\theta}\ \mu\mathcal{R} \qquad \overset{\mu}{\Longrightarrow}_4 \qquad \dfrac{\begin{array}{c}\mu(\Pi'\theta, \Pi_S)\\ \Gamma\theta \longrightarrow (\bar{y} \rhd B\,S\,\bar{u})\theta\end{array} \quad \begin{array}{c}\Xi'\\ (\bar{y} \rhd B\,S\,\bar{u})\theta \longrightarrow (\bar{y} \rhd S\,\bar{u})\theta\end{array}}{\Gamma\theta \longrightarrow (\bar{y} \rhd S\,\bar{u})\theta}$$

In Step 3, we apply the inductive hypothesis $\mu(\Pi', \Pi_S)\theta = \mu(\Pi'\theta, \Pi_S)$. From the construction of Ξ and Ξ' as defined in Definition 3.19, we can verify that $\Xi\theta = \Xi'$.

∎

Lemma 3.22. *The derivations* $\nu(\Pi, \Pi_S)\theta$ *and* $\nu(\Pi\theta, \Pi_S)$ *are the same derivation.*

Proof By induction on the measure ht(Π). Most cases are similar to the proof of Lemma 3.21. The only non-trivial case is when $\mathcal{C}\,p = \bar{y} \rhd p\,\bar{u}$, i.e., $\mathcal{C}\,S = \bar{y} \rhd S\,\bar{u}$. In this case, we have:

$$\begin{tikzcd}
\overset{\Pi}{\Gamma \longrightarrow \bar{y} \rhd S_i \bar{u}} \arrow[r, Rightarrow, "\nu", "1"'] \arrow[d, Rightarrow, "2\ \theta"] & \dfrac{\overset{\Pi}{\Gamma \longrightarrow \bar{y} \rhd S\,\bar{u}} \quad \overset{\Pi_S}{S\,\bar{x} \longrightarrow B\,S\,\bar{x}}}{\Gamma \longrightarrow \bar{y} \rhd p\,\bar{u}}\ \nu\mathcal{R} \arrow[d, Rightarrow, "3\ \theta"] \\
\overset{\Pi\theta}{\Gamma\theta \longrightarrow (\bar{y} \rhd S\,\bar{u})\theta} \arrow[r, Rightarrow, "\nu", "4"'] & \dfrac{\overset{\Pi\theta}{\Gamma\theta \longrightarrow (\bar{y} \rhd S\,\bar{u})\theta} \quad \overset{\Pi_S}{S\,\bar{x} \longrightarrow B\,S\,\bar{x}}}{\Gamma\theta \longrightarrow (\bar{y} \rhd p\,\bar{u})\theta}\ \nu\mathcal{R}
\end{tikzcd}$$

In Step 3, notice that since $\bar{x}$ are new eigenvariables, and hence not in the domain of θ (by the definition of substitution of derivations), $\Pi_S\theta = \Pi_S$. ∎

The unfolding of derivations obviously commutes with weakening of global signatures.

Lemma 3.23. *The derivation $\mu(\Sigma : \Pi, \Pi_S)$ and $\Sigma : \mu(\Pi, \Pi_S)$ are the same derivation.*

Lemma 3.24. *The derivation $\nu(\Sigma : \Pi, \Pi_S)$ and $\Sigma : \nu(\Pi, \Pi_S)$ are the same derivation.*

3.5 Logical equivalence

Logical equivalence is defined via implications, i.e., $B \equiv C$ if and only if $B \supset C \wedge C \supset B$ is provable. For all known logics, logical equivalence is a congruence, that is, it is preserved by arbitrary context. Since we have introduced a new connective ∇ we need to verify whether this property still holds. We define a *context* as a formula with a hole [.], denoted by $C[.]$. Given a context $C[.]$, $C[A]$ denotes a formula obtained from $C[.]$ by replacing the hole [.] with A. Note that this replacement is a textual replacement, hence there can be variable capture, i.e., free variables in A become bound in $C[A]$.

Proposition 3.25. *The logical equivalence relation $\equiv$ is a congruence relation, i.e., if $A \equiv B$ then $C[A] \equiv C[B]$ for arbitrary context $C[.]$.*

Proof The proof is by induction on the size of the context $C[.]$, i.e, the number of logical connectives in $C[.]$. The non-trivial case is of course when $C[.] = \nabla x.C'[.]$. By induction hypothesis we know that $C'[A] \equiv C'[B]$. We distinguish two cases based on the occurrence of x in A and B. If x is not free in both A and B, by Lemma 3.14 we have a derivation $x \rhd (C'[A] \equiv C'[B])$ and hence by applying $\nabla\mathcal{R}$ rule we also have a derivation of $\nabla x.(C'[A] \equiv C'[B])$. By Proposition 2.1 we can push ∇ under implication and conjunction, therefore we get a derivation of $\nabla x.C'[A] \equiv \nabla x.C'[B]$. Otherwise, suppose x is free in A or B. The free variable is treated as an eigenvariable. In this case we apply Lemma 3.12 to weaken the scope of x so that we have a derivation of $x \rhd C'[A] \equiv C'[B]$ and hence also a derivation of $\nabla x.C'[A] \equiv \nabla x.C'[B]$. ∎

3.6 Horn definitions and ∇

We observe that the main difference between ∇ and $\forall$ is in the left-behavior. We therefore expect that ∇ and $\forall$ are interchangeable when the formula we are trying to prove does not make any use of left-rules, especially eq$\mathcal{L}$. In this section we look at the properties of *cut-free proofs* (i.e., proofs which do not make use of the *mc* rule), under a class of "essentially" Horn definitions, which is characterized by the absence of implications in the body of the definitions.

Definition 3.26. *An* hc-goal *(named for Horn clauses) is a formula built from the base set of logic connectives* $\top$*,* $\wedge$*,* $\vee$*, and* $\exists$*. An* $\mathsf{hc}^{\forall}$-goal *is a formula built from these connectives and* $\forall$*; an* hc^{∇}-goal *is a formula built from the base set and* ∇*; and an* $\mathsf{hc}^{\forall\nabla}$-goal *is a formula admitting those connectives as well as both* $\forall$ *and* ∇*. A definition is an* hc-definition *(resp.,* $\mathsf{hc}^{\forall}$-definition*,* hc^{∇}-definition*, and* $\mathsf{hc}^{\forall\nabla}$-definition*) if the body of all of its clauses are* hc-goals *(resp.,* $\mathsf{hc}^{\forall}$-goals*,* hc^{∇}-goals*, and* $\mathsf{hc}^{\forall\nabla}$-goals*).*

Notice that all of these kind of definitions are trivially stratifiable. We shall now show that when definitions are essentially Horn clauses, the difference between ∇ and $\forall$ cannot actually be observed. In particular, we show that ∇ and $\forall$ can be interchanged for $\mathsf{hc}^{\forall\nabla}$-definitions and $\mathsf{hc}^{\forall\nabla}$-goals without affecting provability. In proving this statement inductively we need a stronger hypothesis, that is, we can interchange the scope of variables in this case (either global or local) without affecting provability. We consider only the non co-inductive definitions, i.e., definitions which contain only regular or inductive definitions.

Lemma 3.27. *Let* $\mathbf{D}$ *be a non co-inductive* $\mathsf{hc}^{\forall\nabla}$*-definition, and let* G *be an* $\mathsf{hc}^{\forall\nabla}$*-goal. The sequent* $\Sigma\,;\,.\longrightarrow (\sigma_1, x, \sigma_2) \triangleright G$ *is cut-free provable if and only if the sequent*

$$\Sigma, h\,;\,.\longrightarrow (\sigma_1\sigma_2) \triangleright G[(h\,\sigma_1)/x]$$

is cut-free provable. Moreover, given a derivation Π *of the first sequent, there is a derivation* Π' *of the second sequent such that* $\mathrm{ht}(\Pi') \leq \mathrm{ht}(\Pi)$*, and vice versa.*

Proof One direction is an easy consequence of Lemma 3.12. In the other direction, we would like to construct the derivation Π' from the derivation Π. Let us assume that x is not in Σ. We examine the following non-trivial case.

Suppose Π ends with $\forall\mathcal{R}$

$$\cfrac{\begin{matrix}\Pi_1 \\ \Sigma, f\,;\,.\longrightarrow \sigma_1 x \sigma_2 \triangleright G'[(f\,\sigma_1 x \sigma_2)/y]\end{matrix}}{\Sigma\,;\,.\longrightarrow \sigma_1 x \sigma_2 \triangleright \forall y.G'}\;\forall\mathcal{R}.$$

By induction hypothesis there is a derivation Π_1' of

$$\Sigma, f, h\,;\,.\longrightarrow \sigma_1\sigma_2 \triangleright G'[(f\,\sigma_1 (h\,\sigma_1)\,\sigma_2)/y, (h\sigma_1)/x]$$

such that $\mathrm{ht}(\Pi_1') \leq \mathrm{ht}(\Pi_1)$. We then apply the substitution $[(\lambda\sigma_1\lambda x\lambda\sigma_2.f'\,\sigma_1\sigma_2)/f]$ to remove the dependency of f on x. The derivation Π' is then constructed as follows.

$$\dfrac{\begin{array}{c}\Pi_1'[(\lambda\sigma_1\lambda x\lambda\sigma_2.f'\,\sigma_1\sigma_2)/f]\\ \Sigma, h, f'\,;\,.\longrightarrow \sigma_1\sigma_2 \rhd G'[(f'\,\sigma_1\sigma_2)/y, (h\,\sigma_1)/x]\end{array}}{\Sigma, h\,;\,.\longrightarrow \sigma_1\sigma_2 \rhd \forall y.G'[(h\,\sigma_1)/x]}\;\forall\mathcal{R}$$

Note that substitution does not increase the height of the derivation Π_1', therefore $\mathrm{ht}(\Pi') = \mathrm{ht}(\Pi_1'\theta) + 1 \leq \mathrm{ht}(\Pi)$.

∎

Notice that we restrict the definition $\mathbf{D}$ to a non co-inductive one. This is because in a proof involving co-induction, there could be a replacement of a co-inductive predicate with arbitrary invariants which may not be Horn goals.

The following proposition is an expected consequence of the preceeding lemma.

PROPOSITION 3.28. *Let* $\mathbf{D}$ *be a* non co-inductive $\mathsf{hc}^{\forall\nabla}$*-definition and let* $\mathbf{D}'$ *be the non co-inductive* $\mathsf{hc}^{\forall\nabla}$*-definition resulting from replacing some occurrences of* $\forall$ *and* ∇ *in in the body of clauses of* $\mathbf{D}$ *with* ∇ *and* $\forall$*, respectively. Similarly, let* G *be an* $\mathsf{hc}^{\forall\nabla}$*-goal and let* G' *be the* $\mathsf{hc}^{\forall\nabla}$*-goal resulting from replacing some occurrences of* $\forall$ *and* ∇ *in* G *with* ∇ *and* $\forall$*, respectively. If the sequent* $\Sigma\,;\,\cdot \longrightarrow \sigma \rhd G$ *is cut-free provable using definition* $\mathbf{D}$ *then the sequent* $\Sigma\,;\,\cdot \longrightarrow \sigma \rhd G'$ *is cut-free provable using definition* $\mathbf{D}'$*.*

Proof Let Π be a derivation of $\Sigma\,;\,\cdot \longrightarrow \sigma \rhd G$. We construct a derivation Π' of $\Sigma\,;\,\cdot \longrightarrow \sigma \rhd G'$ by induction on the measure $\mathrm{ht}(\Pi)$. The non-trivial cases are when Π ends with the introduction rule for the connective being interchanged.

Suppose $G = \forall x.H$, $G' = \nabla x.H'$ and Π ends with $\forall\mathcal{R}$.

$$\dfrac{\begin{array}{c}\Pi_1\\ \Sigma, h\,;\,.\longrightarrow \sigma \rhd H[(h\,\sigma)/x]\end{array}}{\Sigma\,;\,.\longrightarrow \sigma \rhd \forall x.H}\;\forall\mathcal{R}$$

By Lemma 3.27 there is a derivation Π_1' of $\Sigma\,;\,.\longrightarrow (\sigma, x)\rhd H$ such that $\mathrm{ht}(\Pi_1') \leq \mathrm{ht}(\Pi_1)$. We can therefore apply the induction hypothesis to Π_1' to get a derivation Π_2 of $\Sigma\,;\,.\longrightarrow (\sigma, x) \rhd H'$. The derivation Π' is therefore

$$\dfrac{\begin{array}{c}\Pi_2\\ \Sigma\,;\,.\longrightarrow (\sigma, x) \rhd H'\end{array}}{\Sigma\,;\,.\longrightarrow \sigma \rhd \nabla x.H'}\;\nabla\mathcal{R}$$

The case where $G = \nabla x.H$, $G' = \forall x.H'$ and Π ends with $\nabla\mathcal{R}$ is done analogously, since Lemma 3.27 works on both directions. ∎

3.7 Proof search

In the search for a cut-free proof of a sequent, there can be more than one choice of the applicable rules at a time. Some of these rules can always be applied without affecting the cut-free provability of the sequent; we call these rules *asynchronous* rules, and the other rules are called *synchronous*. The more general notion of synchrony and asynchrony in proof search has been already studied in the literature (see e.g., [3]), and these notions are usually tied to connectives rather than rules. Our attempt to classify rules as synchronous and asynchronous for logical connectives is of more limited scope. We do not attempt, for example, to devise a notion of uniform provability [36, 3], although the following results will be used to establish certain uniform proof search in limited settings.

DEFINITION 3.29. *A rule $\mathcal{R}$ is an* asynchronous rule *if for every sequent $\Gamma \longrightarrow \mathcal{C}$ such that $\mathcal{R}$ is applicable to the sequent and such that there is a cut-free proof of the sequent, there is a cut-free proof of the same sequent ending with $\mathcal{R}$.*

PROPOSITION 3.30. *The rules $\wedge\mathcal{R}$, $\wedge\mathcal{L}^*$, $\vee\mathcal{L}$, $\supset\mathcal{R}$, $\top\mathcal{R}$, $\bot\mathcal{L}$, $\exists\mathcal{L}$, $\forall\mathcal{R}$, $\nabla\mathcal{L}$, $\nabla\mathcal{R}$, $def\mathcal{L}$, $def\mathcal{R}$, $\mathrm{eq}\mathcal{L}$, $\mathrm{eq}\mathcal{R}$, $\nu\mathcal{L}$, $\mu\mathcal{R}$, $\mathrm{eq}\mathcal{L}^{csu}$, $def\mathcal{L}^{=}$ and $def\mathcal{L}^{csu}$ are asynchronous rules.*

Proof We make use of the cut-elimination result for Linc (Corollary 4.21 in Chapter 4). Suppose we have a cut-free derivation Π of the sequent $\Gamma \longrightarrow \mathcal{C}$. For each case we construct a cut-free derivation Π' ending with the respected rule. The cases with $\mathrm{eq}\mathcal{R}$, $\top\mathcal{R}$, $\bot\mathcal{L}$ are trivial. The rules $def\mathcal{L}^{=}$ and $def\mathcal{L}^{csu}$ are derivable from $def\mathcal{L}$, $\mathrm{eq}\mathcal{L}$, $\exists\mathcal{L}$ and $\mathrm{eq}\mathcal{L}^{csu}$, and hence their asynchrony follows from those rules. The cases with $\nu\mathcal{L}$ and $\mu\mathcal{R}$ are the same as $def\mathcal{L}$ and $def\mathcal{R}$, but in these cases we use the fact that $def\mathcal{L}$ and $def\mathcal{R}$ are derived rules for co-inductive and inductive definitions (Proposition 2.5). The case with $\mathrm{eq}\mathcal{L}^{csu}$ is the same as $\mathrm{eq}\mathcal{L}$. It remains to check the following cases.

$\wedge\mathcal{R}$**:** Suppose $\mathcal{C}$ is $\mathcal{B} \wedge \mathcal{D}$. We construct a derivation Ξ as follows.

$$\dfrac{\dfrac{\begin{matrix}\Pi\\ \Gamma \longrightarrow \mathcal{B}\wedge\mathcal{D}\end{matrix} \quad \dfrac{\dfrac{}{\mathcal{B}\longrightarrow\mathcal{B}}\,init}{\mathcal{B}\wedge\mathcal{D}\longrightarrow\mathcal{B}}\,\wedge\mathcal{L}}{\Gamma\longrightarrow\mathcal{B}}\,mc \qquad \dfrac{\begin{matrix}\Pi\\ \Gamma \longrightarrow \mathcal{B}\wedge\mathcal{D}\end{matrix} \quad \dfrac{\dfrac{}{\mathcal{D}\longrightarrow\mathcal{D}}\,init}{\mathcal{B}\wedge\mathcal{D}\longrightarrow\mathcal{D}}\,\wedge\mathcal{L}}{\Gamma\longrightarrow\mathcal{D}}\,mc}{\Gamma\longrightarrow\mathcal{B}\wedge\mathcal{D}}\,\wedge\mathcal{R}$$

By cut-elimination, there is a cut-free derivation Π_1 of $\Gamma \longrightarrow \mathcal{B}$ and a cut-free derivation Π_2 of $\Gamma \longrightarrow \mathcal{D}$. Therefore we construct the derivation Π' as follows.

$$\dfrac{\begin{matrix}\Pi_1\\ \Gamma\longrightarrow\mathcal{B}\end{matrix} \quad \begin{matrix}\Pi_2\\ \Gamma\longrightarrow\mathcal{D}\end{matrix}}{\Gamma\longrightarrow\mathcal{B}\wedge\mathcal{D}}\,\wedge\mathcal{R}$$

$\wedge\mathcal{L}^*$**:** Suppose Γ is $\mathcal{B}\wedge\mathcal{D},\Gamma'$. We construct the derivation Π' by cut-elimination on the following derivation.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathcal{B}\longrightarrow\mathcal{B}}\,\mathit{init}}{\mathcal{B},\mathcal{D}\longrightarrow\mathcal{B}}\,w\mathcal{L}\qquad\dfrac{\dfrac{}{\mathcal{D}\longrightarrow\mathcal{D}}\,\mathit{init}}{\mathcal{B},\mathcal{D}\longrightarrow\mathcal{D}}\,w\mathcal{L}}{\mathcal{B},\mathcal{D}\longrightarrow\mathcal{B}\wedge\mathcal{D}}\,\wedge\mathcal{R}\qquad\begin{array}{c}\Pi\\ \mathcal{B}\wedge\mathcal{D},\Gamma'\longrightarrow\mathcal{C}\end{array}}{\mathcal{B},\mathcal{D},\Gamma'\longrightarrow\mathcal{C}}\,mc}{\mathcal{B}\wedge\mathcal{D},\Gamma'\longrightarrow\mathcal{C}}\,\wedge\mathcal{L}^*$$

$\vee\mathcal{L}$**:** Symmetric to the $\wedge\mathcal{R}$-case.

$\supset\mathcal{R}$**:** Suppose $\mathcal{C}$ is $\mathcal{B}\supset\mathcal{D}$. The derivation Π' is obtained by cut-elimination on the following derivation.

$$\dfrac{\dfrac{\begin{array}{c}\Pi\\ \Gamma\longrightarrow\mathcal{B}\supset\mathcal{D}\end{array}\qquad\dfrac{\dfrac{}{\mathcal{B}\longrightarrow\mathcal{B}}\,\mathit{init}\qquad\dfrac{\dfrac{}{\mathcal{D}\longrightarrow\mathcal{D}}\,\mathit{init}}{\mathcal{D},\mathcal{B}\longrightarrow\mathcal{D}}\,w\mathcal{L}}{\mathcal{B}\supset\mathcal{D},\mathcal{B}\longrightarrow\mathcal{D}}\,\supset\mathcal{L}}{\Gamma,\mathcal{B}\longrightarrow\mathcal{D}}\,mc}{\Gamma\longrightarrow\mathcal{B}\supset\mathcal{D}}\,\supset\mathcal{R}$$

$\exists\mathcal{L}$**:** Suppose Γ is $\bar{x}\triangleright\exists y.By,\Gamma'$. Then Π' is obtained by cut-elimination on the derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Sigma,h\,;\,\bar{x}\triangleright B\ (h\,\bar{x})\longrightarrow\bar{x}\triangleright B\ (h\,\bar{x})}\,\mathit{init}}{\Sigma,h\,;\,\bar{x}\triangleright B\ (h\,\bar{x})\longrightarrow\bar{x}\triangleright\exists y.By}\,\exists\mathcal{R}\qquad\begin{array}{c}h\!:\!\Pi\\ \Sigma,h\,;\,\bar{x}\triangleright\exists y.By,\Gamma'\longrightarrow\mathcal{C}\end{array}}{\Sigma,h\,;\,\bar{x}\triangleright B\ (h\,\bar{x}),\Gamma'\longrightarrow\mathcal{C}}\,mc}{\Sigma\,;\,\bar{x}\triangleright\exists y.By,\Gamma'\longrightarrow\mathcal{C}}\,\exists\mathcal{L}$$

$\forall\mathcal{R}$**:** Symmetric to the $\exists\mathcal{L}$ case.

$\nabla\mathcal{L}$**:** Suppose Γ is $\bar{x}\triangleright\nabla y.By,\Gamma'$. Then Π' is obtained by cut-elimination on the following derivation.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\bar{x}y\triangleright By\longrightarrow\bar{x}y\triangleright By}\,\mathit{init}}{\bar{x}y\triangleright By\longrightarrow\bar{x}\triangleright\nabla y.By}\,\nabla\mathcal{R}\qquad\begin{array}{c}\Pi\\ \bar{x}\triangleright\nabla y.By,\Gamma'\longrightarrow\mathcal{C}\end{array}}{\bar{x}y\triangleright By,\Gamma'\longrightarrow\mathcal{C}}\,mc}{\bar{x}\triangleright\nabla y.By,\Gamma'\longrightarrow\mathcal{C}}\,\nabla\mathcal{L}$$

$\nabla\mathcal{R}$**:** Symmetric to the $\nabla\mathcal{L}$ case.

def$\mathcal{L}$: Suppose Γ is $\bar{x} \triangleright p\,\bar{t}, \Gamma'$, where the predicate $p\,\bar{y}$ is defined by $B\,\bar{y}$. We construct Π' by cut-elimination on the following derivation.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\bar{x} \triangleright B\,\bar{t} \longrightarrow \bar{x} \triangleright B\,\bar{t}}\ \mathit{init}}{\bar{x} \triangleright B\,\bar{t} \longrightarrow \bar{x} \triangleright p\,\bar{t}}\ \mathit{def}\mathcal{R} \qquad \begin{array}{c}\Pi \\ \bar{x} \triangleright p\,\bar{t}, \Gamma' \longrightarrow \mathcal{C}\end{array}}{\bar{x} \triangleright B\,\bar{t}, \Gamma' \longrightarrow \mathcal{C}}\ \mathit{mc}}{\bar{x} \triangleright p\,\bar{t}, \Gamma' \longrightarrow \mathcal{C}}\ \mathit{def}\mathcal{L}$$

def$\mathcal{R}$: Symmetric to the *def$\mathcal{L}$* case.

eq$\mathcal{L}$: Suppose Γ is $\bar{x} \triangleright s = t, \Gamma$. Then we take Π' as the derivation

$$\dfrac{\left\{\begin{array}{c}\Pi^{\rho} \\ \Sigma\rho\,;\,\Gamma'\rho \longrightarrow \mathcal{C}\rho\end{array}\right\}_{\rho}}{\Sigma\,;\,\bar{x} \triangleright s = t, \Gamma' \longrightarrow \mathcal{C}}\ \mathrm{eq}\mathcal{L}$$

where Π^{ρ} is obtained by cut-elimination on the derivation

$$\dfrac{\dfrac{}{\cdot\,;\,\cdot \longrightarrow (\bar{x} \triangleright s = t)\rho}\ \mathrm{eq}\mathcal{R} \qquad \begin{array}{c}\Pi\rho \\ \Sigma\rho\,;\,(\bar{x} \triangleright s = t)\rho, \Gamma'\rho \longrightarrow \mathcal{C}\rho\end{array}}{\Sigma\rho\,;\,\Gamma'\rho \longrightarrow \mathcal{C}\rho}\ \mathit{mc}$$

∎

Chapter 4

Cut Elimination for Linc

In this chapter we prove the cut-elimination theorem for Linc, from which the consistency of the logic follows. The classic proof of cut-elimination for Gentzen's LK and LJ is done by induction on the size of the cut formula, i.e., the number of logical connectives in the formula. His cut-elimination proof is essentially a procedure for permuting cut-rule over other rules. During the permutation, a cut on a compound formula is reduced to several cuts on formulas of smaller size, which by induction hypothesis can be removed. For example, the derivation

$$\dfrac{\dfrac{\overset{\Pi_1}{\Delta \longrightarrow B_1} \quad \overset{\Pi_2}{\Delta \longrightarrow B_2}}{\Delta \longrightarrow B_1 \wedge B_2}\,\wedge\mathcal{R} \qquad \dfrac{\overset{\Pi'}{B_1, \Gamma \longrightarrow C}}{B_1 \wedge B_2, \Gamma \longrightarrow C}\,\wedge\mathcal{L}}{\Delta, \Gamma \longrightarrow C}\,mc$$

reduces to

$$\dfrac{\overset{\Pi_1}{\Delta \longrightarrow B_1} \quad \overset{\Pi'}{B_1, \Gamma \longrightarrow C}}{\Delta, \Gamma \longrightarrow C}\,mc \quad .$$

In the case of Linc, the use of definitions and (co-)induction complicates the reduction of cut. The eq$\mathcal{L}$ rule uses substitutions in the premise, hence when permuting up the cut rule over eq$\mathcal{L}$ we may need to apply the substitutions to some premise derivations above the cut. The way we handle this issue is similar to that of $FO\lambda^{\Delta\mathbb{N}}$ (see Chapter 3, Section 3.1). The induction and co-induction cases however pose a different problem. Consider for example a cut involving the induction rules

$$\dfrac{\dfrac{\overset{\Pi_1}{\Delta \longrightarrow B\,p\,t}}{\Delta \longrightarrow p\,t}\,\mu\mathcal{R} \qquad \dfrac{\overset{\Pi_S}{B\,S\,y \longrightarrow S\,y} \quad \overset{\Pi}{S\,t, \Gamma \longrightarrow C}}{p\,t, \Gamma \longrightarrow C}\,\mu\mathcal{L}}{\Delta, \Gamma \longrightarrow C}\,mc \quad .$$

There are at least two problems in reducing this cut. First, any permutation of the cut will necessarily involve a cut with S which can be of larger size than p, and hence simple induction on the size of 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, or at least will not be obvious.

Our proof of cut-elimination uses the technique of reducibility originally due to Tait. The method was extended by Martin-Löf [27] to the setting of natural deduction, and to sequent calculus by McDowell and Miller for the logic $FO\lambda^{\Delta\mathbb{N}}$. Our proof of cut-elimination is based on the latter approach. The original idea of Martin-Löf was to use derivations directly as a measure and to define a well-founded ordering on them. The basis for the ordering relation is a set of reduction rules that are used to eliminate the applications of cut rule. Two orderings are defined on derivations: *normalizability* and *reducibility* (called computability in [27]). The well-foundedness of normalizability ordering implies that the process of applying the reduction rules to a derivation will eventually terminate in a cut-free derivation of the same sequent. Reducibility ordering is a superset of normalizability ordering and hence its well-foundedness implies the well-foundedness of normalizability ordering. The main part of the proof is in showing that all derivations in Linc are reducible, and hence normalizable.

We prove a stronger lemma from which cut-elimination follows as a corollary. In a simplified form, this lemma basically says that given any derivation Π of

$$B_1, \dots, B_n, \Gamma \longrightarrow C$$

and reducible derivations $\Pi_1, \dots, \Pi_n$ of $\Delta \longrightarrow B_1, \dots, \Delta_n \longrightarrow B_n$ $(n \geq 0)$, the derivation Ξ

$$\dfrac{\overset{\Pi_1}{\Delta_1 \longrightarrow B_1} \quad \cdots \quad \overset{\Pi_n}{\Delta_n \longrightarrow B_n} \quad \overset{\Pi}{B_1, \dots, B_n, \Gamma \longrightarrow C}}{\Delta_1, \dots, \Delta_n, \Gamma \longrightarrow C}\; mc$$

is reducible. The cut-elimination theorem is then just a particular instance of this lemma, that is, the case where $n = 0$. The proof proceeds by induction on the height of Π with subordinate inductions on n and on the (well-founded) reduction tree of $\Pi_1, \dots, \Pi_n$.

Most of the reduction rules are variants of Gentzen's reduction rules, except, of course, for the cases involving induction and co-induction. We outline here the reduction rules for the cut on $\mu\mathcal{L}/\mu\mathcal{R}$ and $\nu\mathcal{L}/\nu\mathcal{R}$ pairs. In both cases we make use of the notion of unfolding of derivations (Chapter 3, Section 3.4) to define the reduction rules. In the $\mu\mathcal{L}/\mu\mathcal{R}$ case above, the cut is reduced to

$$\dfrac{\dfrac{\overset{\mu(\Pi_1, \Pi_S)}{\Delta \longrightarrow B\,S\,t} \quad \overset{\Pi_S[t/y]}{B\,S\,t \longrightarrow S\,t}}{\Delta \longrightarrow S\,t}\; mc \qquad \overset{\Pi}{S\,t, \Gamma \longrightarrow C}}{\Delta, \Gamma \longrightarrow C}\; mc$$

We recall that the derivation $\mu(\Pi_1, \Pi_S)$ is constructed inductively from Π_1 by replacing, among others, the subderivation of the form

$$\dfrac{\overset{\Pi_2}{\Delta' \longrightarrow B\,p\,u}}{\Delta' \longrightarrow p\,u}\; \mu\mathcal{R} \qquad \text{with} \qquad \dfrac{\overset{\mu(\Pi_2, \Pi_S)}{\Delta' \longrightarrow B\,S\,u} \quad \overset{\Pi_S[u/y]}{B\,S\,u \longrightarrow S\,u}}{\Delta' \longrightarrow S\,u}\; mc$$

where $\mu(\Pi_2, \Pi_S)$ is constructed inductively from Π_2. There are possibly more cuts produced in the unfolding process, but those are of smaller rank (i.e., the height of Π_S is smaller than the height of the original derivation) and hence are reducible by the outer induction hypothesis.

The $\nu\mathcal{R}/\nu\mathcal{L}$ case is more complicated. Suppose we have the derivation Ξ

$$\dfrac{\dfrac{\overset{\Pi_1}{\Delta \longrightarrow S\,t} \qquad \overset{\Pi_S}{S\,y \longrightarrow B\,S\,y}}{\Delta \longrightarrow p\,t}\,\nu\mathcal{R} \qquad \dfrac{\overset{\Pi}{B\,p\,t, \Gamma \longrightarrow C}}{p\,t, \Gamma \longrightarrow C}\,\nu\mathcal{L}}{\Delta, \Gamma \longrightarrow C}\,mc$$

The objective of the reduction rule is to reduce the height of the right premise of the cut. Therefore, we need to do the unfolding on the left premise.

$$\dfrac{\dfrac{\overset{\Pi_1}{\Delta \longrightarrow S\,t} \qquad \overset{\nu(\Pi_S, \Pi_S)[t/y]}{S\,t \longrightarrow B\,p\,t}}{\Delta \longrightarrow B\,p\,t}\,mc \qquad \overset{\Pi}{B\,p\,t, \Gamma \longrightarrow C}}{\Delta, \Gamma \longrightarrow C}\,mc$$

In constructing the derivation $\nu(\Pi_S, \Pi_S)$, subderivations of Π_S of the form

$$\overset{\Psi}{\Delta \longrightarrow S\,u} \qquad \text{are replaced by} \qquad \dfrac{\overset{\Psi}{\Delta \longrightarrow S\,u} \quad \overset{\Pi_S}{S\,x \longrightarrow B\,S\,x}}{\Delta \longrightarrow p\,u}\,\nu\mathcal{R}\ .$$

This is the delicate point of the proof since there could be a potentially infinite unwinding of derivations as we push up the cut. Notice that unlike in the inductive case, the unfolding process uses only the derivations in the left premise of the cut, and hence the outer induction hypothesis will not help in establishing the reducibility of the unfolded derivation. We solve this problem by building into the reducibility ordering a closure condition which will allow us to establish the reducibility of the unfolded derivation $\mu(\Pi_S, \Pi_S)$, given the reducibility of Π_S.

4.1 Cut reduction

Here we define a reduction relation between derivations, which is an adaptation of the reduction rules used in Gentzen's original Hauptsatz [18].

Definition 4.1. *We define a* reduction *relation between derivations. The redex is always a derivation Ξ ending with the multicut rule*

$$\dfrac{\overset{\Pi_1}{\Sigma\,;\Delta_1 \longrightarrow \mathcal{B}_1} \quad \cdots \quad \overset{\Pi_n}{\Sigma\,;\Delta_n \longrightarrow \mathcal{B}_n} \quad \overset{\Pi}{\Sigma\,;\mathcal{B}_1, \dots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}}{\Sigma\,;\Delta_1, \dots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\,mc\ .$$

We refer to the judgments $\mathcal{B}_1, \dots, \mathcal{B}_n$ produced by the mc as cut judgments*.*

If $n = 0$, Ξ reduces to the premise derivation Π.

For $n > 0$ we specify the reduction relation based on the last rule of the premise derivations. If the rightmost premise derivation Π ends with a left rule acting on a cut judgment $\mathcal{B}_i$, then the last rule of Π_i and the last rule of Π together determine the reduction rules that apply. We classify these rules according to the following criteria: we call the rule an essential *case when Π_i ends with a right rule; if it ends with a left* *rule, it is a* left-commutative *case; if* Π_i *ends with the init rule, then we have an* axiom *case; a* multicut *case arises when it ends with the* mc *rule. When* Π *does not end with a left rule acting on a cut judgment, then its last rule is alone sufficient to determine the reduction rules that apply. If* Π *ends in a rule acting on a judgment other than a cut judgment, then we call this a* right-commutative *case. A* structural *case results when* Π *ends with a contraction or weakening on a cut judgment. If* Π *ends with the init rule, this is also an axiom case; similarly a multicut case arises if* Π *ends in the* mc *rule.*

For simplicity of presentation, we always show $i = 1$.

<u>Essential cases:</u>

$\wedge\mathcal{R}/\wedge\mathcal{L}$: *If* Π_1 *and* Π *are*

$$\dfrac{\begin{matrix}\Pi_1' \\ \Delta_1 \longrightarrow \mathcal{B}_1'\end{matrix} \quad \begin{matrix}\Pi_1'' \\ \Delta_1 \longrightarrow \mathcal{B}_1''\end{matrix}}{\Delta_1 \longrightarrow \mathcal{B}_1' \wedge \mathcal{B}_1''}\wedge\mathcal{R} \qquad \dfrac{\begin{matrix}\Pi' \\ \mathcal{B}_1', \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{matrix}}{\mathcal{B}_1' \wedge \mathcal{B}_1'', \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}\wedge\mathcal{L} \quad ,$$

then Ξ *reduces to*

$$\dfrac{\begin{matrix}\Pi_1' \\ \Delta_1 \longrightarrow \mathcal{B}_1'\end{matrix} \quad \begin{matrix}\Pi_2 \\ \Delta_2 \longrightarrow \mathcal{B}_2\end{matrix} \quad \cdots \quad \begin{matrix}\Pi_n \\ \Delta_n \longrightarrow \mathcal{B}_n\end{matrix} \quad \begin{matrix}\Pi' \\ \mathcal{B}_1', \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{matrix}}{\Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\, mc \quad .$$

The case for the other $\wedge\mathcal{L}$ *rule is symmetric.*

$\vee\mathcal{R}/\vee\mathcal{L}$: *If* Π_1 *and* Π *are*

$$\dfrac{\begin{matrix}\Pi_1' \\ \Delta_1 \longrightarrow \mathcal{B}_1'\end{matrix}}{\Delta_1 \longrightarrow \mathcal{B}_1' \vee \mathcal{B}_1''}\vee\mathcal{R} \qquad \dfrac{\begin{matrix}\Pi' \\ \mathcal{B}_1', \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{matrix} \quad \begin{matrix}\Pi'' \\ \mathcal{B}_1'', \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{matrix}}{\mathcal{B}_1' \vee \mathcal{B}_1'', \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}\vee\mathcal{L} \quad ,$$

then Ξ *reduces to*

$$\dfrac{\begin{matrix}\Pi_1' \\ \Delta_1 \longrightarrow \mathcal{B}_1'\end{matrix} \quad \begin{matrix}\Pi_2 \\ \Delta_2 \longrightarrow \mathcal{B}_2\end{matrix} \quad \cdots \quad \begin{matrix}\Pi_n \\ \Delta_n \longrightarrow \mathcal{B}_n\end{matrix} \quad \begin{matrix}\Pi' \\ \mathcal{B}_1', \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{matrix}}{\Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\, mc \quad .$$

The case for the other $\vee\mathcal{R}$ *rule is symmetric.*

$\supset\mathcal{R}/\supset\mathcal{L}$: *Suppose* Π_1 *and* Π *are*

$$\dfrac{\begin{matrix}\Pi_1' \\ \mathcal{B}_1', \Delta_1 \longrightarrow \mathcal{B}_1''\end{matrix}}{\Delta_1 \longrightarrow \mathcal{B}_1' \supset \mathcal{B}_1''}\supset\mathcal{R} \qquad \dfrac{\begin{matrix}\Pi' \\ \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{B}_1'\end{matrix} \quad \begin{matrix}\Pi'' \\ \mathcal{B}_1'', \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{matrix}}{\mathcal{B}_1' \supset \mathcal{B}_1'', \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}\supset\mathcal{L} \quad .$$

Let Ξ_1 *be*

$$\dfrac{\dfrac{\left\{\begin{array}{c}\Pi_i\\ \Delta_i \longrightarrow \mathcal{B}_i\end{array}\right\}_{i\in\{2..n\}} \quad \begin{array}{c}\Pi'\\ \mathcal{B}_2,\dots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{B}'_1\end{array}}{\Delta_2,\dots,\Delta_n,\Gamma\longrightarrow \mathcal{B}'_1}\,mc \qquad \begin{array}{c}\Pi'_1\\ \mathcal{B}'_1,\Delta_1\longrightarrow\mathcal{B}''_1\end{array}}{\Delta_1,\dots,\Delta_n,\Gamma\longrightarrow\mathcal{B}''_1}\,mc \quad .$$

Then Ξ *reduces to*

$$\dfrac{\begin{array}{c}\Xi_1\\ \ldots\longrightarrow\mathcal{B}''_1\end{array} \quad \left\{\begin{array}{c}\Pi_i\\ \Delta_i \longrightarrow \mathcal{B}_i\end{array}\right\}_{i\in\{2..n\}} \quad \begin{array}{c}\Pi''\\ \mathcal{B}''_1,\{\mathcal{B}_i\}_{i\in\{2..n\}},\Gamma\longrightarrow\mathcal{C}\end{array}}{\begin{array}{c}\Delta_1,\dots,\Delta_n,\Gamma,\Delta_2,\dots,\Delta_n,\Gamma\longrightarrow\mathcal{C}\\ \hline\hline \Delta_1,\dots,\Delta_n,\Gamma\longrightarrow\mathcal{C}\end{array}\,c\mathcal{L}}\,mc \quad .$$

We use the double horizontal lines to indicate that the relevant inference rule (in this case, c$\mathcal{L}$*) may need to be applied zero or more times.*
$\forall\mathcal{R}/\forall\mathcal{L}$*: If* Π_1 *and* Π *are*

$$\dfrac{\begin{array}{c}\Pi'_1\\ \Sigma,h\,;\,\Delta_1\longrightarrow\bar{z}\rhd B'_1[(h\,\bar{z})/x]\end{array}}{\Sigma\,;\,\Delta_1\longrightarrow\bar{z}\rhd\forall x.B'_1}\,\forall\mathcal{R} \qquad \dfrac{\begin{array}{c}\Pi'\\ \Sigma\,;\,\bar{z}\rhd B'_1[t/x],\mathcal{B}_2,\dots,\mathcal{B}_n,\Gamma\longrightarrow\mathcal{C}\end{array}}{\Sigma\,;\,\bar{z}\rhd\forall x.B'_1,\mathcal{B}_2,\dots,\mathcal{B}_n,\Gamma\longrightarrow\mathcal{C}}\,\forall\mathcal{L} \quad ,$$

then Ξ *reduces to*

$$\dfrac{\begin{array}{c}\Pi'_1[\lambda\bar{z}.t/h]\\ \Sigma\,;\,\Delta_1\longrightarrow\bar{z}\rhd B'_1[t/x]\end{array} \quad \left\{\begin{array}{c}\Pi_i\\ \Sigma\,;\,\Delta_i \longrightarrow \mathcal{B}_i\end{array}\right\}_{i\in\{2..n\}} \quad \begin{array}{c}\Pi'\\ \ldots\longrightarrow\mathcal{C}\end{array}}{\Sigma\,;\,\Delta_1,\dots,\Delta_n,\Gamma\longrightarrow\mathcal{C}}\,mc \quad .$$

$\exists\mathcal{R}/\exists\mathcal{L}$*: If* Π_1 *and* Π *are*

$$\dfrac{\begin{array}{c}\Pi'_1\\ \Sigma\,;\,\Delta_1\longrightarrow\bar{z}\rhd B'_1[t/x]\end{array}}{\Sigma\,;\,\Delta_1\longrightarrow\bar{z}\rhd\exists x.B'_1}\,\exists\mathcal{R} \qquad \dfrac{\begin{array}{c}\Pi'\\ \Sigma,h\,;\,\bar{z}\rhd B'_1[(h\,\bar{z})/x],\mathcal{B}_2,\dots,\mathcal{B}_n,\Gamma\longrightarrow\mathcal{C}\end{array}}{\Sigma\,;\,\bar{z}\rhd\exists x.B'_1,\mathcal{B}_2,\dots,\mathcal{B}_n,\Gamma\longrightarrow\mathcal{C}}\,\exists\mathcal{L} \quad ,$$

then Ξ *reduces to*

$$\dfrac{\begin{array}{c}\Pi'_1\\ \Sigma\,;\,\Delta_1\longrightarrow\bar{z}\rhd B'_1[t/x]\end{array} \quad \ldots \quad \begin{array}{c}\Pi'[\lambda\bar{z}.t/h]\\ \Sigma\,;\,\bar{z}\rhd B'_1[t/x],\mathcal{B}_2,\dots,\Gamma\longrightarrow\mathcal{C}\end{array}}{\Sigma\,;\,\Delta_1,\dots,\Delta_n,\Gamma\longrightarrow\mathcal{C}}\,mc \quad .$$

$\nabla\mathcal{R}/\nabla\mathcal{L}$: *If* Π_1 *and* Π *are*

$$\dfrac{\overset{\Pi'_1}{\Delta_1 \longrightarrow \bar{z}y \triangleright B'_1[y/x]}}{\Delta_1 \longrightarrow \bar{z} \triangleright \nabla x.B'_1}\;\nabla\mathcal{R} \qquad\qquad \dfrac{\overset{\Pi'}{\bar{z}y \triangleright B'_1[y/x], \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}}{\bar{z} \triangleright \nabla x.B'_1, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}\;\exists\mathcal{L}\quad,$$

then Ξ *reduces to*

$$\dfrac{\overset{\Pi'_1}{\Delta_1 \longrightarrow \bar{z}y \triangleright B'_1[y/x]} \quad \ldots \quad \overset{\Pi'}{\bar{z}y \triangleright B'_1[y/x], \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}}{\Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\;mc\quad.$$

$*/\mu\mathcal{L}$: *Suppose* Π *is the derivation*

$$\dfrac{\overset{\Pi_S}{\bar{x}\,;\, D\,S\,\bar{x} \longrightarrow S\,\bar{x}} \quad \overset{\Pi'}{\Sigma\,;\, \bar{z} \triangleright S\,\bar{t}, \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}}{\Sigma\,;\, \bar{z} \triangleright p\,\bar{t}, \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}\;\mu\mathcal{L}$$

where $p\,\bar{x} \overset{\mu}{=} B\,p\,\bar{x}$. *Then* Ξ *reduces to*

$$\dfrac{\overset{\mu(\Pi_1, \Pi_S)}{\Sigma\,;\, \Delta_1 \longrightarrow \bar{z} \triangleright S\,\bar{t}} \quad \ldots \quad \overset{\Pi'}{\Sigma\,;\, \bar{z} \triangleright S\,\bar{t}, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}}{\Sigma\,;\, \Delta_1, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\;mc$$

$\nu\mathcal{R}/\nu\mathcal{L}$: *Suppose* Π_1 *and* Π *are*

$$\dfrac{\overset{\Pi'_1}{\Sigma\,;\, \Delta_1 \longrightarrow \bar{z} \triangleright S\,\bar{t}} \quad \overset{\Pi_S}{\bar{x}\,;\, S\,\bar{x} \longrightarrow D\,S\,\bar{x}}}{\Sigma\,;\, \Delta_1 \longrightarrow \bar{z} \triangleright p\,\bar{t}}\;\nu\mathcal{R} \qquad\qquad \dfrac{\overset{\Pi'}{\Sigma\,;\, \bar{z} \triangleright D\,p\,\bar{t}, \Gamma \longrightarrow \mathcal{C}}}{\Sigma\,;\, \bar{z} \triangleright p\,\bar{t}, \ldots, \Gamma \longrightarrow \mathcal{C}}\;\nu\mathcal{L}$$

where $\bar{t} = t_1, \ldots, t_m$. *Apply Lemma 3.16 to* Π_S *to get a derivation*

$$\overset{\Pi'_S}{h_1, \ldots, h_m\,;\, \bar{z} \triangleright S\,\bar{u} \longrightarrow \bar{z} \triangleright D\,S\,\bar{u}}$$

where $\bar{u} = (h_1\,\bar{z}), \ldots, (h_m\,\bar{z})$. *Let* Ξ_1 *be the derivation*

$$\dfrac{\overset{\Pi'_1}{\Sigma\,;\, \Delta_1 \longrightarrow \bar{z} \triangleright S\,\bar{t}} \quad \overset{\Pi'_S\theta}{\Sigma\,;\, \bar{z} \triangleright S\,\bar{t} \longrightarrow \bar{z} \triangleright D\,S\,\bar{t}}}{\Sigma\,;\, \Delta_1 \longrightarrow \bar{z} \triangleright D\,S\,\bar{t}}\;mc$$

where $\theta = [(\lambda\bar{z}.t_1)/h_1, \ldots, (\lambda\bar{z}.t_m)/h_m]$. *Then* Ξ *reduces to*

$$\frac{\begin{array}{c}\nu(\Xi_1,\Pi_S)\\ \Sigma\,;\,\Delta_1 \longrightarrow \bar{z}\rhd D\,p\,\bar{t}\end{array} \quad \left\{\begin{array}{c}\Pi_j\\ \Sigma\,;\,\Delta_j \longrightarrow \mathcal{B}_j\end{array}\right\}_{j\in\{2,\ldots,n\}} \quad \begin{array}{c}\Pi'\\ \Sigma\,;\,\bar{z}\rhd D\,p\,\bar{t},\ldots,\Gamma \longrightarrow \mathcal{C}\end{array}}{\Sigma\,;\,\Delta_1,\ldots,\Delta_n,\Gamma \longrightarrow \mathcal{C}}\;mc$$

def$\mathcal{R}$/def$\mathcal{L}$: *Let* $p\,\bar{x} \stackrel{\triangle}{=} D\,p\,\bar{x}$ *be a regular definition. Suppose* Π_1 *and* Π *are*

$$\frac{\begin{array}{c}\Pi_1'\\ \Delta_1 \longrightarrow \bar{z}\rhd D\,p\,\bar{t}\end{array}}{\Delta_1 \longrightarrow \bar{z}\rhd p\,\bar{t}}\;\textit{def}\mathcal{R} \qquad\qquad \frac{\begin{array}{c}\Pi'\\ \bar{z}\rhd D\,p\,\bar{t},\ldots,\Gamma \longrightarrow \mathcal{C}\end{array}}{\bar{z}\rhd p\,\bar{t},\ldots,\Gamma \longrightarrow \mathcal{C}}\;\textit{def}\mathcal{L} \quad .$$

Then Ξ *reduces to*

$$\frac{\begin{array}{c}\Pi_1'\\ \Delta_1 \longrightarrow \bar{z}\rhd D\,p\,\bar{t}\end{array} \quad \ldots \quad \begin{array}{c}\Pi'\\ \bar{z}\rhd D\,p\,\bar{t},\ldots,\Gamma \longrightarrow \mathcal{C}\end{array}}{\Delta_1,\ldots,\Delta_n,\Gamma \longrightarrow \mathcal{C}}\;mc \quad .$$

eq$\mathcal{R}$/eq$\mathcal{L}$: *Suppose* Π_1 *and* Π *are*

$$\frac{}{\Sigma\,;\,\Delta_1 \longrightarrow \bar{z}\rhd s = t}\;\mathrm{eq}\mathcal{R} \qquad\qquad \frac{\left\{\begin{array}{c}\Pi^{\rho}\\ \Sigma\rho\,;\,\mathcal{B}_2\rho,\ldots,\mathcal{B}_n\rho,\Gamma\rho \longrightarrow \mathcal{C}\rho\end{array}\right\}_{\rho}}{\Sigma\,;\,\bar{z}\rhd s = t,\mathcal{B}_2,\ldots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{C}}\;\mathrm{eq}\mathcal{L} \quad .$$

Then by the definition of eq$\mathcal{R}$ *rule,* s *and* t *are equal terms (modulo* λ*-conversion), and hence are unifiable by the empty substitution. Note that in this case* $\Pi^{\epsilon} \in \{\Pi^{\rho}\}_{\rho}$. *Therefore* Ξ *reduces to*

$$\cfrac{\cfrac{\left\{\begin{array}{c}\Pi_i\\ \Sigma\,;\,\Delta_i \longrightarrow \mathcal{B}_i\end{array}\right\}_{i\in\{2..n\}} \quad \begin{array}{c}\Pi^{\epsilon}\\ \Sigma\,;\,\mathcal{B}_2,\ldots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{C}\end{array}}{\Sigma\,;\,\Delta_2,\ldots,\Delta_n,\Gamma \longrightarrow \mathcal{C}}\;mc}{\Sigma\,;\,\Delta_1,\Delta_2,\ldots,\Delta_n,\Gamma \longrightarrow \mathcal{C}}\;\mathrm{w}\mathcal{L} \quad .$$

<u>Left-commutative cases:</u> *In the following cases, we suppose that* Π *ends with a left rule, other than* $\{\mathrm{c}\mathcal{L}, \mathrm{w}\mathcal{L}, \mu\mathcal{L}\}$, *acting on* $\mathcal{B}_1$.

$\bullet\mathcal{L}/\circ\mathcal{L}$: *Suppose* Π_1 *is*

$$\frac{\left\{\begin{array}{c}\Pi_1^{i}\\ \Sigma_i\,;\,\Delta_1^{i} \longrightarrow \mathcal{B}_1\end{array}\right\}}{\Sigma\,;\,\Delta_1 \longrightarrow \mathcal{B}_1}\;\bullet\mathcal{L} \quad ,$$

where $\bullet\mathcal{L}$ *is any left rule except* $\supset\mathcal{L}$, $\mathrm{eq}\mathcal{L}$, *or* $\mu\mathcal{L}$. *Note that in this case we have* $\Sigma_i \supseteq \Sigma$ *for each* i. *Then* Ξ *reduces to*

$$\dfrac{\left\{\dfrac{\begin{matrix}\Pi_1^i\\ \Sigma_i\,;\,\Delta_1^i \longrightarrow \mathcal{B}_1\end{matrix} \quad \left\{\begin{matrix}\Pi'_j\\ \Sigma_i\,;\,\Delta_j \longrightarrow \mathcal{B}_j\end{matrix}\right\}_{j\in\{2..n\}} \quad \begin{matrix}\Pi'\\ \Sigma_i\,;\,\mathcal{B}_1,\dots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{C}\end{matrix}}{\Sigma_i\,;\,\Delta_1^i,\Delta_2,\dots,\Delta_n,\Gamma \longrightarrow \mathcal{C}}\,mc\right\}}{\Sigma\,;\,\Delta_1,\Delta_2,\dots,\Delta_n,\Gamma \longrightarrow \mathcal{C}}\,\bullet\mathcal{L}\ .$$

The derivations Π'_j *and* Π' *are obtained from* Π_j *and* Π, *respectively, by applying Lemma 3.11 (weakening of signature).*

$\supset\mathcal{L}/\circ\mathcal{L}$: *Suppose* Π_1 *is*

$$\dfrac{\begin{matrix}\Pi'_1\\ \Delta'_1 \longrightarrow \mathcal{D}'_1\end{matrix} \quad \begin{matrix}\Pi''_1\\ \mathcal{D}''_1,\Delta'_1 \longrightarrow \mathcal{B}_1\end{matrix}}{\mathcal{D}'_1 \supset \mathcal{D}''_1,\Delta'_1 \longrightarrow \mathcal{B}_1}\supset\mathcal{L}\ .$$

Let Ξ_1 *be*

$$\dfrac{\begin{matrix}\Pi''_1\\ \mathcal{D}''_1,\Delta'_1 \longrightarrow \mathcal{B}_1\end{matrix} \quad \begin{matrix}\Pi_2\\ \Delta_2 \longrightarrow \mathcal{B}_2\end{matrix} \quad \cdots \quad \begin{matrix}\Pi_n\\ \Delta_n \longrightarrow \mathcal{B}_n\end{matrix} \quad \begin{matrix}\Pi\\ \mathcal{B}_1,\dots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{C}\end{matrix}}{\mathcal{D}''_1,\Delta'_1,\Delta_2,\dots,\Delta_n,\Gamma \longrightarrow \mathcal{C}}\,mc\ .$$

Then Ξ *reduces to*

$$\dfrac{\dfrac{\begin{matrix}\Pi'_1\\ \Delta'_1 \longrightarrow \mathcal{D}'_1\end{matrix}}{\Delta'_1,\Delta_2,\dots,\Delta_n,\Gamma \longrightarrow \mathcal{D}'_1}\,w\mathcal{L} \quad \begin{matrix}\Xi_1\\ \mathcal{D}''_1,\Delta'_1,\Delta_2,\dots,\Delta_n,\Gamma \longrightarrow \mathcal{C}\end{matrix}}{\mathcal{D}'_1 \supset \mathcal{D}''_1,\Delta'_1,\Delta_2,\dots,\Delta_n,\Gamma \longrightarrow \mathcal{C}}\supset\mathcal{L}\ .$$

$\mu\mathcal{L}/\circ\mathcal{L}$: *Suppose* Π_1 *is*

$$\dfrac{\begin{matrix}\Pi_S\\ \bar{x}\,;\,D\,S\,\bar{x} \longrightarrow S\,\bar{x}\end{matrix} \quad \begin{matrix}\Pi'_1\\ \Sigma\,;\,\bar{z} \triangleright S\,\bar{t},\Delta'_1 \longrightarrow \mathcal{B}_1\end{matrix}}{\Sigma\,;\,\bar{z} \triangleright p\,\bar{t},\Delta'_1 \longrightarrow \mathcal{B}_1}\,\mu\mathcal{L}$$

where $p\,\bar{x} \overset{\mu}{=} D\,p\,\bar{x}$. *Let* Ξ_1 *be*

$$\dfrac{\overset{\Pi_1'}{\Sigma\,;\,\bar{z} \triangleright S\,\bar{t}, \Delta_1' \longrightarrow \mathcal{B}_1} \quad \dots \quad \overset{\Pi_n}{\Sigma\,;\,\Delta_n \longrightarrow \mathcal{B}_n} \quad \overset{\Pi}{\Sigma\,;\,\mathcal{B}_1, \dots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}}{\Sigma\,;\,\bar{z} \triangleright S\,\bar{t}, \Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ mc \quad .$$

Then Ξ *reduces to*

$$\dfrac{\overset{\Pi_S}{\bar{x}\,;\,D\,S\,\bar{x} \longrightarrow S\,\bar{x}} \quad \overset{\Xi_1}{\Sigma\,;\,\bar{z} \triangleright S\,\bar{t}, \Delta_1', \dots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}}{\Sigma\,;\,\bar{z} \triangleright p\,\bar{t}, \Delta_1', \dots, \Delta_n \longrightarrow \mathcal{C}}\ \mu\mathcal{L}$$

eq$\mathcal{L}/\circ\mathcal{L}$: *Suppose* Π_1 *is*

$$\dfrac{\left\{\begin{matrix}\Pi_1^\rho \\ \Sigma\rho\,;\,\Delta_1'\rho \longrightarrow \mathcal{B}_1\rho\end{matrix}\right\}}{\Sigma\,;\,\bar{z} \triangleright s = t, \Delta_1' \longrightarrow \mathcal{B}_1}\ \mathrm{eq}\mathcal{L} \quad ,$$

then Ξ *reduces to*

$$\dfrac{\left\{\dfrac{\overset{\Pi_1^\rho}{\Sigma\rho\,;\,\Delta_1'\rho \longrightarrow \mathcal{B}_1\rho} \quad \left\{\begin{matrix}\Pi_i\rho \\ \Sigma\rho\,;\,\Delta_i\rho \longrightarrow \mathcal{B}_i\rho\end{matrix}\right\}_{i\in\{2..n\}} \quad \overset{\Pi\rho}{\dots \longrightarrow \mathcal{C}\rho}}{\Sigma\rho\,;\,\Delta_1'\rho, \Delta_2\rho, \dots, \Delta_n\rho, \Gamma\rho \longrightarrow \mathcal{C}\rho}\ mc\right\}}{\bar{z} \triangleright s = t, \Delta_1', \Delta_2, \dots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ \mathrm{eq}\mathcal{L} \quad .$$

Right-commutative cases:

$-/\circ\mathcal{L}$: *Suppose* Π *is*

$$\dfrac{\left\{\begin{matrix}\Pi^i \\ \Sigma_i\,;\,\mathcal{B}_1, \dots, \mathcal{B}_n, \Gamma^i \longrightarrow \mathcal{C}\end{matrix}\right\}}{\Sigma\,;\,\mathcal{B}_1, \dots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}\ \circ\mathcal{L} \quad ,$$

where $\circ\mathcal{L}$ *is any left rule other than* $\supset\mathcal{L}$, eq$\mathcal{L}$, *or* $\mu\mathcal{L}$ *(but including* c$\mathcal{L}$ *or* w$\mathcal{L}$*) acting on a judgment other than* $\mathcal{B}_1, \dots, \mathcal{B}_n$. *Note that* $\Sigma_i \supseteq \Sigma$ *since in the case* $\exists\mathcal{L}$ *a new eigenvariable is introduced in the premise. The derivation* Ξ *reduces to*

$$\dfrac{\left\{\dfrac{\overset{\Pi_1'}{\Sigma_i\,;\,\Delta_1 \longrightarrow \mathcal{B}_1} \quad \cdots \quad \overset{\Pi_n'}{\Sigma_i\,;\,\Delta_n \longrightarrow B_n'} \quad \overset{\Pi^i}{\Sigma_i\,;\,\mathcal{B}_1, \dots, \mathcal{B}_n, \Gamma^i \longrightarrow \mathcal{C}}}{\Sigma_i\,;\,\Delta_1, \dots, \Delta_n, \Gamma^i \longrightarrow \mathcal{C}}\ mc\right\}}{\Sigma\,;\,\Delta_1, \dots, \Delta_n, \Gamma \longrightarrow \mathcal{C}}\ \circ\mathcal{L} \quad ,$$

where $\Pi_1', \dots, \Pi_n'$ *are obtained from applying Lemma 3.11 to* $\Pi_1, \dots, \Pi_n$, *respectively.*

$-/\supset\mathcal{L}$: *Suppose* Π *is*

$$\dfrac{\overset{\Pi'}{\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma'\longrightarrow\mathcal{D}'}\quad\overset{\Pi''}{\mathcal{B}_1,\ldots,\mathcal{B}_n,\mathcal{D}'',\Gamma'\longrightarrow\mathcal{C}}}{\mathcal{B}_1,\ldots,\mathcal{B}_n,\mathcal{D}'\supset\mathcal{D}'',\Gamma'\longrightarrow\mathcal{C}}\supset\mathcal{L}\quad.$$

Let Ξ_1 *be*

$$\dfrac{\overset{\Pi_1}{\Delta_1\longrightarrow\mathcal{B}_1}\quad\cdots\quad\overset{\Pi_n}{\Delta_n\longrightarrow\mathcal{B}_n}\quad\overset{\Pi'}{\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma'\longrightarrow\mathcal{D}'}}{\Delta_1,\ldots,\Delta_n,\Gamma'\longrightarrow\mathcal{D}'}\,mc$$

and Ξ_2 *be*

$$\dfrac{\overset{\Pi_1}{\Delta_1\longrightarrow\mathcal{B}_1}\quad\cdots\quad\overset{\Pi_n}{\Delta_n\longrightarrow\mathcal{B}_n}\quad\overset{\Pi''}{\mathcal{B}_1,\ldots,\mathcal{B}_n,\mathcal{D}'',\Gamma'\longrightarrow\mathcal{C}}}{\Delta_1,\ldots,\Delta_n,\mathcal{D}'',\Gamma'\longrightarrow\mathcal{C}}\,mc\quad.$$

Then Ξ *reduces to*

$$\dfrac{\overset{\Xi_1}{\Delta_1,\ldots,\Delta_n,\Gamma'\longrightarrow\mathcal{D}'}\quad\overset{\Xi_2}{\Delta_1,\ldots,\Delta_n,\mathcal{D}'',\Gamma'\longrightarrow\mathcal{C}}}{\Delta_1,\ldots,\Delta_n,\mathcal{D}'\supset\mathcal{D}'',\Gamma'\longrightarrow\mathcal{C}}\supset\mathcal{L}\quad.$$

$-/\mu\mathcal{L}$: *Suppose* Π *is*

$$\dfrac{\overset{\Pi_S}{\bar{x}\,;\,D\,S\,\bar{x}\longrightarrow S\,\bar{x}}\quad\overset{\Pi'}{\Sigma\,;\,\mathcal{B}_1,\ldots,\mathcal{B}_n,\bar{z}\triangleright S\,\bar{t},\Gamma'\longrightarrow\mathcal{C}}}{\Sigma\,;\,\mathcal{B}_1,\ldots,\mathcal{B}_n,\bar{z}\triangleright p\,\bar{t},\Gamma'\longrightarrow\mathcal{C}}\,\mu\mathcal{L}\quad,$$

where $p\,\bar{x}\overset{\mu}{=}D\,p\,\bar{x}$. *Let* Ξ_1 *be*

$$\dfrac{\overset{\Pi_1}{\Sigma\,;\,\Delta_1\longrightarrow\mathcal{B}_1}\quad\cdots\quad\overset{\Pi_n}{\Sigma\,;\,\Delta_n\longrightarrow\mathcal{B}_n}\quad\overset{\Pi'}{\Sigma\,;\,\mathcal{B}_1,\ldots,\mathcal{B}_n,\bar{z}\triangleright S\,\bar{t},\Gamma'\longrightarrow\mathcal{C}}}{\Sigma\,;\,\Delta_1,\ldots,\Delta_n,\bar{z}\triangleright S\,\bar{t},\Gamma'\longrightarrow\mathcal{C}}\,mc\quad.$$

Then Ξ *reduces to*

$$\dfrac{\overset{\Pi_S}{\bar{x}\,;\,D\,S\,\bar{x}\longrightarrow S\,\bar{x}}\quad\overset{\Xi}{\Sigma\,;\,\Delta_1,\ldots,\Delta_n,\bar{z}\triangleright S\,\bar{t},\Gamma'\longrightarrow\mathcal{C}}}{\Sigma\,;\,\Delta_1,\ldots,\Delta_n,\bar{z}\triangleright p\,\bar{t},\Gamma'\longrightarrow\mathcal{C}}\,\mu\mathcal{L}\quad.$$

$-/\mathrm{eq}\mathcal{L}$: *If* Π *is*

$$\dfrac{\left\{\overset{\Pi^{\rho}}{\Sigma\rho\,;\,\mathcal{B}_1\rho,\ldots,\mathcal{B}_n\rho,\Gamma'\rho\longrightarrow\mathcal{C}\rho}\right\}}{\Sigma\,;\,\mathcal{B}_1,\ldots,\mathcal{B}_n,\bar{z}\triangleright s=t,\Gamma'\longrightarrow\mathcal{C}}\,\mathrm{eq}\mathcal{L}\quad,$$

then Ξ *reduces to*

$$\dfrac{\left\{\dfrac{\left\{\begin{array}{c}\Pi_i\rho\\ \Sigma\rho\,;\,\Delta_i\rho\longrightarrow\mathcal{B}_i\rho\end{array}\right\}_{i\in\{1..n\}}\quad\begin{array}{c}\Pi^{\rho}\\ \Sigma\rho\,;\,\mathcal{B}_i\rho,\ldots,\Gamma'\rho\longrightarrow\mathcal{C}\rho\end{array}}{\Sigma\rho\,;\,\Delta_1\rho,\ldots,\Delta_n\rho,\Gamma'\rho\longrightarrow\mathcal{C}\rho}\,mc\right\}}{\Sigma\,;\,\Delta_1,\ldots,\Delta_n,\bar{z}\rhd s=t,\Gamma'\longrightarrow\mathcal{C}}\,\mathrm{eq}\mathcal{L}\quad.$$

$-/\circ\mathcal{R}$*: If* Π *is*

$$\dfrac{\left\{\begin{array}{c}\Pi^i\\ \Sigma_i\,;\,\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma^i\longrightarrow\mathcal{C}^i\end{array}\right\}}{\Sigma\,;\,\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma\longrightarrow\mathcal{C}}\,\circ\mathcal{R}\quad,$$

where $\circ\mathcal{R}$ *is any right rule except* $\nu\mathcal{R}$*, then* Ξ *reduces to*

$$\dfrac{\left\{\dfrac{\begin{array}{c}\Pi'_1\\ \Sigma_i\,;\,\Delta_1\longrightarrow\mathcal{B}_1\end{array}\quad\cdots\quad\begin{array}{c}\Pi'_n\\ \Sigma_i\,;\,\Delta_n\longrightarrow\mathcal{B}_n\end{array}\quad\begin{array}{c}\Pi^i\\ \Sigma_i\,;\,\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma^i\longrightarrow\mathcal{C}^i\end{array}}{\Sigma_i\,;\,\Delta_1,\ldots,\Delta_n,\Gamma^i\longrightarrow\mathcal{C}^i}\,mc\right\}}{\Sigma\,;\,\Delta_1,\ldots,\Delta_n,\Gamma\longrightarrow\mathcal{C}}\,\circ\mathcal{R}\quad,$$

where $\Pi'_1,\ldots,\Pi'_n$ *are obtained from* $\Pi_1,\ldots,\Pi_n$ *by Lemma 3.11.*
$-/\nu\mathcal{R}$*: Suppose* Π *is*

$$\dfrac{\begin{array}{c}\Pi'\\ \Sigma\,;\,\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma\longrightarrow\bar{z}\rhd S\,\bar{t}\end{array}\quad\begin{array}{c}\Pi_S\\ \bar{x}\,;\,S\,\bar{x}\longrightarrow D\,S\,\bar{x}\end{array}}{\Sigma\,;\,\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma\longrightarrow\bar{z}\rhd p\,\bar{t}}\,\nu\mathcal{R}\quad,$$

where $p\,\bar{x}\overset{\nu}{=}D\,p\,\bar{x}$*. Let* Ξ_1 *be*

$$\dfrac{\begin{array}{c}\Pi_1\\ \Sigma\,;\,\Delta_1\longrightarrow\mathcal{B}_1\end{array}\quad\cdots\quad\begin{array}{c}\Pi_n\\ \Sigma\,;\,\Delta_n\longrightarrow\mathcal{B}_n\end{array}\quad\begin{array}{c}\Pi'\\ \Sigma\,;\,\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma\longrightarrow\bar{z}\rhd S\,\bar{t}\end{array}}{\Sigma\,;\,\Delta_1,\ldots,\Delta_n,\Gamma\longrightarrow\bar{z}\rhd S\,\bar{t}}\,mc\quad.$$

Then Ξ *reduces to*

$$\dfrac{\begin{array}{c}\Xi_1\\ \Sigma\,;\,\Delta_1,\ldots,\Delta_n,\Gamma\longrightarrow\bar{z}\rhd S\,\bar{t}\end{array}\quad\begin{array}{c}\Pi_S\\ \bar{x}\,;\,S\,\bar{x}\longrightarrow D\,S\,\bar{x}\end{array}}{\Sigma\,;\,\Delta_1,\ldots,\Delta_n,\Gamma\longrightarrow\bar{z}\rhd p\,\bar{t}}\,\nu\mathcal{R}\quad.$$

Multicut cases:
$mc/\circ\mathcal{L}$*: If* Π *ends with a left rule, other than c*$\mathcal{L}$*, w*$\mathcal{L}$ *and* $\mu\mathcal{L}$*, acting on* $\mathcal{B}_1$ *and* Π_1 *ends with a multicut and reduces to* Π'_1*, then* Ξ *reduces to*

$$\dfrac{\begin{array}{c}\Pi'_1\\ \Delta_1\longrightarrow\mathcal{B}_1\end{array}\quad\begin{array}{c}\Pi_2\\ \Delta_2\longrightarrow\mathcal{B}_2\end{array}\quad\cdots\quad\begin{array}{c}\Pi_n\\ \Delta_n\longrightarrow\mathcal{B}_n\end{array}\quad\begin{array}{c}\Pi\\ \mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma\longrightarrow\mathcal{C}\end{array}}{\Delta_1,\ldots,\Delta_n,\Gamma\longrightarrow\mathcal{C}}\,mc\quad.$$

$-/mc$: Suppose Π *is*

$$\dfrac{\left\{\overset{\Pi^j}{\{\mathcal{B}_i\}_{i\in I^j},\Gamma^j \longrightarrow \mathcal{D}^j}\right\}_{j\in\{1..m\}} \quad \overset{\Pi'}{\{\mathcal{D}^j\}_{j\in\{1..m\}},\{\mathcal{B}_i\}_{i\in I'},\Gamma' \longrightarrow \mathcal{C}}}{\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma^1,\ldots,\Gamma^m,\Gamma' \longrightarrow \mathcal{C}}\ mc \qquad ,$$

where $I^1,\ldots,I^m,I'$ *partition the judgments* $\{\mathcal{B}_i\}_{i\in\{1..n\}}$ *among the premise derivations* Π_1, $\ldots$, Π_m,Π'. *For* $1 \leq j \leq m$ *let* Ξ^j *be*

$$\dfrac{\left\{\Delta_i \overset{\Pi_i}{\longrightarrow} \mathcal{B}_i\right\}_{i\in I^j} \quad \overset{\Pi^j}{\{\mathcal{B}_i\}_{i\in I^j},\Gamma^j \longrightarrow \mathcal{D}^j}}{\{\Delta_i\}_{i\in I^j},\Gamma^j \longrightarrow \mathcal{D}^j}\ mc \qquad .$$

Then Ξ *reduces to*

$$\dfrac{\left\{\ldots \overset{\Xi^j}{\longrightarrow} \mathcal{D}^j\right\}_{j\in\{1..m\}} \quad \left\{\Delta_i \overset{\Pi_i}{\longrightarrow} \mathcal{B}_i\right\}_{i\in I'} \quad \ldots \overset{\Pi'}{\longrightarrow} \mathcal{C}}{\Delta_1,\ldots,\Delta_n,\Gamma^1,\ldots\Gamma^m,\Gamma' \longrightarrow \mathcal{C}}\ mc \qquad .$$

Structural case:
$-/c\mathcal{L}$: If Π *is*

$$\dfrac{\overset{\Pi'}{\mathcal{B}_1,\mathcal{B}_1,\mathcal{B}_2,\ldots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{C}}}{\mathcal{B}_1,\mathcal{B}_2,\ldots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{C}}\ c\mathcal{L} \qquad ,$$

then Ξ *reduces to*

$$\dfrac{\dfrac{\Delta_1 \overset{\Pi_1}{\longrightarrow} \mathcal{B}_1 \quad \left\{\Delta_i \overset{\Pi_i}{\longrightarrow} \mathcal{B}_i\right\}_{i\in\{1..n\}} \quad \overset{\Pi'}{\mathcal{B}_1,\mathcal{B}_1,\mathcal{B}_2,\ldots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{C}}}{\Delta_1,\Delta_1,\Delta_2,\ldots,\Delta_n,\Delta_n,\Gamma \longrightarrow \mathcal{C}}\ mc}{\dfrac{}{\Delta_1,\Delta_2,\ldots,\Delta_n,\Gamma \longrightarrow \mathcal{C}}}\ c\mathcal{L} \qquad .$$

$-/\mathrm{w}\mathcal{L}$: If Π *is*

$$\dfrac{\overset{\Pi'}{\mathcal{B}_2,\ldots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{C}}}{\mathcal{B}_1,\mathcal{B}_2,\ldots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{C}}\ \mathrm{w}\mathcal{L} \qquad ,$$

then Ξ *reduces to*

$$\dfrac{\dfrac{\Delta_2 \overset{\Pi_2}{\longrightarrow} \mathcal{B}_2 \quad \ldots \quad \Delta_n \overset{\Pi_n}{\longrightarrow} \mathcal{B}_n \quad \overset{\Pi'}{\mathcal{B}_2,\ldots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{C}}}{\Delta_2,\ldots,\Delta_n,\Gamma \longrightarrow \mathcal{C}}\ mc}{\dfrac{}{\Delta_1,\Delta_2,\ldots,\Delta_n,\Gamma \longrightarrow \mathcal{C}}}\ \mathrm{w}\mathcal{L} \qquad .$$

Axiom cases:
init/ $\circ\mathcal{L}$: Suppose Π *ends with a left-rule acting on* $\mathcal{B}_1$ *and* Π_1 *ends with the init rule. Then it must be the case that* $\Delta_1 = \{\mathcal{B}_1\}$ *and* Ξ *reduces to*

$$\dfrac{\begin{matrix}\Pi_2 \\ \Delta_2 \longrightarrow \mathcal{B}_2\end{matrix} \quad \cdots \quad \begin{matrix}\Pi_n \\ \Delta_n \longrightarrow \mathcal{B}_n\end{matrix} \quad \begin{matrix}\Pi \\ \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{matrix}}{\mathcal{B}_1, \Delta_2, \ldots, \Delta_n, \Gamma \longrightarrow \mathcal{C}} \; mc \quad .$$

$-$/init: If Π *ends with the init rule, then* $n = 1$*,* Γ *is the empty multiset, and* $\mathcal{C}$ *must be a cut formula, i.e.,* $\mathcal{C} = \mathcal{B}_1$*. Therefore* Ξ *reduces to* Π_1*.*

An inspection of the rules of the logic and this definition will reveal that every derivation ending with a multicut has a reduct. Because we use a multiset as the left side of the sequent, there may be ambiguity as to whether a formula occurring on the left side of the rightmost premise to a multicut rule is in fact a cut formula, and if so, which of the left premises corresponds to it. As a result, several of the reduction rules may apply, and so a derivation may have multiple reducts.

The following lemmas show that the reduction relation is preserved by some of the transformation of derivations defined in previous chapter. Observe that the redexes of a derivation are not affected by substitution or weakening of signatures, since both transformations do not change the last rule of a derivation.

Lemma 4.2. *Let* Π *be a derivation of* $\Sigma\,;\,\Gamma \longrightarrow \mathcal{C}$ *ending with a mc. Let* Σ' *be a signature different from* Σ*. Then* Π *reduces to a derivation* Π' *if and only if* $\Sigma' : \Pi$ *reduces to* $\Sigma' : \Pi'$*.*

Proof By induction on ht(Π) and inspection on the reduction rules. Induction hypothesis is needed in the case *mc/* $\circ\mathcal{L}$. In the cases $*/\mu\mathcal{L}$ and $\nu\mathcal{R}/\nu\mathcal{L}$ we make use of Lemma 3.23 and Lemma 3.24, respectively. ∎

Lemma 4.3. *Let* Π *be a derivation of* $\Sigma\,;\,\Gamma \longrightarrow \mathcal{C}$ *ending with a mc and let* θ *be a substitution. Then* Π *reduces to a derivation* Π' *if and only if* $\Pi\theta$ *reduces to* $\Pi'\theta$*.*

Proof By induction on ht(Π), Lemma 3.21 and Lemma 3.22. ∎

Lemma 4.4. *Let* $p\,\bar{x} \overset{\mu}{=} D\,p\,\bar{x}$ *be an inductive definition and let* Π_S *be a derivation of* $\bar{x}\,;\,D\,S\,\bar{x} \longrightarrow S\,\bar{x}$ *for some invariant* S*. Let* $\mathcal{C}\,p$ *be a non-atomic judgment in which* p *occurs strictly positively. Let* Π *and* Π' *be two derivations of the same sequent* $\Sigma\,;\,\Gamma \longrightarrow \mathcal{C}\,p$*, and* Π *ends with a mc rule. Then the derivation* Π *reduces to a derivation* Π' *if and only the derivation* $\mu(\Pi, \Pi_S)$ *reduces to* $\mu(\Pi', \Pi_S)$*.*

Proof By case analysis on the reduction rules. The case analysis can be much simplified by the following observation. First, the reduction rules are driven only by outermost connectives in the formulas in the sequent. Second, the unfolding of a derivation affects only the right-handside of the sequents appearing in the derivation (or more specifically, only the branches containing major premises). By a quick inspection on the definition of reduction rules in Definition 4.1, we see that the only non-trivial case to consider is the right-commutative case $-/\circ\mathcal{R}$. Since $\mathcal{C}\,p$ is non-atomic (and assuming that it has at least one occurrence of p, otherwise it is trivial since $\Pi = \mu(\Pi, \Pi_S)$ in this case), the only cases we need to verify is when its topmost logical connective is either $\wedge$, $\vee$, $\supset$, $\forall$, $\exists$, ∇. In these cases, the unfolding does not change the topmost connective, therefore any reduction rule that applies to Π also applies to $\mu(\Pi, \Pi_S)$ and vice versa. Lemma 3.21, 3.22, 3.23 and 3.24 are used when substitutions are involved (right/left commutative cases with eq$\mathcal{L}$) and when global signature weakening are involved (right/left commutative cases with $\forall$ and $\exists$). ∎

LEMMA 4.5. *Let $p\,\bar{x} \overset{\nu}{=} D\,p\,\bar{x}$ be a co-inductive definition and let Π_S be a derivation of $\bar{x}\,;\, S\,\bar{x} \longrightarrow D\,S\,\bar{x}$ for some invariant S. Let $\mathcal{C}\,p$ be a non-atomic judgment in which p occurs strictly positively. Let Π and Π' be two derivations of the sequent $\Sigma\,;\,\Gamma \longrightarrow \mathcal{C}\,S$, where Π ends with a mc rule. Then the derivation Π reduces to a derivation Π' if and only the derivation $\nu(\Pi, \Pi_S)$ reduces to $\nu(\Pi', \Pi_S)$.*

Proof Analogous to the proof of Lemma 4.4. ∎

4.2 Normalizability and reducibility

We now define two properties of derivations: normalizability and reducibility. Each of these properties implies that the derivation can be reduced to a cut-free derivation of the same end-sequent.

4.2.1 Normalizability

DEFINITION 4.6. *We define the set of* normalizable *derivations to be the smallest set that satisfies the following conditions:*

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

These clauses assert that a given derivation is normalizable provided certain (perhaps infinitely many) other derivations are normalizable. If we call these other derivations the predecessors of the given derivation, then a derivation is normalizable if and only if the tree of the derivation and its successive predecessors is well-founded. In this case, the well-founded tree is called the normalization *of the derivation.*

The set of all normalizable derivations is denoted by $\aleph$.

Since a normalization is well-founded, it has an associated induction principle: for any property P of derivations, if for every derivation Π in the normalization, P holds for every predecessor of Π implies that P holds for Π, then P holds for every derivation in the normalization.

LEMMA 4.7. *If there is a normalizable derivation of a sequent, then there is a cut-free derivation of the sequent.*

Proof Let Π be a normalizable derivation of the sequent $\Gamma \longrightarrow \mathcal{B}$. We show by induction on the normalization of Π that there is a cut-free derivation of $\Gamma \longrightarrow \mathcal{B}$.

1. If Π ends with a multicut, then any of its reducts is one of its predecessors and so is normalizable. But the reduct is also a derivation of $\Gamma \longrightarrow \mathcal{B}$, so by the induction hypothesis this sequent has a cut-free derivation.

2. Suppose Π ends with a rule other than multicut. Since we are given that Π is normalizable, by definition the premise derivations are normalizable. These premise derivations are the predecessors of Π, so by the induction hypothesis there are cut-free derivations of the premises. Thus there is a cut-free derivation of $\Gamma \longrightarrow \mathcal{B}$.

∎

The next two lemmas are also proved by induction on the normalization of the given derivation.

Lemma 4.8. *If* Π *is a normalizable derivation, then for any substitution* θ*,* $\Pi\theta$ *is normalizable.*

Proof We prove this lemma by induction on the normalization of Π.

1. If Π ends with a multicut, then $\Pi\theta$ also ends with a multicut. By Lemma 4.3 every reduct of $\Pi\theta$ corresponds to a reduct of Π, therefore by induction hypothesis every reduct of $\Pi\theta$ is normalizable, and hence $\Pi\theta$ is normalizable.

2. Suppose Π ends with a rule other than multicut and has premise derivations $\{\Pi_i\}$. By Definition 3.1 each premise derivation in $\Pi\theta$ is either Π_i or $\Pi_i\theta$. Since Π is normalizable, Π_i is normalizable, and so by the induction hypothesis $\Pi_i\theta$ is also normalizable. Thus $\Pi\theta$ is normalizable. ∎

Lemma 4.9. *If* Π *is a normalizable derivation, then for any signature* Σ *the derivation* $\Sigma : \Pi$ *is normalizable.*

Proof By induction on the normalization of Π. The proof is analogous to the proof of Lemma 4.8, but in the case where Π ends with *mc* we use Lemma 4.2 instead. ∎

4.2.2 Generated Sets

In defining the notion of reducibility in the following section, we will need to construct a closed set in defining the reducibility of derivations ending with co-induction rules. The main closure condition is that related to the co-inductive invariants used in the derivation. We call the least of such closed set the generated set. Its main use will be in proving that reducibility is preserved under co-inductive unfolding of derivations.

Let $\mathcal{L}$ denote the set of all derivations and $\mathcal{P}(\mathcal{L})$ denote the power set of $\mathcal{L}$. Then the set $\mathcal{P}(\mathcal{L})$ together with order relation $\subseteq$ (set inclusion), greatest lower bound operation $\cap$ and least upper bound operation $\cup$ defines a complete lattice. The generated set will be defined as the least fixed point of a monotone function on this lattice. The precise definition follows.

DEFINITION 4.10. *Let $p\,\bar{x} \overset{\nu}{=} B\,p\,\bar{x}$ be a co-inductive definition. Let $\mathcal{R}$ be a set of derivations such that for all $\Psi \in \mathcal{R}$, it holds that $\mathrm{lvl}(\Psi) < \mathrm{lvl}(p)$. Let Φ be a derivation of $\Sigma\,;\Gamma \longrightarrow \mathcal{C}$ such that $\mathrm{lvl}(\mathcal{C}) \leq \mathrm{lvl}(p)$ and let Π_S be a derivation of $\bar{x}\,; S\,\bar{x} \longrightarrow B\,S\,\bar{x}$ for some invariant S, such that $\mathrm{lvl}(S) \leq \mathrm{lvl}(p)$. We define a function $\mathcal{W}_{\mathcal{R},\Phi,\Pi_S} : \mathcal{P}(\mathcal{L}) \rightarrow \mathcal{P}(\mathcal{L})$ (abbreviated as $\mathcal{W}$ below) as follows, given an input set $\mathcal{K}$.*

1. *$\Phi \in \mathcal{W}(\mathcal{K})$,*

2. *for every derivation $\Xi \in \mathcal{K}$ and for every substitution θ the derivation $\Xi\theta$ is in $\mathcal{W}(\mathcal{K})$,*

3. *for every derivation $\Xi \in \mathcal{K}$ of the sequent $\Sigma'\,;\Gamma \longrightarrow \mathcal{D}$ and for every signature Σ'' such that Σ'' and Σ' are disjoint, the derivation $\Sigma'' : \Xi$ is in $\mathcal{W}(\mathcal{K})$,*

4. *for every derivation Ξ of $\Sigma'\,;\Delta \longrightarrow \mathcal{D}$ in $\mathcal{K}$,*

   (a) if $\mathcal{D} = \bar{z} \triangleright S\,\bar{u}$ then

$$\dfrac{\overset{\Xi}{\Sigma'\,;\Delta \longrightarrow \bar{z} \triangleright S\,\bar{u}} \qquad \overset{\Pi'_S}{\Sigma'\,;\bar{z} \triangleright S\,\bar{u} \longrightarrow \bar{z} \triangleright B\,S\,\bar{u}}}{\Sigma'\,;\Delta \longrightarrow \bar{z} \triangleright B\,S\,\bar{u}}\ mc\ ,$$

   where Π'_S is a raised instance of Π_S, is in $\mathcal{W}(\mathcal{K})$,

   (b) otherwise,

   i. if Ξ ends with $\supset\mathcal{R}$

$$\dfrac{\overset{\Xi'}{\Delta, \mathcal{D}_1 \longrightarrow \mathcal{D}_2}}{\Delta \longrightarrow \mathcal{D}_1 \supset \mathcal{D}_2}\ \supset\mathcal{R}$$

   then $\Xi' \in \mathcal{W}(\mathcal{K})$ and for every substitution θ and signature Σ'' and for every derivation Ψ of $\Delta' \longrightarrow \mathcal{D}_1\theta$ in $\mathcal{R}$, the derivation

$$\dfrac{\overset{\Psi}{\Gamma \longrightarrow \mathcal{D}_1\theta} \qquad \overset{\Sigma'':\Xi'\theta}{\mathcal{D}_1\theta, \Delta\theta \longrightarrow \mathcal{D}_2\theta}}{\Delta', \Delta\theta \longrightarrow \mathcal{D}_2\theta}\ mc$$

   is in $\mathcal{W}(\mathcal{K})$.

   ii. if Ξ ends with mc, then every reduct of Π is in $\mathcal{W}(\mathcal{C})$,

   iii. if Ξ ends with any other rule then its major premises are in $\mathcal{W}(\mathcal{C})$.

The $(\mathcal{R}, \Phi, \Pi_S)$-generated set *is defined as the least fixed point of $\mathcal{W}$.*

From now on, when we write $\mathcal{W}_{\mathcal{R},\Phi,\Pi_S}$, it is understood that the derivations Φ and Π_S take the form as indicated in Definition 4.10, and that the underlying coinductive definition and invariant are assumed to be given.

We define the ordinal powers of $\mathcal{W}$ as follows

$$\begin{array}{lcl} \mathcal{W}^0 & = & \mathcal{W}(\emptyset), \\ \mathcal{W}^{n+1} & = & \mathcal{W}(\mathcal{W}^n) \end{array}$$

and $\mathcal{W}^\omega = \bigcup\{\mathcal{W}^k \mid k < \omega\}$.

Lemma 4.11. *The function $\mathcal{W}_{\mathcal{R},\Phi,\Pi_S}$ in Definition* 4.10 *is*

1. *monotone, i.e., for any sets $\mathcal{K}_1 \subseteq \mathcal{K}_2$, $\mathcal{W}_{\mathcal{R},\Phi,\Pi_S}(\mathcal{K}_1) \subseteq \mathcal{W}_{\mathcal{R},\Phi,\Pi_S}(\mathcal{K}_2)$,*
2. *and continuous, i.e., for any chain*

$$\mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \mathcal{K}_3 \subseteq ....$$

we have

$$\mathcal{W}_{\mathcal{R},\Phi,\Pi_S}(\cup\{\mathcal{K}_i \mid i < \omega\}) = \bigcup\{\mathcal{W}_{\mathcal{R},\Phi,\Pi_S}(\mathcal{K}_i) \mid i < \omega\}.$$

Proof We abbreviate $\mathcal{W}_{\mathcal{R},\Phi,\Pi_S}$ as simply $\mathcal{W}$.

1. Let $\Psi \in \mathcal{W}(\mathcal{K}_1)$. If $\Psi = \Phi$ then $\Psi \in \mathcal{W}(\mathcal{K}_2)$. Otherwise, Ψ is obtained from a derivation $\Psi' \in \mathcal{K}_1$ by applying one of the operations (1) - (4) in Definition 4.10. Since $\Psi' \in \mathcal{K}_2$, we can apply the same operation to get Ψ in $\mathcal{W}(\mathcal{K}_2)$.
2. We show that $\cup\{\mathcal{W}(\mathcal{K}_i) \mid i < \omega\} \subseteq \mathcal{W}(\cup\{\mathcal{K}_i \mid i < \omega\})$ and $\mathcal{W}(\cup\{\mathcal{K}_i \mid i < \omega\}) \subseteq \cup\{\mathcal{W}(\mathcal{K}_i) \mid i < \omega\}$.

   (a) for every $j < \omega$,

$$\begin{array}{rcl} \mathcal{K}_j \subseteq \cup\{\mathcal{K}_i \mid i < \omega\} & \Rightarrow & \text{(by monotonicity of } \mathcal{W}) \\ & & \mathcal{W}(\mathcal{K}_j) \subseteq \mathcal{W}(\cup\{\mathcal{K}_i \mid i < \omega\}) \\ & \Rightarrow & \cup\{\mathcal{W}(\mathcal{K}_i) \mid i < \omega\} \subseteq \mathcal{W}(\cup\{\mathcal{K}_i \mid i < \omega\}) \end{array}$$

   (b) Let Ψ be an element of $\mathcal{W}(\cup\{\mathcal{K}_i \mid i < \omega\})$. If $\Psi = \Phi$ then obviously $\Psi \in \cup\{\mathcal{W}(\mathcal{K}_i) \mid i < \omega\}$. Otherwise, Ψ is obtained from an element Ψ' in $\cup\{\mathcal{K}_i \mid i < \omega\}$ by some operation defined in Definition 4.10. Since Ψ' is also in some $\mathcal{K}_k$, we have $\Psi \in \mathcal{W}(\mathcal{K}_k) \subseteq \cup\{\mathcal{W}(\mathcal{K}_i) \mid i < \omega\}$.

∎

It is a known fact that for every continous function $\mathcal{W}$ (see [4] for example), the least fixed point of $\mathcal{W}$ is $\mathcal{W}^\omega$. Therefore, the $(\mathcal{R}, \Phi, \Pi_S)$-generated set is exactly $\mathcal{W}^\omega_{\mathcal{R},\Phi,\Pi_S}$. This characterization will be useful in proving some properties of $\mathcal{W}_{\mathcal{R},\Phi,\Pi_S}$. In particular, the fact that $\mathcal{W}^\omega_{\mathcal{R},\Phi,\Pi_S}$ is the least fixed point implies that we can use ordinary induction to prove its properties.

The proof for the following lemma is immediate by induction on the construction of generated sets, Definition 4.10 and Definition 3.2 (definition of major premises).

LEMMA 4.12. *Let $\mathcal{W}^{\omega}_{\mathcal{R},\Phi,\Pi_S}$ be a generated set with the underlying co-inductive definition $p\,\bar{x} \stackrel{\nu}{=} B\,p\,\bar{x}$. Then for every $\Xi \in \mathcal{W}^{\omega}_{\mathcal{R},\Phi,\Pi_S}$, $\mathrm{lvl}(\Xi) \leq \mathrm{lvl}(p)$.*

LEMMA 4.13. *Let $\mathcal{W}_1 = \mathcal{W}_{\mathcal{R},\Phi_1,\Pi_S}$ and let $\mathcal{W}_2 = \mathcal{W}_{\mathcal{R},\Phi_2,\Pi_S}$. If $\Phi_1 \in \mathcal{W}^{\omega}_2$ then $\mathcal{W}^{\omega}_1 \subseteq \mathcal{W}^{\omega}_2$.*

Proof It is enough to show that for every $i < \omega$, $\mathcal{W}^{i}_1 \subseteq \mathcal{W}^{\omega}_2$. The case where $i = 0$ is trivial. Suppose $\mathcal{W}^{i}_1 \subseteq \mathcal{W}^{\omega}_2$. We show that $\mathcal{W}^{i+1}_1 \subseteq \mathcal{W}^{\omega}_2$. Note that the only difference between $\mathcal{W}_1$ and $\mathcal{W}_2$ is in the operation (1) in Definition 4.10. Let $\Xi \in \mathcal{W}^{i+1}_1$. If Ξ is Φ_1 then $\Xi \in \mathcal{W}^{\omega}_2$ by inductive hypothesis. Otherwise, suppose Ξ is obtained from $\Xi' \in \mathcal{W}^{i}_1$ by some operation (2) - (4) in Definition 4.10. Then we can perform the same operation on $\mathcal{W}^{\omega}_2$, that is, we have $\Xi \in \mathcal{W}_2(\mathcal{W}^{\omega}_2)$. But we know that $\mathcal{W}^{\omega}_2$ is a fixed point of $\mathcal{W}_2$, and therefore $\Xi \in \mathcal{W}^{\omega}_2$. ∎

4.2.3 Reducibility

The inductive definition of reducibility is done by induction on the level of the derivation: in the definition of reducibility for derivations of level i we assume that reducibility is already defined for all levels $j < i$. (Recall from Definition 2.3 that the level of a derivation is defined to be the level of the consequent of its end-sequent.) In the particular case involving co-induction, a generated set is constructed from reducible derivations of level $j < i$. We shall denote the set of all reducible derivations with $\Re$. The notation $\Re_i$ denotes the set of all reducible derivations of level i, while $\Re_{<i}$ ($\Re_{\leq i}$) denotes the set of reducible derivations of level smaller than i (respectively, smaller or equal to i). In the following definition, we refer to a derivation Π as an i-level derivation if $\mathrm{lvl}(\Pi) = i$.

DEFINITION 4.14. *For any i, we define the set of* reducible *i-level derivations to be the smallest set of i-level derivations that satisfies the following conditions:*

1. *If an i-level derivation Π ends with a multicut, then it is i-reducible if every reduct of Π is i-reducible.*

2. *Suppose the i-level derivation Π ends with the implication right rule*

$$\dfrac{\begin{array}{c}\Pi' \\ \mathcal{B}, \Gamma \longrightarrow \mathcal{C}\end{array}}{\Gamma \longrightarrow \mathcal{B} \supset \mathcal{C}} \supset\mathcal{R} \quad .$$

Let j be the level of $\mathcal{B}$ and let k be the level of $\mathcal{C}$. Then the derivation is i-reducible if the premise derivation Π' is k-reducible and, for every substitution θ, signature Σ', multiset Δ of formulas, and j-reducible derivation Π'' of $\Delta \longrightarrow \mathcal{B}\theta$, the derivation

$$\dfrac{\begin{array}{c}\Pi'' \\ \Delta \longrightarrow \mathcal{B}\theta\end{array} \qquad \begin{array}{c}\Sigma' : \Pi'\theta \\ \mathcal{B}\theta, \Gamma\theta \longrightarrow \mathcal{C}\theta\end{array}}{\Delta, \Gamma\theta \longrightarrow \mathcal{C}\theta}\, mc$$

is k-reducible.

3. *If the derivation ends with the $\supset\mathcal{L}$ rule or the $\mu\mathcal{L}$ rule, then it is i-reducible if the right premise derivation is i-reducible and the other premise derivations are normalizable.*

4. *Suppose Π ends with $\nu\mathcal{R}$ rule*

$$\dfrac{\begin{matrix}\Pi' \\ \Sigma\,;\,\Gamma \longrightarrow \bar{z} \triangleright S\,\bar{t}\end{matrix} \quad \begin{matrix}\Pi_S \\ \bar{x}\,;\,S\,\bar{x} \longrightarrow B\,S\,\bar{x}\end{matrix}}{\Sigma\,;\,\Gamma \longrightarrow \bar{z} \triangleright p\,\bar{t}}\ \nu\mathcal{R}$$

where $p\,\bar{x} \overset{\nu}{=} B\,p\,\bar{x}$. Let j be the level of Π' and let k be the level of Π_S. Then Π is i-reducible if Π' is j-reducible, Π_S is k-reducible, and for every derivation Ψ in the $(\Re_{<i}, \Pi', \Pi_S)$-generated set, Ψ is $\mathrm{lvl}(\Psi)$-reducible.

5. *If the derivation ends with any other rule, with the premise derivations $\{\Pi_k\}_{k\in I}$ for some index set I, then it is i-reducible if every derivation Π_k is $\mathrm{lvl}(\Pi_k)$-reducible.*

These clauses assert that a given derivation is reducible provided certain (perhaps infinitely many) other derivations are reducible. If we call these other derivations the predecessors of the given derivation, then a derivation is reducible only if the tree of the derivation and its successive predecessors is well-founded. In this case, the well-founded tree is called the reduction *of the derivation.*

In defining reducibility for a derivation of (writing the signatures explicitly)

$$\Sigma\,;\,\Gamma \longrightarrow \bar{z} \triangleright B \supset C$$

ending with $\supset\mathcal{R}$ we quantify over reducible derivations of

$$\Sigma\theta \cup \Sigma'\,;\,\Delta \longrightarrow (\bar{z} \triangleright B)\theta,$$

for some new signature Σ'. This is legitimate since we are defining reducibility for a derivation having level $\max(\mathrm{lvl}(B)+1, \mathrm{lvl}(C))$, so the set of reducible derivations having level $\mathrm{lvl}(B)$ is already defined (substitution does not change the level).

For a derivation ending with $\supset\mathcal{L}$ or $\mu\mathcal{L}$, some premise derivations may have consequents with a higher level than that of the consequent of the conclusion. As a result, we cannot use the reducibility of those premise derivations to define the reducibility of the derivation as a whole, since the reducibility of the premise derivations may not yet be defined. Thus we use the weaker notion of normalizability for those premise derivations.

In the $\nu\mathcal{R}$ case, the reducibility of Π is defined only if the generated set induced by its premises contains only reducible derivations. Lemma 4.12 ensures that the derivations in the generated set has lower or equal level as Π. Also observe that the consequent of the premise to the rule *def*$\mathcal{L}$ and *def*$\mathcal{R}$ cannot have a higher level than the consequent of the conclusion because of the level restriction on definitional clauses. Finally, as with normalizations, reductions have associated induction principles.

From now on, we shall refer to an i-reducible derivation, for some level i, as simply reducible derivation.

Lemma 4.15. *If a derivation is reducible, then it is normalizable.*

Proof By induction on the reduction tree and Definition 4.6 (normalizability). ∎

Lemma 4.16. *If* Π *is a reducible derivation, then for any substitution* θ*,* $\Pi\theta$ *is reducible.*

Proof We prove this lemma by induction on the reduction of Π.

1. If Π ends with a multicut, then $\Pi\theta$ also ends with a multicut. By Lemma 4.3 every reduct of $\Pi\theta$ is an instance of a reduct of Π. Hence by induction hypothesis all reducts of $\Pi\theta$ are reducible.

2. If Π ends with the implication right rule then Π and $\Pi\theta$ are

$$\dfrac{\begin{array}{c}\Pi'\\ \Sigma\,;\mathcal{B},\Gamma\longrightarrow\mathcal{C}\end{array}}{\Sigma\,;\Gamma\longrightarrow\mathcal{B}\supset\mathcal{C}}\supset\mathcal{R}\qquad\qquad \dfrac{\begin{array}{c}\Pi'\theta\\ \Sigma\theta\,;\mathcal{B}\theta,\Gamma\theta\longrightarrow\mathcal{C}\theta\end{array}}{\Sigma\theta\,;\Gamma\theta\longrightarrow(\mathcal{B}\supset\mathcal{C})\theta}\supset\mathcal{R}\quad .$$

$\Pi\theta$ is reducible if $\Pi'\theta$ is reducible and, for every substitution θ', signature Σ', multiset Δ, and reducible derivation Π'', the derivation Ξ

$$\dfrac{\begin{array}{c}\Pi''\\ \Sigma\theta\theta'\,;\Delta\longrightarrow\mathcal{B}\theta\theta'\end{array}\qquad\begin{array}{c}\Sigma':\Pi'\theta\theta'\\ \Sigma\theta\theta'\,;\mathcal{B}\theta\theta',\Gamma\theta\theta'\longrightarrow\mathcal{C}\theta\theta'\end{array}}{\Sigma\theta\theta'\,;\Delta,\Gamma\theta\theta'\longrightarrow\mathcal{C}\theta\theta'}\;mc$$

is also reducible. $\Pi'\theta$ is reducible by the induction hypothesis, and Ξ is reducible since it is a predecessor of Π. Therefore $\Pi\theta$ is reducible.

3. If Π ends with $\supset\mathcal{L}$ or $\mu\mathcal{L}$, then the right premise derivation is reducible and the other premise derivations are normalizable. By Definition 3.1 each premise derivation in $\Pi\theta$ is obtained by applying θ to a premise derivation in Π. By the induction hypothesis the right premise derivation of $\Pi\theta$ is reducible, and by Lemma 4.8 the other premise derivations are normalizable. Thus $\Pi\theta$ is reducible.

4. Suppose Π ends with $\nu\mathcal{R}$,

$$\dfrac{\begin{array}{c}\Pi'\\ \Sigma\,;\Delta\longrightarrow\bar{z}\rhd S\,\bar{t}\end{array}\quad\begin{array}{c}\Pi_S\\ \bar{x}\,;S\,\bar{x}\longrightarrow D\,S\,\bar{x}\end{array}}{\Sigma\,;\Delta\longrightarrow\bar{z}\rhd p\,\bar{t}}\;\nu\mathcal{R}$$

where $p\,\bar{x}\stackrel{\nu}{=}D\,p\,\bar{x}$. Let $\mathcal{W}$ be the $(\Re_{<i},\Pi',\Pi_S)$-generated set where $i=\mathrm{lvl}(p)$. Then Π' and Π_S are reducible, and the generated set $\mathcal{W}$ contains only reducible derivations. The derivation $\Pi\theta$ is

$$\dfrac{\begin{array}{c}\Pi'\theta\\ \Sigma\theta\,;\Delta\theta\longrightarrow(\bar{z}\rhd S\,\bar{t})\theta\end{array}\quad\begin{array}{c}\Pi_S\\ \bar{x}\,;S\,\bar{x}\longrightarrow D\,S\,\bar{x}\end{array}}{\Sigma\theta\,;\Delta\theta\longrightarrow(\bar{z}\rhd p\,\bar{t})\theta}\;\nu\mathcal{R}\quad .$$

The derivation $\Pi'\theta$ is reducible by inductive hypothesis. It remains to show that the $(\Re_{<i}, \Pi'\theta, \Pi_S)$-generated set $\mathcal{W}'$ contains only reducible derivations. From Definition 4.10 we see that $\Pi'\theta \in \mathcal{W}$, and therefore by Lemma 4.13 $\mathcal{W}' \subseteq \mathcal{W}$ and hence all derivations in $\mathcal{W}'$ are reducible.

5. Suppose Π ends with any other rule and has premise derivations $\{\Pi_i\}$. By Definition 3.1 each premise derivation in $\Pi\theta$ is either Π_i or $\Pi_i\theta$. Since Π is reducible, Π_i is reducible, and so by the induction hypothesis $\Pi_i\theta$ is also reducible. Thus $\Pi\theta$ is reducible. ∎

Lemma 4.17. *Let* Π *be a reducible derivation of* $\Sigma\,;\Gamma \longrightarrow \mathcal{C}$. *Let* Σ' *be a signature different from* Σ. *Then the derivation* $\Sigma' : \Pi$ *is reducible.*

Proof By induction on the reduction of Π. The proof is analogous to the proof of Lemma 4.16. Note that in the case where Π ends with $\nu\mathcal{R}$, the weakening of global signature is already included in the definition of generated set (Definition 4.10 (3)). ∎

4.2.4 Reducibility of unfolded derivations

The following lemmas state that reducibility is preserved by (co)inductive unfolding, under certain assumptions.

Lemma 4.18. Inductive unfolding. *Let* $p\,\bar{x} \overset{\mu}{=} B\,p\,\bar{x}$ *be an inductive definition. Let* Π_S *be a reducible derivation of* $\bar{x}\,; B\,S\,\bar{x} \longrightarrow S\,\bar{x}$. *Let* Π *be a reducible derivation of* $\Sigma\,;\Gamma \longrightarrow \mathcal{C}\,p$ *such that every occurrence of* p *in* $\mathcal{C}\,p$ *is strictly positive. Suppose the following statements hold*

1. *for every reducible derivation* Ξ *of* $\Sigma'\,;\Delta \longrightarrow \bar{z} \triangleright B\,S\,\bar{u}$ *the derivation*

$$\cfrac{\begin{matrix}\Xi \\ \Sigma'\,;\Delta \longrightarrow \bar{z} \triangleright B\,S\,\bar{u}\end{matrix} \qquad \begin{matrix}\Pi'_S \\ \Sigma'\,;\bar{z} \triangleright B\,S\,\bar{u} \longrightarrow \bar{z} \triangleright S\,\bar{u}\end{matrix}}{\Sigma'\,;\Delta \longrightarrow \bar{z} \triangleright S\,\bar{u}}\,mc$$

is reducible, where Π'_S *is a raised instance of* Π_S *and*

2. *for every* $\mathcal{D}$, *the derivation* Id *of* $\mathcal{D} \longrightarrow \mathcal{D}$ *is reducible, and for every reducible derivation* Ξ *of* $\Delta \longrightarrow \mathcal{D}$ *the derivation*

$$\cfrac{\begin{matrix}\Xi \\ \Delta \longrightarrow \mathcal{D}\end{matrix} \quad \begin{matrix}Id \\ \mathcal{D} \longrightarrow \mathcal{D}\end{matrix}}{\Delta \longrightarrow \mathcal{D}}\,mc$$

is reducible.

Then the derivation $\mu(\Pi, \Pi_S)$ *of* $\Sigma\,;\Gamma \longrightarrow \mathcal{C}\,S$ *is reducible.*

Proof By induction on the reduction of Π. We show the non-trivial cases, assuming that $\mathcal{C}\,S \neq \mathcal{C}\,p$.

1. Suppose Π ends with *init* rule on $\bar{z} \triangleright p\,\bar{u}$. Then $\mu(\Pi, \Pi_S)$ is the derivation

$$\dfrac{\overset{\Pi_S}{\bar{x}\,;\, B\,p\,\bar{x} \longrightarrow S\,\bar{x}} \quad \overset{Id}{\Sigma\,;\,\bar{z} \triangleright S\,\bar{u} \longrightarrow \bar{z} \triangleright S\,\bar{u}}}{\Sigma\,;\,\bar{z} \triangleright p\,\bar{u} \longrightarrow \bar{z} \triangleright S\,\bar{u}}\ \mu\mathcal{L} \quad .$$

The reducibility of $\mu(\Pi, \Pi_S)$ follows immediately from the assumptions.

2. Suppose Π ends with $\supset \mathcal{R}$, that is, $\mathcal{C}\,p = \mathcal{C}_1\,p \supset \mathcal{C}_2\,p$.

$$\dfrac{\overset{\Pi'}{\Gamma, \mathcal{C}_1\,p \longrightarrow \mathcal{C}_2\,p}}{\Gamma \longrightarrow \mathcal{C}_1\,p \supset \mathcal{C}_2\,p} \supset \mathcal{R} \quad .$$

By the restriction on $\mathcal{C}\,p$, it must be the case that $\mathcal{C}\,S = \mathcal{C}_1\,p \supset \mathcal{C}_2\,S$ since p is vacuous in $\mathcal{C}_1\,p$. By the definition of reducibility, the derivation Π' is reducible and for substitution θ, signature Σ', multiset of formulas Δ, and reducible derivation Ψ of $\Delta \longrightarrow (\mathcal{C}_1\,p)\theta$, the derivation Ξ

$$\dfrac{\overset{\Psi}{\Delta \longrightarrow (\mathcal{C}_1\,p)\theta} \qquad \overset{\Sigma' : \Pi'\theta}{\Gamma\theta, (\mathcal{C}_1\,p)\theta \longrightarrow (\mathcal{C}_2\,p)\theta}}{\Delta \longrightarrow (\mathcal{C}_2\,p)\theta}\ mc$$

is reducible. We want to show that the derivation $\mu(\Pi, \Pi_S)$

$$\dfrac{\overset{\mu(\Pi', \Pi_S)}{\Gamma, \mathcal{C}_1\,p \longrightarrow \mathcal{C}_2\,S}}{\Gamma \longrightarrow \mathcal{C}_1\,p \supset \mathcal{C}_2\,S} \supset \mathcal{R}$$

is reducible. This reduces to showing that $\mu(\Pi', \Pi_S)$ is reducible and that

$$\dfrac{\overset{\Psi}{\Delta \longrightarrow (\mathcal{C}_1\,p)\theta} \qquad \overset{\Sigma' : \mu(\Pi', \Pi_S)\theta}{\Gamma\theta, (\mathcal{C}_1\,p)\theta \longrightarrow (\mathcal{C}_2\,S)\theta}}{\Delta \longrightarrow (\mathcal{C}_2\,S)\theta}\ mc$$

is reducible. The first follows from induction hypothesis on Π'. For the second derivation, we know from Lemma 3.21 and Lemma 3.23 that

$$\Sigma' : \mu(\Pi', \Pi_S)\theta = \Sigma' : \mu(\Pi'\theta, \Pi_S) = \mu(\Sigma' : \Pi'\theta, \Pi_S) = \mu(\Xi, \Pi_S).$$

We can apply induction hypothesis on Ξ, since it is a predecessor of Π. Therefore, the derivation $\mu(\Pi, \Pi_S)$ is reducible.

3. Suppose Π ends with $\mu\mathcal{R}$ rule on $\bar{z} \triangleright p\,\bar{u}$.

$$\dfrac{\overset{\Pi'}{\Sigma\,;\,\Gamma \longrightarrow \bar{z} \triangleright B\,p\,\bar{u}}}{\Sigma\,;\,\Gamma \longrightarrow \bar{z} \triangleright p\,\bar{u}}\ \mu\mathcal{R} \quad .$$

Then $\mu(\Pi, \Pi_S)$ is the derivation

$$\dfrac{\dfrac{\mu(\Pi', \Pi_S)}{\Sigma\,;\Gamma \longrightarrow \bar{z} \rhd B\,S\,\bar{u}} \qquad \dfrac{\Pi'_S}{\Sigma\,;\,\bar{z} \rhd B\,S\,\bar{u} \longrightarrow \bar{z} \rhd S\,\bar{u}}}{\Sigma\,;\Gamma \longrightarrow \bar{z} \rhd S\,\bar{u}}\;mc \quad ,$$

where Π'_S is a raised instance of Π_S. The derivation $\mu(\Pi', \Pi_S)$ is reducible by inductive hypothesis. This, together with the assumptions of the lemma, implies that $\mu(\Pi, \Pi_S)$ is reducible.

4. Suppose Π ends with *mc*.

$$\dfrac{\dfrac{\Pi_1}{\Delta_1 \longrightarrow \mathcal{D}_1} \quad \cdots \quad \dfrac{\Pi_n}{\Delta_n \longrightarrow \mathcal{D}_m} \quad \dfrac{\Pi'}{\mathcal{D}_1, \ldots, \mathcal{D}_m, \Gamma' \longrightarrow \mathcal{C}\,p}}{\Delta_1, \ldots, \Delta_m, \Gamma' \longrightarrow \mathcal{C}\,p}\;mc \quad .$$

Then $\mu(\Pi, \Pi_S)$ is the derivation

$$\dfrac{\dfrac{\Pi_1}{\Delta_1 \longrightarrow \mathcal{D}_1} \quad \cdots \quad \dfrac{\Pi_n}{\Delta_n \longrightarrow \mathcal{D}_m} \quad \dfrac{\mu(\Pi', \Pi_S)}{\mathcal{D}_1, \ldots, \mathcal{D}_m, \Gamma' \longrightarrow \mathcal{C}\,S}}{\Delta_1, \ldots, \Delta_m, \Gamma' \longrightarrow \mathcal{C}\,S}\;mc \quad .$$

By the definition of reducibility, every reduct of Π is reducible. We need to show that every reduct of $\mu(\Pi, \Pi_S)$ is reducible.

From Lemma 4.4, we know that for the case where $(\mathcal{C}\,p)$ is not atomic every reduct of $\mu(\Pi, \Pi_S)$ corresponds to some reduct of Π. Therefore inductive hypothesis can be applied to show the reducibility of each reduct of $\mu(\Pi, \Pi_S)$. If $(\mathcal{C}\,p)$ is atomic, that is, suppose $(\mathcal{C}\,p) = \bar{z} \rhd p\,\bar{u}$, we have the following two cases.

Suppose Π' is the derivation

$$\dfrac{\dfrac{\Pi''}{\Sigma\,;\,\mathcal{D}_1, \ldots, \mathcal{D}_m, \Gamma' \longrightarrow \bar{z} \rhd B\,p\,\bar{u}}}{\Sigma\,;\,\mathcal{D}_1, \ldots, \mathcal{D}_m, \Gamma' \longrightarrow \bar{z} \rhd p\,\bar{u}}\;\mu\mathcal{R} \quad .$$

Let Ξ_1 be the derivation

$$\dfrac{\left\{ \dfrac{\Pi_j}{\Sigma\,;\,\Delta_j \longrightarrow \mathcal{D}_j} \right\}_{j \in \{1, \ldots, m\}} \qquad \dfrac{\Pi''}{\Sigma\,;\,\mathcal{D}_1, \ldots, \Gamma' \longrightarrow \bar{z} \rhd B\,p\,\bar{u}}}{\Sigma\,;\,\Delta_1, \ldots, \Delta_m, \Gamma' \longrightarrow \bar{z} \rhd B\,p\,\bar{u}}\;mc$$

then the derivation

$$\dfrac{\dfrac{\Xi_1}{\Sigma\,;\,\Delta_1, \ldots, \Delta_m, \Gamma' \longrightarrow \bar{z} \rhd B\,p\,\bar{u}}}{\Sigma\,;\,\Delta_1, \ldots, \Delta_m, \Gamma' \longrightarrow \bar{z} \rhd p\,\bar{u}}\;\mu\mathcal{R}$$

is a reduct of Π (by the reduction rule $-/\mu\mathcal{R}$), and therefore by the definition of reducibility both this reduct and Ξ_1 are reducible predecessors of Π. Let Ψ be the derivation

$$\dfrac{\begin{array}{c}\mu(\Pi'',\Pi_S)\\ \Sigma\,;\mathcal{D}_1,\dots,\Gamma' \longrightarrow \bar{z} \rhd B\,S\,\bar{u}\end{array} \quad \begin{array}{c}\Pi'_S\\ \Sigma\,;\bar{z} \rhd B\,S\,\bar{u} \longrightarrow \bar{z} \rhd S\,\bar{u}\end{array}}{\Sigma\,;\mathcal{D}_1,\dots,\Gamma' \longrightarrow \bar{z} \rhd S\,\bar{u}}\,mc$$

Then the derivation $\mu(\Pi,\Pi_S)$ is the following

$$\dfrac{\left\{\begin{array}{c}\Pi_j\\ \Sigma\,;\Delta_j \longrightarrow \mathcal{D}_j\end{array}\right\}_{j\in\{1,\dots,m\}} \quad \begin{array}{c}\Psi\\ \Sigma\,;\mathcal{D}_1,\dots,\Gamma' \longrightarrow \bar{z} \rhd S\,\bar{u}\end{array}}{\Sigma\,;\Delta_1,\dots,\Delta_m,\Gamma' \longrightarrow \bar{z} \rhd S\,\bar{u}}\,mc \quad .$$

The only applicable reduction rule to $\mu(\Pi,\Pi_S)$ is $-/mc$, which gives us the reduct Ξ

$$\dfrac{\begin{array}{c}\Psi'\\ \Sigma\,;\Delta_1,\dots,\Delta_m,\Gamma' \longrightarrow \bar{z} \rhd B\,S\,\bar{u}\end{array} \quad \begin{array}{c}\Pi'_S\\ \Sigma\,;\bar{z} \rhd B\,S\,\bar{u} \longrightarrow \bar{z} \rhd S\,\bar{u}\end{array}}{\Sigma\,;\Delta_1,\dots,\Delta_m,\Gamma' \longrightarrow \bar{z} \rhd S\,\bar{u}}\,mc \quad ,$$

where Ψ' is the derivation

$$\dfrac{\left\{\begin{array}{c}\Pi_j\\ \Sigma\,;\Delta_j \longrightarrow \mathcal{D}_j\end{array}\right\}_{j\in\{1,\dots,m\}} \quad \begin{array}{c}\mu(\Pi'',\Pi_S)\\ \Sigma\,;\mathcal{D}_1,\dots,\Gamma' \longrightarrow \bar{z} \rhd B\,S\,\bar{u}\end{array}}{\Sigma\,;\Delta_1,\dots,\Delta_m,\Gamma' \longrightarrow \bar{z} \rhd B\,S\,\bar{u}}\,mc$$

Notice that Ψ' is exactly $\mu(\Xi_1,\Pi_S)$, and is reducible by inductive hypothesis. Therefore the assumption on Π_S applies, and the reduct Ξ is reducible.

Otherwise, suppose Π' ends with *init*, then $\mathcal{D}_1 = \mathcal{C}\,p$ and Π is the derivation

$$\dfrac{\begin{array}{c}\Pi_1\\ \Sigma\,;\Delta_1 \longrightarrow \bar{z} \rhd p\,\bar{u}\end{array} \quad \dfrac{}{\Sigma\,;\bar{z} \rhd p\,\bar{u} \longrightarrow \bar{z} \rhd p\,\bar{u}}\,init}{\Sigma\,;\Delta_1 \longrightarrow \bar{z} \rhd p\,\bar{u}}\,mc \quad .$$

The only reduct of Π is Π_1 since the only applicable reduction is $-/init$. On the other hand, the derivation $\mu(\Pi,\Pi_S)$ is

$$\dfrac{\begin{array}{c}\Pi_1\\ \Sigma\,;\Delta_1 \longrightarrow \bar{z} \rhd p\,\bar{u}\end{array} \quad \dfrac{\begin{array}{c}\Pi_S\\ \bar{x}\,;B\,S\,\bar{x} \longrightarrow S\,\bar{x}\end{array} \quad \begin{array}{c}Id\\ \Sigma\,;\bar{z} \rhd S\,\bar{u} \longrightarrow \bar{z} \rhd S\,\bar{u}\end{array}}{\Sigma\,;\bar{z} \rhd p\,\bar{u} \longrightarrow \bar{z} \rhd S\,\bar{u}}\,\mu\mathcal{L}}{\Sigma\,;\Delta_1 \longrightarrow \bar{z} \rhd S\,\bar{u}}\,mc$$

Its only reduct is (by $*/\mu\mathcal{L}$)

$$\frac{\overset{\mu(\Pi_1, \Pi_S)}{\Sigma\,;\,\Delta_1 \longrightarrow \bar{z} \rhd S\,\bar{u}} \quad \overset{Id}{\Sigma\,;\,\bar{z} \rhd S\,\bar{u} \longrightarrow \bar{z} \rhd S\,\bar{u}}}{\Sigma\,;\,\Delta_1 \longrightarrow \bar{z} \rhd S\,\bar{u}}\ mc \quad .$$

The derivation $\mu(\Pi_1, \Pi_S)$ is reducible by inductive hypothesis (Π_1 is a predecessor of Π) and therefore assumption on Id applies, and the above reduct is reducible.

∎

Lemma 4.19. Co-inductive unfolding. *Let $p\,\bar{x} \overset{\nu}{=} B\,p\,\bar{x}$ be a co-inductive definition. Let Π_S be a reducible derivation of $\bar{x}\,;\,S\,\bar{x} \longrightarrow B\,S\,\bar{x}$ for some invariant S such that $\mathrm{lvl}(S) \leq \mathrm{lvl}(p)$. Let $\mathcal{C}\,p$ be a judgment such that $\mathrm{lvl}(\mathcal{C}\,p) \leq \mathrm{lvl}(p)$, and let Π be a reducible derivation of $\Sigma\,;\,\Gamma \longrightarrow \mathcal{C}\,S$.*

Suppose there is a $(\Re_{<i}, \Phi, \Pi_S)$-generated set $\mathcal{W}$, where $i = \mathrm{lvl}(p)$ and Φ is some reducible derivation, such that all derivations in $\mathcal{W}$ are reducible and $\Pi \in \mathcal{W}$. Then the derivation $\nu(\Pi, \Pi_S)$ of $\Sigma\,;\,\Gamma \longrightarrow \mathcal{C}\,p$ is reducible.

Proof By induction on the reduction of Π and case analysis on $\mathcal{C}$. If $\mathcal{C}\,S = \mathcal{C}\,p$ then $\nu(\Pi, \Pi_S) = \Pi$, which is reducible by assumption. Otherwise, suppose $\mathcal{C}\,S \neq \mathcal{C}\,p$. We have two cases to consider.

Suppose $\mathcal{C}\,p = \bar{z} \rhd p\,\bar{u}$. Then $\mathcal{C}\,S = \bar{z} \rhd S\,\bar{u}$ and $\nu(\Pi, \Pi_S)$ is the derivation

$$\frac{\overset{\Pi}{\Sigma\,;\,\Gamma \longrightarrow \bar{z} \rhd S\,\bar{u}} \quad \overset{\Pi_S}{\bar{x}\,;\,S\,\bar{x} \longrightarrow B\,S\,\bar{x}}}{\Sigma\,;\,\Gamma \longrightarrow \bar{z} \rhd p\,\bar{u}}\ \nu\mathcal{R} \quad .$$

Let $\mathcal{G}$ be the $(\Re_{<i}, \Pi, \Pi_S)$-generated set. By the definition of reducibility, the above derivation is reducible if its premises are reducible and the derivations in $\mathcal{G}$ are reducible. The reducibility of the premise derivations is immediate from the assumptions. Since $\Pi \in \mathcal{W}$, by Lemma 4.13 $\mathcal{G} \subseteq \mathcal{W}$, and therefore the derivations in $\mathcal{G}$ are reducible.

Otherwise $\mathcal{C}\,p$ is non-atomic judgment and there are several subcases, depending on the last rule in Π.

1. Suppose Π ends with *mc*.

$$\frac{\overset{\Pi_1}{\Delta_1 \longrightarrow \mathcal{D}_1} \quad \cdots \quad \overset{\Pi_n}{\Delta_n \longrightarrow \mathcal{D}_m} \quad \overset{\Pi'}{\mathcal{D}_1, \ldots, \mathcal{D}_m, \Gamma' \longrightarrow \mathcal{C}\,S}}{\Delta_1, \ldots, \Delta_n, \Gamma' \longrightarrow \mathcal{C}\,S}\ mc$$

Then $\nu(\Pi, \Pi_S)$ is the derivation

$$\frac{\overset{\Pi_1}{\Delta_1 \longrightarrow \mathcal{D}_1} \quad \cdots \quad \overset{\Pi_n}{\Delta_n \longrightarrow \mathcal{D}_m} \quad \overset{\nu(\Pi', \Pi_S)}{\mathcal{D}_1, \ldots, \mathcal{D}_m, \Gamma' \longrightarrow \mathcal{C}\,p}}{\Delta_1, \ldots, \Delta_n, \Gamma' \longrightarrow \mathcal{C}\,p}\ mc \quad .$$

The derivation $\nu(\Pi, \Pi_S)$ is reducible if every reduct of $\nu(\Pi, \Pi_S)$ is reducible. From Lemma 4.5, it follows that every reduct of $\nu(\Pi, \Pi_S)$ is of the form $\nu(\Xi, \Pi_S)$ where Ξ is a reduct of Π. Since all reducts of Π are in $\mathcal{W}$ by Definition 4.10, we can apply the inductive hypothesis to show that $\nu(\Xi, \Pi_S)$ is reducible, and hence $\nu(\Pi, \Pi_S)$ is also reducible.

2. Suppose Π ends with $\supset \mathcal{R}$.

$$\dfrac{\begin{matrix}\Pi' \\ \Gamma, \mathcal{C}_1\, p \longrightarrow \mathcal{C}_2\, S\end{matrix}}{\Gamma \longrightarrow \mathcal{C}_1\, p \supset \mathcal{C}_2\, S} \supset \mathcal{R}$$

Note that p is vacuous in $\mathcal{C}\, p$, therefore $\mathcal{C}_1\, S = \mathcal{C}_1\, p$. The reducibility of Π implies that Π' is reducible and for any reducible derivation Ψ (which is in $\Re_{<i}$ because $\mathrm{lvl}(\mathcal{C}_1\, p) < \mathrm{lvl}(p) = i$) and for any substitution θ and signature Σ', the derivation Ξ

$$\dfrac{\begin{matrix}\Psi \\ \Delta' \longrightarrow (\mathcal{C}_1\, p)\theta\end{matrix} \qquad \begin{matrix}\Sigma' : \Pi'\theta \\ \Gamma, (\mathcal{C}_1\, p)\theta \longrightarrow (\mathcal{C}_2\, S)\theta\end{matrix}}{\Delta', \Gamma\theta \longrightarrow (\mathcal{C}_2\, S)\theta} \, mc$$

is reducible. By Definition 4.10, $\Pi', \Xi \in \mathcal{W}$. The derivation $\nu(\Pi, \Pi_S)$ is

$$\dfrac{\begin{matrix}\nu(\Pi', \Pi_S) \\ \Gamma, \mathcal{C}_1\, p \longrightarrow \mathcal{C}_2\, p\end{matrix}}{\Gamma \longrightarrow \mathcal{C}_1\, p \supset \mathcal{C}_2\, p} \supset \mathcal{R} \quad .$$

To show that $\nu(\Pi, \Pi_S)$ is reducible, we need to show that $\nu(\Pi', \Pi_S)$ is reducible, and for every Δ',θ,Σ' and reducible derivation Ψ of $\Delta' \longrightarrow (\mathcal{C}_1\, p)\theta$, the derivation Ξ'

$$\dfrac{\begin{matrix}\Psi \\ \Delta' \longrightarrow (\mathcal{C}_1\, p)\theta\end{matrix} \qquad \begin{matrix}\Sigma' : \nu(\Pi', \Pi_S)\theta \\ \Gamma, (\mathcal{C}_1\, p)\theta \longrightarrow (\mathcal{C}_2\, p)\theta\end{matrix}}{\Delta', \Gamma\theta \longrightarrow (\mathcal{C}_2\, p)\theta} \, mc$$

is reducible. From Lemma 3.24 and Lemma 3.22 we see that $\Sigma' : \nu(\Pi', \Pi_S)\theta = \nu(\Sigma' : \Pi'\theta, \Pi_S)$. So Ξ' is exactly $\nu(\Xi, \Pi_S)$. Therefore we apply induction hypothesis to Π' and to Ξ to get the reducible derivations $\nu(\Pi', \Pi_S)$ and Ξ, respectively.

3. Suppose Π ends with $\wedge\mathcal{R}$.

$$\dfrac{\begin{matrix}\Pi_1 \\ \Gamma \longrightarrow \mathcal{C}_1\, S\end{matrix} \qquad \begin{matrix}\Pi_2 \\ \Gamma \longrightarrow \mathcal{C}_2\, S\end{matrix}}{\Gamma \longrightarrow \mathcal{C}_1\, S \wedge \mathcal{C}_2\, S} \wedge\mathcal{R}$$

Since Π is reducible, both Π_1 and Π_2 are reducible predecessor of Π, and both are also in the set $\mathcal{W}$ by the definition of generated set. Thus it follows from inductive hypothesis that the derivation $\nu(\Pi,\Pi_S)$

$$\frac{\begin{array}{c}\nu(\Pi_1,\Pi_S)\\ \Gamma\longrightarrow\mathcal{C}_1\,p\end{array}\qquad\begin{array}{c}\nu(\Pi_2,\Pi_S)\\ \Gamma\longrightarrow\mathcal{C}_2\,p\end{array}}{\Gamma\longrightarrow\mathcal{C}_1\,p\wedge\mathcal{C}_2\,p}\;\wedge\mathcal{R}$$

is reducible. The cases where Π ends with $\vee\mathcal{R}$, $\exists\mathcal{R}$, $\forall\mathcal{R}$ or $\nabla\mathcal{R}$ (and the corresponding left-rules) are treated similarly since in these cases all premise derivations are in the reduction of Π.

4. Suppose Π ends with $\supset\mathcal{L}$.

$$\frac{\begin{array}{c}\Pi_1\\ \Gamma'\longrightarrow\mathcal{D}\end{array}\qquad\begin{array}{c}\Pi_2\\ \Gamma',\mathcal{E}\longrightarrow\mathcal{C}\,S\end{array}}{\Gamma',\mathcal{D}\supset\mathcal{E}\longrightarrow\mathcal{C}\,S}\;\supset\mathcal{L}\quad.$$

By the definition of reducibility, the derivation Π_2 is reducible and is the major premise of Π, and hence in $\mathcal{W}$. The derivation Π_1 is normalizable. By inductive hypothesis, $\nu(\Pi_2,\Pi_S)$ is reducible. Therefore the derivation $\nu(\Pi,\Pi_S)$

$$\frac{\begin{array}{c}\Pi_1\\ \Gamma'\longrightarrow\mathcal{D}\end{array}\qquad\begin{array}{c}\nu(\Pi_2,\Pi_S)\\ \Gamma',\mathcal{E}\longrightarrow\mathcal{C}\,p\end{array}}{\Gamma',\mathcal{D}\supset\mathcal{E}\longrightarrow\mathcal{C}\,p}\;\supset\mathcal{L}$$

is reducible. The case where Π ends with $\mu\mathcal{L}$ is treated similarly since in this case unfolding is also done only in the major premise.

5. Suppose Π ends with a left rule $\circ\mathcal{L}$ other than $\supset\mathcal{L}$ and $\mu\mathcal{L}$. In this case all premise derivations are major premises and inductive hypothesis can be applied to show the reducibility of $\nu(\Pi,\Pi_S)$.

∎

4.3 Cut elimination

LEMMA 4.20. *For any derivation* Π *of* $\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma\longrightarrow\mathcal{C}$, *reducible derivations*

$$\begin{array}{c}\Pi_1\\ \Delta_1\longrightarrow\mathcal{B}_1,\end{array}\quad\ldots\quad,\begin{array}{c}\Pi_n\\ \Delta_n\longrightarrow\mathcal{B}_n\end{array}$$

where $n\geq 0$, *for any substitutions* $\delta_1,\ldots,\delta_n,\gamma$ *and signatures* $\Sigma_1,\ldots,\Sigma_n,\Sigma$, *such that* $\mathcal{B}_i\delta_i=\mathcal{B}_i\gamma$, *for every* $i\in\{1,\ldots,n\}$, *the derivation* Ξ

$$\frac{\begin{array}{c}\Sigma_1:\Pi_1\delta_1\\ \Delta_1\delta_1\longrightarrow\mathcal{B}_1\delta_1\end{array}\quad\cdots\quad\begin{array}{c}\Sigma_n:\Pi_n\delta_n\\ \Delta_n\delta_n\longrightarrow\mathcal{B}_n\delta_n\end{array}\quad\begin{array}{c}\Sigma:\Pi\gamma\\ \mathcal{B}_1\gamma,\ldots,\mathcal{B}_n\gamma,\Gamma\gamma\longrightarrow\mathcal{C}\gamma\end{array}}{\Delta_1\delta_1,\ldots,\Delta_n\delta_n,\Gamma\gamma\longrightarrow\mathcal{C}\gamma}\;mc$$

is reducible.

Proof The proof is by induction on ind(Π) with subordinate induction on ht(Π), on n and on the reductions of $\Pi_1, \ldots, \Pi_n$. The proof does not rely on the order of the inductions on reductions. Thus when we need to distinguish one of the Π_i, we shall refer to it as Π_1 without loss of generality. The derivation Ξ is reducible if all its reducts are reducible.

If $n = 0$, then Ξ reduces to $\Sigma : \Pi\gamma$, thus in this case we show that $\Sigma : \Pi\gamma$ is reducible. Since reducibility is preserved by substitution and signature weakening (Lemma 4.16 and Lemma 4.17) it is enough to show that Π is reducible. This is proved by a case analysis of the last rule in Π. For each case, the result follows easily from the outer induction hypothesis and Definition 4.14. The $\supset \mathcal{R}$ case requires that substitution for variables and weakening of global signatures do not increase the measures of a derivation (Lemma 3.6 and Lemma 3.11). In the cases for $\supset \mathcal{L}$ and $\mu\mathcal{L}$ we need the additional information that reducibility implies normalizability (Lemma 4.15). The case for $\nu\mathcal{R}$ requires special attention. Let $p\,\bar{x} \overset{\nu}{=} D\,p\,\bar{x}$ be a co-inductive definition. Suppose Π is the derivation

$$\frac{\begin{array}{c}\Pi' \\ \Gamma \longrightarrow \bar{z} \triangleright S\,\bar{t}\end{array} \quad \begin{array}{c}\Pi_S \\ S\,\bar{x} \longrightarrow D\,S\,\bar{x}\end{array}}{\Gamma \longrightarrow \bar{z} \triangleright p\,\bar{t}}\ \nu\mathcal{R}$$

for some invariant S. Let $l = \mathrm{lvl}(p)$. To show that Π is reducible we must show that its premises are reducible and the $(\Re_{<l}, \Pi', \Pi_S)$-generated set contains only reducible derivations. Reducibility of premises of Π is immediate from inductive hypotheses. Since $\mathrm{ind}(\Pi_S) \leq \mathrm{ind}(\Pi)$ and $\mathrm{ht}(\Pi_S) < \mathrm{ht}(\Pi)$ and substitution and weakening of signatures does not increase these measures, by inductive hypothesis, for every reducible derivation Ψ of $\Delta \longrightarrow \bar{y} \triangleright S\,\bar{u}$ for some Δ, $\bar{y}$ and $\bar{u}$, the derivation

$$\frac{\begin{array}{c}\Psi \\ \Delta \longrightarrow \bar{y} \triangleright S\,\bar{u}\end{array} \quad \begin{array}{c}\Pi'_S \\ \bar{y} \triangleright S\,\bar{u} \longrightarrow \bar{z} \triangleright B\,S\,\bar{u}\end{array}}{\Delta \longrightarrow \bar{y} \triangleright B\,S\,\bar{u}}\ mc \qquad (*)$$

is reducible. Recall that Π'_S is obtained from $\Pi_S^{\uparrow\hat{\tau}(\bar{y})}$ (see Lemma 3.16) by some substitution and hence $\mathrm{ht}(\Pi'_S) \leq \mathrm{ht}(\Pi_S)$ and $\mathrm{ind}(\Pi'_S) \leq \mathrm{ind}(\Pi_S)$. We will make use of this fact to establish the reducibility of all derivations in the $(\Re_{<l}\Pi', \Pi_S)$-generated set, which we shall refer to here as $\mathcal{G}$. This is done by induction of the construction of the generated set (see Definition 4.10). For each natural number j, the state of the construction is denoted by $\mathcal{W}^j$, and $\mathcal{G}$ is the ordinal limit, i.e., $\mathcal{W}^\omega$. The base case follows immediately from outer inductive hypothesis. For the inductive case, assume that all derivations in $\mathcal{W}^j$ are reducible. To show that all derivations in $\mathcal{W}^{j+1}$ are reducible, it is enough to check that each case (1) - (4) in Definition 4.10 preserves reducibility. The cases (1), (2) and (3) follow from outer inductive hypothesis (Π' is reducible), Lemma 4.16, and Lemma 4.17, respectively. For the case (4.a), since all derivations in $\mathcal{W}^j$ are reducible, it follows from the fact (*) that the resulting derivations from applying (4.a) are reducible. The remaining cases (4.b) follows from the definition of major premises (Definition 3.2) and the definition of reducibility.

For $n > 0$, we analyze all possible reductions and show for each case the reduct is reducible. Some cases follow immediately from inductive hypothesis. We show here the non-trivial cases.

$\supset\mathcal{R}/\supset\mathcal{L}$: Suppose Π_1 and Π are

$$\dfrac{\begin{array}{c}\Pi'_1\\ \Delta_1, \mathcal{B}'_1 \longrightarrow \mathcal{B}''_1\end{array}}{\Delta_1 \longrightarrow \mathcal{B}'_1 \supset \mathcal{B}''_1}\supset\mathcal{R} \qquad \dfrac{\begin{array}{cc}\Pi' & \Pi''\\ \mathcal{B}_2, \ldots, \Gamma \longrightarrow \mathcal{B}'_1 & \mathcal{B}''_1, \mathcal{B}_2, \ldots, \Gamma \longrightarrow \mathcal{C}\end{array}}{\mathcal{B}'_1 \supset \mathcal{B}''_1, \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}\supset\mathcal{L} \quad .$$

The derivation Ξ_1

$$\dfrac{\begin{array}{cccc}\Sigma_1 : \Pi_2\delta_2 & & \Sigma_n : \Pi_n\delta_n & \Sigma : \Pi'\gamma\\ \Delta_2\delta_2 \longrightarrow \mathcal{B}_2\delta_2 & \ldots & \Delta_n\delta_n \longrightarrow \mathcal{B}_n\delta_n & \mathcal{B}_2\gamma, \ldots, \mathcal{B}_n\gamma, \Gamma\gamma \longrightarrow \mathcal{B}'_1\gamma\end{array}}{\Delta_2\delta_2, \ldots, \Delta_n\delta_n, \Gamma\gamma \longrightarrow \mathcal{B}'_1\gamma}\,mc$$

is reducible by induction hypothesis since $\mathrm{ind}(\Pi') \leq \mathrm{ind}(\Pi)$ and $\mathrm{ht}(\Pi') < \mathrm{ht}(\Pi)$. Since Π_1 is reducible, by Definition 4.14 the derivation Ξ_2

$$\dfrac{\begin{array}{cc}\Xi_1 & \Sigma' : \Pi_1\delta_1\\ \Delta_2\delta_2, \ldots, \Gamma\gamma \longrightarrow \mathcal{B}'_1\gamma & \mathcal{B}'_1\delta_1, \Delta_1\delta_1 \longrightarrow \mathcal{B}''_1\delta_1\end{array}}{\Delta_1\delta_1, \ldots, \Delta_n\delta_n, \Gamma\gamma \longrightarrow \mathcal{B}''_1\delta_1}\,mc$$

is a predecessor of Π_1 and therefore is reducible. The reduct of Ξ in this case is the following derivation

$$\dfrac{\dfrac{\begin{array}{ccc}\Xi_2 & \left\{\begin{array}{c}\Sigma_i : \Pi_i\delta_i\\ \Delta_i\delta_i \longrightarrow \mathcal{B}_i\delta_i\end{array}\right\}_{i\in\{2..n\}} & \begin{array}{c}\Sigma' : \Pi''\gamma\\ \mathcal{B}''_1\gamma, \ldots, \mathcal{B}_n\gamma, \Gamma\gamma \longrightarrow \mathcal{C}\gamma\end{array}\\ \ldots \longrightarrow \mathcal{B}''_1\delta_1 & & \end{array}}{\Delta_1\delta_1, \ldots, \Delta_n\delta_n, \Gamma\gamma, \Delta_2\delta_2, \ldots, \Delta_n\gamma, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,mc}{\Delta_1\delta_1, \ldots, \Delta_n\delta_n, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,c\mathcal{L}$$

which is reducible by induction hypothesis and Definition 4.14 (reducibility).
$\forall\mathcal{L}/\forall\mathcal{R}$: Suppose Π_1 and Π are

$$\dfrac{\begin{array}{c}\Pi'_1\\ \Delta_1 \longrightarrow \bar{z} \rhd B'_1[(h\ \bar{z})/x]\end{array}}{\Delta_1 \longrightarrow \bar{z} \rhd \forall x.B'_1}\forall\mathcal{R} \qquad \dfrac{\begin{array}{c}\Pi'\\ \bar{z} \rhd B'_1[t/x], \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}\end{array}}{\bar{z} \rhd \forall x.B'_1, \mathcal{B}_2, \ldots, \mathcal{B}_n, \Gamma \longrightarrow \mathcal{C}}\forall\mathcal{L}$$

Since we identify derivations that differ only in the choice of intermediate eigenvariables that are not free in the end sequents, we can choose a variable h such that it is not free in the domains and ranges of δ_1 and γ and different from Σ_1 and Σ. We can safely assume that the bound variables $\bar{z}$ and x are different from free variables in δ_1,γ, Σ_1 and Σ.

This way we can push the substitution inside binders. The derivation Ξ is thus

$$\cfrac{\cfrac{\begin{array}{c}\Sigma_1 : \Pi_1'\delta_1\\ \Delta_1\delta_1 \longrightarrow \bar{z} \rhd B_1'\delta_1[(h\ \bar{z})/x]\end{array}}{\Delta_1\delta_1 \longrightarrow \bar{z} \rhd \forall x.B_1'\delta_1}\,\forall\mathcal{R} \quad \dots \quad \cfrac{\begin{array}{c}\Sigma' : \Pi'\gamma\\ \bar{z} \rhd B_1'\gamma[t\gamma/x],\dots,\Gamma\gamma \longrightarrow \mathcal{C}\gamma\end{array}}{\bar{z} \rhd \forall x.B_1'\gamma,\dots,\Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,\forall\mathcal{L}}{\Delta_1\delta_1,\dots,\Delta_n\delta_n,\Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,mc$$

Let $\delta_1' = \delta_1 \circ [\lambda\bar{z}.t\gamma/h]$. The reduct of Ξ in this case is

$$\cfrac{\begin{array}{c}\Sigma' : \Pi_1'\delta_1'\\ \Delta_1\delta_1 \longrightarrow \bar{z} \rhd B_1'\delta_1[t\gamma/x]\end{array} \quad \dots \quad \begin{array}{c}\Pi'\gamma\\ \bar{z} \rhd B_1'\gamma[t\gamma/x],\dots,\Gamma\gamma \longrightarrow \mathcal{C}\gamma\end{array}}{\Delta_1\delta_1,\dots,\Delta_n\delta_n,\Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,mc$$

which is reducible by induction hypothesis.
eq$\mathcal{R}$/eq$\mathcal{L}$: Suppose Π_1 and Π are

$$\cfrac{}{\Delta_1 \longrightarrow \bar{z} \rhd s = t}\,\mathrm{eq}\mathcal{R} \qquad\qquad \cfrac{\left\{\begin{array}{c}\Pi^\rho\\ \mathcal{B}_2\rho,\dots,\mathcal{B}_n\rho,\Gamma\rho \longrightarrow \mathcal{C}\rho\end{array}\right\}_\rho}{\bar{z} \rhd s = t,\dots,\mathcal{B}_n,\Gamma \longrightarrow \mathcal{C}}\,\mathrm{eq}\mathcal{L}$$

Then Ξ is the derivation

$$\cfrac{\cfrac{}{\Delta_1\delta_1 \longrightarrow (\bar{z} \rhd s = t)\delta_1}\,\mathrm{eq}\mathcal{R} \quad \cdots \quad \cfrac{\left\{\begin{array}{c}\Sigma' : \Pi^{\gamma\circ\rho'}\\ \mathcal{B}_2\gamma\rho',\dots,\mathcal{B}_n\gamma\rho',\Gamma\rho' \longrightarrow \mathcal{C}\gamma\rho'\end{array}\right\}_{\rho'}}{(\bar{z} \rhd s = t)\gamma,\dots,\mathcal{B}_n\gamma,\Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,\mathrm{eq}\mathcal{L}}{\Delta_1\delta_1,\dots,\Delta_n\delta_n,\Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,mc$$

The eq$\mathcal{R}$ tells us that s and t are unifiable via empty substitution (i.e., they are the same normal terms). The reduct of Ξ

$$\cfrac{\begin{array}{c}\Pi_2\delta_2\\ \Delta_2\delta_2 \longrightarrow \mathcal{B}_2\delta_2\end{array} \quad \dots \quad \begin{array}{c}\Sigma' : \Pi^\gamma\\ \mathcal{B}_2\gamma,\dots,\Gamma\gamma \longrightarrow \mathcal{C}\gamma\end{array}}{\Delta_1\delta_1,\dots,\Delta_n\delta_n,\Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,mc$$

is therefore reducible by induction hypothesis.
$*/\mu\mathcal{L}$: Suppose Π is the derivation

$$\cfrac{\begin{array}{c}\Pi_S\\ D\,S\,\bar{x} \longrightarrow S\,\bar{x}\end{array} \quad \begin{array}{c}\Pi'\\ \bar{z} \rhd S\,\bar{t},\Gamma \longrightarrow \mathcal{C}\end{array}}{\bar{z} \rhd p\,\bar{t},\Gamma \longrightarrow \mathcal{C}}\,\mu\mathcal{L}$$

Let $\bar{z} \triangleright p\,\bar{u}$ be the result of applying δ_1 to $\bar{z} \triangleright p\,\bar{t}$. Then Ξ is the derivation

$$\dfrac{\begin{array}{c}\Sigma_1 \!:\! \Pi_1\delta_1\\ \Delta_1\delta_1 \longrightarrow \bar{z} \triangleright p\,\bar{u}\end{array} \quad \cdots \quad \begin{array}{c}\Sigma_n \!:\! \Pi_n\delta_n\\ \Delta_n\delta_n \longrightarrow \mathcal{B}_n\delta_n\end{array} \quad \dfrac{\begin{array}{c}\Pi_S\\ D\,S\,\bar{x} \longrightarrow S\,\bar{x}\end{array} \quad \begin{array}{c}\Sigma \!:\! \Pi'\gamma\\ \bar{z} \triangleright S\,\bar{u}, \Gamma\gamma \longrightarrow \mathcal{C}\gamma\end{array}}{\bar{z} \triangleright p\,\bar{u}, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,\mu\mathcal{L}}{\Delta_1\delta_1, \ldots, \Delta_n\delta_n, \Gamma\gamma \longrightarrow C\gamma}\,mc$$

The derivation Ξ reduces to the derivation Ξ'

$$\dfrac{\begin{array}{c}\Sigma_1 \!:\! \mu(\Pi_1, \Pi_S)\delta_1\\ \Delta_1\delta_1 \longrightarrow \bar{z} \triangleright S\,\bar{u}\end{array} \quad \cdots \quad \begin{array}{c}\Sigma_n \!:\! \Pi_n\delta_n\\ \Delta_n\delta_n \longrightarrow \mathcal{B}_n\delta_n\end{array} \quad \begin{array}{c}\Pi'\gamma\\ \bar{z} \triangleright S\,\bar{u}, \Gamma\gamma \longrightarrow \mathcal{C}\gamma\end{array}}{\Delta_1\delta_1, \ldots, \Delta_n\delta_n, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,mc \quad .$$

Notice that we have used the fact that

$$\mu(\Sigma_1 \!:\! \Pi_1\delta_1, \Pi_S) = \Sigma_1 \!:\! \mu(\Pi_1\delta_1, \Pi_S) = \Sigma_1 \!:\! \mu(\Pi_1, \Pi_S)\delta_1$$

in the derivation above, which follows from Lemma 3.21 and Lemma 3.23. Therefore, in order to prove that Ξ' is reducible, it remains to show that the unfolding of Π_1 produces a reducible derivation.

We observe the following facts. Let Π'_S be a raised instance of Π_S, i.e., Π'_S is a derivation of some sequent $\bar{y} \triangleright D\,S\,\bar{w} \longrightarrow \bar{y} \triangleright S\,\bar{w}$. By Lemma 3.18 we have $\mathrm{ht}(\Pi'_S) \leq \mathrm{ht}(\Pi_S) < \mathrm{ht}(\Pi)$ and $\mathrm{ind}(\Pi'_S) \leq \mathrm{ind}(\Pi_S) < \mathrm{ind}(\Pi)$. Therefore, by the outer induction hypothesis, the result of cutting Π'_S with any reducible derivation must produce another reducible derivation. Precisely, given any reducible derivation Ψ of $\Delta \longrightarrow y \triangleright D\,S\,\bar{w}$, the derivation

$$\dfrac{\begin{array}{c}\Psi\\ \Delta \longrightarrow \bar{y} \triangleright D\,S\,\bar{w}\end{array} \quad \begin{array}{c}\Pi'_S\\ \bar{y} \triangleright D\,S\,\bar{w} \longrightarrow \bar{y} \triangleright S\,\bar{w}\end{array}}{\Delta \longrightarrow \bar{y} \triangleright S\,\bar{w}}\,mc$$

is reducible. Additionally, from Definition 3.8, we clearly have $\mathrm{ind}(Id_{\mathcal{D}}) = 0 < \mathrm{ind}(\Pi)$ for any judgment $\mathcal{D}$, and therefore at this stage, we have also established that for every reducible derivation Ψ' of $\Delta' \longrightarrow \mathcal{D}'$, the derivation

$$\dfrac{\begin{array}{c}\Psi'\\ \Delta' \longrightarrow \mathcal{D}'\end{array} \quad \begin{array}{c}Id_{\mathcal{D}'}\\ \mathcal{D}' \longrightarrow \mathcal{D}'\end{array}}{\Delta' \longrightarrow \mathcal{D}'}\,mc$$

is reducible.

These facts give sufficient conditions for concluding the reducibility of $\mu(\Pi_1, \Pi_S)$ by applying Lemma 4.18. Therefore, applying the induction hypothesis to Π', we establish the reducibility of Ξ'.

$\nu\mathcal{R}/\nu\mathcal{L}$: Suppose Π_1 and Π are

$$\dfrac{\overset{\Pi'_1}{\Delta_1 \longrightarrow \bar{y} \rhd S\,\bar{t}} \quad \overset{\Pi_S}{S\,\bar{x} \longrightarrow D\,S\,\bar{x}}}{\Delta_1 \longrightarrow \bar{y} \rhd p\,\bar{t}}\,\nu\mathcal{R} \qquad\qquad \dfrac{\overset{\Pi'}{\bar{y} \rhd D\,p\,\bar{t}, \mathcal{B}_2, \ldots, \Gamma \longrightarrow \mathcal{C}}}{\bar{y} \rhd p\,\bar{t}, \mathcal{B}_2, \ldots, \Gamma \longrightarrow \mathcal{C}}\,\nu\mathcal{L}$$

where $p\,\bar{x} \overset{\nu}{=} D\,p\,\bar{x}$. Suppose $(\bar{y} \rhd p\,\bar{t})\delta_1 = (\bar{y} \rhd p\,\bar{t})\gamma = \bar{z} \rhd p\,\bar{u}$. Then Ξ is the derivation

$$\dfrac{\dfrac{\overset{\Sigma_1 : \Pi'_1\delta_1}{\Delta_1\delta_1 \longrightarrow \bar{z} \rhd S\,\bar{u}} \quad \overset{\Pi_S}{S\,\bar{x} \longrightarrow D\,S\,\bar{x}}}{\Delta_1\delta_1 \longrightarrow \bar{z} \rhd p\,\bar{u}}\,\nu\mathcal{R} \quad \cdots \quad \dfrac{\overset{\Sigma : \Pi'\gamma}{\bar{z} \rhd D\,p\,\bar{u}, \ldots, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}}{\bar{z} \rhd p\,\bar{u}, \ldots, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,\nu\mathcal{L}}{\Delta_1\delta_1, \ldots, \Delta_n\delta_n, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,mc$$

Let $l = \mathrm{lvl}(p)$ and let $\mathcal{W}$ be the $(\Re_{<l}, \Pi'_1, \Pi_S)$-generated set. Since Π_1 is reducible, by Definition 4.14, the derivations in $\mathcal{W}$ are predecessors of Π_1 and are reducible. By the definition of generated set (Definition 4.10), $\mathcal{W}$ contains the derivation $\Sigma_1 : \Pi'_1\delta_1 \in \mathcal{W}$ and the derivation Ξ_1

$$\dfrac{\overset{\Sigma_1 : \Pi'_1\delta_1}{\Delta_1\delta_1 \longrightarrow \bar{z} \rhd S\,\bar{u}} \quad \overset{\Pi'_S}{\bar{z} \rhd S\,\bar{u} \longrightarrow \bar{z} \rhd D\,S\,\bar{u}}}{\Delta_1\delta_1 \longrightarrow \bar{z} \rhd D\,S\,\bar{u}}\,mc \quad .$$

where Π'_S is a raised instance of Π_S. Therefore by Lemma 4.19 the derivation $\nu(\Xi_1, \Pi_S)$ is reducible. The reduct of Ξ is the derivation

$$\dfrac{\overset{\nu(\Xi_1, \Pi_S)}{\Delta_1\delta_1 \longrightarrow \bar{z} \rhd D\,p\,\bar{u}} \quad \cdots \quad \overset{\Sigma_n : \Pi_n\delta_n}{\Delta_n\delta_n \longrightarrow \mathcal{B}_n\delta_n} \quad \overset{\Sigma : \Pi'\gamma}{\bar{z} \rhd D\,p\,\bar{u}, \ldots, \mathcal{B}_n\gamma, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}}{\Delta_1\delta_1, \ldots, \Delta_n\delta_n, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,mc.$$

Its reducibility follows from the reducibility of $\nu(\Xi_1, \Pi_S)$ and outer induction hypothesis.
$\supset \mathcal{L}/\circ\mathcal{L}$: Suppose Π_1 is

$$\dfrac{\overset{\Pi'_1}{\Delta'_1 \longrightarrow \mathcal{D}'_1} \quad \overset{\Pi''_1}{\mathcal{D}''_1, \Delta'_1 \longrightarrow \mathcal{B}_1}}{\mathcal{D}'_1 \supset \mathcal{D}''_1, \Delta'_1 \longrightarrow \mathcal{B}_1}\supset\mathcal{L}$$

Since Π_1 is reducible, it follows from Definition 4.14 that Π'_1 is normalizable and Π''_1 is reducible. Let Ξ_1 be the derivation

$$\dfrac{\overset{\Sigma_1 : \Pi''_1\delta_1}{\mathcal{D}''_1\delta_1, \Delta'_1\delta_1 \longrightarrow \mathcal{B}_1\delta_1} \quad \overset{\Sigma_2 : \Pi_2\delta_2}{\Delta_2\delta_2 \longrightarrow \mathcal{B}_2\delta_2} \quad \cdots \quad \overset{\Sigma : \Pi\gamma}{\mathcal{B}_1\delta_1, \ldots, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}}{\mathcal{D}''_1\delta_1, \Delta'_1\delta_1, \Delta_2\delta_2, \ldots, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}\,mc \quad .$$

Ξ_1 is reducible by induction hypothesis on the reduction of Π_1 (Π_1'' is a predecessor of Π_1). The reduct of Ξ in this case is the derivation

$$\dfrac{\dfrac{\dfrac{\Sigma_1 : \Pi_1'\delta_1}{\Delta_1'\delta_1 \longrightarrow \mathcal{D}_1'\delta_1}}{\Delta_1'\delta_1, \Delta_2\delta_2, \ldots, \Gamma\gamma \longrightarrow \mathcal{D}_1'\delta_1}\, w\mathcal{L} \qquad \dfrac{\Xi_1}{\mathcal{D}_1''\delta_1, \Delta_1'\delta_1, \Delta_2\delta_2, \ldots, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}}{(\mathcal{D}_1' \supset \mathcal{D}_1'')\delta_1, \Delta_1'\delta_1, \Delta_2\delta_2, \ldots, \Gamma\gamma \longrightarrow \mathcal{C}\gamma} \supset \mathcal{L} \quad .$$

Since Π_1' is normalizable and substitution and signature weakening preserves normalizability, by Definition 4.6 the left premise of the reduct is normalizable, and hence the reduct is reducible.

eq$\mathcal{L}$/ $\circ$ $\mathcal{L}$: Suppose Π_1 is

$$\dfrac{\left\{ \dfrac{\Pi^{\rho}}{\Delta_1'\rho \longrightarrow \mathcal{B}_1\rho} \right\}_{\rho}}{\bar{z} \rhd s = t, \Delta_1' \longrightarrow \mathcal{B}_1}\, \mathrm{eq}\mathcal{L}$$

Then Ξ is the derivation

$$\dfrac{\dfrac{\left\{ \dfrac{\Sigma_1 : \Pi^{\delta_1 \circ \rho'}}{\Delta_1'\delta_1\rho' \longrightarrow \mathcal{B}_1\delta_1\rho'} \right\}_{\rho'}}{\Delta_1'\delta_1 \longrightarrow \mathcal{B}_1\delta_1}\, \mathrm{eq}\mathcal{L} \qquad \dfrac{\Sigma_2 : \Pi_2\delta_2}{\Delta_2\delta_2 \longrightarrow \mathcal{B}_2\delta_2} \quad \cdots \quad \dfrac{\Sigma : \Pi\gamma}{\mathcal{B}_1\gamma, \ldots, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}}{(\bar{z} \rhd s = t)\delta_1, \Delta_1'\delta_1, \Delta_2\delta_2, \ldots, \Gamma\gamma \longrightarrow \mathcal{C}\gamma}\, mc$$

Notice that each premise derivation $\Pi^{\delta_1 \circ \rho'}$ of $\Pi_1\delta_1$ is a also a premise derivation of Π_1, and therefore it is a predecessor of Π_1. Let $\Xi^{\rho'}$ be the derivation

$$\dfrac{\dfrac{\Sigma_1 : \Pi_1^{\delta_1 \circ \rho'}}{\Delta_1'\delta_1\rho' \longrightarrow \mathcal{B}_1\delta_1\rho'} \quad \dfrac{(\Sigma_2 : \Pi_2\delta_2)\rho'}{\Delta_2\delta_2\rho' \longrightarrow \mathcal{B}_2\delta_2\rho'} \quad \cdots \quad \dfrac{(\Sigma : \Pi\gamma)\rho'}{\mathcal{B}_1\gamma\rho', \ldots, \Gamma\gamma\rho' \longrightarrow \mathcal{C}\gamma\rho'}}{\Delta_1'\delta_1\rho', \Delta_2\delta_2\rho', \ldots, \Gamma\gamma\rho' \longrightarrow \mathcal{C}\gamma\rho'}\, mc.$$

Observe that in the derivation $\Sigma_2 : \Pi_2\delta_2$, by the definition of weakening of signature, the variables in Σ_2 are not free in the end sequent of $\Pi_2\theta_2$. Now since substitution affects only the free variables in the end sequent, we can equally write the derivation $(\Sigma_2 : \Pi_2\delta_2)\rho'$ as $\Sigma_2' : \Pi_2\delta_2\rho'$ where Σ_2' is a signature containing variables in $\Sigma_2'\rho'$ but not already in the free variables of the end sequent of $\Pi_2\delta_2\rho'$. Having observed this, we can now apply the induction hypothesis (on the reduction of Π_1) to establish the reducibility of $\Xi^{\rho'}$. The reduct of Ξ

$$\dfrac{\left\{\begin{array}{c}\Xi^{\rho'}\\ \Delta_1'\delta_1\rho',\ldots,\Gamma\gamma\rho' \longrightarrow \mathcal{C}\gamma\rho'\end{array}\right\}_{\rho'}}{(\bar z \triangleright s = t)\delta_1, \Delta_1'\delta_1,\ldots,\Gamma\gamma \longrightarrow \mathcal{C}\gamma}\ \mathrm{eq}\mathcal{L}$$

is then reducible by Definition 4.14.

$\mu\mathcal{L}/\circ\mathcal{L}$**:** Suppose Π_1 is

$$\dfrac{\begin{array}{c}\Pi_S\\ D\,S\,\bar x \longrightarrow S\,\bar x\end{array} \quad \begin{array}{c}\Pi_1'\\ \bar y \triangleright S\,\bar t, \Delta_1' \longrightarrow \mathcal{B}_1\end{array}}{\bar y\triangleright p\,\bar t,\Delta_1'\longrightarrow\mathcal{B}_1}\ \mu\mathcal{L} \quad .$$

Since Π_1 is reducible, it follows from the definition of reducibility that Π_1' is reducible predecessor of Π_1 and Π_S is normalizable. Suppose $\bar z\triangleright p\,\bar u = (\bar y\triangleright p\,\bar t)\delta_1 = (\bar y\triangleright p\,\bar t)\gamma$. Let Ξ_1 be the derivation

$$\dfrac{\begin{array}{c}\Sigma_1 : \Pi_1'\delta_1\\ \bar z\triangleright S\,\bar u, \Delta_1'\delta_1 \longrightarrow \mathcal{B}_1\delta_1\end{array} \quad \cdots \quad \begin{array}{c}\Sigma_n : \Pi_n\delta_n\\ \Delta_n\delta_n \longrightarrow \mathcal{B}_n\delta_n\end{array} \quad \begin{array}{c}\Sigma : \Pi\gamma\\ \mathcal{B}_1\gamma,\ldots,\mathcal{B}_n\gamma,\Gamma\gamma \longrightarrow \mathcal{C}\gamma\end{array}}{\bar z\triangleright S\,\bar u, \Delta_1'\delta_1,\ldots,\Delta_n\delta_n,\Gamma\gamma \longrightarrow \mathcal{C}\gamma}\ mc \quad .$$

Ξ_1 is reducible by induction on the reduction of Π_1, therefore the reduct of Ξ

$$\dfrac{\begin{array}{c}\Pi_S\\ D\,S\,\bar x \longrightarrow S\,\bar x\end{array} \quad \begin{array}{c}\Xi_1\\ \bar z\triangleright S\,\bar u, \Delta_1'\delta_1,\ldots,\Delta_n\delta_n,\Gamma\gamma \longrightarrow \mathcal{C}\gamma\end{array}}{\bar z\triangleright p\,\bar u, \Delta_1'\delta_1,\ldots,\Delta_n\delta_n,\Gamma\gamma \longrightarrow \mathcal{C}\gamma}\ \mu\mathcal{L}$$

is reducible.

$-/\supset\mathcal{L}$**:** Suppose Π is

$$\dfrac{\begin{array}{c}\Pi'\\ \mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma' \longrightarrow \mathcal{D}'\end{array} \quad \begin{array}{c}\Pi''\\ \mathcal{B}_1,\ldots,\mathcal{B}_n,\mathcal{D}'',\Gamma' \longrightarrow \mathcal{C}\end{array}}{\mathcal{B}_1,\ldots,\mathcal{B}_n,\mathcal{D}'\supset\mathcal{D}'',\Gamma' \longrightarrow \mathcal{C}}\ \supset\mathcal{L} \quad .$$

Let Ξ_1 be

$$\dfrac{\begin{array}{c}\Sigma_1 : \Pi_1\delta_1\\ \Delta_1\delta_1 \longrightarrow \mathcal{B}_1\delta_1\end{array} \quad \cdots \quad \begin{array}{c}\Sigma_n : \Pi_n\delta_n\\ \Delta_n\delta_n \longrightarrow \mathcal{B}_n\delta_n\end{array} \quad \begin{array}{c}\Sigma : \Pi'\gamma\\ \mathcal{B}_1\gamma,\ldots,\mathcal{B}_n\gamma,\Gamma'\gamma \longrightarrow \mathcal{D}'\gamma\end{array}}{\Delta_1\delta_1,\ldots,\Delta_n\delta_n,\Gamma'\gamma \longrightarrow \mathcal{D}'\gamma}\ mc$$

and Ξ_2 be

$$\dfrac{\overset{\Sigma_1:\Pi_1\delta_1}{\Delta_1\delta_1 \longrightarrow \mathcal{B}_1\delta_1} \quad \cdots \quad \overset{\Sigma_n:\Pi_n\delta_n}{\Delta_n\delta_n \longrightarrow \mathcal{B}_n\delta_n} \quad \overset{\Sigma:\Pi''\gamma}{\mathcal{B}_1\gamma,\ldots,\mathcal{B}_n\gamma,\mathcal{D}''\gamma,\Gamma'\gamma \longrightarrow \mathcal{C}\gamma}}{\Delta_1\delta_1,\ldots,\Delta_n\delta_n,\mathcal{D}''\gamma,\Gamma'\gamma \longrightarrow \mathcal{C}\gamma}\; mc \quad .$$

Both Ξ_1 and Ξ_2 are reducible by induction hypothesis. Therefore the reduct of Ξ

$$\dfrac{\overset{\Xi_1}{\Delta_1\delta_1,\ldots,\Delta_n\delta_n,\Gamma'\gamma \longrightarrow \mathcal{D}'\gamma} \qquad \overset{\Xi_2}{\Delta_1\delta_1,\ldots,\Delta_n\delta_n,\mathcal{D}''\gamma,\Gamma'\gamma \longrightarrow \mathcal{C}\gamma}}{\Delta_1\delta_1,\ldots,\Delta_n\delta_n,(\mathcal{D}' \supset \mathcal{D}'')\gamma,\Gamma'\gamma \longrightarrow \mathcal{C}\gamma}\; \supset\mathcal{L} \quad .$$

is reducible (recall that reducibility of Ξ_1 implies its normalizability by Lemma 4.8).
$-/\nu\mathcal{R}$**:** Suppose Π is

$$\dfrac{\overset{\Pi'}{\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma \longrightarrow \bar{y} \triangleright S\,\bar{t}} \quad \overset{\Pi_S}{S\,\bar{x} \longrightarrow D\,S\,\bar{x}}}{\mathcal{B}_1,\ldots,\mathcal{B}_n,\Gamma \longrightarrow \bar{y} \triangleright p\,\bar{t}}\; \nu\mathcal{R} \quad ,$$

where $p\,\bar{x} \overset{\nu}{=} D\,p\,\bar{x}$. Suppose $\bar{z} \triangleright p\,\bar{u} = (\bar{y} \triangleright p\,\bar{t})\delta_1 = (\bar{y} \triangleright p\,\bar{t})\gamma$. Let Ξ_1 be the derivation

$$\dfrac{\overset{\Sigma_1:\Pi_1\delta_1}{\Delta_1\delta_1 \longrightarrow \mathcal{B}_1\delta_1} \quad \cdots \quad \overset{\Sigma:\Pi_n\delta_n}{\Delta_n\delta_n \longrightarrow \mathcal{B}_n\delta_n} \quad \overset{\Sigma:\Pi'\gamma}{\mathcal{B}_1\gamma,\ldots,\mathcal{B}_n\gamma,\Gamma\gamma \longrightarrow \bar{z} \triangleright S\,\bar{u}}}{\Delta_1\delta_1,\ldots,\Delta_n\delta_n,\Gamma\gamma \longrightarrow \bar{z} \triangleright S\,\bar{u}}\; mc \quad .$$

The derivations $\Sigma:\Pi'\gamma$, Π_S, Ξ_1 and the derivation

$$\dfrac{\overset{\Psi}{\Delta' \longrightarrow \bar{a} \triangleright S\,\bar{w}} \quad \overset{\Pi'_S}{\bar{a} \triangleright S\,\bar{w} \longrightarrow \bar{a} \triangleright D\,S\,\bar{w}}}{\Delta' \longrightarrow \bar{a} \triangleright D\,S\,\bar{w}}\; mc \quad ,$$

where Ψ is any reducible derivation and Π'_S is a raised instance of Π_S, are reducible by induction hypothesis on the length of Π. Again, we use the same arguments as in the case where $n = 0$ to show that the $(\Re_{<\mathrm{lvl}(p)}, \Xi_1, \Pi_S)$-generated set contains only reducible derivations. Therefore by Definition 4.14, the reduct Ξ

$$\dfrac{\overset{\Xi_1}{\Delta_1\delta_1,\ldots,\Delta_n\delta_n,\Gamma\gamma \longrightarrow \bar{z} \triangleright S\,\bar{u}} \quad \overset{\Pi_S}{S\,\bar{x} \longrightarrow D\,S\,\bar{x}}}{\Delta_1\delta_1,\ldots,\Delta_n\delta_n,\Gamma\gamma \longrightarrow \bar{z} \triangleright p\,\bar{u}}\; \nu\mathcal{R}$$

is reducible.
mc/ $\circ\mathcal{L}$**:** Suppose Π_1 ends with a *mc*. Then any reduct of $\Sigma_1:\Pi_1\delta_1$ corresponds to a predecessor of Π_1 by Lemma 4.3 and Lemma 4.2. Therefore the reduct of Ξ is reducible by induction on the reduction of Π_1.
$-/$*init*: Ξ reduces to $\Sigma_1:\Pi_1\delta_1$. Since Π_1 is reducible, by Lemma 4.16 and Lemma 4.17, $\Sigma_1:\Pi_1\delta_1$ is reducible and hence Ξ is reducible. ∎

Corollary 4.21. *Given a fixed stratified definition, a sequent has a proof in* Linc *if and only if it has a cut-free proof.*

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

Corollary 4.22. *The logic* Linc *is consistent.*

Proof Suppose otherwise, that is, there is a formula C such that there is a proof Π_1 of C and another proof Π_2 for $C \supset \bot$. By cut-elimination, Π_2 must end with $\supset \mathcal{R}$, that is, Π_2 is

$$\dfrac{\begin{matrix}\Pi_2' \\ C \longrightarrow \bot\end{matrix}}{\longrightarrow C \supset \bot} \supset \mathcal{R}$$

Cutting Π_1 with Π_2' we get a derivation of $\cdot \longrightarrow \bot$, and applying the cut-elimination procedure we get a cut-free derivation of $\cdot \longrightarrow \bot$. But there cannot be such derivation since there is no right-introduction rule for $\bot$, contradiction. ∎

4.4 Conclusion

The proof of cut-elimination for Linc closely resembles that of $FO\lambda^{\Delta\mathbb{N}}$. In particular, the notion of reducibility is defined by induction on the level of the derivation as in $FO\lambda^{\Delta\mathbb{N}}$. We argue that the side condition on $\nu\mathcal{R}$ is imposed by the use of this framework of reducibility. More precisely, suppose we lift the side condition on $\nu\mathcal{R}$. Then given a co-inductive definition $p\,x \overset{\nu}{=} B\,p\,x$ and a derivation Π

$$\dfrac{\begin{matrix}\Pi' \\ \Gamma \longrightarrow S\,t\end{matrix} \quad \begin{matrix}\Pi_S \\ S\,x \longrightarrow B\,S\,x\end{matrix}}{\Gamma \longrightarrow p\,t} \nu\mathcal{R}$$

where $\mathrm{lvl}(S\,x) > \mathrm{lvl}(p)$, the reducibility of Π cannot depend on the reducibility of its premises. Therefore the only way to make the reducibility argument go through is by directly defining the co-inductive unfolding of Π' as the predecessor of Π. But we see there is a problem with this scheme of reducibility. Suppose we are given a co-inductive definition $p \overset{\nu}{=} p$. Let Π be the derivation

$$\dfrac{\begin{matrix}\Pi' \\ \longrightarrow p\end{matrix} \quad \dfrac{}{p \longrightarrow p}\, \mathit{init}}{\longrightarrow p} \nu\mathcal{R}$$

Let Π^1 be the derivation

$$\dfrac{\begin{matrix}\Pi' \\ \longrightarrow p\end{matrix} \quad \dfrac{}{p \longrightarrow p}\, \mathit{init}}{\longrightarrow p} \mathit{mc}$$

Then Π is reducible if the result of unfolding Π^1 is reducible, that is, if the following derivation Π^2

$$\cfrac{\cfrac{\cfrac{\Pi'}{\longrightarrow p} \quad \cfrac{}{p \longrightarrow p}\,\mathit{init}}{\longrightarrow p}\,\mathit{mc} \quad \cfrac{}{p \longrightarrow p}\,\mathit{init}}{\longrightarrow p}\,\nu\mathcal{R}$$

is reducible. Now, Π^2 is reducible if the following derivation Π^3 is reducible.

$$\cfrac{\cfrac{\cfrac{\Pi^2}{\longrightarrow p} \quad \cfrac{}{p \longrightarrow p}\,\mathit{init}}{\longrightarrow p}\,\mathit{mc} \quad \cfrac{}{p \longrightarrow p}\,\mathit{init}}{\longrightarrow p}\,\nu\mathcal{R}$$

We can easily see that this leads to infinite descending chains and hence this notion of reducibility is not well-founded. In our formulation of reducibility, this infinite descend is avoided since the unfolded derivation Π^2 is not a predecessor of Π. However, this does not mean that the side condition on $\nu\mathcal{R}$ cannot be removed. Doing so would probably require a very different proof technique than the one we currently use.

Chapter 5

Reasoning about Logical Specifications in Linc

In this chapter, we illustrate the use of Linc to encode logical specifications and to reason about them. Both first-order and higher-order encodings are shown. The examples of first-order encoding are given in Section 5.1, Section 5.2 and Section 5.3. In these examples the ∇ quantifier plays no significant role, therefore we shall omit writing explicitly the local signatures. Section 5.1 shows the examples of encoding natural numbers in Linc. In particular, we show that the natural number induction rule in $FO\lambda^{\Delta\mathbb{N}}$ is derivable in Linc. Section 5.2 presents some examples of lists, both finite and infinite, and show how we can derive the (co)-induction principle for lists and use it to derive properties about functions on lists. Section 5.3 shows an encoding of CCS [39] with fixed point operator. We illustrate the use of co-induction proof technique in proving the *similarity* of processes in CCS.

The next two sections illustrate the encodings involving higher-order abstract syntax. Section 5.4 shows an encoding of a simple object-logic, Horn logic with universal quantification. We show an example of reasoning about the provability of generic judgments in the object-logic which makes an essential use of ∇ in interpreting the object-logic universal quantification. We also discuss in this section a limitation of ∇ in dealing with certain inductive proofs involving higher-order abstract syntax. Section 5.5 presents an encoding of the lazy λ-calculus and the notion of *applicative bisimulation* [1]. Section 5.6 concludes this chapter and discusses some related work.

5.1 Natural numbers

We introduce a type nt to encode natural numbers. The type nt has the following constructors:

$$z : nt \qquad s : nt \rightarrow nt$$

which denote the natural number zero and the successor function, respectively. The membership predicate is encoded as the inductive predicate nat with the following definition clause

$$nat\ x \overset{\mu}{=} (x = z) \vee \exists y.(x = (s\ y)) \wedge nat\ y.$$

Given this definition, we can derive the natural number induction rule as in $FO\lambda^{\Delta\mathbb{N}}$ (i.e., the $nat\mathcal{L}$, see [28]).

Proposition 5.1. *The rule nat$\mathcal{L}$*

$$\frac{\longrightarrow B\,z \quad i\,;B\,i \longrightarrow B\,(s\ i) \quad B\,I, \Gamma \longrightarrow C}{nat\ I, \Gamma \longrightarrow C}\ nat\mathcal{L}$$

is derivable in Linc.

Proof We show that there is a derivation from the given premises of $nat\mathcal{L}$ to its conclusion, using the rules in Linc.

$$
\dfrac{
\dfrac{
\dfrac{\longrightarrow B\,z}{j\,;\, j = z \longrightarrow B\,j}\;\mathrm{eq}\mathcal{L}^{csu}
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{i\,;\,B\,i \longrightarrow B\,(s\,i)}{j,k\,;\,j=(s\,k), B\,k \longrightarrow B\,j}\;\mathrm{eq}\mathcal{L}^{csu}}{j,k\,;\,j=(s\,k)\wedge B\,k \longrightarrow B\,j}\;\wedge\mathcal{L}^{*}}{j\,;\,\exists k.j=(s\,k)\wedge B\,k \longrightarrow B\,j}\;\exists\mathcal{L}
}{j\,;\,(j=z)\vee \exists k.j=(s\,k)\wedge B\,k \longrightarrow B\,j}\;\vee\mathcal{L}
\qquad
B\,I,\Gamma \longrightarrow C
}{nat\;I,\Gamma \longrightarrow C}\;\mu\mathcal{L}
$$

■

Given the derivability of $nat\mathcal{L}$ in Linc, all previous examples in $FO\lambda^{\Delta\mathbb{N}}$ can be trivially carried over to Linc without much effort. For example, we can show that Linc with the definition nat above encodes the intuitionistic version of Peano's arithmetic, a result which is shown for $FO\lambda^{\Delta\mathbb{N}}$ in [30]. We illustrate the use of this derived rule in the following example.

Example 5.2. The sum function can be encoded using the predicate $sum : nt \to nt \to nt \to o$, defined inductively as follows.

$$
\begin{aligned}
sum\;I\;J\;K \overset{\mu}{=}\; & (I = z \wedge J = K) \vee \\
& \exists M \exists N. I = (s\,M) \wedge K = (s\,N) \wedge sum\;M\;J\;N.
\end{aligned}
$$

Note that since sum is already an inductive definition we can prove certain properties about it directly by structural induction on its definition, that is, if the predicate sum appears on the assumptions on the properties to be proved. Some other properties must still be proved with the assumption that the arguments of sum are natural numbers. Consider for example proving the functionality and the totality of sum on its first two arguments. The functionality property is stated formally as follows (we leave out the type nt)

$$\forall I \forall J \forall M sum\;I\;J\;M \supset \forall N. sum\;I\;J\;N \supset M = N$$

and the totality property is specified as

$$\forall I. nat\;I \supset \forall J. nat\;J \supset \exists K. sum\;I\;J\;K.$$

The former property is proved by induction on the predicate sum while the latter is by induction on nat . In both cases, the induction invariants are simply the right hand-side of the outermost implications, i.e., the corresponding invariants for the above cases are:

$$
\begin{aligned}
D_1 &= \lambda I \lambda J \lambda M. \forall N. sum\;I\;J\;N \supset M = N \\
D_2 &= \lambda I \forall J. nat\;J \supset \exists K. sum\;I\;J\;K.
\end{aligned}
$$

The informal proof for the functionality property is done by induction on the first argument of sum, followed by case analyses on the definition of sum. The formal proofs

$$\dfrac{\dfrac{\begin{array}{c}\Pi_z\\ \longrightarrow D\ z\ J\ J\end{array} \quad \begin{array}{c}\Pi_s\\ D\ I\ J\ M \longrightarrow D\ (s\,I)\ J\ (s\,M)\end{array} \quad \dfrac{}{D\ I\ J\ M \longrightarrow D\ I\ J\ M}\ \mathit{init}}{\mathit{sum}\ I\ J\ M \longrightarrow D\ I\ J\ M}\ \mathit{sum}_{\mathcal{L}}}{\longrightarrow \forall I \forall J \forall M \mathit{sum}\ I\ J\ M \supset D\ I\ J\ M}\ \forall\mathcal{R};\supset\mathcal{R}$$

where

$$D = \lambda I \lambda J \lambda M.\forall N.\mathit{sum}\ I\ J\ N \supset M = N,$$

Π_z is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{J = N \longrightarrow J = N}\ \mathit{init}}{z = z \wedge J = N \longrightarrow J = N}\ \wedge\mathcal{L} \quad \dfrac{\dfrac{\dfrac{}{z = (s\,M) \longrightarrow J = N}\ \mathrm{eq}\mathcal{L}}{\begin{array}{r} z = (s\,M) \wedge N = (s\,N') \wedge \\ \mathit{sum}\,M\,J\,N' \end{array} \longrightarrow J = N}\ \wedge\mathcal{L}}{\exists M \exists N'.\left[\begin{array}{r} z = (s\,M) \wedge N = (s\,N') \wedge \\ \mathit{sum}\,M\,J\,N' \end{array}\right] \longrightarrow J = N}\ \exists\mathcal{L}}{\begin{array}{c}(z = z \wedge J = N)\ \vee \\ \exists M \exists N'.z = (s\,M) \wedge N = (s\,N') \wedge \mathit{sum}\ M\ J\ N'\end{array} \longrightarrow J = N}\ \vee\mathcal{L}}{\mathit{sum}\,z\,J\,N \longrightarrow J = N}\ \mathit{def}\mathcal{L}}{\longrightarrow \mathit{sum}\,z\,J\,N \supset J = N}\ \supset\mathcal{R}}{\longrightarrow \forall N.\mathit{sum}\,z\,J\,N \supset J = N}\ \forall\mathcal{R}$$

and Π_s is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\cdots, s\,I = z \longrightarrow s\,M = N}\ \mathrm{eq}\mathcal{L}}{\cdots, \begin{array}{c} s\,I = z \wedge \\ J = N \end{array} \longrightarrow s\,M = N}\ \wedge\mathcal{L} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathit{sum}\,I\,J\,Q \longrightarrow \mathit{sum}\,I\,J\,Q}\ \mathit{init} \quad \dfrac{\dfrac{}{\longrightarrow s\,M = s\,M}\ \mathrm{eq}\mathcal{R}}{M = Q \longrightarrow s\,M = s\,Q}\ \mathrm{eq}\mathcal{L}}{\mathit{sum}\,I\,J\,Q \supset M = Q, \mathit{sum}\,I\,J\,Q \longrightarrow s\,M = s\,Q}\ \supset\mathcal{L}}{D\,I\,J\,M, \mathit{sum}\,I\,J\,Q \longrightarrow s\,M = s\,Q}\ \forall\mathcal{L}}{\cdots, \begin{array}{l} s\,I = s\,P \wedge \\ N = (s\,Q) \wedge \\ \mathit{sum}\ P\ J\ Q \end{array} \longrightarrow s\,M = N}\ \wedge\mathcal{L}^*;\mathrm{eq}\mathcal{L}}{\cdots, \exists P \exists Q.\left[\begin{array}{l} s\,I = s\,P \wedge \\ N = (s\,Q) \wedge \\ \mathit{sum}\ P\ J\ Q \end{array}\right] \longrightarrow s\,M = N}\ \exists\mathcal{L}}{D_1\,I\,J\,M, \left[\begin{array}{r} s\,I = z \wedge J = N \vee \\ \exists P \exists Q.s\,I = s\,P \wedge N = s\,Q \wedge \mathit{sum}\ P\ J\ Q \end{array}\right] \longrightarrow s\,M = N}\ \vee\mathcal{L}}{D\,I\,J\,M, \mathit{sum}\,(s\,I)\,J\,N \longrightarrow (s\,M) = N}\ \mathit{def}\mathcal{L}}{D\,I\,J\,M \longrightarrow \forall N.\mathit{sum}\ (s\,I)\ J\ N \supset s\,M = N}\ \forall\mathcal{R};\supset\mathcal{R}$$

Fig. 5.1. A formal proof of the functionality of *sum*.

$$
\dfrac{\dfrac{\dfrac{\begin{matrix}\Pi_z\\ \longrightarrow F\ z\end{matrix}\quad\begin{matrix}\Pi_s\\ F\ I \longrightarrow F\ (s\,I)\end{matrix}\quad\dfrac{}{F\ I \longrightarrow \forall J.\mathit{nat}\ J \supset \exists K.\mathit{sum}\ I\ J\ K}\,\mathit{init}}{\mathit{nat}\ I \longrightarrow \forall J.\mathit{nat}\ J \supset \exists K.\mathit{sum}\ I\ J\ K}\,\mathit{nat}\mathcal{L}}{\longrightarrow \mathit{nat}\ I \supset \forall J.\mathit{nat}\ J \supset \exists K.\mathit{sum}\ I\ J\ K}\supset\mathcal{R}}{\longrightarrow \forall I.\mathit{nat}\ I \supset \forall J.\mathit{nat}\ J \supset \exists K.\mathit{sum}\ I\ J\ K}\,\forall\mathcal{R}
$$

where F is $\lambda I.\forall J.\mathit{nat}\ J \supset \exists K.\mathit{sum}\ I\ J\ K$, the derivations Π_z is

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathit{nat}\ J \longrightarrow z = z}\,\mathrm{eq}\mathcal{R}\quad\dfrac{}{\mathit{nat}\ J \longrightarrow J = J}\,\mathrm{eq}\mathcal{R}}{\mathit{nat}\ J \longrightarrow z = z \wedge J = J}\,\wedge\mathcal{R}}{\mathit{nat}\ J \longrightarrow (z = z \wedge J = J)\ \vee\ \exists M \exists N.z = (s\,M) \wedge J = (s\,N) \wedge \mathit{sum}\ M\ J\ N}\,\vee}{\mathit{nat}\ J \longrightarrow \mathit{sum}\ z\ J\ J}\,\mathit{def}\mathcal{R}}{\mathit{nat}\ J \longrightarrow \exists K.\mathit{sum}\ z\ J\ K}\,\exists\mathcal{R}}{\longrightarrow \mathit{nat}\ J \supset \exists K.\mathit{sum}\ z\ J\ K}\supset\mathcal{R}}{\longrightarrow \forall J.\mathit{nat}\ J \supset \exists K.\mathit{sum}\ z\ J\ K}\,\forall\mathcal{R}
$$

Π_s is

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathit{nat}\ J \longrightarrow \mathit{nat}\ J}\,\mathit{init}\quad\dfrac{\dfrac{\begin{matrix}\Pi_s'\\ \mathit{sum}\ I\ J\ K, \mathit{nat}\ J \longrightarrow \mathit{sum}\ (s\,I)\ J\ (s\,K)\end{matrix}}{\mathit{sum}\ I\ J\ K, \mathit{nat}\ J \longrightarrow \exists K.\mathit{sum}\ (s\,I)\ J\ K}\,\exists\mathcal{R}}{\exists K.\mathit{sum}\ I\ J\ K, \mathit{nat}\ J \longrightarrow \exists K.\mathit{sum}\ (s\,I)\ J\ K}\,\exists\mathcal{L}}{\mathit{nat}\ J \supset \exists K.\mathit{sum}\ I\ J\ K, \mathit{nat}\ J \longrightarrow \exists K.\mathit{sum}\ (s\,I)\ J\ K}\supset\mathcal{L}}{\forall J.\mathit{nat}\ J \supset \exists K.\mathit{sum}\ I\ J\ K, \mathit{nat}\ J \longrightarrow \exists K.\mathit{sum}\ (s\,I)\ J\ K}\,\forall\mathcal{L}}{\forall J.\mathit{nat}\ J \supset \exists K.\mathit{sum}\ I\ J\ K \longrightarrow \mathit{nat}\ J \supset \exists K.\mathit{sum}\ (s\,I)\ J\ K}\supset\mathcal{R}}{\forall J.\mathit{nat}\ J \supset \exists K.\mathit{sum}\ I\ J\ K \longrightarrow \forall J.\mathit{nat}\ J \supset \exists K.\mathit{sum}\ (s\,I)\ J\ K}\,\forall\mathcal{R}
$$

and Π_s' is

$$
\dfrac{\dfrac{\begin{array}{c}\dfrac{\dfrac{\dfrac{}{\cdots \longrightarrow s\,I = s\,I}\,\mathrm{eq}\mathcal{R}\quad\dfrac{}{\cdots \longrightarrow s\,I = s\,I}\,\mathrm{eq}\mathcal{R}}{\cdots \longrightarrow s\,I = s\,I \wedge s\,K = s\,K}\,\wedge\mathcal{R}\qquad\dfrac{}{\mathit{sum}\,I\,J\,K, \mathit{nat}\ J \longrightarrow \mathit{sum}\,I\,J\,K}\,\mathit{init}}{\mathit{sum}\ I\ J\ K, \mathit{nat}\ J \longrightarrow s\,I = s\,I \wedge s\,K = s\,K \wedge \mathit{sum}\ M\ J\ K}\,\wedge\mathcal{R}\\ \hline\hline \mathit{sum}\ I\ J\ K, \mathit{nat}\ J \longrightarrow \exists M \exists N.s\,I = s\,M \wedge s\,K = s\,N \wedge \mathit{sum}\ M\ J\ N\end{array}\,\exists\mathcal{R}}{\mathit{sum}\ I\ J\ K, \mathit{nat}\ J \longrightarrow \begin{array}{l}(s\,I = z \wedge J = s\,K)\ \vee\\ \exists M \exists N.s\,I = s\,M \wedge s\,K = s\,N \wedge \mathit{sum}\ M\ J\ N\end{array}}\,\vee\mathcal{R}}{\mathit{sum}\ I\ J\ K, \mathit{nat}\ J \longrightarrow \mathit{sum}\ (s\,I)\ J\ (s\,K)}\,\mathit{def}\mathcal{R}
$$

Fig. 5.2. A formal proof of the totality of *sum*.

given in Figure 5.1 and Figure 5.2 follow exactly this structure of informal proofs. In Figure 5.1 we make use of a derived induction rule for *sum*:

$$\dfrac{\longrightarrow D\ z\ J\ J \qquad D\ I\ J\ K \longrightarrow D\ (s\,I)\ J\ (s\,K) \qquad D\ R\ S\ T, \Gamma \longrightarrow C}{\mathit{sum}\,R\,S\,T, \Gamma \longrightarrow C}\ \mathit{sum}_{\mathcal{L}}$$

where I, J, K are eigenvariables in their respected sequents. This rule can be derived in Linc in the similar way as we derive $\mathit{nat}\mathcal{L}$.

∎

5.2 Lists

Lists over some fixed type α are encoded as the type *lst*, with the usual constructors nil : *lst* for empty list and :: : $\alpha \to \mathit{lst} \to \mathit{lst}$. We consider some functions on lists and their properties, for both finite lists and infinite lists.

$$\dfrac{\dfrac{}{\longrightarrow z = z}\ \mathrm{eq}\mathcal{R} \qquad \dfrac{\dfrac{}{\longrightarrow x :: L = x :: L}\ \mathrm{eq}\mathcal{R}}{L_1 = L_2 \longrightarrow x :: L1 = x :: L2}\ \mathrm{eq}\mathcal{L}^{csu} \qquad \dfrac{}{\cdots, L_1 = L_2 \longrightarrow L_1 = L_2}\ \mathit{init}}{\mathit{list}\ L_1, \mathit{list}\ L_2, \mathit{eqx}\ L_1\ L_2 \longrightarrow L_1 = L_2}\ \mathit{eqx}_{\mathcal{L}}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\longrightarrow \top}\ \top\mathcal{R}}{\longrightarrow \mathit{eqx}\,\mathrm{nil}\,\mathrm{nil}}\ \mathit{def}\mathcal{R}^{=} \qquad \dfrac{\dfrac{}{\mathit{eqx}\,L_1'\,L_2' \longrightarrow \mathit{eqx}\,L_1'\,L_2'}\ \mathit{init}}{\mathit{eqx}\,(x :: L_1')\,(x :: L_2') \longrightarrow \mathit{eqx}\,L_1'\,L_2'}\ \mathrm{def}\mathcal{L}^{csu} \qquad \dfrac{}{\mathit{eqx}\,L\,L, \mathit{list}\ L \longrightarrow \mathit{eqx}\,L\,L}\ \mathit{init}}{\mathit{list}\ L, \mathit{list}\ L \longrightarrow \mathit{eqx}\ L\ L}\ \mathit{list}_{\mathcal{L}}}{\mathit{list}\ L_1, \mathit{list}\ L_2, L_1 = L_2 \longrightarrow \mathit{eqx}\ L_1\ L_2}\ \mathrm{eq}\mathcal{L}^{csu}$$

Fig. 5.3. Freeness property of lists.

5.2.1 Finite lists

Finite lists over α are generated by the constructors nil and ::, from the elements of the type α. The membership predicate is defined as follows.

$$\mathit{list}\ L \overset{\mu}{=} L = \mathrm{nil} \lor \exists x \exists L'.L = x :: L' \land \mathit{list}\ L'$$

Its corresponding induction rule is

$$\frac{\longrightarrow D\ \mathrm{nil} \quad D\ L \longrightarrow D\ (x :: L) \quad D\ l, \Gamma \longrightarrow C}{\mathit{list}\ l, \Gamma \longrightarrow C}\ \mathit{list}_{\mathcal{L}}$$

List is an example of *free* inductive data type (just as natural number is), that is, two lists are (extensionally) equal if and only if they are syntactically equal. More precisely, we can define the equality predicate for lists as follows.

$$\begin{array}{rcl} \mathit{eqx}\ L_1\ L_2 & \overset{\mu}{=} & (L_1 = \mathrm{nil} \wedge L_2 = \mathrm{nil}) \vee \\ & & (\exists x \exists L_1' \exists L_2'. L_1 = (x :: L_1') \wedge L_2 = (x :: L_2') \wedge \mathit{eqx}\ L_1'\ L_2'). \end{array}$$

We can equally write this definition as a patterned definition as follows

$$\mathit{eqx}\ \mathrm{nil}\ \mathrm{nil} \overset{\mu}{=} \top, \quad \mathit{eqx}\ (X :: L_1')\ (X :: L_2') \overset{\mu}{=} \mathit{eqx}\ L_1'\ L_2'.$$

This patterned definition is used when we apply $\mathit{def}\mathcal{L}^{csu}$.

The freeness property above can be stated formally as

$$\forall L1 \forall L2. \mathit{list}\ L1 \supset \mathit{list}\ L2 \supset (\mathit{eqx}\ L1\ L2 \equiv (L1 = L2)).$$

Its two subproofs are shown in Figure 5.3. In one of the subproofs, we make use of a derived induction rule for *eq*, that is,

$$\frac{\longrightarrow D\ \mathrm{nil}\ \mathrm{nil} \quad D\, L_1\, L_2 \longrightarrow D\, (x :: L_1)\, (x :: L_2) \quad D\, l\, l', \Gamma \longrightarrow C}{\mathit{eqx}\ l\ l', \Gamma \longrightarrow C}\ \mathit{eqx}_{\mathcal{L}}$$

This rule can be shown to be derivable in Linc, following the same scheme for deriving $\mathit{nat}\mathcal{L}$ and *sum* as seen previously. The induction invariant in the first proof in the figure is $\lambda L_1 \lambda L2.L1 = L2$, while in the second it is $\lambda L.\mathit{eqx}\ L\ L$.

We next consider the append function on finite lists and prove one of its properties, i.e., associativity.

Example 5.3. *Associativity of app.* The append function can be encoded as the following definition clauses.

$$\begin{array}{rcl} \mathrm{app}\, L_1\, L_2\, L_3 & \overset{\mu}{=} & (L_1 = \mathrm{nil} \wedge L_2 = L_3) \vee \\ & & \exists x, L_1', L_3'. L_1 = (x :: L_1') \wedge L_3 = (x :: L_3') \wedge \mathrm{app}\, L_1'\, L_2\, L_3'. \end{array}$$

We can alternatively see this as coding the patterned definition clause

$$\mathrm{app}\, \mathrm{nil}\, L\, L \overset{\mu}{=} \top, \quad \mathrm{app}\, (X :: L_1)\, L_2\, (X :: L_3) \overset{\mu}{=} \mathrm{app}\, L_1\, L_2\, L_3.$$

The associated induction principle for app is as follows.

$$\frac{\longrightarrow D\ \mathrm{nil}\ L\ L \quad D\ L_1\ L_2\ L_3 \longrightarrow D\ (x :: L_1)\ L_2\ (X :: L_3) \quad D\ l_1\ l_2\ l_3, \Gamma \longrightarrow C}{\mathrm{app}\, l_1\, l_2\, l_3, \Gamma \longrightarrow C}\ \mathrm{app}_{\mathcal{L}}$$

The associativity of *app* is stated formally as follows.

$$\begin{gathered}\forall L_1 \forall L_2 \forall L_{12} \forall L_3 \forall L_4.(\mathrm{app}\, L_1\, L_2\, L_{12} \wedge \mathrm{app}\, L_{12}\, L_3\, L_4) \supset \\ \forall L_{23}.\mathrm{app}\, L_2\, L_3\, L_{23} \supset \mathrm{app}\, L_1\, L_{23}\, L_4.\end{gathered} \tag{5.1}$$

Proving this formula requires us to prove first that the definition of append is functional, that is,

$$\forall L_1 \forall L_2 \forall L_3 \forall L_4.\mathrm{app}\, L_1\, L_2\, L_3 \wedge \mathrm{app}\, L_1\, L_2\, L_4 \supset L_3 = L_4.$$

This is done by induction on $\mathrm{app}\, L_1\, L_2\, L_3$. The invariant in this case is

$$I = \lambda L_1 \lambda L_2 \lambda L_3.\forall R.\mathrm{app}\, L_1\, L_2\, R \supset R = L_3.$$

It is a simple case analysis to check that this is the right invariant:

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\Pi_1 \\ \longrightarrow I\,\mathrm{nil}\,L\,L\end{matrix} \quad \begin{matrix}\Pi_2 \\ I\,L_1'\,L_2\,L_3' \longrightarrow I\,(X :: L_1')\,L_2\,(X :: L_3')\end{matrix} \quad \begin{matrix}\Pi_3 \\ I\,L_1\,L_2\,L_3, \ldots \longrightarrow L_3 = L_4\end{matrix}}{\mathrm{app}\, L_1\, L_2\, L_3, \mathrm{app}\, L_1\, L_2\, L_4 \longrightarrow L_3 = L_4}\, \mathit{app}_{\mathcal{L}}}{\longrightarrow \mathrm{app}\, L_1\, L_2\, L_3 \wedge \mathrm{app}\, L_1\, L_2\, L_4 \supset L_3 = L_4}\, \supset\mathcal{R}; \wedge\mathcal{L}^*}{\longrightarrow \forall L_1 \forall L_2 \forall L_3 \forall L_4.\mathrm{app}\, L_1\, L_2\, L_3 \wedge \mathrm{app}\, L_1\, L_2\, L_4 \supset L_3 = L_4}\, \forall\mathcal{R}$$

where Π_1, Π_2 and Π_3 are the following derivation.

$$\dfrac{\dfrac{\dfrac{}{\longrightarrow \mathrm{nil} = \mathrm{nil}}\, \mathrm{eq}\mathcal{R}}{\mathrm{app}\,\mathrm{nil}\,\mathrm{nil}\,L \longrightarrow L = \mathrm{nil}}\, \mathit{def}\mathcal{L}^{csu}}{\longrightarrow \forall L.\mathrm{app}\,\mathrm{nil}\,\mathrm{nil}\,L \supset L = \mathrm{nil}}\, \forall\mathcal{R}; \supset\mathcal{R}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{app}\, L_1'\, L_2\, L' \longrightarrow \mathrm{app}\, L_1'\, L_2\, L'}\, \mathit{init} \quad \dfrac{\dfrac{}{\mathrm{app}\, L_1'\, L_2\, L'' \longrightarrow X :: L'' = X :: L''}\, \mathit{init}}{\mathrm{app}\, L_1'\, L_2\, L', L' = L_3' \longrightarrow X :: L' = X :: L_3'}\, \mathrm{eq}\mathcal{L}^{csu}}{\mathrm{app}\, L_1'\, L_2\, L' \supset L' = L_3', \mathrm{app}\, L_1'\, L_2\, L' \longrightarrow X :: L' = X :: L_3'}\, \supset\mathcal{L}}{\forall R.\mathrm{app}\, L_1'\, L_2\, R \supset R = L_3', \mathrm{app}\, L_1'\, L_2\, L' \longrightarrow X :: L' = X :: L_3'}\, \forall\mathcal{L}}{I\, L_1'\, L_2\, L_3', \mathrm{app}\, X :: L_1'\, L_2\, L \longrightarrow L = X :: L_3'}\, \mathit{def}\mathcal{L}^{csu}}{I\, L_1'\, L_2\, L_3' \longrightarrow \forall L.\mathrm{app}\, X :: L_1'\, L_2\, X :: L_3' \supset L = X :: L_3'}\, \forall\mathcal{R}; \supset\mathcal{R}$$

$$\dfrac{\dfrac{\dfrac{}{\mathrm{app}\, L_1\, L_2\, L_4 \longrightarrow \mathrm{app}\, L_1\, L_2\, L_4}\, \mathit{init} \quad \dfrac{}{L_3 = L_4, \mathrm{app}\, L_1\, L_2\, L_4 \longrightarrow L_3 = L_4}\, \mathit{init}}{\mathrm{app}\, L_1\, L_2\, L_4 \supset L_4 = L_3, \mathrm{app}\, L_1\, L_2\, L_4 \longrightarrow L_3 = L_4}\, \supset\mathcal{L}}{\forall R.\mathrm{app}\, L_1\, L_2\, R \supset R = L_3, \mathrm{app}\, L_1\, L_2\, L_4 \longrightarrow L_3 = L_4}\, \forall\mathcal{L}$$

We are now ready to prove the associativity of append. The problem of associativity is reduced to proving the following sequent

$$\mathrm{app}\, L_1\, L_2\, L_{12},\ \mathrm{app}\, L_{12}\, L_3\, L_4,\ \mathrm{app}\, L_2\, L_3\, L_{23} \longrightarrow \mathrm{app}\, L_1\, L_{23}\, L_4. \tag{5.2}$$

We then proceed by induction on the list L_1, that is, we apply the $\mathrm{app}_{\mathcal{L}}$ rule to the hypothesis $\mathrm{app}\, L_1\, L_2\, L_{12}$ using the invariant

$$S = \lambda L_1 \lambda L_2 \lambda L_{12}.\forall L_3 \forall L_4.\mathrm{app}\, L_{12}\, L_3\, L_4 \supset \forall L_{23}.\mathrm{app}\, L_2\, L_3\, L_{23} \supset \mathrm{app}\, L_1\, L_{23}\, L_4.$$

Applying the $\mu\mathcal{L}$ rule, followed by $\vee\mathcal{L}$, to sequent (5.2) reduces the sequent to the following sub-goals

(i) $\longrightarrow S\ \mathrm{nil}\, L_2\, L_3$,

(ii) $S\ L_1'\, L_2\, L_3' \longrightarrow S\ (X :: L_1)\, L_2\, (X :: L_3)$,

(ii) $S\ L_1\, L_2\, L_{12},\ \mathrm{app}\, L_{12}\, L_3\, L_4,\ \mathrm{app}\, L_2\, L_3\, L_{23} \longrightarrow \mathrm{app}\, L_1\, L_{23}\, L_4$,

The proof for the first sequent is straightforward. The proof for the second sequent is done by a series of case analysis. The third sequent reduces to

$$\mathrm{app}\, L_{12}\, L_3\, L_4, \mathrm{app}\, L_{12}\, L_3\, L_{23} \longrightarrow \mathrm{app}\,\mathrm{nil}\, L_{23}\, L_4.$$

This follows from the functionality of append (which identifies L_4 and L_{23}) and *defR*. ∎

5.2.2 Infinite lists

Unlike finite lists, infinite lists are not characterized by the finite constructions from its constructors. Instead, it is characterized by the *destructor operations* (also called *observation*) on lists, i.e., taking the *head* and the *tail* of the list. In a list of the form $x :: L$, the element x is the head and L is the tail of the list. This leads to a different notion of equivalence of lists, that is, two lists are equivalent if they are observationally equivalent. This notion of equivalence is typically defined via *bisimulation* (a more general notion, involving transition systems, will be given in Section 5.3). In the case of lists, the observables are the empty list nil and the constructor ::. The equivalence relation for the (possibly) infinite lists is defined as follows.

$$\begin{array}{rcl} \mathit{eqi}\, L_1\, L_2 & \stackrel{\nu}{=} & (L_1 = \mathrm{nil} \supset L_2 = \mathrm{nil}) \wedge (L_2 = \mathrm{nil} \supset L_1 = \mathrm{nil}) \wedge \\ & & (\forall X \forall L_1'.L_1 = X :: L_1' \supset \exists L_2'.L_2 = X :: L_2' \wedge \mathit{eqi}\ L_1'\ L_2') \\ & & (\forall X \forall L_2'.L_2 = X :: L_2' \supset \exists L_1'.L_1 = X :: L_1' \wedge \mathit{eqi}\ L_2'\ L_1') \end{array}$$

It can be verified that the relation defined by *eqi* obeys the usual equality laws, i.e., reflexive, symmetric and transitive.

One interesting property concerning equality of infinite lists is that it can be defined via finite approximation. That is, two infinite lists are equal (in the above sense) if and only if every finite prefix of one list is (syntactically) equal to the prefix of the same length of the other list. This is also known as the *take lemma* [5]. We show the proof of the take lemma in the following example.

Example 5.4. *The take lemma.* Let us define the *tk* predicate as follows.

$$\begin{aligned} tk\, z\, L\, \mathrm{nil} &\stackrel{\triangle}{=} \top. \\ tk\, N\, \mathrm{nil}\, \mathrm{nil} &\stackrel{\triangle}{=} \top. \\ tk\, (s\, N)\, (X :: L)\, (X :: R) &\stackrel{\triangle}{=} tk\, N\, L\, R. \end{aligned}$$

Given a natural number n and a list L, the *tk* predicate "outputs" a list R such that R is the first n elements of L. The definition consisting of *eqi* and *tk* can be stratified by assigning *eqi* with a level higher than the level of *tk*.

Let $eqf\, L_1\, L_2$ denotes the formula $\forall N.nat\ N \supset \forall L.tk\, N\, L_1\, L \equiv tk\, N\, L_2\, L$. The take lemma can be stated formally as $eqi\, L_1\, L_2 \equiv eqf\, L_1\, L_2$. We first prove the forward direction, that is,

$$eqi\, L_1\, L_2 \supset \forall N.nat\ N \supset \forall L.tk\, N\, L_1\, L \equiv tk\, N\, L_2\, L.$$

Proving this formula reduces to proving the sequent

$$eqi\, L_1\, L_2, nat\ N \longrightarrow \forall L.tk\, N\, L_1\, L \equiv tk\, N\, L_2\, L.$$

This is done by induction on N with the invariant

$$I = \lambda N.\forall L_1 \forall L_2.eqi\, L_1\, L_2 \supset \forall L.tk\, N\, L_1\, L \equiv tk\, N\, L_2\, L.$$

The proof for the base case is as follows.

$$\dfrac{\dfrac{\dfrac{}{eqi\, L_1\, L_2 \longrightarrow \top}\ \top\mathcal{R}}{eqi\, L_1\, L_2 \longrightarrow tk\, z\, L_2\, \mathrm{nil}}\ def\mathcal{R}^{=} \qquad \dfrac{\dfrac{}{eqi\, \mathrm{nil}\, L_2 \longrightarrow \top}\ \top\mathcal{R}}{eqi\, \mathrm{nil}\, L_2 \longrightarrow tk\, z\, L_2\, \mathrm{nil}}\ def\mathcal{R}^{=}}{eqi\, L_1\, L_2, tk\, z\, L_1\, L \longrightarrow tk\, z\, L_2\, L}\ def\mathcal{L}^{csu}$$

The inductive cases are given by the sequents

$$I\, N, eqi\, L_1\, L_2, tk\, (s\, N)\, L_1\, L \longrightarrow tk\, (s\, N)\, L_2\, L$$

and

$$I\, N, eqi\, L_1\, L_2, tk\, (s\, N)\, L_2\, L \longrightarrow tk\, (s\, N)\, L_1\, L.$$

We show the proof for the first sequent, since the proof for the latter can be obtained from the first by substitution (exchanging L_1 and L_2) and the symmetry of *eqi*. The proof starts with case analyses ($def\mathcal{L}^{csu}$) on *tk*, followed by *eqi*. The detail of the formal proof is rather involved, but it is basically some mechanical checking of cases. We look at the particular case where both L_1 and L_2 are instantiated to non-nil terms, i.e., $L_1 = X :: L_1'$ and $L_2 = X :: L_2'$ for some X, L_1' and L_2'. This subproof is given in the following.

$$\dfrac{\dfrac{}{\ldots, eqi\,L_1'\,L_2' \longrightarrow eqi\,L_1'\,L_2'}\,init \qquad \begin{array}{c}\Pi\\ \forall L.tk\,N\,L_1'\,L \equiv tk\,N\,L_2'\,L, \cdots \longrightarrow tk\,N\,L_2'\,L'\end{array}}{\dfrac{eqi\,L_1'\,L_2' \supset \forall L.tk\,N\,L_1\,L \equiv tk\,N\,L_2\,L, eqi\,L_1'\,L_2', tk\,N\,L_1'\,L' \longrightarrow tk\,N\,L_2'\,L'}{I\,N, eqi\,L_1'\,L_2', tk\,N\,L_1'\,L' \longrightarrow tk\,N\,L_2'\,L'}\,\forall\mathcal{L}}\,\supset\mathcal{R}$$

Here the subproof Π is

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\cdots, tk\,N\,L_1'\,L' \longrightarrow tk\,N\,L_1'\,L'}\,init \qquad \dfrac{}{\cdots, tk\,N\,L_2'\,L' \longrightarrow tk\,N\,L_2'\,L'}\,init}{tk\,N\,L_1'\,L' \supset tk\,N\,L_2'\,L, \cdots \longrightarrow tk\,N\,L_2'\,L'}\,\supset\mathcal{L}}{tk\,N\,L_1'\,L' \equiv tk\,N\,L_2'\,L, \cdots \longrightarrow tk\,N\,L_2'\,L'}\,\wedge\mathcal{L}}{\forall L.tk\,N\,L_1'\,L \equiv tk\,N\,L_2'\,L, \cdots \longrightarrow tk\,N\,L_2'\,L'}\,\forall\mathcal{L}.$$

The other direction, i.e., $eqf\,L_1\,L_2 \supset eqi\,L_1\,L_2$ is proved by co-induction. The invariant in this case is $\lambda L_1 \lambda L_2.eqf\,L_1\,L_2$. Applying $\nu\mathcal{R}$, followed by applications of asynchronous rules, give us the following premises

1. $eqf\,\mathrm{nil}\,L_2 \longrightarrow L_2 = \mathrm{nil}$
2. $eqf\,L_1\,\mathrm{nil} \longrightarrow L_1 = \mathrm{nil}$
3. $eqf(X :: L_1')\,L_2 \longrightarrow \exists L_2'.L_2 = (X :: L_2') \wedge eqf\,L_1'\,L_2'$.
4. $eqf\,L_1\,(X :: L_1') \longrightarrow \exists L_1'.L_1 = (X :: L_1') \wedge eqf\,L_1'\,L_2'$.
5. $eqf\,L_1\,L_2 \longrightarrow eqf\,L_1\,L_2$.

We show the proof for the first and the third sequents. The second and the fourth are symmetric to the first and the third. The proof for the first sequent is given in the following.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\longrightarrow \top}\,\top\mathcal{R}}{\longrightarrow nat\;(s\,z)}\,def\mathcal{R}^{=} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\longrightarrow \top}\,\top\mathcal{R}}{\longrightarrow tk\,(s\,z)\,\mathrm{nil}\,\mathrm{nil}}\,def\mathcal{R}^{=} \qquad \dfrac{\dfrac{}{\longrightarrow \mathrm{nil} = \mathrm{nil}}\,\mathrm{eq}\mathcal{R}}{tk\,(s\,z)\,L_2\,\mathrm{nil} \longrightarrow L_2 = \mathrm{nil}}\,def\mathcal{L}^{csu}}{tk\,(s\,z)\,\mathrm{nil}\,\mathrm{nil} \supset tk\,(s\,z)\,L_2\,\mathrm{nil} \longrightarrow L_2 = \mathrm{nil}}\,\supset\mathcal{L}}{tk\,(s\,z)\,\mathrm{nil}\,\mathrm{nil} \equiv tk\,(s\,z)\,L_2\,\mathrm{nil} \longrightarrow L_2 = \mathrm{nil}}\,\wedge\mathcal{L}}{\forall L.tk\,(s\,z)\,\mathrm{nil}\,L \equiv tk\,(s\,z)\,L_2\,L \longrightarrow L_2 = \mathrm{nil}}\,\forall\mathcal{L}}{nat\;(s\,z) \supset \forall L.tk\,(s\,z)\,\mathrm{nil}\,L \equiv tk\,(s\,z)\,L_2\,L \longrightarrow L_2 = \mathrm{nil}}\,\supset\mathcal{L}}{\forall N.nat\;N \supset \forall L.tk\,N\,\mathrm{nil}\,L \equiv tk\,N\,L_2\,L \longrightarrow L_2 = \mathrm{nil}}\,\forall\mathcal{L}$$

In the proof of the third sequent, we make use of the following lemma

$$eqf\,(X :: L)\,(X :: R) \supset eqf\,L\,R,$$

which essentially says that the set denoted by *eqf* is downward-closed. The proof of this lemma is given below. We refer to this proof as Ξ.

$$\begin{array}{cl}
\begin{array}{cc} \Xi_1 & \Xi_2 \\ \cdots \longrightarrow tk\,M\,L\,L' \supset tk\,M\,R\,L' & \cdots \longrightarrow tk\,M\,R\,L' \supset tk\,M\,L\,L' \end{array} & \\ \hline
eqf\,(X :: L)\,(X :: R), nat\ M \longrightarrow tk\,M\,L\,L' \equiv tk\,M\,R\,L' & \wedge\mathcal{R} \\ \hline\hline
eqf\,(X :: L)\,(X :: R) \longrightarrow eqf\,L\,R & \forall\mathcal{R};\supset\mathcal{R}
\end{array}$$

The proof Ξ_1 is symmetric to Ξ_2, therefore we show here only Ξ_1, which is the following derivation.

$$\begin{array}{cl}
\begin{array}{cc}
\begin{array}{cl} & \\ \hline\hline nat\ M \longrightarrow nat\ (s\,M) & def\mathcal{R}^{=}; init \end{array}
&
\begin{array}{cl}
\Xi_3 & \\
tk\,(s\,M)\,(X :: L)\,(X :: L') \supset tk\,(s\,M)\,(X :: R)\,(X :: L'), nat\ M, tk\,M\,L\,L' \longrightarrow tk\,M\,R\,L' & \\ \hline
tk\,(s\,M)\,(X :: L)\,(X :: L') \equiv tk\,(s\,M)\,(X :: R)\,(X :: L'), nat\ M, tk\,M\,L\,L' \longrightarrow tk\,M\,R\,L' & \\ \hline
\forall P.[\ tk\,(s\,M)\,(X :: L)\,P \equiv tk\,(s\,M)\,(X :: R)\,P], nat\ M, tk\,M\,L\,L' \longrightarrow tk\,M\,R\,L' & \forall\mathcal{L}
\end{array}
\end{array} & \\ \hline
nat\ (s\,M) \supset \forall P.tk\,(s\,M)\,(X :: L)\,P \equiv tk\,(s\,M)\,(X :: R)\,P\ , nat\ M, tk\,M\,L\,L' \longrightarrow tk\,M\,R\,L' & \supset\mathcal{L} \\ \hline
eqf\,(X :: L)\,(X :: R), nat\ M, tk\,M\,L\,L' \longrightarrow tk\,M\,R\,L' & \forall\mathcal{L} \\ \hline
eqf\,(X :: L)\,(X :: R), nat\ M \longrightarrow tk\,M\,L\,L' \supset tk\,M\,R\,L' & \supset\mathcal{R}
\end{array}$$

where Ξ_3 is the derivation

$$\begin{array}{cl}
\begin{array}{cc}
\begin{array}{c} \\ \hline\hline tk\,M\,L\,L' \longrightarrow tk\,(s\,M)\,(X :: L)\,(X :: L') \end{array}
&
\begin{array}{cl}
& \\ \hline
tk\,M\,R\,L', \cdots \longrightarrow tk\,M\,R\,L' & init \\ \hline
tk\,(s\,M)\,(X :: R)\,(X :: L'), nat\ M, tk\,M\,L\,L' \longrightarrow tk\,M\,R\,L' & def\mathcal{L}^{csu}
\end{array}
\end{array} & \\ \hline
tk\,(s\,M)\,(X :: L)\,(X :: L') \supset tk\,(s\,M)\,(X :: R)\,(X :: L'), nat\ M, tk\,M\,L\,L' \longrightarrow tk\,M\,R\,L' & \supset\mathcal{L}
\end{array}$$

The complete proof of sequent (3) is given in Figure 5.4.

∎

Example 5.5. *Co-recursive append.* The co-recursive append requires case analysis on all arguments.

$$\begin{array}{rcl}
\text{coapp}\ L_1\ L_2\ L_3 & \overset{\nu}{=} & (L_1 = \text{nil} \wedge L_2 = \text{nil} \wedge L_3 = \text{nil}) \vee \\
& & (L_1 = \text{nil} \wedge \exists x \exists L_2' \exists L_3'.(L_2 = (x :: L_2') \wedge L3 = (x :: L_3') \\
& & \quad \wedge\ \text{coapp nil}\ L_2'\ L_3') \vee \\
& & (\exists x \exists L_1' \exists L_3'.L_1 = (x :: L_1') \wedge L_3 = (x :: L_3') \\
& & \quad \wedge\ \text{coapp}\ L_1'\ L_2\ L_3').
\end{array}$$

$$
\cfrac{
  \cfrac{
    \cfrac{
      \cfrac{}{\longrightarrow \top}\,\top\mathcal{R}
    }{\longrightarrow nat\;(s\,z)}\,def\mathcal{R}^{=}
    \qquad
    \cfrac{
      \begin{array}{c}\Pi\\ tk\,(s\,z)\,(X :: L_1')\,(X :: \mathrm{nil}) \equiv\\ \qquad tk\,(s\,z)\,L_2\,(X :: \mathrm{nil}),\ eqf(X :: L_1')\,L_2 \longrightarrow \cdots\end{array}
    }{
      \begin{array}{l}\forall L.tk\,(s\,z)\,(X :: L_1')\,L \equiv\\ \qquad tk\,(s\,z)\,L_2\,L,\ eqf(X :: L_1')\,L_2 \longrightarrow \cdots\end{array}
    }\,\forall\mathcal{L}
  }{
    \begin{array}{l}nat\;(s\,z) \supset\\ \quad \forall L.tk\,(s\,z)\,(X :: L_1')\,L \equiv\\ \qquad tk\,(s\,z)\,L_2\,L,\ eqf(X :: L_1')\,L_2 \longrightarrow \cdots\end{array}
  }\,\supset\mathcal{L}
}{
  \cfrac{eqf(X :: L_1')\,L_2, eqf(X :: L_1')\,L_2 \longrightarrow \exists L_2'.L_2 = (X :: L_2') \wedge eqf L_1'\,L_2'}{eqf(X :: L_1')\,L_2 \longrightarrow \exists L_2'.L_2 = (X :: L_2') \wedge eqf L_1'\,L_2'}\,c\mathcal{L}
}\,\forall\mathcal{L}
$$

Π :

$$
\cfrac{
  \cfrac{
    \cfrac{}{\longrightarrow \top}\,\top\mathcal{R}
  }{\longrightarrow tk\,(s\,z)\,(X :: L_1')\,(X :: \mathrm{nil})}\,def\mathcal{R}^{=}
  \qquad
  \begin{array}{c}\Pi_1\\ tk\,(s\,z)\,L_2\,(X :: \mathrm{nil}),\\ eqf(X :: L_1')\,L_2 \quad \longrightarrow \cdots\end{array}
}{
  \cfrac{
    \begin{array}{r}tk\,(s\,z)\,(X :: L_1')\,(X :: \mathrm{nil}) \supset\\ tk\,(s\,z)\,L_2\,(X :: \mathrm{nil}),\\ eqf(X :: L_1')\,L_2\end{array} \longrightarrow \exists L_2'.L_2 = (X :: L_2') \wedge eqf L_1'\,L_2'
  }{
    \begin{array}{r}tk\,(s\,z)\,(X :: L_1')\,(X :: \mathrm{nil}) \equiv\\ tk\,(s\,z)\,L_2\,(X :: \mathrm{nil}),\\ eqf(X :: L_1')\,L_2\end{array} \longrightarrow \exists L_2'.L_2 = (X :: L_2') \wedge eqf L_1'\,L_2'
  }\,\wedge\mathcal{L}
}\,\supset\mathcal{L}
$$

Π_1 :

$$
\cfrac{
  \cfrac{
    \cfrac{
      \cfrac{}{\cdots \longrightarrow X :: L_3 = X :: L_3}\,\mathrm{eq}\mathcal{R}
      \qquad
      \begin{array}{c}\Xi\\ eqf(X :: L_1')\,(X :: L_3) \longrightarrow eqf L_1'\,L_3\end{array}
    }{eqf(X :: L_1')\,(X :: L_3) \quad \longrightarrow (X :: L_3) = (X :: L_3) \wedge eqf L_1'\,L_3}\,\wedge\mathcal{R}
  }{eqf(X :: L_1')\,(X :: L_3) \quad \longrightarrow \exists L_2'.(X :: L_3) = (X :: L_2') \wedge eqf L_1'\,L_2'}\,\exists\mathcal{R}
}{
  \begin{array}{l}tk\,(s\,z)\,L_2\,(X :: \mathrm{nil}),\\ \qquad eqf(X :: L_1')\,L_2\end{array} \quad \longrightarrow \exists L_2'.L_2 = (X :: L_2') \wedge eqf L_1'\,L_2'
}\,def\mathcal{L}^{csu}
$$

Fig. 5.4. A proof of the take lemma.

The corresponding associativity property is stated analogously to the inductive one and as in the inductive case, the main statement reduces to proving the sequent

$$\text{coapp } l_1\ l_2\ l_{12},\ \text{coapp } l_{12}\ l_3\ l_4,\ \text{coapp } l_2\ l_3\ l_{23} \longrightarrow \text{coapp } l_1\ l_{23}\ l_4.$$

We apply the $\nu\mathcal{R}$ rule to coapp $l_1\ l_{23}\ l_4$, using the simulation

$$I = \lambda l_1 \lambda l_2 \lambda l_{12}.\exists l_{23} \exists l_3 \exists l_4.\text{coapp } l_{12}\ l_3\ l_4 \wedge\ \text{coapp } l_2\ l_3\ l_{23} \wedge\ \text{coapp } l_1\ l_{23}\ l_4.$$

Subsequent steps of the proof involve mainly case analysis on coapp $l_{12}\ l_3\ l_4$. As in the inductive case, we have to prove the sub-cases when l_{12} is nil. However, unlike in the former case, case analyses on the arguments of coapp suffices. ∎

5.3 Abstract transition systems

We consider a subset of the calculus of communicating systems (CCS) [39] with a fixed point operator μ. The process expressions of CCS are generated by the following grammar.

$$\mathtt{P} ::= 0 \mid X \mid \alpha.\mathtt{P} \mid \mathtt{P}|\mathtt{Q} \mid \mathtt{P} + \mathtt{Q} \mid \mu X.P$$
$$\alpha ::= x \mid \bar{x} \mid \tau.$$

Processes can make transitions and perform actions. The process 0 is the inert process which does not make any transition. The '.' (dot) operator is the action prefix. The process $\alpha.P$ can perform an action α and changes its state into process P. Actions come in pairs, that is, for every action a, there is a complement action $\bar{a}$. There is a special action, τ, which denotes silent transition and which does not have any complementary action. An action is identified with its double-complement, i.e., $a = \bar{\bar{a}}$. The $+$ is the choice operator and $|$ is the parallel composition. The variable X above denotes a process variable. We consider only closed process expressions, that is, all occurrences of process variables are bound by the μ operator. The μ operator is the fixed point operator. The operational meaning of these operators is given by the operational semantics in Figure 5.5.

$$\frac{}{A.P \xrightarrow{A} P} \qquad \frac{P \xrightarrow{A} P'}{P \mid Q \xrightarrow{A} P' \mid Q} \qquad \frac{Q \xrightarrow{A} Q'}{P \mid Q \xrightarrow{A} P \mid Q'} \qquad \frac{P[\mu X.P/X] \xrightarrow{A} Q}{\mu X.P \xrightarrow{A} Q}$$

$$\frac{P \xrightarrow{A} R}{P + Q \xrightarrow{A} R} \qquad \frac{Q \xrightarrow{A} R}{P + Q \xrightarrow{A} R} \qquad \frac{P \xrightarrow{\bar{A}} R \quad Q \xrightarrow{A} R}{P \mid Q \xrightarrow{\tau} R \mid S}$$

Fig. 5.5. One-step transition for CCS

The process expressions can be translated into higher-order abstract syntax in the following way. We introduce the types p and a to denote processes and actions. The operators above can then be given the following types:

$$\begin{array}{l} \tau : a, \quad \bar{\ } : a \rightarrow a, \quad . : a \rightarrow p \rightarrow p, \\ | : p \rightarrow p \rightarrow p, \quad + : p \rightarrow p \rightarrow p, \quad \mu : (p \rightarrow p) \rightarrow p. \end{array}$$

The one-step transition judgment is encoded as the predicate $. \xrightarrow{\cdot} . : p \rightarrow a \rightarrow p$. The one-step transition semantics for CCS in Figure 5.5 is translated to the definition clauses in Figure 5.6.

$$\begin{array}{rl}
A.P \xrightarrow{A} P & \overset{\mu}{=} \top. \\
P + Q \xrightarrow{A} P' & \overset{\mu}{=} P \xrightarrow{A} P'. \\
P + Q \xrightarrow{A} Q' & \overset{\mu}{=} Q \xrightarrow{A} Q'. \\
P|Q \xrightarrow{A} P'|Q & \overset{\mu}{=} P \xrightarrow{A} P'. \\
P|Q \xrightarrow{A} P|Q' & \overset{\mu}{=} Q \xrightarrow{A} Q'. \\
\mu X.P\,X \xrightarrow{A} Q & \overset{\mu}{=} P\,(\mu X.P\,X) \xrightarrow{A} Q. \\
P|Q \xrightarrow{\tau} P'|Q' & \overset{\mu}{=} \exists A \exists B.comp\ A\ B \wedge P \xrightarrow{A} P' \wedge Q \xrightarrow{B} Q' \\
comp\ A\ \bar{A} & \overset{\mu}{=} \top. \\
comp\ \bar{A}\ A & \overset{\mu}{=} \top.
\end{array}$$

Fig. 5.6. Encoding of one-step transition of CCS in Linc.

Given this encoding, we can easily see that to infer a one-step transition, we only need to use right-introduction rules. In the following example we consider proving a *negative* statement about the transition system, in which the left-rules (in particular the induction rule) are used.

Example 5.6. The process $\mu X.X$ clearly does not make any transition since it has no action-prefix. This is stated formally as the formula

$$\forall A \forall Q.\mu X.X \xrightarrow{A} Q \supset \bot.$$

Finding a proof for this formula reduces to finding a proof for the sequent

$$\mu X.X \xrightarrow{A} Q \longrightarrow \bot.$$

Clearly, applying the $\mathit{def}\mathcal{L}^{csu}$ rule will not lead us to a proof, since it will result in the same sequent in the premise. We need to use induction. The informal proof is done by induction on the structure of the one-step derivation. In each case, pattern matching on the clause suffices to show that $\mu X.X$ does not make any transition. This is proved formally by using the invariant $S = \lambda P \lambda A \lambda Q.P = \mu X.X \supset \bot$.

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\Pi \\ B\,(\mu X.X)\,A\,Q \longrightarrow \bot\end{matrix}}{B\,P\,A\,Q, P = \mu X.X \longrightarrow \bot}\;\mathrm{eq}\mathcal{L}}{B\,P\,A\,Q \longrightarrow P = \mu X.X \supset \bot}\;\supset\mathcal{R} \qquad \dfrac{\dfrac{}{\longrightarrow \mu X.X = \mu X.X}\;\mathrm{eq}\mathcal{R} \qquad \dfrac{}{\bot \longrightarrow \bot}\;\bot\mathcal{L}}{\mu X.X = \mu X.X \supset \bot \longrightarrow \bot}\;\supset\mathcal{L}}{\mu X.X \xrightarrow{A} Q \longrightarrow \bot}\;\mu\mathcal{L}$$

The formula $B\,P\,A\,Q$ denotes the body of the definition clause corresponding to the patterned definition in Figure 5.6. The premise derivation Π is constructed by a series of case analyses on the definition of one-step transition. Most cases fail to pattern-match, and the only case that successfully pattern-match yields the following sequent (using the sixth clause in Figure 5.6)

$$\mu X.X = \mu X.X \supset \bot \longrightarrow \bot$$

which is trivially provable. ∎

5.3.1 Bisimulation

An important notion in reasoning about processes is that of *bisimulation*. That is, given two processes P and Q, any transition of P can be simulated by a transition of Q such that their continuations continue to simulate each other. If we define the behavior of a process as the actions it is capable of performing (at current state or some “future” state), then bisimulation gives us a way to say when two processes are behaviorally equivalent. The original motivation of studying bisimulation [39] is precisely this, that it will provide us with an equational theory of behaviors. We consider here encoding simulation (which is, roughly speaking, one-half of bisimulation) and bisimulation. Both are encoded as coinductive definitions given in Figure 5.7. In the following we show a simple example of checking simulation between processes. In Chapter 6 we shall consider more comprehensive examples of bisimulation in π-calculus, an extension of CCS.

Example 5.7. Consider proving the simulation $\mathit{sim}\ (\ \mu x.a.x)\ (\mu x.(a.x \mid a.x))$. This is proved using the following simulation predicate

$$S \coloneqq \lambda P \lambda Q.(P = \mu x.a.x) \wedge \exists Q'.Q \xrightarrow{a} Q \mid Q'.$$

$$\begin{array}{lcl} \mathit{sim}\ P\ \ Q & \stackrel{\triangle}{=} & \forall A \forall P'.P \xrightarrow{A} P' \supset \exists Q'.Q \xrightarrow{A} Q' \wedge \mathit{sim}\ P\ '\ Q' \\ \mathit{bisim}\ P\ \ Q & \stackrel{\triangle}{=} & [\forall A \forall P'.P \xrightarrow{A} P' \supset \exists Q'.Q \xrightarrow{A} Q' \wedge \mathit{bisim}\ P\ '\ Q'] \wedge \\ & & [\forall A \forall Q'.Q \xrightarrow{A} Q' \supset \exists P'.P \xrightarrow{A} P' \wedge \mathit{bisim}\ Q\ '\ P'] \end{array}$$

Fig. 5.7. Simulation and bisimulation for CCS

Intuitively, it is enough to characterize $\mu x.(a.x \mid a.x)$ by saying that it preserves the capability of making an a-transition whenever it makes an a-transition. The proof is as follows. Let $R = \mu x.a.x$ and $T = \mu x.(a.x \mid a.x)$.

$$\dfrac{\begin{array}{c}\Pi_1 \\ \longrightarrow R = \mu x.a.x \wedge \exists Q'.T \xrightarrow{a} T \mid Q'\end{array} \quad \begin{array}{c}\Pi_2 \\ P = \mu x.a.x \wedge \exists Q'.(Q \xrightarrow{a} Q \mid Q') \longrightarrow B\ S\ P\ Q\end{array}}{\longrightarrow \mathit{sim}\ R\ T}\ \nu\mathcal{R}$$

where $B\ S\ P\ Q$ is the formula

$$\forall A \forall P'.P \xrightarrow{A} P' \supset \exists Q_1.Q \xrightarrow{A} Q_1 \wedge [P' = \mu x.a.x \wedge \exists Q_2.Q_1 \xrightarrow{a} Q_1 \mid Q_2]$$

The first premise is provable by instantiating Q' with $a.T$. The second premise derivation reduces to the following derivation, after some simplification steps.

$$\dfrac{(Q \xrightarrow{a} Q \mid Q') \longrightarrow (Q \xrightarrow{a} Q \mid Q') \wedge [\mu x.a.x = \mu x.a.x \wedge \exists Q_2.((Q \mid Q') \xrightarrow{a} (Q \mid Q') \mid Q_2)]}{(Q \xrightarrow{a} Q \mid Q') \longrightarrow \exists Q_1.(Q \xrightarrow{a} Q_1) \wedge [\mu x.a.x = \mu x.a.x \wedge \exists Q_2.(Q_1 \xrightarrow{a} Q_1 \mid Q_2)]}\ \exists\mathcal{R}$$

We show here the subderivation for the third conjunct, the others are trivially provable.

$$\dfrac{\dfrac{\dfrac{}{(Q \xrightarrow{a} Q \mid Q') \longrightarrow (Q \xrightarrow{a} (Q \mid Q'))}\ \mathit{init}}{(Q \xrightarrow{a} Q \mid Q') \longrightarrow ((Q \mid Q') \xrightarrow{a} (Q \mid Q') \mid Q')}\ \mathit{def}\mathcal{R}}{(Q \xrightarrow{a} Q \mid Q') \longrightarrow \exists Q_2.((Q \mid Q') \xrightarrow{a} (Q \mid Q') \mid Q_2)}\ \exists\mathcal{R}$$

■

5.4 Object logic

In this section we consider an encoding of a simple object-logic, that is, Horn logic with universal quantification. The static structure of the object-level quantification is encoded via abstraction at the meta-level, similar to our previous encoding of the μ-operator. However, in the previous encoding we essentially deal only with closed terms,

$$\begin{array}{rcl}
pv\ \hat{\top} & \stackrel{\triangle}{=} & \top \\
pv\ (G \mathbin{\&} G') & \stackrel{\triangle}{=} & pv\ G \wedge pv\ G' \\
pv\ (\hat{\forall}\, G) & \stackrel{\triangle}{=} & \nabla x. pv\ (Gx) \\
pv\ (\hat{\exists}\, G) & \stackrel{\triangle}{=} & \exists x. pv\ (Gx) \\
pv\ A & \stackrel{\triangle}{=} & atom\ A \wedge \exists D. prog\ D \wedge bc(D, A) \\
bc(A, A) & \stackrel{\triangle}{=} & atom\ A \\
bc(G \supset D, A) & \stackrel{\triangle}{=} & bc(D, A) \wedge pv\ G \\
bc(\hat{\forall}\, D, A) & \stackrel{\triangle}{=} & \exists t.\ bc(D\ t, A)
\end{array}$$

Fig. 5.8. Interpreter for an object-level logic.

and hence there is no need to interpret object-level variables at the meta-level. With object-logic universal quantification the issue is different, since the universally quantified goal can be replaced by another goal with (object-level) eigenvariables. We illustrate an example in which we use ∇ to interpret the object-logic eigenvariables and show that the use of ∇ captures the intensional aspect of object-logic eigenvariables, that is, to act as fresh constant in (object-logic) proof construction.

The provability of the object-logic is encoded using two predicates: one for encoding the first-order provability and the other for backchaining. The completeness of this proof system follows from the completeness of uniform provability for intuitionistic logic. We introduce the type *obj* to denote object-logic formulas. The object-logic connectives are encoded as the constants

$$\hat{\top} : obj, \quad \& : obj \to obj \to obj, \quad \Rightarrow : obj \to obj \to obj$$
$$\hat{\forall} : (i \to obj) \to obj, \quad \hat{\exists} : (i \to obj) \to obj$$

which denote, respectively, object-level true, conjunction, implication, universal quantifier and existential quantifier. The object-level implication is only used in the program clause and never in the goal. The type i ranges over first-order object-level terms. The predicate $pv\ \cdot$ of type $obj \to o$ is used to indicate first-order provability and $bc(\cdot, \cdot)$ of type $obj \to obj \to o$ is used to indicate backchaining. The definition clauses in Figure 5.8 encodes provability for a first-order logic programming language that is restricted to $\mathsf{hc}^{\forall}$ and is parametrized by the predicates *atom* (describing object-level atomic formulas) and *prog* (describing object-level logic programs clauses). We illustrate in the following example reasoning about provability of this object-logic in Linc.

Example 5.8. We recall the motivating example given in [28, 38]. Consider the problem of proving the formula

$$\forall u \forall v [q\ \langle u, t_1 \rangle\ \langle v, t_2 \rangle\ \langle v, t_3 \rangle],$$

where q is a three place predicate, $\langle\cdot,\cdot\rangle$ is used to form pairs, t_1 and t_2 are some first-order terms, and the only assumptions for the predicate q are the (universal closure of the) three atomic formulas: $q\ X\ X\ Y$, $q\ X\ Y\ X$ and $q\ Y\ X\ X$. Clearly, this query succeeds only if terms t_2 and t_3 are equal. Notice that while the object-level logic here is $\mathsf{hc}^{\forall}$ (since our motivating example is concerned with the provability of a universally quantified formula), the meta-level definition is hc^{∇}.

The query that captures our intended example is the following formula

$$\forall x, y, z[pv\ (\hat{\forall} u \hat{\forall} v[q\ \langle u, x\rangle\ \langle v, y\rangle\ \langle v, z\rangle]) \supset y = z]$$

along with the definition consisting of the clauses in Figure 5.8 and the following object-logic program clauses.

$$\begin{array}{l} prog\ (\hat{\forall} x \hat{\forall} y.q\ x\ x\ y) \stackrel{\triangle}{=} \top. \\ prog\ (\hat{\forall} x \hat{\forall} y.q\ x\ y\ x) \stackrel{\triangle}{=} \top. \\ prog\ (\hat{\forall} x \hat{\forall} y.q\ y\ x\ x) \stackrel{\triangle}{=} \top. \\ atom\ \ (q\ X\ Y\ Z) \stackrel{\triangle}{=} \top. \end{array}$$

Attempting a proof of this formula leads to the following sequent (after applying some right rules and a pair of $def\mathcal{L}^{csu}$ and $\nabla\mathcal{L}$ rules):

$$X, Y, Z\,;\ (s, r) \triangleright pv\ (q\ \langle s, X\rangle\ \langle r, Y\rangle\ \langle r, Z\rangle) \longrightarrow \triangleright Y = Z.$$

A series of $def\mathcal{L}^{csu}$ rules will now need to be applied in order to work through the encoding of the object-level interpreter. In the end, three separate unification problems will be attempted, one for each of the three ways to prove the predicate q. In particular, the $def\mathcal{L}^{csu}$ rule will attempt to unify the term $\lambda s\lambda r.(q\ \langle s, X\rangle\ \langle r, Y\rangle\ \langle r, Z\rangle)$ with each of the following three terms:

$$\begin{array}{c} \lambda s\lambda r.(q\ (X'\ s\ r)\ (X'\ s\ r)\ (Y'\ s\ r)) \\ \lambda s\lambda r.(q\ (X'\ s\ r)\ (Y'\ s\ r)\ (X'\ s\ r)) \\ \lambda s\lambda r.(q\ (Y'\ s\ r)\ (X'\ s\ r)\ (X'\ s\ r)) \end{array}$$

The first two unification problems fail and hence the corresponding occurrences of $def\mathcal{L}^{csu}$ succeed. The third of these unification problems is solvable, however, with X' instantiated to $\lambda s\lambda r.\langle r, Z\rangle$, Y' instantiated to $\lambda s\lambda r.\langle s, Z\rangle$, Y instantiated to Z (or vice versa), and X uninstantiated. As a result, this third premise is the sequent $\cdot\,;\,\cdot \longrightarrow Y = Y$, which is provable using eq$\mathcal{R}$.

The more common approach to encoding object-logic provability into a meta-logic uses the meta-level universal quantifier instead of the ∇ for the clause encoding the provability of object-level universal quantification: that is, the clause

$$pv\ (\hat{\forall} x.G\ x) \stackrel{\triangle}{=} \forall x[pv\ (G\ x)].$$

is used instead. In this case, attempting a proof of this formula reduces to an attempt to prove the sequent

$$X, Y, Z\,;\, \triangleright pv\ (q\ \langle s_1, X\rangle\ \langle s_2, Y\rangle\ \langle r, Z\rangle) \longrightarrow \triangleright Y = Z,$$

and were s_1 and s_2 are two terms. To complete the proof, these two terms must be chosen to be different. While this sequent can be proved, doing so requires the assumption that there are two such distinct terms (the domain is non-empty and not a singleton). Our encoding using ∇ allows this (meta-level) proof to be completed in a more natural way without this assumption. ∎

The extensional nature of $\hat{\forall}$, that is, $pv\ \hat{\forall} x.P\,x$ implies $pv\ P\,t$ for arbitrary term t, can be proved indirectly in Linc as a consequence of the meta-properties of Linc.

Proposition 5.9. *If $pv\ \hat{\forall} x.P\,x$ is provable then $\forall t.pv\ P\,t$ is provable.*

Proof Proving the goal $pv\ \hat{\forall} x.P\,x$ reduces to proving the subgoal $\nabla x.pv\ P\,x$. Note that since the definition for pv is hc^{∇}, by Proposition 3.28, in proving hc^{∇} goals, ∇ can be interchanged with $\forall$ without affecting provability. Therefore if we have a proof of $\nabla x.pv\ P\,x$ then we also have a proof of $\forall x.pv\ P\,x$. ∎

However, we cannot directly prove the implication

$$pv\ \hat{\forall} x.P\,x \supset \forall x.pv\ P\,x \tag{5.3}$$

in Linc, even with induction. To see why, consider the following proof attempt with induction.

$$\dfrac{\dfrac{\dfrac{\dfrac{\cdots \qquad x \triangleright S\,(P\,x) \longrightarrow \forall x.pv\ P\,x}{x \triangleright pv\ P\,x \longrightarrow \forall x.pv\ P\,x}\,\mu\mathcal{L}}{\nabla x.pv\ P\,x \longrightarrow \forall x.pv\ P\,x}\,\nabla\mathcal{L}}{pv\ \hat{\forall} x.P\,x \longrightarrow \forall x.pv\ P\,x}\,def\mathcal{L}^{csu}}{\longrightarrow pv\ \hat{\forall} x.P\,x \supset \forall x.pv\ P\,x}\,\supset\mathcal{R}$$

Note that the first three rules used in the above derivation are all asynchronous rules, therefore any proof of the formula (5.3) can be transformed to a proof which ends in those rules. However such a proof cannot exist since on the left we have an extra local signature guarding the formula $S\,(P\,x)$ which does not match the local signature of the right-hand side of the sequent, regardless of what invariant S we choose. This example illustrates the fact that our current formulation of induction and co-induction do not interact much with ∇.

5.5 The lazy λ-calculus

We consider an untyped version of the pure λ-calculus with lazy evaluation [1], following the usual HOAS style, i.e., object-level λ-operator and application are encoded as constants lam : $(tm \rightarrow tm) \rightarrow tm$ and @ : $tm \rightarrow tm \rightarrow tm$, where tm is the syntactic category of object-level λ-terms. The evaluation relation is encoded as the following inductive definition

$$\begin{aligned}\mathrm{lam}\,M \Downarrow \mathrm{lam}\,M &\overset{\mu}{=} \top. \\ M\,@\,N \Downarrow T &\overset{\mu}{=} M \Downarrow (\mathrm{lam}\,P) \wedge (P\,N) \Downarrow T.\end{aligned}$$

Notice that object-level substitution is realized via β-reduction in the meta-logic.

Applicative simulation

We show some simple properties about applicative simulation of λ-expressions in this language. Simulation is encoded as the (stratified) co-inductive definition

$$\mathit{sim}\ R\ S \overset{\nu}{=} \forall T.R \Downarrow \mathrm{lam}\,T \supset \exists U.S \Downarrow \mathrm{lam}\,U \wedge \forall P.\mathit{sim}\ (T\,P)\ (U\,P).$$

Consider the reflexivity property of simulation, i.e., $\forall s.\mathit{sim}\ s\ s$. This is proved co-inductively by using the simulation $\lambda x \lambda y.[x = y]$. After applying $\forall\mathcal{R}$ and $\nu\mathcal{R}$, it remains to prove the sequents $\longrightarrow [s = s]$, and

$$x = y \longrightarrow \forall x_1.x \Downarrow \mathrm{lam}\,x_1 \supset (\exists x_2.y \Downarrow \mathrm{lam}\,x_2 \wedge \forall x_3.(x_1\,x_3) = (x_2\,x_3)) \ .$$

The first sequent is provable by an application of eq$\mathcal{R}$ rule. The second sequent is proved as follows.

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{z \Downarrow \mathrm{lam}\,x_1 \longrightarrow z \Downarrow \mathrm{lam}\,x_1}\ \mathit{init} \qquad \cfrac{\cfrac{}{z \Downarrow \mathrm{lam}\,x_1 \longrightarrow (x_1\,x_3) = (x_1\,x_3)}\ \mathrm{eq}\mathcal{R}}{z \Downarrow \mathrm{lam}\,x_1 \longrightarrow \forall x_3.(x_1\,x_3) = (x_1\,x_3)}\ \forall\mathcal{R}}{z \Downarrow \mathrm{lam}\,x_1 \longrightarrow (z \Downarrow \mathrm{lam}\,x_1 \wedge \forall x_3.(x_1\,x_3) = (x_1\,x_3))}\ \wedge\mathcal{R}}{z \Downarrow \mathrm{lam}\,x_1 \longrightarrow (\exists x_2.z \Downarrow \mathrm{lam}\,x_2 \wedge \forall x_3.(x_1\,x_3) = (x_2\,x_3))}\ \exists\mathcal{R}}{x = y, x \Downarrow \mathrm{lam}\,x_1 \longrightarrow (\exists x_2.y \Downarrow \mathrm{lam}\,x_2 \wedge \forall x_3.(x_1\,x_3) = (x_2\,x_3))}\ \mathrm{eq}\mathcal{L}}{x = y \longrightarrow x \Downarrow \mathrm{lam}\,x_1 \supset (\exists x_2.y \Downarrow \mathrm{lam}\,x_2 \wedge \forall x_3.(x_1\,x_3) = (x_2\,x_3))}\ \supset\mathcal{R}}{x = y \longrightarrow \forall x_1.x \Downarrow \mathrm{lam}\,x_1 \supset (\exists x_2.y \Downarrow \mathrm{lam}\,x_2 \wedge \forall x_3.(x_1\,x_3) = (x_2\,x_3))}\ \forall\mathcal{R}$$

The transitivity property is expressed as the formula

$$\forall r \forall s \forall t.\mathit{sim}\ r\ s \wedge \mathit{sim}\ s\ t \supset \mathit{sim}\ r\ t.$$

Its proof involves co-induction on *sim r t* with the following simulation

$$S := \lambda u \lambda v.\exists w.\mathit{sim}\ u\ w \wedge \mathit{sim}\ w\ v,$$

followed by case analyses (i.e., *def*$\mathcal{L}$ and eq$\mathcal{L}$ rules) on *sim r s* and *sim s t*. The rest of the proof is basically a series of manipulation of logical connectives.

$$\cfrac{\cfrac{\cfrac{\Pi}{S\,u\,v \longrightarrow B\,S\,u\,v} \qquad \cfrac{\cfrac{\cfrac{}{\mathit{sim}\ r\ s, \mathit{sim}\ s\ t \longrightarrow \mathit{sim}\ r\ s}\ \mathit{init} \qquad \cfrac{}{\mathit{sim}\ r\ s, \mathit{sim}\ s\ t \longrightarrow \mathit{sim}\ s\ t}\ \mathit{init}}{\mathit{sim}\ r\ s, \mathit{sim}\ s\ t \longrightarrow \mathit{sim}\ r\ s \wedge \mathit{sim}\ s\ t}\ \wedge\mathcal{R}}{\mathit{sim}\ r\ s, \mathit{sim}\ s\ t \longrightarrow \exists w.\mathit{sim}\ r\ w \wedge \mathit{sim}\ w\ t}\ \exists\mathcal{R}}{\mathit{sim}\ r\ s, \mathit{sim}\ s\ t \longrightarrow \mathit{sim}\ r\ t}\ \nu\mathcal{R}}{\longrightarrow \forall r \forall s \forall t.\mathit{sim}\ r\ s \wedge \mathit{sim}\ s\ t \supset \mathit{sim}\ r\ t}\ \forall\mathcal{R}; \supset\mathcal{R}$$

$B\,S\,u\,v$ is the formula

$$\forall x.u \Downarrow \mathrm{lam}\, x \supset \exists y.v \Downarrow \mathrm{lam}\, y \wedge \forall z.S\,(x\,z)\,(y\,z).$$

The derivation Π is given in Figure 5.9.

Divergence

The existence of a divergent term can be proved formally in Linc, using the following encoding of divergence.

$$\begin{array}{lcl}\mathrm{divrg}\ T & \stackrel{\nu}{=} & (\exists T_1 \exists T_2.T = (T_1 @ T_2) \wedge \mathrm{divrg}\ T_1) \vee \\ & & (\exists T_1 \exists T_2.T = (T_1 @ T_2) \wedge \exists E.T_1 \Downarrow \mathrm{lam}\, E \wedge \mathrm{divrg}\ (E\,T_2)).\end{array}$$

Let Ω be the term $(\mathrm{lam}\, x.(x @ x)) @ (\mathrm{lam}\, x.(x @ x))$. We show that divrg Ω holds. The proof is straightforward by co-induction using the simulation $S := \lambda s.s = \Omega$. Applying the $\nu\mathcal{R}$ produces the sequents $\longrightarrow \Omega = \Omega$ and $T = \Omega \longrightarrow S_1 \ \vee \ S_2$ where

$$S_1 := \exists T_1 \exists T_2.T = (T_1 @ T_2) \wedge (S\,T_1),\ \text{and}$$

$$S_2 := \exists T_1 \exists T_2.T = (T_1 @ T_2) \wedge \exists E.T_1 \Downarrow \mathrm{lam}\, E \wedge S\,(E\,T_2).$$

Clearly, only the second disjunct is provable, i.e., by instantiating T_1 and T_2 with the same term $\mathrm{lam}\, x.(x @ x)$, and E with the function $\lambda x.(x @ x)$.

5.6 Conclusion and related work

We have seen several examples of reasoning about logical specifications in Linc. Both first-order and higher-order encodings are shown. We have shown that the natural number induction in $FO\lambda^{\Delta\mathbb{N}}$ can be derived in Linc. This implies that applications and examples that were previously done in $FO\lambda^{\Delta\mathbb{N}}$, e.g., [28, 31, 57], can be carried out in Linc without any essential modification. For example, the encoding of CCS in Section 5.3 has been studied in [31]. However, in their encoding simulation and bisimulation are encoded indirectly via natural number induction because $FO\lambda^{\Delta\mathbb{N}}$ does not support co-induction. In our example, simulation and bisimulation are encoded directly as co-inductive definitions. Part of Section 5.5 has appeared in [42] and Section 5.4 has appeared in [38]. We note that the current formulation of (co)-induction does not allow certain proofs which require reference to local signatures, as we have seen in the object-logic encoding. This limitation implies that certain forms of induction over higher-order abstract syntax do not benefit from the presence of ∇. The subject-reduction theorem for functional languages, for example, does not admit simple formalization in Linc, although it can still be done via the encoding style as in [12, 28]. We shall see in Chapter 6, there are still interesting uses of ∇ in reasoning with (co-)induction.

$$
\begin{array}{cl}
\dfrac{}{\longrightarrow \mathrm{lam}\,x \Downarrow \mathrm{lam}\,x}\,def\mathcal{R}
\quad
\begin{array}{cl}
\Pi_1 & \\
\forall n.sim\ (x\,n)\ (m\,n), & \\
(\mathrm{lam}\,m) \Downarrow (\mathrm{lam}\,m) \supset & \\
\exists p.v \Downarrow \mathrm{lam}\,p \wedge \forall n.sim\ (m\,n)\ (p\,n) \quad \longrightarrow \cdots & \\
\hline
\forall n.sim\ (x\,n)\ (m\,n), & \forall\mathcal{L} \\
\forall r.(\mathrm{lam}\,m) \Downarrow (\mathrm{lam}\,r) \supset & \\
\exists p.v \Downarrow \mathrm{lam}\,p \wedge \forall n.sim\ (r\,n)\ (p\,n) \quad \longrightarrow \cdots & \\
\hline
\forall n.sim\ (x\,n)\ (m\,n), sim\ (\mathrm{lam}\,m)\ v \longrightarrow \cdots & def\mathcal{L} \\
\hline
w \Downarrow \mathrm{lam}\,m, \forall n.sim\ (x\,n)\ (m\,n), sim\ w\ v \longrightarrow \cdots & def\mathcal{L}^{csu} \\
\hline\hline
\exists m.w \Downarrow \mathrm{lam}\,m \wedge \forall n.sim\ (x\,n)\ (m\,n), sim\ w\ v \longrightarrow \cdots & \exists\mathcal{L}; \wedge\mathcal{L}
\end{array} & \\
\hline
\mathrm{lam}\,x \Downarrow \mathrm{lam}\,x \supset \exists m.w \Downarrow \mathrm{lam}\,m \wedge \forall n.sim\ (x\,n)\ (m\,n), sim\ w\ v \longrightarrow \cdots & \supset \mathcal{L} \\
\hline
\forall r.\mathrm{lam}\,x \Downarrow \mathrm{lam}\,r \supset \exists m.w \Downarrow \mathrm{lam}\,m \wedge \forall n.sim\ (r\,n)\ (m\,n), sim\ w\ v \longrightarrow \cdots & \forall\mathcal{L} \\
\hline
sim\ (\mathrm{lam}\,x)\ w, sim\ w\ v \longrightarrow \exists y.v \Downarrow \mathrm{lam}\,y \wedge \forall z.S\,(x\,z)\,(y\,z) & def\mathcal{L} \\
\hline\hline
S\,(\mathrm{lam}\,x)\,v \longrightarrow \exists y.v \Downarrow \mathrm{lam}\,y \wedge \forall z.S\,(x\,z)\,(y\,z) & \exists\mathcal{L}; \wedge\mathcal{L} \\
\hline
S\,u\,v, u \Downarrow \mathrm{lam}\,x \longrightarrow \exists y.v \Downarrow \mathrm{lam}\,y \wedge \forall z.S\,(x\,z)\,(y\,z) & def\mathcal{L}^{csu} \\
\hline
S\,u\,v \longrightarrow \forall x.u \Downarrow \mathrm{lam}\,x \supset \exists y.v \Downarrow \mathrm{lam}\,y \wedge \forall z.S\,(x\,z)\,(y\,z) & \forall\mathcal{R}; \supset\mathcal{R}
\end{array}
$$

Π_1 :

$$
\begin{array}{cl}
\dfrac{}{\longrightarrow (\mathrm{lam}\,m) \Downarrow (\mathrm{lam}\,m)}\,def\mathcal{R}
\quad
\begin{array}{cl}
\dfrac{}{\cdots \longrightarrow \mathrm{lam}\,p \Downarrow \mathrm{lam}\,p}\,def\mathcal{R} \quad \begin{array}{c} \Pi_2 \\ \cdots \longrightarrow \forall z.S\,(x\,z)\,(p\,z) \end{array} & \\
\hline
\forall n.sim\ (x\,n)\ (m\,n), \qquad \mathrm{lam}\,p \Downarrow \mathrm{lam}\,p \wedge & \\
\forall n.sim\ (m\,n)\ (p\,n) \quad \longrightarrow \quad \forall z.S\,(x\,z)\,(p\,z) & \\
\hline
\forall n.sim\ (x\,n)\ (m\,n), \qquad \exists y.\mathrm{lam}\,p \Downarrow \mathrm{lam}\,y \wedge & \exists\mathcal{R} \\
\forall n.sim\ (m\,n)\ (p\,n) \quad \longrightarrow \quad \forall z.S\,(x\,z)\,(y\,z) & \\
\hline
\forall n.sim\ (x\,n)\ (m\,n), & def\mathcal{L}^{csu} \\
v \Downarrow \mathrm{lam}\,p, \qquad \exists y.v \Downarrow \mathrm{lam}\,y \wedge & \\
\forall n.sim\ (m\,n)\ (p\,n) \quad \longrightarrow \quad \forall z.S\,(x\,z)\,(y\,z) & \\
\hline
\forall n.sim\ (x\,n)\ (m\,n), \qquad \exists y.v \Downarrow \mathrm{lam}\,y \wedge & \wedge\mathcal{L} \\
v \Downarrow \mathrm{lam}\,p \wedge \forall n.sim\ (m\,n)\ (p\,n) \quad \longrightarrow \quad \forall z.S\,(x\,z)\,(y\,z) & \\
\hline
\forall n.sim\ (x\,n)\ (m\,n), \qquad \exists y.v \Downarrow \mathrm{lam}\,y \wedge & \exists\mathcal{L} \\
\exists p.v \Downarrow \mathrm{lam}\,p \wedge \forall n.sim\ (m\,n)\ (p\,n) \quad \longrightarrow \quad \forall z.S\,(x\,z)\,(y\,z) &
\end{array} & \\
\hline
(\mathrm{lam}\,m) \Downarrow (\mathrm{lam}\,m) \supset & \supset \mathcal{L} \\
\forall n.sim\ (x\,n)\ (m\,n), \quad \exists p.v \Downarrow \mathrm{lam}\,p \wedge \forall n.sim\ (m\,n)\ (p\,n) \quad \longrightarrow \exists y.v \Downarrow \mathrm{lam}\,y \wedge \forall z.S\,(x\,z)\,(y\,z) &
\end{array}
$$

Π_2 :

$$
\begin{array}{cl}
\dfrac{}{\cdots \longrightarrow sim\ (x\,z)\ (m\,z)}\,init \quad \dfrac{}{\cdots \longrightarrow sim\ (m\,z)\ (p\,z)}\,init & \\
\hline
sim\ (x\,z)\ (m\,z), sim\ (m\,z)\ (p\,z) \longrightarrow sim\ (x\,z)\ (m\,z) \wedge sim\ (m\,z)\ (p\,z) & \wedge\mathcal{R} \\
\hline\hline
\forall n.sim\ (x\,n)\ (m\,n), \forall n.sim\ (m\,n)\ (p\,n) \longrightarrow sim\ (x\,z)\ (m\,z) \wedge sim\ (m\,z)\ (p\,z) & \forall\mathcal{L} \\
\hline
\forall n.sim\ (x\,n)\ (m\,n), \forall n.sim\ (m\,n)\ (p\,n) \longrightarrow \exists q.sim\ (x\,z)\ q \wedge sim\ q\ (p\,z) & \exists\mathcal{R} \\
\hline
\forall n.sim\ (x\,n)\ (m\,n), \forall n.sim\ (m\,n)\ (p\,n) \longrightarrow \forall z.\exists q.sim\ (x\,z)\ q \wedge sim\ q\ (p\,z) & \forall\mathcal{R}
\end{array}
$$

Fig. 5.9. A proof of transitivity of applicative simulation.

Chapter 6

Encoding π-calculus

In this chapter we consider an encoding of the operational semantics of the π-calculus [40]. Central to the calculus is the notion of *names* which represent communication links between processes. The dynamics of names in π-calculus, i.e., the creation of new names and the changes in the scoping of names as the process evolves, construe a notion of mobility in which the movement of processes is interpreted as the changing of communication links. The notion of new name generation is not unique to π-calculus. It also appears in the area of, for example, security protocol, where the creation of fresh value (*nonces*) is common in the specification of the protocols. A common approach to encoding fresh names generation in logic is via the use of eigenvariable (or universal quantifier). This approach is taken in the encoding of π-calculus in [34], nonces in security protocols [6], references to locations in imperative programming languages [8, 35] and constructors hidden within abstract data types [32]. In these works, the main interest is in encoding *computation*, while we are also interested in doing *reasoning* about computation. This approach does not scale easily to suit our purpose because of the following reason. Suppose we would like to verify that a certain computation, F, follows from some other computation Gxy under the assumption that x and y are different fresh names. In other words, we would like to say that if Gxy holds genericly (with respect to x and y) then F is true. This type of statement often occurs in checking bisimulation where the closure with respect to one-step transitions is a requirement. Using the $\forall$ encoding above we would have to prove the sequent $\forall x \forall y. Gxy \longrightarrow F$. However, the assumption that x and y are distinct are not enforced in this sequent, since there might be a proof in which x and y are identified. Thus we do not capture faithfully the intended assumption on genericity of x and y, at least not without additional non-logical encoding. Our solution to this problem is to use ∇, instead of $\forall$, to encode fresh names generation. Using this new encoding, the above sequent becomes $\nabla x \nabla y. Gxy \longrightarrow F$, which can be proved only by instantiating x and y to different names.

We are mainly interested in reasoning about the bisimilarity and equivalence of processes. We first consider an encoding of the finite π-calculus (all execution of processes always terminate) and study the bisimulation relations and congruence in this subcalculus. The full π-calculus which includes non-terminating processes is studied later in the chapter. In Section 6.1 we introduce the syntax of π-calculus, for the finite case, and its encoding into higher-order abstract syntax. The encoding of the operational semantics of one-step transitions is given in Section 6.2 in which the adequacy of the encoding is also shown. Section 6.3 studies the encoding of a variant of bisimulation, the strong late bisimulation. This notion of bisimulation gives rise to the strong congruence relation, which is the subject of Section 6.4. In Section 6.5 we consider the π-calculus with replications, which allows us to express some non-terminating processes. Here we see that co-induction proof method is needed to prove the bisimilarity of processes. We show an example on how the informal co-inductive proof can be carried over to formal proofs in Linc. The correctness of the encoding of bisimulation and congruence for the full π-calculus is also shown. Section 6.6 concludes this chapter with some conjectures and related work.

6.1 Finite late π-calculus

In this section, we consider the finite late π-calculus as defined in [40], that is, the fragment of π-calculus without recursion (or replication). The main emphasis in this section is thus on the treatment of names in logic. The syntax of processes is defined as follows

$$\mathtt{P} ::= 0 \mid \bar{x}y.\mathtt{P} \mid x(y).\mathtt{P} \mid \tau.\mathtt{P} \mid (x)\mathtt{P} \mid [x = y]\mathtt{P} \mid \mathtt{P}|\mathtt{Q} \mid \mathtt{P} + \mathtt{Q}.$$

We use the notation $\mathtt{P}$, $\mathtt{Q}$, $\mathtt{R}$, $\mathtt{S}$ and $\mathtt{T}$ to denote processes. Names are denoted by a, b, c, d, x, y, z. The occurrence of y in the process $x(y).\mathtt{P}$ and $(y)\mathtt{P}$ is a binding occurrence, with $\mathtt{P}$ as its scope. The set of free names in $\mathtt{P}$ is denoted by $\mathrm{fn}(\mathtt{P})$, the set of bound names is denoted by $\mathrm{bn}(\mathtt{P})$. We write $\mathrm{n}(\mathtt{P})$ for the set $\mathrm{fn}(\mathtt{P}) \cup \mathrm{bn}(\mathtt{P})$.

The one-step transition in π-calculus is denoted by $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$, where $\mathtt{P}$ and $\mathtt{Q}$ are processes and α is an action. The kinds of actions are *the silent action* τ, *the free output action* $\bar{x}y$, *the bound input action* $x(y)$ and *the bound output action* $\bar{x}(y)$. The name y in $x(y)$ and $\bar{x}(y)$ is a binding occurrence. Just like we did with processes, we use $\mathrm{fn}(\alpha)$, $\mathrm{bn}(\alpha)$ and $\mathrm{n}(\alpha)$ to denote free names, bound names, and names in α. An action without binding occurrences of names is a *free action*, otherwise it is a *bound action*.

We encode the syntax of process expressions using higher-order syntax as follows. We shall require three primitive syntactic categories: n for names, p for processes, and a for actions, and the constructors corresponding to the operators in π-calculus. We assume an infinite set of constants of type n. Other constants and their types are given in Table 6.1. We abbreviate $\nu\lambda x.P$ as simply $\nu x.P$. Notice that when τ is written as a prefix, it has type $p \to p$, and when it is written as an action, it has type a.

The one-step judgment of π-calculus is given the type o' in Linc. We distinguish two different kinds of one-step transitions: those with free actions and those with bound actions. The former is encoded using the constant $\cdot \xrightarrow{\cdot} \cdot$ of type $p \to a \to p \to o'$, and the latter using the constant $\cdot \overset{\cdot}{\rightharpoonup} \cdot$ of type $p \to (n \to a) \to (n \to p) \to o'$. We need an additionaly constant $\diamond : nt \to o' \to o$ to coerce the type o' into o. The type nt denotes natural numbers, which we use here to distinguish the two kinds of one-step transitions. Formally speaking, the free one-step transition $P \xrightarrow{\bar{x}z} Q$ is translated to $\diamond\ 0\ (P \xrightarrow{\bar{x}z} Q)$ and the bound transition $P \xrightarrow{x(y)} Q$ is translated to $\diamond\ 1\ (P \overset{\uparrow x}{\rightharpoonup} \lambda y.Q)$. However, to simplify the presentation, when writing down these translations we shall leave the $\diamond$ and the indices 0 and 1 implicit. The precise translation between π-calculus syntax and HOAS is given in the following definition.

π-calculus syntax	HOAS	
Names	n	
Actions	a	
Processes	p	
$\bar{x}y$	$\uparrow xy$	$\uparrow: n \to n \to a$
τ	τ	$\tau : a$
$x(y)$	$\lambda y \downarrow x\, y$	$\downarrow: n \to n \to a$
$\bar{x}(y)$	$\lambda y \uparrow x\, y$	
0	0	$0 : \mathit{proc}$
$\tau.\mathtt{P}$	$\tau\, P$	$\tau : p \to p$
$\bar{x}y.\mathtt{P}$	$\mathit{out}\ x\ y\ P$	$\mathit{out} : n \to n \to p \to p$
$x(y).\mathtt{P}$	$\mathit{in}\ x\ \lambda y.P\, y$	$\mathit{in} : n \to (n \to p) \to p$
$\mathtt{P} + \mathtt{Q}$	$P + Q$	$+ : p \to p \to p$
$\mathtt{P} \mid \mathtt{Q}$	$P \mid Q$	$\mid\, : p \to p \to p$
$[x = y]\mathtt{P}$	$\mathit{match}\ x\ y\ P$	$\mathit{match} : n \to n \to p \to p$
$(x)\mathtt{P}$	$\nu\lambda x.P\, x$	$\nu : (n \to p) \to p$

Table 6.1. Signatures for π-calculus

Definition 6.1. *We define a translation function $\langle . \rangle$ from process expressions to $\beta\eta$-long normal terms of type* p *as follows.*

$$\begin{array}{rcl@{\qquad}rcl}
\langle 0 \rangle & = & 0 & \langle \tau.\mathtt{P} \rangle & = & \tau\,\langle \mathtt{P} \rangle \\
\langle \bar{x}y.\mathtt{P} \rangle & = & \mathit{out}\ x\ y\ \langle \mathtt{P} \rangle & \langle x(y).\mathtt{P} \rangle & = & \mathit{in}\ x\ \lambda y.\langle \mathtt{P} \rangle \\
\langle \mathtt{P} + \mathtt{Q} \rangle & = & \langle \mathtt{P} \rangle + \langle \mathtt{Q} \rangle & \langle \mathtt{P} | \mathtt{Q} \rangle & = & \langle \mathtt{P} \rangle \mid \langle \mathtt{Q} \rangle \\
\langle [x = y]\mathtt{P} \rangle & = & \mathit{match}\ x\ y\ \langle \mathtt{P} \rangle & \langle (x)\mathtt{P} \rangle & = & \nu\lambda x.\langle \mathtt{P} \rangle
\end{array}$$

The one-step transition judgments is translated to atomic formulas as follows (we overload the symbol $\langle . \rangle$).

$$\begin{array}{rcl@{\qquad}rcl}
\langle \mathtt{P} \xrightarrow{\bar{x}y} \mathtt{Q} \rangle & = & \langle \mathtt{P} \rangle \xrightarrow{\uparrow xy} \langle \mathtt{Q} \rangle & \langle \mathtt{P} \xrightarrow{\tau} \mathtt{Q} \rangle & = & \langle \mathtt{P} \rangle \xrightarrow{\tau} \langle \mathtt{Q} \rangle \\
\langle \mathtt{P} \xrightarrow{x(y)} \mathtt{Q} \rangle & = & \langle \mathtt{P} \rangle \overset{\lambda y \downarrow x\, y}{\rightharpoonup} \lambda y.\langle \mathtt{Q} \rangle & \langle \mathtt{P} \xrightarrow{\bar{x}(y)} \mathtt{Q} \rangle & = & \langle \mathtt{P} \rangle \overset{\lambda y \uparrow x\, y}{\rightharpoonup} \lambda y.\langle \mathtt{Q} \rangle
\end{array}$$

Lemma 6.2. *The function $\langle . \rangle$ is a bijection.*

Proof By induction on the structure of process expressions and the structure of normal λ-terms. ∎

Given the above bijection, we shall omit writing explicitly the function symbol $\langle . \rangle$ when referring to p-term obtained via the translation.

6.2 One-step transitions

The operational semantics of one-step transition is given in Figure 6.1 [40]. There are symmetric counterparts of the rules SUM, PAR, CLOSE and COM which are not shown in the figure but can be easily derived. Figure 6.2 (taken from [37]) contains the inference rules specifying the transitions in Figure 6.1 using higher-order abstract syntax. The variables in these rules denote schema variables: these schema variables have primitive types such as a, n, and p as well as functional types such as $n \rightarrow a$ and $n \rightarrow p$. Notice that for clarity we have η-expanded some terms in Figure 6.1. These inference rules can trivially be written as definition clauses. These clauses are presented in Figure 6.3. Here, schema variables are universally quantified (implicitly) at the top-level of such clauses.

Notice that the complicated side conditions in the original specification of π-calculus are no longer present, as they are now treated directly and declaratively by the meta-logic. For example, the side condition that $x \neq y$ in the open rule is implicit, since x is outside the scope of y and therefore cannot be instantiated with y. The restriction operator is interpreted at the meta-level as the ∇ quantifier. The use of ∇, instead of $\forall$, allows to prove *negative* statements about the transitions, as illustrated in Example 6.3. To simplify the presentation, we mix the syntax of π-calculus and HOAS and use the usual abbreviations in writing process expressions: when a name z is used as a prefix, it denotes the prefix $z(w)$ where w is vacuous in its scope; when a name $\bar{z}$ is used as a prefix it denotes the output prefix $\bar{z}a$ for some fixed constant a. We also abbreviate $(y)\bar{x}y.P$ as $\bar{x}(y).P$ and the process term 0 is omitted if it appears as the continuation

$$\frac{}{\tau.\mathtt{P} \xrightarrow{\tau} \mathtt{P}}\ \mathrm{TAU{-}ACT} \qquad \frac{}{\bar{x}y.\mathtt{P} \xrightarrow{\bar{x}y} \mathtt{P}}\ \mathrm{OUTPUT{-}ACT}$$

$$\frac{}{x(z).\mathtt{P} \xrightarrow{x(w)} \mathtt{P}[w/z]}\ \mathrm{INPUT{-}ACT}, w \notin \mathrm{fn}((z)\mathtt{P}) \qquad \frac{\mathtt{P} \xrightarrow{\alpha} \mathtt{P}'}{[x = x]\mathtt{P} \xrightarrow{\alpha} \mathtt{P}'}\ \mathrm{MATCH}$$

$$\frac{\mathtt{P} \xrightarrow{\alpha} \mathtt{P}'}{\mathtt{P} + \mathtt{Q} \xrightarrow{\alpha} \mathtt{P}'}\ \mathrm{SUM} \qquad \frac{\mathtt{P} \xrightarrow{\alpha} \mathtt{P}'}{\mathtt{P} \mid \mathtt{Q} \xrightarrow{\alpha} \mathtt{P}' \mid \mathtt{Q}}\ \mathrm{PAR}, \mathrm{bn}(\alpha) \cap \mathrm{fn}(\mathtt{Q}) = \emptyset$$

$$\frac{\mathtt{P} \xrightarrow{\bar{x}y} \mathtt{P}', \quad \mathtt{Q} \xrightarrow{x(z)} \mathtt{Q}'}{\mathtt{P} \mid \mathtt{Q} \xrightarrow{\tau} \mathtt{P}' \mid \mathtt{Q}'[y/z]}\ \mathrm{COM} \qquad \frac{\mathtt{P} \xrightarrow{\bar{x}(w)} \mathtt{P}', \quad \mathtt{Q} \xrightarrow{x(w)} \mathtt{Q}'}{\mathtt{P} \mid \mathtt{Q} \xrightarrow{\tau} (w)(\mathtt{P}' \mid \mathtt{Q}')}\ \mathrm{CLOSE}$$

$$\frac{\mathtt{P} \xrightarrow{\alpha} \mathtt{P}'}{(y)\mathtt{P} \xrightarrow{\alpha} (y)\mathtt{P}'}\ \mathrm{RES}, y \notin \mathrm{n}(\alpha) \qquad \frac{\mathtt{P} \xrightarrow{\bar{x}y} \mathtt{P}'}{(y)\mathtt{P} \xrightarrow{\bar{x}(w)} \mathtt{P}'[w/y]}\ \mathrm{OPEN}, y \neq x, w \notin \mathrm{fn}((y)\mathtt{P}')$$

Fig. 6.1. The operational semantics of the late π-calculus.

$$\frac{}{\tau\ P \xrightarrow{\tau} P}\,\tau \qquad \frac{P \xrightarrow{A} Q}{match\ x\ x\ P \xrightarrow{A} Q}\,\text{match} \qquad \frac{P \xrightharpoonup{X} Q}{match\ x\ x\ P \xrightharpoonup{X} Q}\,\text{match}$$

$$\frac{P \xrightarrow{A} R}{P+Q \xrightarrow{A} R}\,\text{sum} \qquad \frac{Q \xrightarrow{A} R}{P+Q \xrightarrow{A} R}\,\text{sum} \qquad \frac{P \xrightharpoonup{X} R}{P+Q \xrightharpoonup{X} R}\,\text{sum} \qquad \frac{Q \xrightharpoonup{X} R}{P+Q \xrightharpoonup{X} R}\,\text{sum}$$

$$\frac{P \xrightarrow{A} P'}{P \mid Q \xrightarrow{A} P' \mid Q}\,\text{par} \qquad \frac{Q \xrightarrow{A} Q'}{P \mid Q \xrightarrow{A} P \mid Q'}\,\text{par}$$

$$\frac{P \xrightharpoonup{X} \lambda y M y}{P \mid Q \xrightharpoonup{X} \lambda n(Mn \mid Q)}\,\text{par} \qquad \frac{Q \xrightharpoonup{X} \lambda y N y}{P \mid Q \xrightharpoonup{X} \lambda n(P \mid Nn)}\,\text{par}$$

$$\frac{\nabla n(Pn \xrightarrow{A} P'n)}{\nu n.Pn \xrightarrow{A} \nu n.P'n}\,\text{res} \qquad \frac{\nabla n(Pn \xrightharpoonup{X} \lambda m\ P'nm)}{\nu n.Pn \xrightharpoonup{X} \lambda m\ \nu n.(P'nm)}\,\text{res} \qquad \frac{\nabla y(My \xrightarrow{\uparrow xy} M'y)}{\nu y.My \xrightharpoonup{\lambda y \uparrow xy} \lambda y M'y}\,\text{open}$$

$$\frac{}{out\ x\ y\ P \xrightarrow{\uparrow xy} P}\,\text{out} \qquad \frac{}{in\ x\ M \xrightharpoonup{\downarrow x} M}\,\text{in}$$

$$\frac{P \xrightharpoonup{\lambda y \downarrow xy} \lambda y\ My \qquad Q \xrightharpoonup{\lambda y \uparrow xy} \lambda y\ Ny}{P \mid Q \xrightarrow{\tau} \nu n.(Mn \mid Nn)}\,\text{close} \qquad \frac{P \xrightharpoonup{\lambda y \uparrow xy} \lambda y My \qquad Q \xrightharpoonup{\lambda y \downarrow xy} \lambda y Ny}{P \mid Q \xrightarrow{\tau} \nu n.(Mn \mid Nn)}\,\text{close}$$

$$\frac{P \xrightharpoonup{\lambda y \downarrow xy} \lambda y My \qquad Q \xrightarrow{\uparrow xy} Q'}{P \mid Q \xrightarrow{\tau} (My) \mid Q'}\,\text{com} \qquad \frac{P \xrightarrow{\uparrow xy} P' \qquad Q \xrightharpoonup{\lambda y \downarrow xy} \lambda y Ny}{P \mid Q \xrightarrow{\tau} P' \mid (Ny)}\,\text{com}$$

Fig. 6.2. The HOAS representation of the late π-calculus.

$$\tau\ P \xrightarrow{\tau} P \triangleq \top. \quad \mathit{in}\ X\ M \xrightharpoonup{\downarrow X} M \triangleq \top. \quad \mathit{out}\ x\ y\ P \xrightarrow{\uparrow xy} P' \triangleq \top.$$

$$\mathit{match}\ x\ x\ P \xrightarrow{A} Q \triangleq P \xrightarrow{A} Q. \qquad \mathit{match}\ x\ x\ P \xrightharpoonup{A} Q \triangleq P \xrightharpoonup{A} Q.$$

$$P + Q \xrightarrow{A} R \triangleq P \xrightarrow{A} R. \qquad P + Q \xrightarrow{A} R \triangleq Q \xrightarrow{A} R.$$

$$P + Q \xrightharpoonup{A} R \triangleq P \xrightharpoonup{A} R. \qquad P + Q \xrightharpoonup{A} R \triangleq Q \xrightharpoonup{A} R.$$

$$P \mid Q \xrightarrow{A} P' \mid Q \triangleq P \xrightarrow{A} P'. \qquad P \mid Q \xrightarrow{A} P \mid Q' \triangleq Q \xrightarrow{A} Q'$$

$$P \mid Q \xrightharpoonup{A} \lambda n(M\,n \mid Q) \triangleq P \xrightharpoonup{A} M. \qquad P \mid Q \xrightharpoonup{A} \lambda n(P \mid N\,n) \triangleq Q \xrightharpoonup{A} N.$$

$$\nu n.Pn \xrightarrow{A} \nu n.Qn \triangleq \nabla n(Pn \xrightarrow{A} Qn). \quad \nu n.Pn \xrightarrow{A} \nu n.Qn \triangleq \nabla n(Pn \xrightarrow{A} Qn).$$

$$\nu y.Py \xrightharpoonup{\uparrow X} \nu y.Qy \triangleq \nabla y(Py \xrightarrow{\uparrow Xy} Qy).$$

$$P \mid Q \xrightarrow{\tau} \nu y.My \mid Ny \triangleq \exists X.P \xrightharpoonup{\downarrow X} M \wedge Q \xrightharpoonup{\uparrow X} T$$

$$P \mid Q \xrightarrow{\tau} \nu y.My \mid Ny \triangleq \exists X.P \xrightharpoonup{\uparrow X} M \wedge Q \xrightharpoonup{\downarrow X} T$$

$$P \mid Q \xrightarrow{\tau} MY \mid Q' \triangleq \exists X.P \xrightharpoonup{\downarrow X} M \wedge Q \xrightarrow{\uparrow XY} Q'$$

$$P \mid Q \xrightarrow{\tau} P' \mid NY \triangleq \exists X.P \xrightharpoonup{\uparrow XY} P' \wedge Q \xrightarrow{\downarrow X} N$$

Fig. 6.3. Definition clauses for one-step transition of π-calculus

of a prefix. We assume that the operators $|$ and $+$ associates the right, e.g., we write $P + Q + R$ to denote $P + (Q + R)$.

Example 6.3. In this example we illustrate how the scoping constraints in the π-calculus is handled at the meta-level. Consider the process $(y)[x = y]\bar{x}z.0$. This process cannot make any transition since the bound variable y denotes a name different from x. We would therefore expect that the following is provable.

$$\forall x \forall z \forall Q \forall \alpha.[((y)[x = y](\bar{x}z.0) \xrightarrow{\alpha} Q) \supset \bot]$$

This type of statement naturally occurs when one is asking whether two processes are bisimilar (see Section 6.3), where it is necessary to know what transitions a process can make and what it cannot.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\{x, z, Q, \alpha\}\,;\, w \triangleright ([x = w](\bar{x}z.0) \xrightarrow{\alpha} Q) \longrightarrow \bot}\ def\mathcal{L}^{csu}}{\{x, z, Q, \alpha\}\,;\, . \triangleright \nabla y.([x = y](\bar{x}z.0) \xrightarrow{\alpha} Q) \longrightarrow \bot}\ \nabla\mathcal{L}}{\{x, z, Q, \alpha\}\,;\, . \triangleright ((y)[x = y](\bar{x}z.0) \xrightarrow{\alpha} Q) \longrightarrow \bot}\ def\mathcal{L}^{csu}}{\{x, z, Q, \alpha\}\,;\, \longrightarrow . \triangleright ((y)[x = y](\bar{x}z.0) \xrightarrow{\alpha} Q) \supset \bot}\ \supset\mathcal{R}$$

Fig. 6.4. The proof of a negation.

To appreciate better the role of ∇, let us look at a different encoding of one-step transition using $\forall$. That is, suppose we replace ∇ with $\forall$ in the inference rules in Figure 6.2. Attempting to prove the above formula given this definition would reduce to attempting a proof of the sequent

$$\{x, z, Q, \alpha\}\,;\, \forall y.([x = y](\bar{x}z.0) \xrightarrow{\alpha} Q) \longrightarrow \bot.$$

Since cut-elimination holds (Corollary 4.21), the only applicable rule is $\forall\mathcal{L}$, followed by $def\mathcal{L}^{csu}$. For the sequent to be provable, x and y would have to be instantiated with different terms so that the $def\mathcal{L}^{csu}$ rule will produce the empty premise. However, we see that there are at least two instantiations of variables that identify them: namely, the substitution $\{w/y,\ w/x,\ \bar{w}z/\alpha,\ 0/Q\}$ gives us

$$\{z\}\,;\, ([w = w](\bar{w}z.0) \xrightarrow{\bar{w}z} 0) \longrightarrow \bot$$

and the substitution $\{x/y,\ \bar{x}z/\alpha,\ 0/Q\}$ gives us

$$\{z\}\,;\,([x = x](\bar{x}z.0) \xrightarrow{\bar{x}z} 0) \longrightarrow \bot.$$

In the first case, the scoping of variables at the object-level is lost at the meta-level, while in the second case, the newness assumption on y is violated. However, these two aspects are captured precisely by ∇, as it is shown in the derivation in Figure 6.4. The success of the topmost instance of $\mathit{def}\mathcal{L}^{csu}$ depends on the failure of the unification problem $\lambda w.x = \lambda w.w$. Notice that the scoping of object variables is maintained at the meta-level by the separation of (global) eigenvariables and (locally bound) generic variables. The "newness" of w is internalized as λ-abstraction and hence it is not subject to any instantiation. ∎

We now prove the adequacy of the encoding of one-step transition. We denote the set of definition clauses in Figure 6.3 with $\mathbf{D}_\pi$. We use the term *π-derivation* to refer to a derivation in the operational semantics of one-step transition of π-calculus. The term *(logic) derivation* is reserved for the derivation in the logic Linc. We shall often use the term *proof* and *logic derivation* interchangeably.

Proposition 6.4. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be processes and* α *an action. The transition* $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$ *is derivable in π-calculus if and only if the sequent* $.\,;\,. \longrightarrow \langle \mathtt{P} \xrightarrow{\alpha} \mathtt{Q}\rangle$ *is provable in* Linc *with definition* $\mathbf{D}_\pi$*.*

Proof
$\Rightarrow$: If $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$ then $.\,;\,. \longrightarrow \langle \mathtt{P} \xrightarrow{\alpha} \mathtt{Q}\rangle$ is provable. The proof is by induction on the π-derivation of $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$. Most cases follow immediately from induction hypothesis. The non-trivial cases are the RES- and OPEN-rules which require the creation of fresh names. Suppose the transition is inferred by the RES rule

$$\frac{\begin{array}{c}\Pi\\ \mathtt{P}' \xrightarrow{\alpha} \mathtt{Q}'\end{array}}{(y)\mathtt{P}' \xrightarrow{\alpha} (y)\mathtt{Q}'}\ \mathrm{RES}, y \notin \mathrm{n}(\alpha),$$

then by induction hypothesis there is a derivation Ξ of the sequent $.\,;\,. \longrightarrow \rhd\langle \mathtt{P} \xrightarrow{\alpha} \mathtt{Q}\rangle$. By Lemma 3.13 there is a derivation Ξ' of $.\,;\,. \longrightarrow y \rhd \langle \mathtt{P}' \xrightarrow{\alpha} \mathtt{Q}'\rangle$. Hence we can construct the following derivation

$$\begin{array}{c}\Xi'\\ \dfrac{\dfrac{.\,;\,. \longrightarrow y \rhd \langle \mathtt{P} \xrightarrow{\alpha} \mathtt{Q}\rangle}{.\,;\,. \longrightarrow \rhd\nabla y.\langle \mathtt{P}' \xrightarrow{\alpha} \mathtt{Q}'\rangle}\ \nabla\mathcal{R}}{.\,;\,. \longrightarrow \langle (y)\mathtt{P}' \xrightarrow{\alpha} (y)\mathtt{Q}'\rangle}\ \mathit{def}\mathcal{R}^{=}\end{array}$$

$\Rightarrow$: If $.\,;\,. \longrightarrow \langle \mathtt{P} \xrightarrow{\alpha} \mathtt{Q}\rangle$ is provable then $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$. In this case, we need to prove a stronger statement: If $.\,;\,. \longrightarrow \bar{a}\rhd\langle \mathtt{P} \xrightarrow{\alpha} \mathtt{Q}\rangle$ is provable then for all renaming substitution

θ such that $\mathrm{dom}(\theta) \subseteq \bar{a}$, the one-step transition $(\mathtt{P} \xrightarrow{\alpha} \mathtt{Q})\theta$ is derivable in π-calculus. The reason for the need for renaming is because we consider judgments as equal modulo α-conversion, or renaming of locally bound names.

Let Ξ of be a derivation of the judgment $.;\, . \longrightarrow \bar{a} \triangleright \langle \mathtt{P} \xrightarrow{\alpha} \mathtt{Q}\rangle$. The proof is by induction on $\mathrm{ht}(\Xi)$. Again, the non-trivial cases are when there is a creation of new names. We look at the case with RES rule. Suppose Π is

$$\dfrac{\dfrac{\begin{array}{c}\Xi' \\ .;\, . \longrightarrow \bar{a}y \triangleright \langle \mathtt{P}' \xrightarrow{\alpha} \mathtt{Q}'\rangle\end{array}}{.;\, . \longrightarrow \bar{a} \triangleright \nabla y.\langle \mathtt{P}' \xrightarrow{\alpha} \mathtt{Q}'\rangle}\ \nabla\mathcal{R}}{.;\, . \longrightarrow \bar{a} \triangleright \langle (y)\mathtt{P}' \xrightarrow{\alpha} (y)\mathtt{Q}'\rangle}\ \mathit{def}\mathcal{R}$$

Then by induction hypothesis for all renaming substitution ρ such that $\mathrm{dom}(\rho) \subseteq \bar{a}y$, $(\mathtt{P}' \xrightarrow{\alpha} \mathtt{Q}')\rho$ is derivable in π-calculus. This includes a renaming which maps y to a fresh name not occuring in $\mathtt{P}$, α and $\mathtt{Q}$. Therefore we have the following derivation in π-calculus

$$\dfrac{\begin{array}{c}\Pi' \\ \mathtt{P}' \xrightarrow{\alpha} \mathtt{Q}'\end{array}}{(y)\mathtt{P}' \xrightarrow{\alpha} (y)\mathtt{Q}'}\ \mathrm{RES}, y \notin \mathrm{n}(\alpha)$$

where Π' is obtained from induction hypothesis. ∎

6.3 Strong bisimilarity

There are many variants of bisimulation studied in the literature related to π-calculus; we will not attempt to cover all of them here. We consider here only *strong bisimulation*, that is, bisimulation in which all actions, observable or not, are considered. More precisely, we consider encoding strong *late* bisimulation. The notion of strong bisimulation for late π-calculus is defined as follows.

DEFINITION 6.5. *[40] A binary relation $\mathcal{S}$ on processes is a* strong late simulation *if it satisfies the following requirements:*

1. *if* $\mathtt{P} \xrightarrow{\alpha} \mathtt{P}'$ *and α is a free action, then for some* $\mathtt{Q}'$, $\mathtt{Q} \xrightarrow{\alpha} \mathtt{Q}'$ *and* $\mathtt{P}'\mathcal{S}\mathtt{Q}'$,

2. *if* $\mathtt{P} \xrightarrow{x(y)} \mathtt{P}'$ *and $y \notin n(P, Q)$, then for some* $\mathtt{Q}'$, $\mathtt{Q} \xrightarrow{x(y)} \mathtt{Q}'$ *and for all w,*

$$\mathtt{P}'[w/y]\ \mathcal{S}\ \mathtt{Q}'[w/y],$$

and

3. *if* $\mathtt{P} \xrightarrow{\bar{x}(y)} \mathtt{P}'$ *and* $y \notin \mathrm{n}(\mathtt{P}, \mathtt{Q})$ *then for some* $\mathtt{Q}'$, $\mathtt{Q} \xrightarrow{\bar{x}(y)} \mathtt{Q}'$ *and* $\mathtt{P}'\ \mathcal{S}\ \mathtt{Q}'$.

The relation $\mathcal{S}$ is a strong late bisimulation *if both $\mathcal{S}$ and its inverse are late simulations. The relation $\dot{\sim}$,* strong late bisimilarity, *on processes is defined by* $\mathtt{P} \dot{\sim} \mathtt{Q}$ *if and only if there exists a late bisimulation $\mathcal{S}$ such that* $\mathtt{P}\mathcal{S}\mathtt{Q}$.

$$\begin{array}{rl}
\mathit{bisim}\ P\ Q \overset{\triangle}{=} & \forall A \forall P'\ [(P \xrightarrow{A} P') \supset \exists Q'.(Q \xrightarrow{A} Q') \wedge \mathit{bisim}\ P'\ Q'] \wedge \\
& \forall A \forall Q'\ [(Q \xrightarrow{A} Q') \supset \exists P'.(P \xrightarrow{A} P') \wedge \mathit{bisim}\ Q'\ P'] \wedge \\
& \forall X \forall P'\ [(P \xrightarrow{\downarrow X} P') \supset \exists Q'.(Q \xrightarrow{\downarrow X} Q') \wedge \forall w.\mathit{bisim}\ (P'w)\ (Q'w)] \wedge \\
& \forall X \forall Q'\ [(Q \xrightarrow{\downarrow X} Q') \supset \exists P'.(P \xrightarrow{\downarrow X} P') \wedge \forall w.\mathit{bisim}\ (Q'w)\ (P'w)] \wedge \\
& \forall X \forall P'\ [(P \xrightarrow{\uparrow X} P') \supset \exists Q'.(Q \xrightarrow{\uparrow X} Q') \wedge \nabla w.\mathit{bisim}\ (P'w)\ (Q'w)] \wedge \\
& \forall X \forall Q'\ [(Q \xrightarrow{\uparrow X} Q') \supset \exists P'.(P \xrightarrow{\uparrow X} P') \wedge \nabla w.\mathit{bisim}\ (Q'w)\ (P'w)]
\end{array}$$

Fig. 6.5. Lazy encoding of strong late bisimulation

We first attempt to do a straightforward encoding of strong bisimulation defined above, that is, by replacing the English words "if ... then ...", "and", "for all", etc., with logical connectives. Let us call this encoding the *lazy encoding* of strong bisimulation (the use of the term "lazy" will be clear later). This encoding is shown in Figure 6.5, and has also appeared in [38]. We denote by $\mathbf{D}_{\pi,\dot{\sim}}$ the definition $\mathbf{D}_{\pi}$ augmented with the *bisim* clause.

Notice that the difference between bound output and bound input is captured by the use of ∇ and $\forall$ quantifiers. We illustrate this in the following example.

Example 6.6. Consider the following processes expressions.

$$P = x(y).(y|\bar{z}),\ Q = x(y).(y.\bar{z} + \bar{z}.y + [y = z]\tau), \quad R = \bar{x}(y).(y|\bar{z}),\ T = \bar{x}(y).(y.\bar{z} + \bar{z}.y).$$

The process P is bisimilar to Q and the process R to T. Note that we need the match prefix in Q since P is capable of performing the silent transition in the case where the input value y is z. We don't need the match prefix in T since y in this case can only be instantiated with a fresh name different from z. We would therefore expect the following sequents to be provable.

$$\cdot\,;\,\cdot \longrightarrow \mathit{bisim}\ x(y).(y|\bar{z})\ \ x(y).(y.\bar{z} + \bar{z}.y + [y = z]\tau)$$

$$\cdot\,;\,\cdot \longrightarrow \mathit{bisim}\ \bar{x}(y).(y|\bar{z})\ \ \bar{x}(y).(y.\bar{z} + \bar{z}.y)$$

$$\cdot\,;\,\cdot \longrightarrow x(y).(y.\bar{z}+\bar{z}.y+[y=z]\tau) \xrightarrow{\downarrow x} \lambda y(y.\bar{z}+\bar{z}.y+[y=z]\tau) \\ \wedge\ \forall w.\mathit{bisim}\ (w|\bar{z})\ (w.\bar{z}+\bar{z}.w+[w=z]\tau) \tag{a.1}$$

$$\cdot\,;\,\cdot \longrightarrow x(y).(y.\bar{z}+\bar{z}.y+[y=z]\tau) \xrightarrow{\downarrow x} \lambda y(y.\bar{z}+\bar{z}.y+[y=z]\tau) \tag{a.2}$$

$$w\,;\,\cdot \longrightarrow \mathit{bisim}\ (w|\bar{z})\ (w.\bar{z}+\bar{z}.w+[w=z]\tau) \tag{a.3}$$

$$w,R,A\,;\,(w|\bar{z}) \xrightarrow{A} R \longrightarrow \exists T.(w.\bar{z}+\bar{z}.w+[w=z]\tau) \xrightarrow{A} T\ \wedge\ \mathit{bisim}\ R\ T \tag{a.4}$$

$$w,T,A\,;\,(w.\bar{z}+\bar{z}.w+[w=z]\tau) \xrightarrow{A} T \longrightarrow \exists R.(w|\bar{z}) \xrightarrow{A} R\ \wedge\ \mathit{bisim}\ T\ R \tag{a.5}$$

$$w,U,X\,;\,(w|\bar{z}) \xrightarrow{\downarrow X} U \longrightarrow \exists V.(w.\bar{z}+\bar{z}.w+[w=z]\tau) \xrightarrow{\downarrow X} V\ \wedge\ \forall n.\mathit{bisim}\ Un\ Vn \tag{a.6}$$

$$w,V,X\,;\,(w.\bar{z}+\bar{z}.w+[w=z]\tau) \xrightarrow{\downarrow X} V \longrightarrow \exists U.(w|\bar{z}) \xrightarrow{\downarrow X} U\ \wedge\ \forall n.\mathit{bisim}\ Vn\ Un \tag{a.7}$$

$$w,U,X\,;\,(w|\bar{z}) \xrightarrow{\uparrow X} U \longrightarrow \exists V.(w.\bar{z}+\bar{z}.w+[w=z]\tau) \xrightarrow{\uparrow X} V\ \wedge\ \nabla n.\mathit{bisim}\ Un\ Vn \tag{a.8}$$

$$w,V,X\,;\,(w.\bar{z}+\bar{z}.w+[w=z]\tau) \xrightarrow{\uparrow X} V \longrightarrow \exists U.(w|\bar{z}) \xrightarrow{\uparrow X} U\ \wedge\ \nabla n.\mathit{bisim}\ Vn\ Un \tag{a.9}$$

$$w\,;\,\cdot \longrightarrow \exists T.(w.\bar{z}+\bar{z}.w+[w=z]\tau) \xrightarrow{\uparrow za} T\ \wedge\ \mathit{bisim}\ (w|0)\ T \tag{a.10}$$

$$w,X',Y,R',M\,;\,w \xrightarrow{\downarrow X'} M,\ \bar{z} \xrightarrow{\uparrow X'Y} R' \longrightarrow \exists T.\left[\begin{array}{r}(w.\bar{z}+\bar{z}.w+[w=z]\tau) \xrightarrow{\tau} T \\ \wedge\ \mathit{bisim}\ (M'Y|R')\ T\end{array}\right] \tag{a.11}$$

$$\cdot\,;\,\cdot \longrightarrow \exists T.(z.\bar{z}+\bar{z}.z+[z=z]\tau) \xrightarrow{\tau} T\ \wedge\ \mathit{bisim}\ (0|0)\ T \tag{a.12}$$

$$w\,;\,\cdot \longrightarrow \exists R.(w|\bar{z}) \xrightarrow{\uparrow za} R\ \wedge\ \mathit{bisim}\ w\ R \tag{a.13}$$

$$w,T,A\,;\,[w=z]\tau \xrightarrow{A} T \longrightarrow \exists R.(w|\bar{z}) \xrightarrow{A} R\ \wedge\ \mathit{bisim}\ T\ R \tag{a.14}$$

$$\cdot\,;\,\cdot \longrightarrow \exists R.(z|\bar{z}) \xrightarrow{\tau} R\ \wedge\ \mathit{bisim}\ 0\ R \tag{a.15}$$

Fig. 6.6. Example of *bisim* with bound input

$$(iii.1) \qquad R, A\,;\; w \triangleright (w|\bar{z}) \xrightarrow{Aw} Rw \longrightarrow w \triangleright \exists T.(w.\bar{z} + \bar{z}.w \xrightarrow{Aw} T \;\wedge\; \mathit{bisim}\ Rw\ T$$

$$(iii.2) \qquad T, A\,;\; w \triangleright (w.\bar{z} + \bar{z}.w) \xrightarrow{Aw} Tw \longrightarrow w \triangleright \exists R.(w|\bar{z}) \xrightarrow{Aw} R \;\wedge\; \mathit{bisim}\ Tw\ R$$

$$(iii.3) \qquad M, X\,;\; w \triangleright (w|\bar{z}) \xrightharpoonup{\downarrow(Xw)} Mw \longrightarrow w \triangleright \exists N \begin{bmatrix} (w.\bar{z} + \bar{z}.w) \xrightharpoonup{\downarrow(Xw)} N \\ \wedge\ \forall n.\mathit{bisim}\ Mwn\ Nn \end{bmatrix}$$

$$(iii.4) \qquad N, X\,;\; w \triangleright (w.\bar{z} + \bar{z}.w) \xrightharpoonup{\downarrow(Xw)} Nw \longrightarrow w \triangleright \exists M \begin{bmatrix} (w.\bar{z} + \bar{z}.w) \xrightharpoonup{\downarrow(Xw)} M \\ \wedge\ \forall n.\mathit{bisim}\ Nwn\ Mn \end{bmatrix}$$

$$(iii.5) \qquad M, X\,;\; w \triangleright (w|\bar{z}) \xrightharpoonup{\uparrow(Xw)} Mw \longrightarrow w \triangleright \exists N \begin{bmatrix} (w.\bar{z} + \bar{z}.w) \xrightharpoonup{\uparrow(Xw)} N \\ \wedge\ \nabla n.\mathit{bisim}\ Mwn\ Nn \end{bmatrix}$$

$$(iii.6) \qquad N, X\,;\; w \triangleright (w.\bar{z} + \bar{z}.w) \xrightharpoonup{\uparrow(Xw)} Nw \longrightarrow w \triangleright \exists M \begin{bmatrix} (w.\bar{z} + \bar{z}.w) \xrightharpoonup{\uparrow(Xw)} M \\ \wedge\ \nabla n.\mathit{bisim}\ Nwn\ Mn \end{bmatrix}$$

$$(iii.7) \qquad M, X, R', Y\,;\; \begin{bmatrix} w \triangleright w \xrightharpoonup{\downarrow X} Mw, \\ w \triangleright \bar{z} \xrightarrow{\uparrow XY} R'w \end{bmatrix} \longrightarrow w \triangleright \exists T. \begin{bmatrix} (w.\bar{z} + \bar{z}.w) \xrightarrow{\tau} T \\ \wedge\ \mathit{bisim}\ (MwY|R'w)\ T \end{bmatrix}$$

$$(iii.8) \qquad R', Y\,;\; w \triangleright \bar{z} \xrightarrow{\uparrow wY} R'w \longrightarrow w \triangleright \exists T.(w.\bar{z} + \bar{z}.w) \xrightarrow{\tau} T \;\wedge\; \mathit{bisim}\ (0|R'w)\ T$$

Fig. 6.7. Example of *bisim* with bound output

Since we have cut-elimination, we need only consider cut-free proofs in proving these sequents. Therefore to prove the first sequent, we first apply *def*$\mathcal{R}$ followed by $\forall\mathcal{R}$ and $\wedge\mathcal{R}$, resulting in the following six sequents.

$$(1)\ \ P', A\,;\ x(y).(y|\bar{z}) \xrightarrow{A} P' \longrightarrow \exists Q'.x(y).(y.\bar{z}+\bar{z}.y+[y=z]\tau) \xrightarrow{A} Q' \ \wedge\ \mathit{bisim}\ P'\ Q'$$

$$(2)\ \ Q', A\,;\ x(y).(y.\bar{z}+\bar{z}.y+[y=z]\tau) \xrightarrow{A} Q' \longrightarrow \exists P'.x(y).(y|\bar{z}) \xrightarrow{A} P' \ \wedge\ \mathit{bisim}\ Q'\ P'$$

$$(3)\ \ M, X\,;\ x(y).(y|\bar{z}) \xrightharpoonup{\downarrow X} M \longrightarrow \exists N. \left(\begin{array}{l} x(y).(y.\bar{z}+\bar{z}.y+[y=z]\tau) \xrightharpoonup{\downarrow X} N\ \wedge \\ \forall w.\mathit{bisim}\ Mw\ Nw \end{array} \right).$$

$$(4)\ \ N, X\,;\ x(y).(y.\bar{z}+\bar{z}.y+[y=z]\tau) \xrightharpoonup{\downarrow X} N \longrightarrow \exists M. \left(\begin{array}{l} x(y).(y|\bar{z}) \xrightharpoonup{\downarrow X} M\ \wedge \\ \forall w.\mathit{bisim}\ Nw\ Mw \end{array} \right).$$

$$(5)\ \ M, X\,;\ x(y).(y|\bar{z}) \xrightharpoonup{\uparrow X} M \longrightarrow \exists N. \left(\begin{array}{l} x(y).(y.\bar{z}+\bar{z}.y+[y=z]\tau) \xrightharpoonup{\uparrow X} N\ \wedge \\ \nabla w.\mathit{bisim}\ Mw\ Nw \end{array} \right).$$

$$(6)\ \ N, X\,;\ x(y).(y.\bar{z}+\bar{z}.y+[y=z]\tau) \xrightharpoonup{\uparrow X} N \longrightarrow \exists N. \left(\begin{array}{l} x(y).(y|\bar{z}) \xrightharpoonup{\uparrow X} M\ \wedge \\ \nabla w.\mathit{bisim}\ Nw\ Mw \end{array} \right).$$

Next, we apply the *def*$\mathcal{L}^{csu}$ to each sequent. Since the outermost constructor of the process P is the input prefix, the only matching definition clause is

$$\mathit{in}\ X'\ M' \xrightharpoonup{\downarrow X'} M' \stackrel{\triangle}{=} \top.$$

The unification succeeds only on sequents (3) and (4) with the respective (most general) unifiers

$$[\lambda y(y|\bar{z})/M, x/X] \quad \text{and} \quad [\lambda y(y.\bar{z}+\bar{z}.y+[y=z]\tau)/N, x/X].$$

It fails on the other sequents, and hence the proofs for those sequents succeed. The corresponding premises of sequents (3) and (4) are the sequents

$$(a)\quad \cdot\,;\,\cdot \longrightarrow \exists N.x(y).(y.\bar{z}+\bar{z}.y+[y=z]\tau) \xrightharpoonup{\downarrow x} N\ \wedge\ \forall w.\mathit{bisim}\ (w|\bar{z})\ Nw \quad \text{and}$$

$$(b)\quad \cdot\,;\,\cdot \longrightarrow \exists M.x(y).(y|\bar{z}) \xrightharpoonup{\downarrow x} M\ \wedge\ \forall w.\mathit{bisim}\ (\bar{z}+\bar{z}.y+[y=z]\tau)\ Mw.$$

We proceed by applying $\exists\mathcal{R}$ to both sequents. It is obvious that for the sequent (a) we should instantiate N with $\lambda y(y.\bar{z}+\bar{z}.y+[y=z]\tau)$ and for the sequent (b) we instantiate M with $\lambda y(y|\bar{z})$. We show here an outline of the proof for the continuation of (a). The proof for (b) is completely symmetric.

A list of premises of (a) is listed in Figure 6.6. Here the sequent $(a.1)$ is the immediate premise of (a), after applying $\exists\mathcal{R}$. The proof branches to sequent $(a.2)$ and $(a.3)$. The sequent $(a.2)$ is provable by one step application of *def*$\mathcal{R}^{=}$. We apply *def*$\mathcal{L}^{csu}$ to $(a.3)$ (followed by instances $\wedge\mathcal{R}$, $\forall\mathcal{R}$ and $\supset\mathcal{R}$) to get the six premises $(a.4)$

- $(a.9)$. We highlight the proof for the sequent $(a.4)$ and $(a.5)$ since they illustrate the important role of eigenvariables in encoding bound input. We see from the definition clauses in Figure 6.2 that applying $\mathit{def}\mathcal{L}^{csu}$ repeatedly to sequent $(a.4)$ leaves us with two premises: $(a.10)$ and $(a.11)$. In the case of $(a.10)$, we infer that $(w|\bar{z})$ has made a free output $\uparrow za$ (a is a constant). This sequent is provable by instantiating T with w. In $(a.11)$, we would like to infer the τ transition from $(w|\bar{z})$. This reduces to inferring that w makes an input transition to some process and $\bar{z}$ makes an output transition *on the same channel*, which is represented here as X'. We continue applying $\mathit{def}\mathcal{L}^{csu}$ to $(a.11)$. We see that in order for the $w \xrightarrow{\downarrow X'} M$ to succeed, X' must be instantiated to w and M to $\lambda y.0$. We are then left with the remaining one-step transition: $\bar{z} \xrightarrow{\uparrow wa} R'$. Again, for this transition to succeed w needs to be instantiated with z, which is what happens when we apply the $\mathit{def}\mathcal{L}^{csu}$. Note that w is an eigenvariable so it can be instantiated when applying $\mathit{def}\mathcal{L}^{csu}$. The resulting sequent is $(a.12)$ which can be verified to be provable. In the symmetric case with $(a.5)$, the subcase where w gets instantiated is given in the sequent $(a.14)$. Here the unification of w and z is driven by the match prefix $[w = z]\tau$. The result of applying $\mathit{def}\mathcal{L}^{csu}$ to $(a.14)$ is the sequent $(a.15)$, where w is instantiated with z, A with τ and T with 0. The resulting sequent can be shown provable.

The sequent $\cdot\,;\,\cdot \longrightarrow \mathit{bisim}\ \bar{x}(y).(y|\bar{z})\ \ \bar{x}(y).(y.\bar{z}+\bar{z}.y)$ is proved in similar fashion, that is, the proof starts with an instance of $\mathit{def}\mathcal{R}^{=}$, followed by asynchronous rules (see Chapter 3, Section 3.7), and $\mathit{def}\mathcal{L}^{csu}$. We get the corresponding six premises as in $(1) - (6)$. In particular, the cases involving bound output are now

$$(5')\ \ M, X\,;\,\bar{x}(y).(y|\bar{z}) \xrightarrow{\uparrow X} M \longrightarrow \exists N.\bar{x}(y).(y.\bar{z}+\bar{z}.y) \xrightarrow{\uparrow X} N\ \wedge\ \nabla w.\mathit{bisim}\ Mw\ Nw$$

$$(6')\ \ N, X\,;\,\bar{x}(y).(y.\bar{z}+\bar{z}.y) \xrightarrow{\uparrow X} N \longrightarrow \exists M.\bar{x}(y).(y|\bar{z}) \xrightarrow{\uparrow X} M\ \wedge\ \nabla w.\mathit{bisim}\ Nw\ Mw$$

We apply $\mathit{def}\mathcal{L}^{csu}$ followed by $\exists\mathcal{R}$ and $\wedge\mathcal{R}$. In the premises of $\mathit{def}\mathcal{L}^{csu}$, the eigenvariable M and X in $(5')$ get instantiated to $\lambda y(y|\bar{z})$ and x, while in $(6')$, N and X get instantiated to $\lambda y(y.\bar{z}+\bar{z}.y)$. In the instances of $\exists\mathcal{R}$, we instantiate N in $(5')$ with the term $\lambda y(y.\bar{z}+\bar{z}.y)$ and M in $(6')$ with $\lambda y(y|\bar{z})$. We thus arrive at the following premise sequents

$$\begin{array}{ll}
(i) & \cdot\,;\,\cdot \longrightarrow \bar{x}(y).(y.\bar{z}+\bar{z}.y) \xrightarrow{\uparrow x} \lambda y(y.\bar{z}+\bar{z}.y) \\
(ii) & \cdot\,;\,\cdot \longrightarrow \nabla w.\mathit{bisim}\ (w|\bar{z})\ (w.\bar{z}+\bar{z}.w) \\
(iii) & \cdot\,;\,\cdot \longrightarrow \bar{x}(y).(y|\bar{z}) \xrightarrow{\uparrow x} \lambda y(y|\bar{z}) \\
(iv) & \cdot\,;\,\cdot \longrightarrow \nabla w.\mathit{bisim}\ (w.\bar{z}+\bar{z}.w)\ (w|\bar{z})
\end{array}$$

Sequents (i) and (ii) are provable by two applications of $\mathit{def}\mathcal{R}^{=}$. We look at the proof of sequent (iii) (the proof of (iv) is basically the same as (iii)). By unfolding the definition of bisim we get the six premises $(iii.1)$ - $(iii.6)$ in Figure 6.7, which are analogous to the sequents $(a.4)$ - $(a.9)$ in Figure 6.6. Notice that the variable w now has local scope. For the sequent $(iii.1)$ to be provable, it is necessary that the process $(w|\bar{z})$ does not make a τ transition. This reduces to showing that the sequent $(iii.7)$, which is obtained from $(iii.1)$ by $\mathit{def}\mathcal{L}^{csu}$, is provable. We first apply $\mathit{def}\mathcal{L}^{csu}$ to the judgment $w \rhd w \xrightarrow{\downarrow Xw} Mw$.

This will unify the variable X with $\lambda w.w$ and M with $\lambda w.0$. The resulting sequent is $(iii.8)$. We then apply $\mathit{def}\mathcal{L}^{csu}$ to $w \triangleright \bar{z} \xrightarrow{\uparrow wY} R'w$. The applicable definition clause is the out-rule (see Figure 6.2). For the $\mathit{def}\mathcal{L}^{csu}$ to succeed we need to identify z and w. But since w has local scope its corresponding unification problem is $\lambda w.w = \lambda w.z$ (recall that local variables are interpreted as λ-abstraction in applying eq$\mathcal{L}$ and $\mathit{def}\mathcal{L}^{csu}$), which has no solution. Therefore the proof for sequent $(iii.8)$ succeeds. Notice the difference with the sequent $(a.11)$ in which the corresponding unification problem is $w = z$, which is solvable since w in that case is an eigenvariable which is subject to instantiation.

∎

We now show that the *bisim* clause does encode bisimulation.

Theorem 6.7. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes. If the sequent*

$$.\,;\,. \longrightarrow \bar{u} \triangleright \mathit{bisim}\ \mathtt{P}\ \mathtt{Q},$$

where free names in $\mathtt{P}$ *and* $\mathtt{Q}$ *are either in* $\bar{u}$ *or global constants, is provable in* Linc *with the definition* $\mathbf{D}_{\pi,\dot{\sim}}$ *then* $\mathtt{P} \dot{\sim} \mathtt{Q}$.

Proof We show that the set

$$\mathcal{S} = \{(\mathtt{R}, \mathtt{T}) \mid .\,;\,. \longrightarrow \bar{w} \triangleright \mathit{bisim}\ \mathtt{R}\ \mathtt{T}\}$$

where free names in $\mathtt{R}$ and $\mathtt{T}$ are either in $\bar{w}$ or global constants, is a strong late bisimulation. Given the cut-elimination theorem, we know that the proof of $\bar{w} \triangleright \mathit{bisim}\ \mathtt{R}\ \mathtt{T}$ must end with an instance of $\mathit{def}\mathcal{R}$. By unfolding the definition clause of *bisim* and applying the introduction rules for the asynchronous logical connectives, we arrive at six sequents in the premise. We show here three of them, the other three are symmetric (by interchanging $\mathtt{R}$ and $\mathtt{T}$):

$$\textbf{(1)}\ \ A, R'\,;\, \bar{w} \triangleright \mathtt{R} \xrightarrow{A\bar{w}} R'\,\bar{w} \longrightarrow \bar{w} \triangleright \exists T'.\mathtt{T} \xrightarrow{A\,\bar{w}} T' \wedge \mathit{bisim}\ (R'\,\bar{w})\ T'$$

$$\textbf{(2)}\ \ X, M\,;\, \bar{w} \triangleright \mathtt{R} \overset{\downarrow(X\bar{w})}{\rightharpoondown} M\,\bar{w} \longrightarrow \bar{w} \triangleright \exists N.\mathtt{T} \overset{\downarrow(X\bar{w})}{\rightharpoondown} N \wedge \forall n.\mathit{bisim}\ (M\,\bar{w}\,n)\ (N\,n)$$

$$\textbf{(3)}\ \ X, M\,;\, \bar{w} \triangleright \mathtt{R} \overset{\uparrow(X\bar{w})}{\rightharpoondown} M\,\bar{w} \longrightarrow \bar{w} \triangleright \exists N.\mathtt{T} \overset{\uparrow(X\bar{w})}{\rightharpoondown} N \wedge \nabla n.\mathit{bisim}\ (M\,\bar{w}\,n)\ (N\,n)$$

Let Π_1, Π_2, Π_3 be the proofs of the sequents (1), (2) and (3), respectively. We need to show that for every transition that $\mathtt{R}$ can perform, $\mathtt{T}$ can make the same transition and their continuations are in $\mathcal{S}$. We distinguish three cases, depending whether α is a free action, a bound input or a bound output. These cases correspond to the sequents (1), (2) and (3) above.

For case (1), suppose we have a transition $\mathtt{R} \xrightarrow{\alpha} \mathtt{R}_1$, then by Proposition 6.4 we have derivation of $.\,;\,. \longrightarrow \mathtt{R} \xrightarrow{\alpha} \mathtt{R}_1$, where the free names $\bar{w}$ are treated as global constants. By Lemma 3.13 we can weaken the scope of $\bar{w}$ so that we have a derivation

Π of $.\,;\,. \longrightarrow \bar{w} \triangleright \mathtt{R} \xrightarrow{\alpha} \mathtt{R}_1$. We instantiate the derivation Π_1 with the substitution $\theta = [\lambda\bar{w}.\mathtt{R}_1/R', \lambda\bar{w}.\alpha/A]$ and then cut this derivation with Π to obtain the derivation Ξ

$$\dfrac{\overset{\Pi}{.\,;\,. \longrightarrow \bar{w} \triangleright \mathtt{R} \xrightarrow{\alpha} \mathtt{R}_1} \qquad \overset{\Pi_1\theta}{.\,;\,\bar{w} \triangleright \mathtt{R} \xrightarrow{\alpha} \mathtt{R}_1 \longrightarrow \bar{w} \triangleright \exists T'.\mathtt{T} \xrightarrow{\alpha} T' \wedge \mathit{bisim}\ \mathtt{R}_1\ T'}}{.\,;\,. \longrightarrow \bar{w} \triangleright \exists T'.\mathtt{T} \xrightarrow{\alpha} T' \wedge \mathit{bisim}\ \mathtt{R}_1\ T'}\ \mathit{mc}$$

By cut-elimination, there is a cut-free proof of the sequent

$$.\,;\,. \longrightarrow \bar{w} \triangleright \exists T'.\mathtt{T} \xrightarrow{\alpha} T' \wedge \mathit{bisim}\ \mathtt{R}_1\ T'.$$

The two last rules of this proof must be instances of $\exists\mathcal{R}$ and $\wedge\mathcal{R}$. Hence there must exist a process term $\mathtt{T}_1$ such that

$$.\,;\,. \longrightarrow \bar{w} \triangleright \mathtt{T} \xrightarrow{\alpha} \mathtt{T}_1 \text{ and } .\,;\,. \longrightarrow \bar{w} \triangleright \mathit{bisim}\ \mathtt{R}_1\ \mathtt{T}_1$$

are provable. By Proposition 6.4 there is a transition $\mathtt{T} \xrightarrow{\alpha} \mathtt{T}_1$, and from the definition of $\mathcal{S}$, clearly $(\mathtt{R}_1, \mathtt{T}_1) \in \mathcal{S}$.

In case (2), we are given a transition $\mathtt{R} \xrightharpoonup{\downarrow x} \lambda y.\mathtt{M}\,y$, and we need to show that there is an abstraction $\mathtt{N}$ (of type $n \to p$) such that $\bar{w} \triangleright \mathtt{T} \xrightharpoonup{\downarrow x} \lambda y.\mathtt{N}\,y$ is provable and that for every name a, $(\mathtt{M}\,a, \mathtt{N}\,a) \in \mathcal{S}$. By similar arguments as in case (1), we show that there is indeed a term $\mathtt{N}$ such that

$$.\,;\,. \longrightarrow \bar{w} \triangleright \mathtt{T} \xrightharpoonup{\downarrow x} \mathtt{N}$$

and

$$.\,;\,. \longrightarrow \bar{w} \triangleright \forall y.\mathit{bisim}\ (\mathtt{M}\,y)\ (\mathtt{N}\,y)$$

are provable. In the second sequent, we can instantiate y with any constant of type name (either global or local). That is, $\bar{w} \triangleright \mathit{bisim}\ (M\ a)\ (\mathtt{N}\,a)$ is provable for every a, and hence $(\mathtt{M}\,a, \mathtt{N}\,a) \in \mathcal{S}$. Case (3) is similar to case (2), the only difference is that in this case we need only to show that $(\mathtt{M}\,a, \mathtt{N}\,a) \in \mathcal{S}$ for a fresh name a not in the free names of $(\mathtt{R}, \mathtt{T})$. Applying the same arguments as above, we arrive at the (cut-free) derivation

$$\dfrac{\overset{\Xi'}{;\, \longrightarrow \bar{w}a \triangleright \mathit{bisim}\ (\mathtt{M}\,a)\ (\mathtt{N}\,a)}}{;\, \longrightarrow \bar{w} \triangleright \nabla y.\mathit{bisim}\ (\mathtt{M}\,y)\ (\mathtt{N}\,y)}\ \nabla\mathcal{R}.$$

Hence $(\mathtt{M}\,a, \mathtt{N}\,a) \in \mathcal{S}$. We have thus shown that $\mathcal{S}$ is indeed a bisimulation set, and therefore the soundness of our encoding. ∎

The converse of Theorem 6.7 is not true, that is, there are some processes that are bisimilar but their encoding is not provable in Linc with $\mathbf{D}_{\pi,\dot{\sim}}$. A counterexample is given below.

Example 6.8. Consider the processes

$$P = y(z).(x + x.z), \quad Q = y(z).(x + x.z + x.[z = a]z).$$

We show that $P \overset{\cdot}{\sim} Q$ but $\mathit{bisim}\ P\ Q$ is not provable. Let

$$\begin{aligned}\mathcal{R} \;=\; & \{(P,Q)\} \bigcup \{(x + x.w, x + x.w + x.[w = a]w) \mid \text{ for some name } w\ \} \\ & \bigcup \{(0, [w = a]w) \mid \text{ for some name } w \neq a\ \} \bigcup \{(a, [a = a]a), (0, 0)\}\end{aligned}$$

and let $\mathcal{S}$ be the symmetric closure of $\mathcal{R}$, i.e., $\mathcal{S} = \{(R,T),(T,R) \mid (R,T) \in \mathcal{R}\}$. We show that $\mathcal{S}$ is a bisimulation set by case analysis on all possible pairs in $\mathcal{S}$. The interesting case is to check that the continuations of $(x + x.w, x + x.w + x.[w = a]w)$ are in $\mathcal{S}$. Let $R = x + x.w$ and $T = x + x.w + [w = a]w$. There are two possible transitions from R: either $R \xrightarrow{x} 0$ or $R \xrightarrow{x} w$. In the first case, we choose for T the transition $T \xrightarrow{x} 0$ (i.e., by selecting the first summand x). In the second case, we choose for T the transition $T \xrightarrow{x} w$. In both cases the continuation pairs are in $\mathcal{S}$.

Conversely, there are three possible transitions from T: $T \xrightarrow{x} 0$, $T \xrightarrow{x} w$ and $T \xrightarrow{x} [w = a]w$. For the first two cases, we choose for R the transition $R \xrightarrow{x} 0$ and $R \xrightarrow{x} w$, respectively. For the third case, if $w = a$, we choose the transition $R \xrightarrow{x} w$, otherwise take the transition $R \xrightarrow{x} 0$. In all cases it is obvious that the continuations are in $\mathcal{S}$.

Let us now attempt to prove the sequent $\cdot\,;\,\cdot \longrightarrow \mathit{bisim}\ P\ Q$, where all the free names in (P,Q) are encoded as constants. It can be checked that, using only asynchronous rules, the proof of this sequent reduces to a proof of the sequent

$$;\, \longrightarrow \forall w.\mathit{bisim}\ (x + x.w)\ (x + x.w + [w = a]w). \tag{6.1}$$

Now we apply $\forall\mathcal{R}$, $\mathit{def}\mathcal{R}$ and the necessary introduction rules for the logical connectives in bisim . We get six sequents as the premise, corresponding to free action, bound input and bound output transitions. Among the premises is the sequent

$$X, R'\,;\,(x + x.w + x.[w = a]w) \xrightharpoonup{\downarrow X} R' \longrightarrow \exists T'.x + x.w \xrightharpoonup{\downarrow X} T' \ \wedge\ \forall n.\mathit{bisim}\ R'n\ T'n.$$

Applying $\mathit{def}\mathcal{L}^{csu}$ on this sequent gives us three premises. The interesting one is the sequent

$$X, R'\,;\,x.[w = a]w \xrightharpoonup{\downarrow X} R' \longrightarrow \exists T'.x + x.w \xrightarrow{\downarrow X} T' \ \wedge\ \forall n.\mathit{bisim}\ R'n\ T'n.$$

Another application of $\mathit{def}\mathcal{L}^{csu}$ gives us

$$w\,;\,\cdot \longrightarrow \exists T'.x + x.w \xrightarrow{\downarrow x} T' \ \wedge\ \forall n.\mathit{bisim}\ [w = a]w\ T'n. \tag{6.2}$$

Notice that w is not instantiated since it plays no active role in inferring the transition from $(x+x.w)$. Now we claim that there is no proof for this sequent. Suppose otherwise, then we must choose a T' to proceed with the proof. There are only two choices, either we instantiate T' with $\lambda n.w$ or $\lambda n.0$. The first choice leads to the premise $w\,;\,\cdot \longrightarrow$ *bisim* $[w = a]w\ x$ while the second leads to $w\,;\,\cdot \longrightarrow$ *bisim* $[w = a]w\ 0$. We first apply $\mathit{def}\mathcal{R}$ and other logical rules to the atom *bisim* $[w = a]w\ x$ to get the following premise (among other ones)

$$w, X', R''\,;\, x \xrightarrow{\downarrow X'} R'' \longrightarrow \exists T''.[w = a] \xrightarrow{\downarrow X'} T'' \ \wedge\ \forall n.\mathit{bisim}\ (R''n)\ (T''n)$$

Applying $\mathit{def}\mathcal{L}^{csu}$ once we get

$$w\,; \longrightarrow \exists T''.[w = a]w \xrightarrow{\downarrow x} T'' \ \wedge\ \forall n.\mathit{bisim}\ 0\ (T''n)$$

where X' is instantiated with x and R'' with $\lambda n.0$. Now notice that w is again not instantiated. This means that there is no way we can prove $[w = a] \xrightarrow{\downarrow x} T''$ for any T'', since w and a are different (recall that eigenvariables are *not* instantiated when applying any right-introduction rules).

The sequent $w\,;\,\cdot \longrightarrow$ *bisim* $([w = a]w)\ 0$ is not provable either since its proof would reduce to a proof of

$$w, Y, U\,;\,[w = a]w \xrightarrow{\downarrow Y} U \longrightarrow \exists V.0 \xrightarrow{\downarrow Y} V \ \wedge\ \forall n.\mathit{bisim}\ (Un)\ (Vn)$$

which in turns would reduce to (by an application of $\mathit{def}\mathcal{L}^{csu}$)

$$\cdot\,;\,\cdot \longrightarrow \exists V.0 \xrightarrow{\downarrow a} V \ \wedge\ \forall n.\mathit{bisim}\ 0\ (Vn).$$

In the latter sequent above, w gets instantiated to a. Clearly this sequent is not provable since the process 0 cannot perform any action. ∎

A complete encoding of late bisimulation

What seems to be the problem in Example 6.8 is that we need to do case analyses on the possible name w can take in the sequent (6.2) *before* we instantiate T'. More generally, our use of eigenvariable in encoding bound input forces lazy instantiation of the variable, i.e., only at the point when it is needed, e.g., in inferring the one-step transition (on the left-hand side of the sequent). We come to a simple solution to this problem by forcing an explicit enumeration of all names in the case involving bound input. For this to work, we need to maintain a list of free names of the processes being checked for their bisimilarity. More specifically, the bisimulation relation is encoded as the following predicate

$$\mathit{lbisim} \quad : \mathit{nlist} \rightarrow p \rightarrow p \rightarrow o.$$

where the type *nlist* denotes the list of names, with the usual constructors: nil for empty list and :: $\ :\ n \rightarrow \mathit{nlist} \rightarrow \mathit{nlist}$ for constructing a list from a name and another list. The relation $P \dot{\sim} Q$ is now encoded as *lbisim* $L\ P\ Q$ where L is a list of names containing free names in P and Q. The definition clause for *lbisim* is given in Figure 6.8. The predicate *lname* X L is used to enumerate all possible instantiation of X with the names in L. Notice that now the *lbisim* clause for input-prefix case has explicit assumptions about the possible name w can take, either it is in the list of free names in L, or it is a new name not in L, P or Q, which in this case is given by the quantifier ∇. Whenever we introduce a new name via ∇, the name has to be added to the list L. We refer to the definition clause given in Figure 6.8 and Figure 6.3 as $\mathbf{D}_{\pi,\dot{\sim}_l}$.

Theorem 6.9. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes. If the sequent*

$$. ; . \longrightarrow \bar{u} \triangleright lbisim\ L\bar{u}\ \mathtt{P}\ \mathtt{Q},$$

where L *is a ground term and the free names in* $\mathtt{P}$ *and* $\mathtt{Q}$ *are in the list* $L\,\bar{u}$*, is provable in* Linc *with the definition* $\mathbf{D}_{\pi,\dot{\sim}_l}$ *then* $\mathtt{P} \dot{\sim} \mathtt{Q}$.

Proof The proof is very similar to the proof of Theorem 6.7, that is, we show that the set

$$\mathcal{S} = \{(\mathtt{R},\mathtt{T}) \mid .; . \longrightarrow \bar{w} \triangleright lbisim\ L\,\bar{w}\ \mathtt{R}\ \mathtt{T}\}$$

where L is a ground term and the free names of $\mathtt{R}$ and $\mathtt{T}$ are included in L, is a bisimulation set. The non-trivial case is when $\mathtt{R}$ makes an input transition. In this case the proof of $\bar{w} \triangleright lbisim\ L\,\bar{w}\ \mathtt{R}\ \mathtt{T}$ reduces to

$$X, M\,;\, \bar{w} \triangleright \mathtt{R} \xrightarrow{\downarrow(X\bar{w})} M\,\bar{w} \longrightarrow \bar{w} \triangleright \exists N. \left[\begin{array}{l} \mathtt{T} \xrightarrow{\downarrow(X\bar{w})} N \\ \wedge\ \forall n. lname\ w\ L\,\bar{w} \supset lbisim\ L\,\bar{w}\ (M\,\bar{w}\,n)\ (N\,n) \\ \wedge\ \nabla n. lbisim\ (n :: L)\ M\bar{w}\ Nn \end{array}\right]$$

Suppose we are given a transition $\bar{w} \triangleright \mathtt{R} \xrightarrow{\downarrow\alpha} \mathtt{R}'$ then by cut and cut-elimination there is some term $\mathtt{T}'$ and name α such that the following sequents are provable.

(1) $\cdot ; \cdot \longrightarrow \bar{w} \triangleright \mathtt{T} \xrightarrow{\downarrow\alpha} \mathtt{T}'$

(2) $h\,;\, \bar{w} \triangleright lname\ h\bar{w}\ L\bar{w} \longrightarrow \bar{w} \triangleright lbisim\ L\bar{w}\ \ \mathtt{R}'(h\bar{w})\ \ \mathtt{T}'(h\bar{w})$

(3) $\cdot ; \cdot \longrightarrow \bar{w}n \triangleright lbisim\ (n :: L\bar{w})\ \mathtt{R}'n\ \mathtt{T}'n$

Since *def*$\mathcal{L}$, eq$\mathcal{L}$ and $\vee\mathcal{L}$ are asynchronous rules, we can rearrange the proof of (2) so that those left-rules are applied before any other rules. The rearranged proof is of the shape

$$\cfrac{\begin{array}{ccc} \Pi_{a_1} & & \Pi_{a_n} \\ \cdot ; \cdot \longrightarrow \bar{w} \triangleright lbisim\ L\bar{w}\ \ \mathtt{R}'a_1\ \ \mathtt{T}'a_1 & \cdots & \cdot ; \cdot \longrightarrow \bar{w} \triangleright lbisim\ L\bar{w}\ \ \mathtt{R}'a_n\ \ \mathtt{T}'a_n \end{array}}{h\,;\, \bar{w} \triangleright lname\ h\bar{w}\ L\bar{w} \longrightarrow \bar{w} \triangleright lbisim\ L\bar{w}\ \ \mathtt{R}'(h\bar{w})\ \ \mathtt{T}'(h\bar{w})}$$

where $a_1, \ldots, a_n$ are the names in $L\bar{w}$. We need to show that the continuation pairs $(\mathtt{R}'a, \mathtt{T}'a) \in \mathcal{S}$ for every a. If $a \in \{a_1, \ldots, a_n\}$ then $(\mathtt{R}'a, \mathtt{T}'a) \in \mathcal{S}$ by the proof Π_a.

$$
\begin{aligned}
\mathit{lbisim}\ L\ P\ Q \overset{\triangle}{=}\ & \forall A \forall P'\ [(P \xrightarrow{A} P') \supset \exists Q'.(Q \xrightarrow{A} Q') \land \mathit{lbisim}\ L\ P'\ Q'] \land \\
& \forall A \forall Q'\ [(Q \xrightarrow{A} Q') \supset \exists P'.(P \xrightarrow{A} P') \land \mathit{lbisim}\ L\ Q'\ P'] \land \\
& \forall X \forall P' \left[\begin{array}{l} (P \xrightharpoonup{\downarrow X} P') \supset \exists Q'.(Q \xrightharpoonup{\downarrow X} Q') \\ \land \forall w.\mathit{lname}\ x\ L \supset \mathit{lbisim}\ L\ (P'w)\ (Q'w) \\ \land \nabla w.\mathit{lbisim}\ (w :: L)\ (P'w)\ (Q'w) \end{array} \right] \land \\
& \forall X \forall Q' \left[\begin{array}{l} (Q \xrightharpoonup{\downarrow X} Q') \supset \exists P'.(P \xrightharpoonup{\downarrow X} P') \\ \land \forall w.\mathit{lname}\ x\ L \supset \mathit{lbisim}\ (Q'w)\ (P'w) \\ \land \nabla w.\mathit{lbisim}\ (w :: L)\ (P'w)\ (Q'w) \end{array} \right] \land \\
& \forall X \forall P' \left[\begin{array}{l} (P \xrightharpoonup{\uparrow X} P') \supset \exists Q'.(Q \xrightharpoonup{\uparrow X} Q') \\ \land \nabla w.\mathit{lbisim}\ (w :: L)\ (P'w)\ (Q'w) \end{array} \right] \land \\
& \forall X \forall Q' \left[\begin{array}{l} (Q \xrightharpoonup{\uparrow X} Q') \supset \exists P'.(P \xrightharpoonup{\uparrow X} P') \\ \land \nabla w.\mathit{lbisim}\ (w :: L)\ (Q'w)\ (P'w) \end{array} \right]
\end{aligned}
$$

$$
\mathit{lname}\ X\ (Y :: L) \overset{\triangle}{=} (X = Y) \lor \mathit{lname}\ X\ L
$$

Fig. 6.8. A complete encoding of strong late bisimulation

Otherwise, a is a fresh name not in $L\bar{w}$,$\mathtt{R}$, and $\mathtt{T}$. In this case, $(\mathtt{R}'a, \mathtt{T}'a) \in \mathcal{S}$ follows from the proof of the sequent (3). ∎

We now prove the converse of Theorem 6.9. For this we need a few more definitions and lemmas. The following definition makes use of multiset. We use $\sqsubseteq$ to denote multiset ordering. We write $\mathcal{P} \sqsubset \mathcal{Q}$ if the multiset $\mathcal{P}$ is strictly smaller than $\mathcal{Q}$, i.e., $\mathcal{P} \sqsubseteq Q$ and $S \neq \mathcal{Q}$.

Definition 6.10. *We define the measures pr and # on normal p-terms as follows.*

- *if P is either $\tau\ P'$, out $x\ y\ P'$ then $pr(P) = 1 + pr(P')$, and $\#(P) = 1 + \#(P')$,*
- *if P is in $x\ P$ then $pr(P) = 1 + pr(P\,y)$ and $\#(P) = 1 + \#(P\,y)$, for some y not free in P,*
- *if P is match $x\ y\ P'$ then $pr(P) = pr(P')$ and $\#(P) = 1 + \#(P')$,*
- *if P is $\nu\ P'$, then $pr(P) = pr(P'y)$ and $\#(P) = 1 + \#(P'y)$ for some y not free in P,*
- *if P is $P_1 + P_2$ or $P_1 \mid P_2$ then $pr(P) = pr(P_1) + pr(P_2)$ and $\#(P) = 1 + \#(P_1) + \#(P_2)$,*
- *otherwise $pr(P) = 0$ and $\#(P) = 0$.*

The measure pr counts the number of prefixes in the term and # counts the number of constructors (except $\mathtt{0}$*). We extend these measures to apply to one-step transition:*

$$pr(P \xrightarrow{A} Q) = pr(P \overset{A'}{\rightharpoonup} Q') = pr(P), \qquad \#(P \xrightarrow{A} Q) = \#(P \overset{A'}{\rightharpoonup} Q') = \#(P).$$

Given a formula B, the measure $\mathcal{M}(B)$ is the multiset

$$\{\#(P) \mid P \xrightarrow{A} Q \text{ or } P \overset{A}{\rightharpoonup} Q \text{ is a subformula occurrence in } B\},$$

where each occurrence of P in B is counted only once. The measure $\mathcal{M}$ is extended to judgments and sequents as follows.

$$\mathcal{M}(\bar{x} \triangleright B) = \mathcal{M}(B), \qquad \mathcal{M}(\Gamma) = \bigcup\{\mathcal{M}(J) \mid J \in \Gamma\}.$$

Lemma 6.11. *If a process* $\mathtt{P}$ *makes a transition to* $\mathtt{Q}$ *then* $pr(\mathtt{P}) > pr(\mathtt{Q})$.

Proof By induction on the π-derivation of $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$. ∎

Definition 6.12. *An occurrence of a one-step-transition predicate $P \xrightarrow{X} Q$ (likewise, $P \overset{X}{\rightharpoonup} Q$) in a judgment $\bar{w} \triangleright C$ is a* ground-head *occurrence if the free variables in P are either in $\bar{w}$ or it is bound by a ∇ quantifier in C. Let $\mathcal{C}$ be a judgment in which the only occurrences of predicate symbols are the one-step-transition predicates. If every* *occurrence of the one-step predicate in $\mathcal{C}$ is ground-head then we say that $\mathcal{C}$ is a* ground-head judgment*. A sequent $\Gamma \longrightarrow \mathcal{C}$ is a ground head sequent if all judgments in Γ are ground head judgments.*

LEMMA 6.13. *Let $\mathcal{A}$ be a ground head judgment. Then for any substitution θ, $\mathcal{A}\theta$ is a ground head judgment and $\mathcal{M}(\mathcal{A}) = \mathcal{M}(\mathcal{A}\theta)$.*

LEMMA 6.14. *Let $\mathcal{A}$ be a ground head judgment. For any raised definition clause $\mathcal{H} \overset{\triangle}{=} \mathcal{B}$ in $\mathbf{D}_{\pi,\dot{\sim}_l}$, if $\mathcal{A}\theta = \mathcal{H}\theta$ then $\mathcal{M}(\mathcal{B}\theta) \sqsubset \mathcal{M}(\mathcal{A})$*

Proof By case analyses on the definition clauses in Figure 6.3 and Lemma 6.13. ∎

In the following we shall need a notion of *partial derivation.* A partial derivation is constructed inductively as derivations, but we do not require that every leaf to be closed, i.e., some leaves may be instances of rules with one or more premises. We call a branch in a partial derivation an *open branch* if its leaf is an instance of a rule with one or more premise, otherwise it is closed. Obviously, a derivation is a partial derivation in which all branches are closed. We then define a partial proof strategy for constructing partial derivations for the particular case of *lbisim* .

DEFINITION 6.15. *The lbisim-(partial) proof strategy for a given sequent is defined as follows.*

Step 1 *Given a sequent $\Gamma \longrightarrow \mathcal{C}$, apply the introduction rules $\exists\mathcal{L}$, $\nabla\mathcal{L}$ and $\wedge\mathcal{L}$ until neither of them is applicable. Let $\Gamma' \longrightarrow \mathcal{C}$ be the resulting sequent.*

Step 2 *Apply* def$\mathcal{L}^{csu}$ *to some atomic judgment in Γ', if there is any, otherwise the strategy terminates. For every premise sequent $\Gamma'' \longrightarrow \mathcal{C}'$, repeat Step 1.*

This strategy is non-terminating in general, however, we see that in a specific case it is terminating, although the resulting partial derivation may not be a valid derivation.

LEMMA 6.16. *Let $\mathcal{S}$ be either the ground head sequent*

$$P', A \,;\, \bar{x} \triangleright P \xrightarrow{A\bar{x}} P'\bar{x} \longrightarrow \mathcal{C} \qquad \text{or} \qquad M, X \,;\, \bar{x} \triangleright P \xrightharpoonup{X\bar{x}} M\bar{x} \longrightarrow \mathcal{C},$$

where $\bar{x}$ are of type n. Then the following statements hold.

(*a*) *The application of lbisim strategy on $\mathcal{S}$ terminates.*

(*b*) *Let Π be the resulting partial derivation after applying the lbisim-strategy. Suppose there is an open branch Ψ in Π. Let $\theta = \theta_1 \circ \ldots \circ \theta_n$ where θ_i is the substitution applied to the premise of of the i-th instance of* def$\mathcal{L}^{csu}$ *in Ψ. Then $P'\theta$ and $A\theta$ (likewise, $M\theta$ and $X\theta$) are closed terms and either one of the sequents*

$$\cdot\,;\,\cdot \longrightarrow (\bar{x} \triangleright P \xrightarrow{A\bar{x}} P'\bar{x})\theta$$
$$\cdot\,;\,\cdot \longrightarrow (\bar{x} \triangleright P \xrightharpoonup{X\bar{x}} M\bar{x})\theta$$

is provable.

Proof As a consequence of Lemma 6.13 and Lemma 6.14, we observe the following invariant in the *lbisim*-strategy: given a ground head sequent $\Gamma \longrightarrow \mathcal{C}$, the resulting sequents from applying *lbisim*-strategy are all ground head sequents. Let size(Γ) be the number of logical connectives in Γ. To show the termination of the *lbisim*-strategy we assign the measure

$$\langle \mathcal{M}(\Gamma), \mathrm{size}(\Gamma) \rangle$$

to a given ground-head sequent $\Gamma \longrightarrow \mathcal{C}$ and show that each step of the strategy produces sequents of smaller measure. Step 1 removes logical connectives and does not introduce any new occurrence of one-step predicate, therefore it clearly reduces size(Γ) while $\mathcal{M}(\Gamma)$ remains unchanged. Step 2 reduces $\mathcal{M}(\Gamma)$ by Lemma 6.13 and Lemma 6.14. Since we start with a fixed ground head sequent, applying this strategy eventually terminates.

The proof of (*b*) is by induction on the length of Ψ. We can write each sequent appearing in the branch Ψ *below* the $i+1$-th application of $def\mathcal{L}^{csu}$ as

$$\Gamma\theta_1 \circ \cdots \circ \theta_i \longrightarrow \mathcal{C}\theta_1 \circ \cdots \theta_i.$$

Let us denote the substitution $\theta_1 \circ \ldots \circ \theta_i$ with θ^i. We claim that for any sequent $\Gamma\theta^i \longrightarrow \mathcal{C}\theta^i$ in Ψ and any judgment $\mathcal{A}\theta^i \in \Gamma\theta^i$, $\mathcal{A}\theta$ is a ground judgment (i.e., no occurrence of eigenvariables) and there is a derivation Ξ of the sequent $\cdot \longrightarrow \mathcal{A}\theta$. In the base case, we have an instance of $def\mathcal{L}^{csu}$ (we show here only the premise that is in Ψ)

$$\frac{\top, \top, \ldots, \top \longrightarrow \mathcal{C}\theta^{n-1} \circ \theta_n}{\mathcal{A}\theta^{n-1}, \top, \ldots, \top \longrightarrow \mathcal{C}\theta^{n-1}}\ def\mathcal{L}^{csu}$$

where $\mathcal{A}\theta^{n-1}\theta_n = \mathcal{A}\theta = \mathcal{H}\theta_n$ for some raised definition clause $\mathcal{H} \stackrel{\triangle}{=} \bar{w} \triangleright \top$. By inspection on the definition clauses of $\mathbf{D}_{\pi,\dot{\sim}_l}$ we see that this instance of $def\mathcal{L}^{csu}$ must have used either one of the definition clauses

$$\tau\ M \xrightarrow{\tau} P \stackrel{\triangle}{=} \top, \quad in\ X\ M \xrightharpoonup{\downarrow X} M \stackrel{\triangle}{=} \top, \quad \text{or} \quad out\ x\ y\ P \xrightarrow{\uparrow xy} P' \stackrel{\triangle}{=} \top$$

or their raised versions. Since $\mathcal{A}\theta^{n-1}$ is a ground-head judgment, applying $def\mathcal{L}^{csu}$ to it using the above clauses will fully instantiate the variables in $\mathcal{A}$ with ground terms, and therefore $\mathcal{A}\theta$ is a ground judgment. We then take Ξ to be the derivation

$$\dfrac{\dfrac{}{\longrightarrow \top}\ \top\mathcal{R}}{\longrightarrow \mathcal{A}\theta}\ def\mathcal{R}^{=}\ .$$

For the inductive cases we have either $def\mathcal{L}^{csu}$, $\wedge\mathcal{L}*$, $\nabla\mathcal{L}$, or $\exists\mathcal{L}$. We assume that the free variables in θ^i and $\Gamma\theta^i$ are different from the new eigenvariables introduced above the sequent $\Gamma\theta^i \longrightarrow \mathcal{C}\theta^i$ and different from the bound variables in the sequent.

1. Suppose $def\mathcal{L}^{csu}$ is applied to $\Gamma\theta^i$

$$\dfrac{\begin{array}{c}\Psi' \\ \mathcal{B}\theta_{i+1}, \Gamma'\theta^i \circ \theta_{i+1} \longrightarrow \mathcal{C}\theta^i \circ \theta_{i+1}\end{array}}{\mathcal{A}\theta^i, \Gamma'\theta^i \longrightarrow \mathcal{C}\theta^i}\ def\mathcal{L}^{csu}$$

for some raised definition clause $\mathcal{H} \overset{\triangle}{=} \mathcal{B}$ such that $\mathcal{H}\theta_{i+1} = \mathcal{A}\theta^i \circ \theta_{i+1}$. Since we can choose the variables in $\mathcal{H}$ to be different from the free variables in θ^i and in the sequent, we have $\mathcal{B}\theta_{i+1} = \mathcal{B}\theta^i \circ \theta_{i+1}$. By induction hypothesis we know that $\mathcal{B}\theta$ is a ground term and there is a derivation of Ξ' of $\cdot \longrightarrow \mathcal{B}\theta$. We see that in most definition clauses for the one-step transition, the variables that appear in the head of the clause also appear in the body. In these cases, clearly $\mathcal{H}\theta$ is a ground judgment and hence $\mathcal{A}\theta$ is also a ground jugdment. The exceptions are with the par- and sum-rules, e.g.,

$$R + T \xrightarrow{Y} R' \overset{\triangle}{=} R \xrightarrow{Y} R'.$$

But since $\mathcal{A}\theta^i$ is a ground-head judgment, unifying $\mathcal{A}\theta^i$ with the above clause will instantiate R and T with ground terms, while the variable $R'\theta$ is a ground term following from the induction hypothesis. Therefore $\mathcal{A}\theta$ is also a ground judgment. We construct the derivation Ξ as follows

$$\dfrac{\begin{array}{c}\Xi' \\ \cdot \longrightarrow \mathcal{B}\theta\end{array}}{\cdot \longrightarrow \mathcal{A}\theta}\ def\mathcal{R}^{=}.$$

The above instance of $def\mathcal{R}^{=}$ is valid since we have $\mathcal{A}\theta^i \circ \theta_{i+1} = \mathcal{H}\theta^i \circ \theta_{i+1}$ and hence $\mathcal{A}\theta = \mathcal{H}\theta_{i+1} \circ \ldots \circ \theta_n$.

2. Suppose $\wedge\mathcal{L}^*$ is applied to $\Gamma\theta^i$

$$\dfrac{\begin{array}{c}\Psi' \\ \mathcal{B}_1\theta^i, \mathcal{B}_2\theta^i, \Gamma'\theta^i \longrightarrow \mathcal{C}\theta^i\end{array}}{\mathcal{B}_1\theta^i \wedge \mathcal{B}_2\theta^i, \Gamma'\theta^i \longrightarrow \mathcal{C}\theta^i}\ \wedge\mathcal{L}^*$$

Then we construct Ξ as follows

$$\dfrac{\begin{array}{c}\Xi_1 \\ \cdot \longrightarrow \mathcal{B}_1\theta\end{array} \quad \begin{array}{c}\Xi_2 \\ \cdot \longrightarrow \mathcal{B}_2\theta\end{array}}{\cdot \longrightarrow \mathcal{B}_1\theta \wedge \mathcal{B}_2\theta}\ \wedge\mathcal{R}$$

where Ξ_1 and Ξ_2 are obtained from induction hypothesis. The judgments $\mathcal{B}_1\theta$ and $\mathcal{B}_2\theta$ are both ground judgments by induction hypothesis, so obviously $\mathcal{B}_1\theta \wedge \mathcal{B}_2\theta$ is also a ground jugdment.

3. Suppose $\exists\mathcal{L}$ is applied to $\Gamma\theta^i$

$$\dfrac{\begin{array}{c}\Psi' \\ \bar{u} \rhd B\theta^i\,(h\bar{u}), \Gamma'\theta^i \longrightarrow \mathcal{C}\theta^i\end{array}}{\bar{u} \rhd \exists y.B\theta^i\,y, \Gamma'\theta^i \longrightarrow \mathcal{C}\theta^i}\ \exists\mathcal{L}$$

By induction hypothesis we get a derivation Ξ' of $\cdot \longrightarrow \bar{u}\rhd(B\,(h\bar{u}))\theta$ where $(B\,(h\bar{u}))$ is a ground judgment. The terms $(\lambda\bar{u}.B)\theta$ and $h\theta$ are ground terms, so obviously $\bar{u} \rhd \exists y.B\theta\,y$ is a ground judgment. We therefore construct Ξ as follows

$$\dfrac{\begin{array}{c}\Xi' \\ \cdot \longrightarrow \bar{u} \rhd (B\,(h\bar{u}))\theta\end{array}}{\cdot \longrightarrow \bar{u} \rhd \exists y.B\theta\,y}\ \exists\mathcal{R}$$

4. Suppose $\nabla\mathcal{L}$ is applied to $\Gamma\theta^i$

$$\dfrac{\begin{array}{c}\Psi' \\ \bar{u}y \rhd (By)\theta^i, \Gamma'\theta^i \longrightarrow \mathcal{C}\theta^i\end{array}}{u \rhd \nabla y.(By)\theta^i, \Gamma'\theta^i \longrightarrow \mathcal{C}\theta^i}\ \nabla\mathcal{L}$$

By induction hypothesis we have a derivation Ξ' of $\cdot \longrightarrow \bar{u}y \rhd (By)\theta$, where $\bar{u}y \rhd (By)\theta$ is a ground judgment. Obviously, $\bar{u}y \rhd \nabla y.(By)\theta$ is a ground judgment. We therefore take Ξ to be the derivation

$$\dfrac{\begin{array}{c}\Xi' \\ \cdot \longrightarrow \bar{u}y \rhd (By)\theta\end{array}}{\cdot \longrightarrow \bar{u} \rhd \nabla y.(By)\theta}\ \nabla\mathcal{R}$$

∎

We note that the use ∇ to encode restriction makes it possible to dualize an open branch of the partial derivation to obtain a valid derivation. If we had used $\forall$, which is not an asynchronous connective on the left, we would not be able to dualize the proof, since on the left-side we would have been required to instantiate it with some name (could be already occurring in the sequent), while on the right we would have been forced to use a fresh eigenvariable.

Theorem 6.17. *If* $\mathtt{P} \overset{\cdot}{\sim} \mathtt{Q}$ *then for any finite list* L *such that* L *contains the free names of* $\mathtt{P}$ *and* $\mathtt{Q}$*, the sequent*

$$\cdot\,;\,\cdot \longrightarrow \mathit{lbisim}\ L\ \mathtt{P}\ \mathtt{Q}$$

is provable in Linc *with the definition* $\mathbf{D}_{\pi,\overset{\cdot}{\sim}_l}$.

Proof The proof is by induction on the measure $\#(\texttt{P}) + \#(\texttt{Q})$. We attempt to prove *lbisim* L $\texttt{P}$ $\texttt{Q}$ by applying $\mathit{def}\mathcal{R}$ followed by instances of $\wedge\mathcal{R}$,$\forall\mathcal{R}$ and $\supset\mathcal{R}$. We get six premises. We show here the construction of the derivations for three of them, the derivations for the other three can be constructed analogously.

$$\begin{array}{ll}
(a) & P', A\,;\, \texttt{P} \xrightarrow{A} P' \longrightarrow \exists Q'.\texttt{Q} \xrightarrow{A} Q' \;\wedge\; \mathit{lbisim}\ L\ P'\ Q' \\
(b) & M, X\,;\, \texttt{P} \xrightarrow{\downarrow X} M \longrightarrow \exists N.\ \texttt{Q} \xrightarrow{\downarrow X} N \;\wedge\; \forall w.\mathit{lname}\ w\ L \supset \mathit{lbisim}\ L\ Mw\ Nw \\
 & \qquad\qquad\qquad\qquad\qquad \wedge\; \nabla w.\mathit{lbisim}\ (w :: L)\ Mw\ Nw \\
(c) & M, X\,;\, \texttt{P} \xrightarrow{\uparrow X} M \longrightarrow \exists N.\texttt{Q} \xrightarrow{\uparrow X} N \;\wedge\; \nabla w.\mathit{lbisim}\ (w :: L)\ Mw\ Nw
\end{array}$$

We apply the *lbisim*-strategy to the above sequents. Let Π_a, Π_b and Π_c be the partial derivations for (a), (b) and (c), respectively. In all cases, if there are no open branches then we are done. Otherwise, suppose there is an open branch Ψ_a in Π_a. The leaf of Ψ_a has the judgment

$$\cdot\,;\,\cdot \longrightarrow \exists Q'.\texttt{Q} \xrightarrow{A\theta} Q' \;\wedge\; \mathit{lbisim}\ L\ P'\theta\ Q' \tag{6.3}$$

in the premise, where θ is the composition of all substitutions appearing in Ψ_a. Since (a) is a ground-head sequent, by Lemma 6.16 there is a derivation of $\cdot\,;\,\cdot \longrightarrow \texttt{P} \xrightarrow{A\theta} P'\theta$. That is, $\texttt{P}$ makes an $A\theta$-transition to $P'\theta$. Hence, by the definition of bisimulation, there is a transition from $\texttt{Q} \xrightarrow{A\theta} R$ such that $P'\theta \dot{\sim} R$, for some process R. By Proposition 6.4 there is a proof Ξ of $\texttt{Q} \xrightarrow{A\theta} R$. From Lemma 6.11 we know that $pr(P'\theta) + pr(R) < pr(\texttt{P}) + pr(\texttt{Q})$, and therefore we can apply the induction hypothesis to get a derivation Π' of *lbisim* L $P'\theta$ R. We thus complete the proof for sequent (6.3) as follows.

$$\dfrac{\dfrac{\begin{matrix}\Xi \\ \longrightarrow \texttt{Q} \xrightarrow{A\theta} R\end{matrix} \qquad \begin{matrix}\Pi' \\ \longrightarrow \mathit{lbisim}\ L\ P'\theta\ R\end{matrix}}{\longrightarrow \texttt{Q} \xrightarrow{A\theta} R \;\wedge\; \mathit{lbisim}\ L\ P'\theta\ R}\ \wedge\mathcal{R}}{\longrightarrow \exists Q'.\texttt{Q} \xrightarrow{A\theta} Q' \;\wedge\; \mathit{lbisim}\ L\ P'\theta\ Q'}\ \exists\mathcal{R}$$

The sequent (b) is proved in a similar fashion. In this case, however, we are given a derivation of *lbisim* L' Ma Na for any name a and any finite list of names L', and we need to construct a derivation for each of the sequents

$$w\,;\,\mathit{lname}\ w\ L \longrightarrow \mathit{lbisim}\ L\ Mw\ Nw \ \text{ and } \ \cdot\,;\,\cdot \longrightarrow w \rhd \mathit{lbisim}\ (w :: L)\ Mw\ Nw.$$

The first is proved by unwinding the predicate *lname* w L by applying $\mathit{def}\mathcal{L}$, $\vee\mathcal{L}$ and eq$\mathcal{L}$. This will result in the premises

$$\longrightarrow \mathit{lbisim}\ L\ Ma_1\ Na_1, \cdots, \longrightarrow \mathit{lbisim}\ L\ Ma_k\ Na_k$$

where $a_1, \ldots, a_k$ are the names in L. These premises are provable from induction hypothesis. The derivation for the sequent $\longrightarrow w \triangleright lbisim\ (w :: L)\ Mw\ Nw$ is obtained from induction hypothesis, i.e., a derivation of $\longrightarrow lbisim\ (c :: L)\ Mc\ Nc$ for some name (constant) c not in L, and then applying Lemma 3.13 to weaken the scope of c to local context. The proof for (c) is similar to this case. ∎

6.4 Strong congruence and distinction

It is a well-known fact that the late bisimulation relation is not a congruence since it is not preserved by input prefix. For a simple example, consider the processes $P = (x|\bar{y})$ and $Q = (x.\bar{y} + \bar{y}.x)$. P and Q are bisimilar but $x(y).P \not\dot{\sim} x(y).Q$ since in the latter case we need to show that for every name a, $P[a/y] \dot{\sim} Q[a/y]$. In particular, a could be x, and hence P can make a τ-transition which cannot be simulated by Q. Part of the reason why the congruence property fails for $\dot{\sim}$ is that π-calculus makes no syntactic distinction between instantiable names and non-instantiable names. In Linc this distinction is made by the use of eigenvariables to encode the former and constants (or variables in local context) to encode the latter. In π-calculus, this lack of syntactic distinction is remedied by considering a variant of bisimulation relation which is closed under all substitutions of names. The resulting relation is shown in [40] to be a congruence. The precise definition follows.

Definition 6.18. *[40] The processes* $\mathtt{P}$ *and* $\mathtt{Q}$ *are* strong late congruent*, written* $\mathtt{P} \sim \mathtt{Q}$*, if* $\mathtt{P}\theta \dot{\sim} \mathtt{Q}\theta$ *for every (name) substitution* θ*.*

In the above definition, strictly speaking we do not need to consider all substitutions. It is enough to consider substitutions the domain of which is included in the set of free variables of $\mathtt{P}$ and $\mathtt{Q}$. To encode strong congruence in Linc we therefore interpret the quantification over "all substitutions" as simply universal quantification over free names in P and Q. This is true for both *bisim* and *lbisim*. The soundness of this encoding follows straightforwardly from the property of universal quantifier and Theorem 6.7 and Theorem 6.9.

Theorem 6.19. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes and let* $\bar{y}$ *be the free names in* $\mathtt{P}$ *and* $\mathtt{Q}$*. If* $\forall \bar{y}.bisim\ \mathtt{P}\ \mathtt{Q}$ *is provable in* Linc *with the definition* $\mathbf{D}_{\pi,\dot{\sim}}$ *then* $\mathtt{P} \sim \mathtt{Q}$*.*

Theorem 6.20. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes and let* $\bar{y}$ *be the free names in* $\mathtt{P}$ *and* $\mathtt{Q}$*. Let* L *be a list of names containing only those in* $\bar{y}$*. If* $\forall \bar{y}.lbisim\ L\ \mathtt{P}\ \mathtt{Q}$ *is provable in* Linc *with the definition* $\mathbf{D}_{\pi,\dot{\sim}_l}$ *then* $\mathtt{P} \sim \mathtt{Q}$*.*

The notion of bisimulation can be further refined by explicitly specifying the distinction among names, e.g., certain names remain distinct after substitutions. This notion is formally defined as follows.

Definition 6.21. *[40] A* distinction *is a finite symmetric and irreflexive relation on names. A substitution* θ respects *a distinction* $\mathtt{D}$ *if* $(x, y) \in \mathtt{D}$ *implies* $x\theta \neq y\theta$*.*

Definition 6.22. *[40] Let* $\mathtt{D}$ *be a distinction. Then* $\mathtt{P}$ *and* $\mathtt{Q}$ *are* strong $\mathtt{D}$-bisimilar*, written* $\mathtt{P} \sim^{\mathtt{D}} \mathtt{Q}$ *if* $\mathtt{P}\theta \dot{\sim} \mathtt{Q}\theta$ *for each* θ *that respects* $\mathtt{D}$.

It is enough to consider distinctions which mention only the free variables in $\mathtt{P}$ and $\mathtt{Q}$ when checking $\mathtt{P} \sim^{\mathtt{D}} \mathtt{Q}$. That is, we have $\mathtt{P} \sim^{\mathtt{D}} \mathtt{Q}$ if and only if $\mathtt{P} \sim^{\mathtt{D}'} \mathtt{Q}$ where $\mathtt{D}' = \mathtt{D} \cap (\mathrm{fn}(\mathtt{P},\mathtt{Q}) \times \mathrm{fn}(\mathtt{P},\mathtt{Q}))$. A class of distinctions that often arises in practice is of the form $(\mathtt{N} \times \mathtt{N}) - \mathtt{I}$ where $\mathtt{N}$ is a finite set of names and $\mathtt{I}$ is the identity relation on names. Let us call such a distinction as *complete distinction.* We abbreviate a complete distinction $(\mathtt{N} \times \mathtt{N}) - \mathtt{I}$ as simply $\mathtt{N}$. It is obvious that any substitution θ that respects $\mathtt{N}$ and whose domain is a subset of $\mathtt{N}$ is a renaming substitution. This suggests that we can encode $\sim^{\mathtt{N}}$ by quantifying the names in $\mathtt{N}$ with ∇ and the rest free variables with $\forall$. The precise statements are given in the following two theorems. The proofs of the theorems make use of the following result (which can be proved analogously to the proofs of Theorem 6.7 and Theorem 6.9):

- If $\nabla\bar{y}.\mathit{bisim}\ \mathtt{P}\ \mathtt{Q}$ is provable then $\mathtt{P}\theta \dot{\sim} \mathtt{Q}\theta$ for every renaming substitution θ such that $\mathrm{dom}(\theta) \subseteq \{\bar{y}\}$.
- If $\nabla\bar{y}.\mathit{lbisim}\ L\ \mathtt{P}\ \mathtt{Q}$ is provable then $\mathtt{P}\theta \dot{\sim} \mathtt{Q}\theta$ for every renaming substitution θ such that $\mathrm{dom}(\theta) \subseteq \{\bar{y}\}$.

Theorem 6.23. *Let* $\mathtt{N} = \{x_1, \ldots, x_n\}$ *be a complete distinction. Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes and let* $\bar{y}$ *be a list of free names in* $\mathtt{P}$ *and* $\mathtt{Q}$ *but not in* N*. If* $\nabla x_1 \ldots \nabla x_n \forall\bar{y}.\mathit{bisim}\ \mathtt{P}\ \mathtt{Q}$ *is provable with the definition* $\mathbf{D}_{\pi,\dot{\sim}}$ *then* $\mathtt{P} \sim^{\mathtt{N}} \mathtt{Q}$.

Theorem 6.24. *Let* $\mathtt{N} = \{x_1, \ldots, x_n\}$ *be a complete distinction. Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes and* L *be a finite list of names containing the free names in* $\mathtt{P}$ *and* $\mathtt{Q}$*, and let* $\bar{y}$ *be a list of free names in* $\mathtt{P}$ *and* $\mathtt{Q}$ *but not in* N*. If* $\nabla x_1 \ldots \nabla x_n \forall\bar{y}.\mathit{lbisim}\ L\ \mathtt{P}\ \mathtt{Q}$ *is provable with the definition* $\mathbf{D}_{\pi,\dot{\sim}_l}$ *then* $\mathtt{P} \sim^{\mathtt{N}} \mathtt{Q}$.

$$\frac{\mathtt{P} \xrightarrow{\alpha} \mathtt{P}'}{!\mathtt{P} \xrightarrow{\alpha} \mathtt{P}' \mid !\mathtt{P}}\ !\mathrm{act}, \mathrm{bn}(\alpha \cap \mathrm{fn}(P) = \emptyset)$$

$$\frac{\mathtt{P} \xrightarrow{\bar{x}y} \mathtt{P}' \quad P \xrightarrow{x(z)} \mathtt{P}''}{!\mathtt{P} \xrightarrow{\tau} (\mathtt{P}' \mid \mathtt{P}''[y/z]) \mid !P}\ !\mathrm{comm} \qquad \frac{P \xrightarrow{\bar{x}(z)} \mathtt{P}' \quad P \xrightarrow{x(z)} \mathtt{P}''}{P \xrightarrow{\tau} (z)(\mathtt{P}' \mid \mathtt{P}'') \mid !P}\ !\mathrm{close}$$

Fig. 6.9. The late transition rules for π-calculus with replication

$$\frac{P \xrightarrow{A} P'}{!P \xrightarrow{A} P' \mid !P}\ !\mathrm{act} \qquad \frac{P \overset{X}{\rightharpoondown} M}{!P \overset{X}{\rightharpoondown} \lambda y(My \mid !P)}\ !\mathrm{act}$$

$$\frac{P \xrightarrow{\uparrow xy} P' \quad P \overset{\downarrow x}{\rightharpoondown} M}{!P \xrightarrow{\tau} (P' \mid My)}\ !\mathrm{comm} \qquad \frac{P \overset{\uparrow x}{\rightharpoondown} M \quad P \overset{\downarrow x}{\rightharpoondown} N}{P \xrightarrow{\tau} \nu z.(Mz \mid Nz) \mid !P}\ !\mathrm{close}$$

Fig. 6.10. The late transition rules for π-calculus with replication (HOAS)

6.5 π-calculus with replication

We now consider an extension to the finite π-calculus which will allow us to represent non-terminating processes. There are at least two ways of encoding non-terminating processes in π-calculus, e.g., via recursive definitions or replications [47]. We consider here the latter approach since it leads to a simpler presentation of the operational semantics. To the syntax of finite π-calculus we add the process expression $!P$. The processes $!P$ can be understood as the infinite parallel composition of P, i.e., $P|P|\cdots|P|\cdots$. Thus it is possible to have a process which retains a copy of itself after making a transition, e.g., $!P \xrightarrow{\alpha} P'|!P$. The operational semantics for one-step transitions of the π-calculus with replication (in HOAS) is given in Figure 6.2 plus the additional rules in Figure 6.10. We use the same symbol to encode replication in HOAS, i.e., $! : p \to p$. The original specification, taken from [47], is given in Figure 6.9

Let us denote the definition clauses for π-calculus with replication with $\mathbf{D}_{!\pi}$. The definition $\mathbf{D}_{!\pi,\dot\sim}$ denotes the definition $\mathbf{D}_{!\pi}$ augmented with the *bisim* clause, and $\mathbf{D}_{!\pi,\dot\sim_l}$ denotes $\mathbf{D}_{!\pi}$ augmented with *lbisim* clause. Note that the *bisim* and *lbisim* clauses have now to be marked as co-inductive definitions in order for the co-induction rules to be applicable.

Example 6.25. Let `P` $=!(z)(\bar{z}a \mid z(y).\bar{x}y)$ and `Q` $=!\tau.\bar{x}a$. The only action `P` can make is the silent action τ since the channel z is restricted internally within the process. It is easy to see that `P` $\xrightarrow{\tau} \bar{x}a \mid$ `P`. That is, the continuation of P is capable of outputting a free name a or making a silent transition. Obviously Q can make the same τ action and results in a bisimilar continuation. Let us try to prove *bisim* `P Q`. The proof strategy as in the finite case (i.e., unfolding the *bisim* clause via *def*$\mathcal{R}$ and applying the necessary introduction rules for logical connectives) will not work here. It is easy to check that applying this strategy, after the first *def*$\mathcal{R}$ on *bisim* (but before the second *def*$\mathcal{R}$ on *bisim*) we arrive at the sequent *bisim* $((z)(0 \mid \bar{x}a) \mid$ `P`$)$ $(\bar{x}a \mid$ `Q`$)$. Since `P` and `Q` still occur in the continuation pair, it is obvious that this strategy is non terminating. We need to use the co-induction proof rules instead.

The informal proof as it is done in π-calculus starts by finding a bisimulation set $\mathcal{S}$ such that $(\mathtt{P},\mathtt{Q}) \in \mathcal{S}$. Let $\mathtt{R}_i$, for any natural number i, be either $(z)(0 \mid \bar{x}a)$ or $(z)(0 \mid 0)$, and $\mathtt{T}_i$ be either $\bar{x}a$ or 0. and let $\mathcal{S}' = \{(\mathtt{R}_1 \mid \cdots \mid \mathtt{R}_n \mid \mathtt{P}, \mathtt{T}_1 \mid \cdots \mid \mathtt{T}_n \mid \mathtt{Q}) \mid n \geq 0\}$. Define $\mathcal{S}$ to be the symmetric closure of $\mathcal{S}'$. It can be verified that $\mathcal{S}$ is a bisimulation set by showing the set is closed with respect to one-step transitions. To prove this formally in Linc we need someway to represent the set $\mathcal{S}$. We code the set $\mathcal{S}$ as the following inductive definition.

$$\mathit{inv}\ \mathtt{P}\ \mathtt{Q} \stackrel{\mu}{=} \top. \quad \mathit{inv}\ \mathtt{Q}\ \mathtt{P} \stackrel{\mu}{=} \top.$$

$$\begin{aligned}
\mathit{inv}\ ((z)(0 \mid 0) \mid M)\ (0 \mid N) &\stackrel{\mu}{=} \mathit{inv}\ M\ N. \\
\mathit{inv}\ (0 \mid N)\ ((z)(0 \mid 0) \mid M) &\stackrel{\mu}{=} \mathit{inv}\ N\ M. \\
\mathit{inv}\ ((z)(0 \mid \bar{x}a) \mid M)\ (\bar{x}a \mid N) &\stackrel{\mu}{=} \mathit{inv}\ M\ N. \\
\mathit{inv}\ (\bar{x}a \mid N)\ ((z)(0 \mid \bar{x}a) \mid M) &\stackrel{\mu}{=} \mathit{inv}\ N\ M.
\end{aligned}$$

Note that the resulting definition from adding *inv* to $\mathbf{D}_{!\pi,\dot{\sim}}$ can be stratified by assigning a level to *inv* such that $\mathrm{lvl}(\mathit{inv}) < \mathrm{lvl}(\mathit{bisim})$. The set of pairs encoded by *inv* can be shown to be symmetric, i.e., the formula $\forall R \forall T. \mathit{inv}\ R\ T \supset \mathit{inv}\ T\ R$ is provable inductively (using the same formula as the induction invariant).

We then apply the $\nu\mathcal{R}$ rule to *bisim* $\mathtt{P}\ \mathtt{Q}$, using the predicate *inv* as the invariant. This application of $\nu\mathcal{R}$ yields the following two premises: $\longrightarrow \mathit{inv}\ \mathtt{P}\ \mathtt{Q}$ and

$$R, T\,;\ \mathit{inv}\ R\ T \longrightarrow B\,R\,T$$

where $B\,R\,T$ is the formula

$$\begin{aligned}
&\forall A \forall P'\ [(R \xrightarrow{A} R') \supset \exists Q'.(T \xrightarrow{A} T') \wedge \mathit{inv}\ R'\ T'] \wedge \\
&\forall A \forall T'\ [(T \xrightarrow{A} T') \supset \exists R'.(R \xrightarrow{A} R') \wedge \mathit{inv}\ T'\ R'] \wedge \\
&\forall X \forall R'\ [(P \xrightharpoonup{\downarrow X} R') \supset \exists T'.(T \xrightharpoonup{\downarrow X} T') \wedge \forall w.\mathit{inv}\ (R'w)\ (T'w)] \wedge \\
&\forall X \forall T'\ [(T \xrightharpoonup{\downarrow X} T') \supset \exists R'.(R \xrightharpoonup{\downarrow X} R') \wedge \forall w.\mathit{inv}\ (R'w)\ (R'w)] \wedge \\
&\forall X \forall R'\ [(R \xrightharpoonup{\uparrow X} R') \supset \exists T'.(T \xrightharpoonup{\uparrow X} T') \wedge \nabla w.\mathit{inv}\ (R'w)\ (T'w)] \wedge \\
&\forall X \forall T'\ [(T \xrightharpoonup{\uparrow X} T') \supset \exists R'.(R \xrightharpoonup{\uparrow X} R') \wedge \nabla w.\mathit{inv}\ (T'w)\ (R'w)]
\end{aligned}$$

The sequent reads, intuitively, that the set defined by *inv* is symmetric and is closed under one-step transitions. This is proved by induction on *inv*. Formally, this is done by applying $\mu\mathcal{L}$ to *inv* $R\ T$, using the invariant

$$\lambda R \lambda T. \mathit{inv}\ R\ T \supset B\,R\,T.$$

The sequents corresponding to the base cases of the induction are

$$\mathit{inv}\ \mathtt{P}\ \mathtt{Q} \longrightarrow B\,\mathtt{P}\,\mathtt{Q} \quad \text{and} \quad \mathit{inv}\ \mathtt{Q}\ \mathtt{P} \longrightarrow B\,\mathtt{Q}\,\mathtt{P}$$

and the inductive cases are given by

$$\mathit{inv}\ R\ T \supset B\,R\,T \longrightarrow \mathit{inv}\ ((z)(0 \mid 0) \mid R)\ (0 \mid T) \supset B\ ((z)(0 \mid 0) \mid R)\ (0 \mid T),$$
$$\mathit{inv}\ R\ T \supset B\,R\,T \longrightarrow \mathit{inv}\ ((z)(0 \mid \bar{x}a) \mid R)\ (\bar{x}a \mid T) \supset B\ ((z)(0 \mid \bar{x}a) \mid R)\ (\bar{x}a \mid T)$$

and their symmetric variants. The proof involves quite a number of cases. We show here one particular case, the other cases can be proved pretty much in the same way. We consider a case for free input, where we have the sequent (after applying some right-introduction rules)

$$R, T, R', A\,;\left\{\begin{array}{c} \mathit{inv}\ R\ T \supset B\,R\,T, \\ \mathit{inv}\ ((z)(0 \mid \bar{x}a) \mid R)\ (\bar{x}a \mid T), \\ ((z)(0 \mid \bar{x}a) \mid R) \xrightarrow{A} R' \end{array}\right\} \longrightarrow \exists T'.(\bar{x}a \mid T) \xrightarrow{A} T' \wedge \mathit{inv}\ R'\ T' \tag{6.4}$$

to prove. Its symmetric case can be proved by using cut, since the predicate *inv* is symmetric. The sequent (6.4) can be simplified by applying $\mathit{def}\mathcal{L}^{csu}$ to the *inv* predicate, followed by an instance of $\supset \mathcal{L}$. The resulting sequent is

$$\left\{\begin{array}{c} B\,R\,T,\ \mathit{inv}\ R\ T \\ ((z)(0 \mid \bar{x}a) \mid R) \xrightarrow{A} R' \end{array}\right\} \longrightarrow \exists T'.(\bar{x}a \mid T) \xrightarrow{A} T' \wedge \mathit{inv}\ R'\ T' \tag{6.5}$$

There are three ways in which the one-step transition in the left-hand side of the sequent (6.4) can be inferred (via $\mathit{def}\mathcal{L}^{csu}$), i.e., either A is $\bar{x}a$ and R' is $((z)(0 \mid 0) \mid R)$, or $R \xrightarrow{A} R''$ and R' is $(z)(0 \mid \bar{x}a) \mid R'')$, or A is τ and $R \xrightarrow{\downarrow X} M$, R' is $((z)(0 \mid 0)|Ma)$ for some X and M. These three cases correspond to the following sequents.

(a) $B\,R\,T, \mathit{inv}\ R\ T \longrightarrow \exists T'.(\bar{x}a \mid T) \xrightarrow{\bar{x}a} T' \wedge \mathit{inv}\ ((z)(0 \mid 0) \mid R)\ T'$

(b) $B\,R\,T, \mathit{inv}\ R\ T, R \xrightarrow{A} R'' \longrightarrow \exists T'.(\bar{x}a \mid T) \xrightarrow{A} T' \wedge \mathit{inv}\ (z)(0 \mid \bar{x}a) \mid R'')\ T'$

(c) $B\,R\,T, \mathit{inv}\ R\ T, R \xrightarrow{\downarrow X} M \longrightarrow \exists T'.(\bar{x}a \mid T) \xrightarrow{\tau} T' \wedge \mathit{inv}\ ((z)(0 \mid 0)|Ma)\ T'$

The proofs for the first and second sequents are given in Figure 6.11. The proof for the third sequent is not given but it is easy to see that it has similar structure as the proof for the second one. ∎

We note that in the above example it is necessary to consider an extension of $\mathbf{D}_{!\pi,\dot{\sim}}$ in order to describe the invariant set. The following results of soundness of the encoding of bisimulation take into account such extensions. We call a stratified definition $\mathbf{D}$ a *consistent extension* of another stratified definition $\mathbf{D}'$ if all the definition clauses appearing in $\mathbf{D}'$ also appear in $\mathbf{D}$.

The proof of the adequacy of the encoding of one-step transitions in the extended calculus is basically the same as in the proof of Proposition 6.4. The additional clauses for replications do not complicate the proof of the proposition. The soundness of the *bisim* and *lbisim* still hold for the π-calculus with replication. The proofs for Theorem 6.27 and Theorem 6.28 follow the same structure as the proofs of Theorem 6.7 and Theorem 6.9.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\cdots \longrightarrow \top}\;\top\mathcal{R}}{\cdots \longrightarrow (\bar{x}a \mid T) \xrightarrow{\bar{x}a} (0 \mid T)}\;\mathit{def}\mathcal{R} \qquad \dfrac{\dfrac{}{\cdots, \mathrm{inv}\ R\ T \longrightarrow \mathrm{inv}\ R\ T}\;\mathit{init}}{\cdots, \mathrm{inv}\ R\ T \longrightarrow \mathrm{inv}\ ((z)(0 \mid 0) \mid R)\ (0 \mid T)}\;\mathit{def}\mathcal{R}}{B\,R\,T, \mathrm{inv}\ R\ T \longrightarrow (\bar{x}a \mid T) \xrightarrow{\bar{x}a} (0 \mid T) \wedge \mathrm{inv}\ ((z)(0 \mid 0) \mid R)\ (0 \mid T)}\;\wedge\mathcal{R}}{B\,R\,T, \mathrm{inv}\ R\ T \longrightarrow \exists T'.(\bar{x}a \mid T) \xrightarrow{\bar{x}a} T' \wedge \mathrm{inv}\ ((z)(0 \mid 0) \mid R)\ T'}\;\exists\mathcal{R}$$

(a)

$$\dfrac{\dfrac{\dfrac{\dfrac{}{R \xrightarrow{A} R'' \longrightarrow R \xrightarrow{A} R''}\;\mathit{init} \qquad \dfrac{\Pi}{\exists V.T \xrightarrow{A} V \wedge \mathrm{inv}\ R''\ V \longrightarrow \cdots}}{R \xrightarrow{A} R'' \supset \exists V.T \xrightarrow{A'} V \wedge \mathrm{inv}\ U\ V, R \xrightarrow{A} R'' \longrightarrow \cdots}\;\supset\mathcal{L}}{\forall U \forall A'\ R \xrightarrow{A} U \supset \exists V.T \xrightarrow{A'} V \wedge \mathrm{inv}\ U\ V, R \xrightarrow{A} R'' \longrightarrow \cdots}\;\forall\mathcal{L};\forall\mathcal{L}}{B\,R\,T, R \xrightarrow{A} R'' \longrightarrow \exists T'.(\bar{x}a \mid T) \xrightarrow{A} T' \wedge \mathrm{inv}\ (z)(0 \mid \bar{x}a) \mid R'')\ T'}\;\wedge\mathcal{L}$$

(b)

where Π is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{T \xrightarrow{A} V \longrightarrow T \xrightarrow{A} V}\;\mathit{init}}{T \xrightarrow{A} V \longrightarrow (\bar{x}a \mid T) \xrightarrow{A} (\bar{x}a \mid V)}\;\mathit{def}\mathcal{R} \qquad \dfrac{\dfrac{}{\mathrm{inv}\ R''\ V \longrightarrow \mathrm{inv}\ R''\ V}\;\mathit{init}}{\mathrm{inv}\ R''\ V \longrightarrow \mathrm{inv}\ (z)(0 \mid \bar{x}a) \mid R'')\ (\bar{x}a \mid V)}\;\mathit{def}\mathcal{R}}{T \xrightarrow{A} V, \mathrm{inv}\ R''\ V \longrightarrow (\bar{x}a \mid T) \xrightarrow{A} (\bar{x}a \mid V) \wedge \mathrm{inv}\ (z)(0 \mid \bar{x}a) \mid R'')\ (\bar{x}a \mid V)}\;\wedge\mathcal{R}}{T \xrightarrow{A} V, \mathrm{inv}\ R''\ V \longrightarrow \exists T'.(\bar{x}a \mid T) \xrightarrow{A} T' \wedge \mathrm{inv}\ (z)(0 \mid \bar{x}a) \mid R'')\ T'}\;\exists\mathcal{R}}{\exists V.T \xrightarrow{A} V \wedge \mathrm{inv}\ R''\ V \longrightarrow \exists T'.(\bar{x}a \mid T) \xrightarrow{A} T' \wedge \mathrm{inv}\ (z)(0 \mid \bar{x}a) \mid R'')\ T'}\;\exists\mathcal{L};\wedge\mathcal{L}$$

Fig. 6.11. Some derivations in Linc with $\mathbf{D}_{!\pi}$

To maintain this same structure, it is necessary that we can always *delay* the application of the $\nu\mathcal{R}$ rule in a derivation (see Chapter 3, Section 3.4).

Proposition 6.26. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be processes and* α *an action. The transition* $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$ *is derivable in* π*-calculus if and only if the sequent* $.\,;\,. \longrightarrow \langle \mathtt{P} \xrightarrow{\alpha} \mathtt{Q}\rangle$*, is provable in* Linc *with any consistent extension of* $\mathbf{D}_{!\pi}$.

Theorem 6.27. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes. If the sequent*

$$.\,;\,. \longrightarrow \bar{u} \triangleright lbisim\ L\bar{u}\ \mathtt{P}\ \mathtt{Q},$$

where L *is a ground term and the free names in* $\mathtt{P}$ *and* $\mathtt{Q}$ *are in the list* $L\,\bar{u}$*, is provable in* Linc *with any consistent extension of* $\mathbf{D}_{!\pi,\dot{\sim}_l}$ *then* $\mathtt{P} \dot{\sim} \mathtt{Q}$.

Theorem 6.28. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes. If the sequent*

$$.\,;\,. \longrightarrow \bar{u} \triangleright lbisim\ L\bar{u}\ \mathtt{P}\ \mathtt{Q},$$

where L *is a ground term and the free names in* $\mathtt{P}$ *and* $\mathtt{Q}$ *are in the list* $L\,\bar{u}$*, is provable in* Linc *with any consistent extension of* $\mathbf{D}_{\pi,\dot{\sim}_l}$ *then* $\mathtt{P} \dot{\sim} \mathtt{Q}$.

The relations given by *bisim* and *lbisim* clauses can be shown to be equivalent relations.

Theorem 6.29. *The following formulas*

$$\forall P. bisim\ P\ P, \quad \forall P \forall Q. bisim\ P\ Q \supset bisim\ Q\ P \text{ and}$$
$$\forall P \forall Q \forall R. bisim\ P\ Q \wedge bisim\ Q\ R \supset bisim\ P\ R$$

are provable in Linc *with any consistent extension of the definition* $\mathbf{D}_{!\pi,\dot{\sim}}$. *Similarly, the formulas*

$$\forall L \forall P. lbisim\ L\ P\ P, \quad \forall L \forall P \forall Q. lbisim\ L\ P\ Q \supset lbisim\ L\ Q\ P \text{ and}$$
$$\forall L \forall P \forall Q \forall R. lbisim\ L\ P\ Q \wedge lbisim\ L\ Q\ R \supset lbisim\ L\ P\ R$$

are provable in Linc *with any consistent extension of* $\mathbf{D}_{!\pi,\dot{\sim}_l}$.

Proof The proof for the reflexivity of *bisim* is done co-inductively with the invariant $\lambda P \lambda Q. P = Q$. The proof for the symmetry of *bisim* is done by case analyses on the *bisim* clause (by $def\mathcal{L}$) and does not involve co-induction. The proof of the transitivity property makes use of co-induction with the invariant $\lambda P \lambda R\ \exists Q. bisim\ P\ Q \wedge bisim\ Q\ R$. In all cases there is no need for using $def\mathcal{L}$ on any predicate other than *bisim*. The proofs are done by synchronizing left and right introduction rules. We show here a subproof for the transitivity property. In this case, after applying instances of $\forall\mathcal{R}$ and an instance of $\nu\mathcal{R}$ followed by instances of $\wedge\mathcal{R}$, we obtain the premises

$$bisim\ P\ Q, bisim\ Q\ R \longrightarrow \exists Q. bisim\ P\ Q \wedge bisim\ Q\ R,$$

which is trivially provable, and

$$
\begin{array}{lll}
(1) & \mathit{bisim}\ P\ Q, \mathit{bisim}\ Q\ R \longrightarrow \forall A \forall P' & [(P \xrightarrow{A} P') \supset \exists R'.(R \xrightarrow{A} R') \\
 & & \wedge\, \exists Q'.\mathit{bisim}\ P'\ R' \wedge \mathit{bisim}\ Q'\ R'] \\
(2) & \mathit{bisim}\ P\ Q, \mathit{bisim}\ Q\ R \longrightarrow \forall X \forall P' & [(P \xrightarrow{\downarrow X} P') \supset \exists R'.(R \xrightarrow{\downarrow X} R') \\
 & & \wedge\, \forall w.\exists Q'.\mathit{bisim}\ (P'w)\ (Q'w) \wedge \mathit{bisim}\ Q'w\ R'w] \\
(3) & \mathit{bisim}\ P\ Q, \mathit{bisim}\ Q\ R \longrightarrow \forall X \forall P' & [(P \xrightarrow{\uparrow X} P') \supset \exists R'.(R \xrightarrow{\uparrow X} R') \\
 & & \wedge\, \nabla w.\exists Q'.\mathit{bisim}\ (P'w)\ (Q'w) \wedge \mathit{bisim}\ Q'w\ R'w]
\end{array}
$$

and their symmetric variants (by interchanging P and R). The proof for sequent (1) is shown in Figure 6.12. Here the rule scheme $\mathcal{R}_1$ denotes the sequence of rules $\forall\mathcal{R}; \forall\mathcal{R}; \supset \mathcal{R}$, $\mathcal{R}_2$ denotes $\mathit{def}\mathcal{L}; \wedge\mathcal{L}$, $\mathcal{R}_3$ denotes $\forall\mathcal{L}; \forall\mathcal{L}; \supset \mathcal{L}$, and $\mathcal{R}_4$ denotes $\exists\mathcal{L}; \wedge\mathcal{L}^*$. The proofs for sequent (2) and (3) follow the same structure. The symmetric variants of (1) $-$ (3) are obtained from the proofs of (1) $-$ (3) via the substitution $[P/R, R/P]$. The proofs for the corresponding formulas involving *lbisim* are done analogously. ∎

6.6 Conclusion and related work

We have shown that the operational semantics of one-step transitions of π-calculus can be encoded naturally in Linc, making use of the higher-order nature of the logic. With the additional feature of definitions in Linc we can reason about certain properties of the transition sytem, namely, bisimulation and congruence relations between processes. The dynamics of names, i.e., the scoping constraint and freshness of names, in π-calculus are captured properly by the use of quantifiers $\forall$ and ∇ in Linc. In particular, the different proof-level binders in Linc (i.e., global and local signatures) generated by $\forall$ and ∇ capture precisely the different nature of names that are implicit in π-calculus, that is, there are names that are instantiable and names that are not. The scoping between names is handled by the alternation of quantifiers. The use of ∇ in the encoding of one-step transition makes it possible to reason about certain negative behavior, e.g., to show that some process does not make any transitions. This aspect of ∇ turns out to be crucial in proving the completeness of one of our encodings of bisimulation.

We have given two encodings of late bisimulation: *bisim* and *lbisim*. Both are sound with respect to the notion of bisimulation in π-calculus, but the first one is not complete while the second is. We retain the *bisim* encoding because we think it looks natural and it does in fact generate an equivalent relation (Theorem 6.29). We would like to think that *bisim* gives rise to the "true" bisimulation, while other variants of bisimulation can be obtained from *bisim* with appropriate *logical theories*. To make our point clear, let us recall the counterexample to the completeness of *bisim* (Example 6.8). For the proof of bisimulation in this example to succeed, it is necessary to do case analyses on the possible name w can take. We take this literally when we consider the fix in *lbisim*, by enumerating all the free names in the processes being checked for their bisimilarity. However, what the informal proof in the example suggests is that this enumeration might be too strong. In fact, what we need to check is only whether w is a or it is not a. This suggests another, cleaner, approach to encoding bisimulation: simply

$$\dfrac{\dfrac{\dfrac{\dfrac{}{P \xrightarrow{A} P' \longrightarrow P \xrightarrow{A} P'}\,\mathit{init} \qquad \dfrac{\begin{array}{c}\Pi \\ Q \xrightarrow{A} Q', \mathit{bisim}\ P'\ Q', \mathit{bisim}\ Q\ R \longrightarrow \cdots\end{array}}{\exists Q'.Q \xrightarrow{A} Q' \wedge \mathit{bisim}\ P'\ Q', \mathit{bisim}\ Q\ R \longrightarrow \cdots}\,\mathcal{R}_4}{\forall A \forall P'.P \xrightarrow{A} P' \supset \exists Q'.Q \xrightarrow{A} Q' \wedge \mathit{bisim}\ P'\ Q', \mathit{bisim}\ Q\ R, P \xrightarrow{A} P' \longrightarrow \cdots}\,\mathcal{R}_3}{\mathit{bisim}\ P\ Q, \mathit{bisim}\ Q\ R, P \xrightarrow{A} P' \longrightarrow \exists R'.R \xrightarrow{A} R' \wedge \exists Q'.\mathit{bisim}\ P'\ Q' \wedge \mathit{bisim}\ Q'\ R'}\,\mathcal{R}_2}{\begin{array}{l}\mathit{bisim}\ P\ Q, \mathit{bisim}\ Q\ R \longrightarrow \forall A \forall P'\ \ [(P \xrightarrow{A} P') \supset \exists R'.(R \xrightarrow{A} R') \\ \qquad\qquad\qquad\qquad\qquad\qquad\quad \wedge \exists Q'.\mathit{bisim}\ P'\ R' \wedge \mathit{bisim}\ Q'\ R']\end{array}}\,\mathcal{R}_1$$

where Π is the derivation

$$\dfrac{\dfrac{\dfrac{}{Q \xrightarrow{A} Q' \longrightarrow Q \xrightarrow{A} Q'}\,\mathit{init} \qquad \begin{array}{c}\Pi_1 \\ \mathit{bisim}\ P'\ Q', \exists R'.R \xrightarrow{A} R' \wedge \mathit{bisim}\ Q'\ R' \longrightarrow \cdots\end{array}}{Q \xrightarrow{A} Q', \cdots, \forall Q' \forall A.Q \xrightarrow{A} Q' \supset \exists R'.R \xrightarrow{A} R' \wedge \mathit{bisim}\ Q'\ R' \longrightarrow \cdots}\,\mathcal{R}_3}{Q \xrightarrow{A} Q', \mathit{bisim}\ P'\ Q', \mathit{bisim}\ Q\ R \longrightarrow \exists R'.R \xrightarrow{A} R' \wedge \exists Q'.\mathit{bisim}\ P'\ Q' \wedge \mathit{bisim}\ Q'\ R'}\,\mathcal{R}_2$$

and Π_1 is

$$\dfrac{\dfrac{\dfrac{}{\cdots \longrightarrow R \xrightarrow{A} R'}\,\mathit{init} \qquad \dfrac{\dfrac{\dfrac{}{\cdots \longrightarrow \mathit{bisim}\ P'\ Q'}\,\mathit{init} \qquad \dfrac{}{\cdots \longrightarrow \mathit{bisim}\ Q'\ R'}\,\mathit{init}}{\cdots \longrightarrow \mathit{bisim}\ P'\ Q' \wedge \mathit{bisim}\ Q'\ R'}\,\wedge\mathcal{R}}{\cdots \longrightarrow \exists Q'.\mathit{bisim}\ P'\ Q' \wedge \mathit{bisim}\ Q'\ R'}\,\exists\mathcal{R}}{\cdots \longrightarrow R \xrightarrow{A} R' \wedge \exists Q'.\mathit{bisim}\ P'\ Q' \wedge \mathit{bisim}\ Q'\ R'}\,\wedge\mathcal{R}}{\mathit{bisim}\ P'\ Q', R' \xrightarrow{A} R', \mathit{bisim}\ Q'\ R' \longrightarrow \exists R'.R \xrightarrow{A} R' \wedge \exists Q'.\mathit{bisim}\ P'\ Q' \wedge \mathit{bisim}\ Q'\ R'}\,\exists\mathcal{R}$$

Fig. 6.12. A proof of transitivity of *bisim*.

add the law of excluded middle on names to the encoding of *bisim*. More precisely, consider the following encoding of late bisimulation

$$\mathit{bisim}_l\ P\ Q \stackrel{\triangle}{=} (\forall x \forall y.(x = y) \vee (x = y \supset \bot)) \supset \mathit{bisim}\ P\ Q.$$

We conjecture that this encoding is sound and complete. Note that we need to mention explicitly the excluded middle in bisim_l because it is not a theorem in intuitionistic logic. This brings forward another interesting conjecture: the *bisim* encoding is sound and complete in *classical logic.* The notion of distinction can also be encoded explicitly as the assumptions in the form $x = y \supset \bot$. That is, a distinction D is encoded as the conjunction

$$F_D \equiv \bigwedge \{x_i = y_i \supset \bot\}_{(x_i, y_i) \in D}.$$

We conjecture that the notion of D-bisimilarity can be faithfully encoded as

$$(\forall x \forall y.(x = y) \vee (x = y \supset \bot)) \supset F_D \supset \mathit{bisim}\ P\ Q.$$

The encoding of one-step transitions of π-calculus has been done in several settings. In [34], processes are encoded directly as formulas in linear logic and the reflexive and transitive closures of one-step transitions is shown to be captured by logical entailment. However, this work considers only the fragment of π-calculus without the restriction operator. Formalization of π-calculus has also been done in the Calculus of Construction in [25, 13], in the *concurrent logical framework* by Watkins, et.al., [7], and in FM-logic [16]. In the latter, the restriction operator is interpreted by the new quantifier [17, 45] in the logic. In these encodings bisimulation is not considered. The encoding of bisimulation for a class of abstract transition systems, without name binding and value passing, has been studied in [31] in the logic $FO\lambda^{\Delta\mathbb{N}}$. The encoding of simulation for finite π-calculus is considered in [37]. In this work, to encode freshness of names, a sort of "counter" is maintained in the encoding of simulation such that each creation of new names results in an increase of the counter. The declarativeness of the encoding is thus somewhat compromised and reasoning about the properties of the encoding is complicated by details concerning the representation of the counter.

Chapter 7

Conclusion and Future Work

7.1 Summary of accomplishments

In this thesis we present the logic Linc which is designed to be used as a meta-logic for specifying and reasoning about operational semantics. The logic Linc is an extension of $FO\lambda^{\Delta\mathbb{N}}$ with the additional features of generalized induction and co-induction proof rules, and a new quantifier ∇. The quantifier ∇ focuses on the intensional reading of universal quantifier, and is mainly used for reasoning about generic judgments. As a connective, the quantifier ∇ is rather weak; it commutes with all other connectives. Its role is essentially in maintaining the scoping of locally bound variables within the formula. The logic Linc also allows quantification over λ-terms which makes it possible to support higher-order abstract syntax. Together with ∇ and the (co-)induction rules, this provides the ability to perform analyses on object systems involving abstractions, as we have shown in the encoding of π-calculus. We have shown that the notion of names in computation systems can be encoded naturally in Linc using quantifiers. We have proved the cut-elimination and consistency of Linc, extending the proof of McDowell and Miller for the logic $FO\lambda^{\Delta\mathbb{N}}$. The proof of cut-elimination is actually based on the technique of reducibility, originally due to Tait and later extended by Martin-Löf. The main technical achievement of our proof of cut-elimination is in formalizing the notion of reducibility in the presence of co-induction. We argue that with the inclusion of co-induction in the framework of reducibility, the co-induction rules must be "stratified" in order to get a well-founded reducibility ordering.

We illustrate the application of Linc in specifying and reasoning about data structures natural numbers and (infinite) lists, abstract transition systems (CCS and π-calculus), object-logic and evaluation of the lazy λ-calculus. These examples illustrate the use of both first-order encoding and higher-order encoding, and the induction and co-induction proof principles. In these encodings, except for the object-logic and π-calculus encodings, ∇ does not play any significant role. Some of these applications have been previously done in $FO\lambda^{\Delta\mathbb{N}}$, where the induction and co-induction proofs are done indirectly via natural number induction. We show that their encoding in Linc admits more direct proofs making use of structural induction and co-induction. We also note that since Linc is a consistent extension of $FO\lambda^{\Delta\mathbb{N}}$, all previous applications done in $FO\lambda^{\Delta\mathbb{N}}$ can be carried out in Linc without any essential modifications. The expressive power of the full Linc is illustrated in the π-calculus example. Here the interpretation of the restriction operator as the quantifier ∇ plays a crucial role in establishing the adequacy result for the encoding of late bisimulation.

7.2 Future work

Our current formulation of induction and co-induction rules is not strong enough to express (co-)inductive proofs which require explicit reference to local signatures. This means that certain forms of judgments, such as the one we have shown in the object-logic encoding, cannot be proved within Linc. One reason for this lack of expressiveness is that the current formulation of (co-)induction treats definition clauses as defining predicates, instead of atomic judgments. This is obvious in the $\mu\mathcal{L}$ rule, where only the right premise of the rule shares the local signature with the conclusion. In order to formalize the (co-)induction rules which act on atomic judgments, we need to redefine the notion of definition to take into account the local signatures. This is not unlike the notion of raised definition given in Definition 2.9. Definition clauses should be relating judgments, instead of formula, e.g., we can have the definition clause of the form

$$xy \triangleright p\,(H\,x\,y) \stackrel{\triangle}{=} y \triangleright \forall x.B\,p\,(H\,x\,y).$$

Notice that the local signatures on the left-hand side and the right-hand side of the definition clause are different. Let us refer to this notion of definition as *generic definition.* As in raised definition, a generic definition actually represents a family of definitions obtained by raising. The above definition clause, for example, can be seen as the representative of the definition clauses

$$\begin{array}{c} xy \triangleright p\,(H\,x\,y) \stackrel{\triangle}{=} y \triangleright \forall x.B\,p\,(H\,x\,y). \\ xyz \triangleright p\,(H_1\,x\,y\,z) \stackrel{\triangle}{=} yz \triangleright \forall x.B\,p\,(H_1\,x\,y\,z) \\ \vdots \end{array}$$

In the formalization of the induction rule $\mu\mathcal{L}$ on generic definition, instead of substituting the inductive predicate with an invariant formula, we need to substitute the invariant (which is now a judgment) for the "atomic judgments" in the body of the definition clause. Since formulas and judgments are of different syntactic categories, in order for the $\mu\mathcal{L}$ rule to be well-formed, the substitution of the inductive predicate with the invariant must be done indirectly via another intermediate defined predicate, whose body is the actual invariant. For example, the induction rule for the above definition can be formalized as follows.

$$\frac{H\,;\, y \triangleright \forall x.B\,q\,(H\,x\,y) \longrightarrow xy \triangleright q\,x\,y \quad \Sigma\,;\, xy \triangleright q\,t, \Gamma \longrightarrow \mathcal{C}}{\Sigma\,;\, xy \triangleright p\,t, \Gamma \longrightarrow \mathcal{C}}\ \mu\mathcal{L}$$

where $xy \triangleright q\,x\,y$ is defined by some invariant $\mathcal{D}$. The co-induction rule $\nu\mathcal{R}$ is defined dually. Of course, this extension from definition to generic definition must be done carefully in order not to destroy cut-elimination.

To get the cut-elimination result for Linc, we need to impose the level restriction on both definitions and the $\nu\mathcal{R}$ rule. It is not clear how the restriction on $\nu\mathcal{R}$ can be removed while retaining the cut-elimination proof using the reducibility technique. It would be interesting to investigate the strong normalization proofs for type systems with co-inductive data types [19, 20] to see how their proofs can be carried over to sequent systems. The level restriction on definitions can probably be weakened to allow arbitrary monotone definitions. It would also be interesting to explore another notion of stratification of definitions using the *regular word assumption* [51] to allow direct inductive proofs involving higher-order abstract syntax.

There is a prototype implementation of Linc in λProlog, available at http://www.cse.psu.edu/˜tiu. This prototype has been used to verify most results in Chapter 5 and Chapter 6. However, this prototype is currently limited to be a proof-checker and the level of automation is still minimal. We consider adding a tactical language in the style of [15] to improve the proof automation. A serious implementation would require more study on the proof search properties of Linc. It is true that with induction and co-induction there is no hope of automation in general. Nevertheless, a large subset of the logic may still admit some uniformity in proof search. A fragment of Linc without ∇ but with (co-)induction has also been separately implemented in the Hybrid system [2, 41].

Another interesting direction for future work is to investigate the connection between explicit co-induction rule with circular proofs, which is particularly attractive from the proof search viewpoint. 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. Alternatively, we could reduce “global” proofs in such a system to “local” proofs in Linc, similarly to [54].

References

[1] S. Abramsky. The lazy lambda calculus. In D. A. Turner, editor, *Research Topics in Functional Programming*, pages 65–116. Addison-Welsey, Reading, MA, 1990.

[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 Conference on Theorem Proving in Higher Order Logics, Hampton, VA, 1-3 August 2002*, volume 2342 of *LNCS*. Springer Verlag, 2002.

[3] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

[4] K. R. Apt and M. H. van Emden. Contributions to the theory of logic programming. *Journal of the ACM*, 29(3):841–862, 1982.

[5] Richard Bird and Philip Wadler. *Introduction to Functional Programming.* Prentice Hall, 1988.

[6] Iliano Cervesato, Nancy A. Durgin, Patrick D. Lincoln, John C. Mitchell, and Andre Scedrov. A meta-notation for protocol analysis. In R. Gorrieri, editor, *Proceedings of the 12th IEEE Computer Security Foundations Workshop — CSFW'99*, pages 55–69, Mordano, Italy, 28–30 June 1999. IEEE Computer Society Press.

[7] Iliano Cervesato, Frank Pfenning, David Walker, and Kevin Watkins. A concurrent logical framework ii: Examples and applications. Technical Report CMU-CS-02-101, Department of Computer Science, Carnegie Mellon University, March 2002. revised May 2003.

[8] Jawahar Chirimar. *Proof Theoretic Approach to Specification Languages.* PhD thesis, University of Pennsylvania, February 1995.

[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] Robin Cockett. Deforestation, program transformation, and cut-elimination. *Electronic Notes in Theoretical Computer Science*, 1, 2001.

[11] Sorin Craciunescu. Proving the equivalence of CLP programs. In *International Conference on Logic Programming , Copenhagen, Denmark 2002 (ICLP 2002)*, 2002.

[12] Joelle Despeyroux, Amy Felty, and Andre Hirschowitz. Higher-order abstract syntax in Coq. In *Second International Conference on Typed Lambda Calculi and Applications*, pages 124–138, April 1995.

[13] Jolle Despeyroux. A higher-order specification of the π-calculus. In *Proc. of the IFIP International Conference on Theoretical Computer Science, IFIP TCS'2000, Sendai, Japan, August 17-19, 2000.*, August 2000.

[14] 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.

[15] Amy Felty. Implementing tactics and tacticals in a higher-order logic programming language. *Journal of Automated Reasoning*, 11(1):43–81, August 1993.

[16] M. J. Gabbay. The π-calculus in FM. In Fairouz Kamareddine, editor, *Thirty-five years of Automath.* Kluwer, 2003.

[17] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2001.

[18] Gerhard Gentzen. Investigations into logical deductions. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131. North-Holland Publishing Co., Amsterdam, 1969.

[19] 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. ftp://ftp.cs.chalmers.se/pub/cs-reports/baastad.92/proc.ps.Z.

[20] 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.

[21] Jean-Yves Girard. A fixpoint theorem in linear logic. Email to the linear@cs.stanford.edu mailing list, February 1992.

[22] Lars Hallnäs. Partial inductive definitions. *Theoretical Computer Science*, 87:115–142, 1991.

[23] John Hannan. Extended natural semantics. *J. of Functional Programming*, 3(2):123–152, April 1993.

[24] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the ACM*, 40(1):143–184, 1993.

[25] D. Hirschkoff. A full formalisation of π-calculus theory in the calculus of constructions. In *Proceedings of TPHOL'97*, number 1275 in LNCS, pages 153–169, 1997.

[26] Gilles Kahn. Natural semantics. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science*, volume 247 of *LNCS*, pages 22–39. Springer-Verlag, March 1987.

[27] 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.

[28] Raymond McDowell. *Reasoning in a Logic with Definitions and Induction.* PhD thesis, University of Pennsylvania, December 1997.

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

[30] 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.

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

[32] Dale Miller. Lexical scoping as universal quantification. In *Sixth International Logic Programming Conference*, pages 268–283, Lisbon, Portugal, June 1989. MIT Press.

[33] Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, pages 321–358, 1992.

[34] Dale Miller. The π-calculus as a theory in linear logic: Preliminary results. In E. Lamma and P. Mello, editors, *Proceedings of the 1992 Workshop on Extensions to Logic Programming*, number 660 in LNCS, pages 242–265. Springer-Verlag, 1993.

[35] Dale Miller. Forum: A multiple-conclusion specification language. *Theoretical Computer Science*, 165(1):201–232, September 1996.

[36] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[37] Dale Miller and Alwen Tiu. Encoding generic judgments. In *Proceedings of FSTTCS*, number 2556 in LNCS, pages 18–32, December 2002.

[38] Dale Miller and Alwen Tiu. A proof theory for generic judgments: An extended abstract. In Phokion Kolaitis, editor, *Proceedings of LICS 2003*, July 2003.

[39] Robin Milner. *Communication and Concurrency.* Prentice-Hall International, 1989.

[40] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, Part II. *Information and Computation*, pages 41–77, 1992.

[41] 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.

[42] Alberto Momigliano and Alwen Tiu. Induction and co-induction in sequent calculus. To appear in the post-proceedings of TYPES 2003, January 2003.

[43] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Theoretical Computer Science: 5th GI-Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183, Karlsruhe, Germany, 1981. SV.

[44] Frank Pfenning. Logical frameworks. In *Handbook of Automated Reasoning*, pages 1063–1147. 2001.

[45] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*. To appear. (A preliminary version appeared in the *Proceedings of the 4th International Symposium on Theoretical Aspects of Computer Software* (TACS 2001), LNCS 2215, Springer-Verlag, 2001, pp 219–242.).

[46] G. Plotkin. A structural approach to operational semantics. DAIMI FN-19, Aarhus University, Aarhus, Denmark, September 1981.

[47] Davide Sangiorgi and David Walker. *PI-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

[48] Luigi Santocanale. A calculus of circular proofs and its categorical semantics. BRICS Report Series RS-01-15, BRICS, Dept. of Comp. Sci., Univ. of Aarhus, May 2001.

[49] Peter Schroeder-Heister. Cut-elimination in logics with definitional reflection. In D. Pearce and H. Wansing, editors, *Nonclassical Logics and Information Processing*, volume 619 of *LNCS*, pages 146–171. Springer, 1992.

[50] 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.

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

[52] Alex K. Simpson. Compositionality via cut-elimination: Hennessy-Milner logic for an arbitrary GSOS. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, pages 420–430, San Diego, California, 26–29 June 1995. IEEE Computer Society Press.

[53] John Slaney. Solution to a problem of Ono and Komori. *Journal of Philosophic Logic*, 18:103–111, 1989.

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

[55] R. F. Stärk. Cut-property and negation as failure. *International Journal of Foundations of Computer Science*, 5(2):129–164, 1994.

[56] W. W. Tait. Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic*, 32(2):198–212, 1967.

[57] Jérémie D. Wajs. *Reasoning about Logic Programs Using Definitions and Induction.* PhD thesis, Pennsylvania State University, 2002.

Vita

Alwen F. Tiu was born in Selat Panjang, Indonesia, on February 5, 1977. In 1998 he received the B.Sc. degree in Computer Science, *cum laude*, from Universitas Indonesia in Depok, Indonesia. From 1998 to 1999 he worked as a programmer at the software house RoInfo Indonesia. In 1999 he enrolled in the International Master Programme on Computational Logic at Technische Universität Dresden, Germany, and graduated with distinction in 2001. In 2001 he enrolled in the Ph. D. program in the Computer Science and Engineering Department at the Pennsylvania State University. Since 2001 he has been employed in the CSE Department of the Pennsylvania State University as a research assistant. He has been a visiting student at Laboratoire d'Informatique, École polytechnique, France since October 2002.