

Session Types as Intuitionistic Linear Propositions

Luis Caires

Frank Pfenning

Bernardo Toninho

December 21, 2011

CMU-CS-11-138

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Several type disciplines for π -calculi have been proposed in which linearity plays a key role, even if their precise relationship with pure linear logic is still not well understood. In this paper, we introduce a type system for the π -calculus that exactly corresponds to the standard sequent calculus proof system for dual intuitionistic linear logic. Our type system is based on a new interpretation of linear propositions as session types, and provides the first purely logical account of all (both shared and linear) features of session types. We show that our type discipline is useful from a programming perspective, and ensures session fidelity, absence of deadlocks, and a tight operational correspondence between π -calculus reductions and cut elimination steps.

Keywords:

1 Introduction

Linear logic has been intensively explored in the analysis of π -calculus models for communicating and mobile system, given its essential ability to deal with resources, effects, and non-interference. The fundamental way it provides for analyzing notions of sharing versus uniqueness, captured by the exponential “!”, seems to have been a source of inspiration for Milner when introducing replication in the π -calculus [22]. Following the early works of Abramsky [1], several authors have exploited variants of π -calculi to express proof reductions (e.g., [5]) or game semantics (e.g., [19]) in systems of linear logic. In the field of concurrency, many research directions have also drawn inspiration from linear logic for developing type-theoretic analyses of mobile processes, motivated by the works of Kobayashi, Pierce, and Turner [21]; a similar influence is already noticeable in the first publications by Honda on session types [16]. Many expressive type disciplines for π -calculi in which linearity frequently plays a key role have been proposed since then (e.g., [20, 18, 26, 15]). However, linearity has been usually employed in such systems in indirect ways, exploiting the fine grained type context management techniques it provides, or the assignment of usage multiplicities to channels [21], rather than the deeper type-theoretic significance of linear logical operators.

In this paper we present a type system for the π -calculus that exactly corresponds to the standard sequent calculus proof system for dual intuitionistic linear logic. The key to our correspondence is a new, perhaps surprising, interpretation of intuitionistic linear logic formulas as a form of session types [16, 18], in which the programming language is a session-typed π -calculus, and the type structure consists precisely of the connectives of intuitionistic linear logic, retaining their standard proof-theoretic interpretation.

In session-based concurrency, processes communicate through so-called session channels, connecting exactly two subsystems, and communication is disciplined by session protocols so that actions always occur in dual pairs: when one partner sends, the other receives; when one partner offers a selection, the other chooses; when a session terminates, no further interaction may occur. New sessions may be dynamically created by invocation of shared servers. Such a model exhibits concurrency in the sense that several sessions, not necessarily causally related, may be executing simultaneously, although races in unshared resources are forbidden; in fact this is the common situation in disciplined concurrent programming idioms. Mobility is also present, since both session and server names may be passed around (delegated) in communications. Session types have been introduced to discipline interactions in session-based concurrency, an important paradigm in communication-centric programming (see [11]).

It turns out that the connectives of intuitionistic linear logic suffice to express all the essential features of finite session disciplines. While in the linear λ -calculus types are assigned to terms (denoting functions and values), in our interpretation types are assigned to names (denoting communication channels) and describe their session protocol. The essence of our interpretation may already be found in the interpretation of the linear logic multiplicatives as behavioral prefix operators. Traditionally, an object of type $A \multimap B$ denotes a linear function that given an object of type A returns an object of type B [14]. In our interpretation, an object of type $A \multimap B$ denotes a session x that first inputs a session channel of type A , and then behaves as B , where B specifies again an interactive behavior, rather than a closed value. Linearity of \multimap is essential, otherwise the behavior of the input session after communication could not be ensured. An object of type $A \otimes B$ denotes a session that first sends a session channel of type A and afterwards behaves as B . But notice that objects of type $A \otimes B$ really consist of two objects: the sent session of type A and the continuation session, of type B . These two sessions are separate and non-interfering, as enforced by the canonical semantics of the linear multiplicative conjunction (\otimes). Our interpretation of $A \otimes B$ appears asymmetric, in the sense that, of course, a channel of type $A \otimes B$ is in general not typable by $B \otimes A$. In fact, the symmetry captured by the proof of $A \otimes B \vdash B \otimes A$ is realized by an appropriately typed process that coerces any session of type $A \otimes B$ to a session of type $B \otimes A$. The other linear constructors are also given compatible interpretations, in particular, the $!A$ type is naturally interpreted as a type of a shared server for sessions of type A , and additive product and sum, to branch and choice session type operators. We thus obtain the first purely logical account of both shared and linear features of session types.

We briefly summarize the contributions of the paper. We describe a system of session types for the π -calculus (Section 3) that corresponds to the sequent calculus for dual intuitionistic linear logic DILL (Section 4). The correspondence is bidirectional and tight, in the sense that (a) any π -calculus computation can be simulated by proof reductions on typing derivations (Theorem 5.3), thus establishing a strong form of subject reduction (Theorem 5.6), and (b) that any proof reduction or conversion corresponds either to a computation step or to a process equivalence on the π -calculus side (Theorems 5.4 and 5.5). An intrinsic consequence of the logical typing is a global progress property, that ensures the absence of deadlock for systems with an arbitrary number of open sessions (Theorem 5.8).

Finally, we illustrate the expressiveness of our system (Section 6) with some examples and discussion.

2 Process Model

We briefly introduce the syntax and operational semantics of the process model: the synchronous π -calculus (see [24]) extended with (binary) guarded choice.

Definition 2.1 (Processes). *Given an infinite set Λ of names (x, y, z, u, v) , the set of processes (P, Q, R) is defined by*

$$P ::= \mathbf{0} \quad | \quad P \mid Q \quad | \quad (\nu y)P \quad | \quad x\langle y \rangle.P \quad | \quad x(y).P \quad | \quad !x(y).P \\ | \quad x.\text{inl}; P \quad | \quad x.\text{inr}; P \quad | \quad x.\text{case}(P, Q)$$

The operators $\mathbf{0}$ (inaction), $P \mid Q$ (parallel composition), and $(\nu y)P$ (name restriction) comprise the static fragment of any π -calculus. We then have $x\langle y \rangle.P$ (send y on x and proceeds as P), $x(y).P$ (receive a name z on x and proceed as P with the input parameter y replaced by z), and $!x(y).P$ which denotes replicated (or persistent) input. The remaining three operators define a minimal labeled choice mechanism, comparable to the n -ary branching constructs found in standard session π -calculi (see eg., [18]). For the sake of minimality and without loss of generality we restrict our model to binary choice. In restriction $(\nu y)P$ and input $x(y).P$ the distinguished occurrence of the name y is binding, with scope the process P . For any process P , we denote the set of *free names* of P by $fn(P)$. A process is *closed* if it does not contain free occurrences of names. We identify process up to consistent renaming of bound names, writing \equiv_α for this congruence. We write $P\{x/y\}$ for the process obtained from P by capture avoiding substitution of x for y in P . Structural congruence expresses basic identities on the structure of processes, while reduction expresses the behavior of processes.

Definition 2.2. Structural congruence ($P \equiv Q$), is the least congruence relation on processes such that

$$\begin{array}{lll} P \mid \mathbf{0} \equiv P & (S\mathbf{0}) & P \equiv_\alpha Q \Rightarrow P \equiv Q & (S\alpha) \\ P \mid Q \equiv Q \mid P & (S|C) & P \mid (Q \mid R) \equiv (P \mid Q) \mid R & (S|A) \\ (\nu x)\mathbf{0} \equiv \mathbf{0} & (S\nu\mathbf{0}) & x \notin fn(P) \Rightarrow P \mid (\nu x)Q \equiv (\nu x)(P \mid Q) & (S\nu|) \\ (\nu x)(\nu y)P \equiv (\nu y)(\nu x)P & (S\nu\nu) & & \end{array}$$

Definition 2.3. Reduction ($P \rightarrow Q$), is the binary relation on processes defined by:

$$\begin{array}{ll} x\langle y \rangle.Q \mid x(z).P \rightarrow Q \mid P\{y/z\} & (RC) \\ x\langle y \rangle.Q \mid !x(z).P \rightarrow Q \mid P\{y/z\} \mid !x(z).P & (R!) \\ x.\text{inl}; P \mid x.\text{case}(Q, R) \rightarrow P \mid Q & (RL) \\ x.\text{inr}; P \mid x.\text{case}(Q, R) \rightarrow P \mid R & (RR) \\ Q \rightarrow Q' \Rightarrow P \mid Q \rightarrow P \mid Q' & (R|) \\ P \rightarrow Q \Rightarrow (\nu y)P \rightarrow (\nu y)Q & (R\nu) \\ P \equiv P', P' \rightarrow Q', Q' \equiv Q \Rightarrow P \rightarrow Q & (R\equiv) \end{array}$$

Notice that reduction is closed (by definition) under structural congruence. Reduction specifies the computations a process performs on its own. To characterize the interactions a process may perform with its environment, we introduce a labeled transition system; the standard early transition system for the π -calculus [24] extended with appropriate labels and transition rules for the choice constructs. A transition $P \xrightarrow{\alpha} Q$ denotes that process P may evolve to process Q by performing the action represented by the label α . Transition labels are given by

$$\alpha ::= \overline{x\langle y \rangle} \mid x(y) \mid \overline{(\nu y)x\langle y \rangle} \mid x.\text{inl} \mid x.\text{inr} \mid \overline{x.\text{inl}} \mid \overline{x.\text{inr}}$$

Actions are input $x(y)$, the left/right offers $x.\text{inl}$ and $x.\text{inr}$, and their matching co-actions, respectively the output $\overline{x\langle y \rangle}$ and bound output $\overline{(\nu y)x\langle y \rangle}$ actions, and the left/ right selections $\overline{x.\text{inl}}$ and $\overline{x.\text{inr}}$. The bound output $\overline{(\nu y)x\langle y \rangle}$ denotes extrusion of a fresh name y along (channel) x . Internal action is denoted by τ , in general an action $\alpha(\bar{\alpha})$ requires a matching $\bar{\alpha}$ (α) in the environment to enable progress, as specified by the transition rules. For a label α , we define the sets $fn(\alpha)$ and $bn(\alpha)$ of free and bound names, respectively, as usual. We denote by $s(\alpha)$ the subject of α (e.g., x in $x\langle y \rangle$).

$$\begin{array}{c}
\frac{P \xrightarrow{\alpha} Q}{(\nu y)P \xrightarrow{\alpha} (\nu y)Q} \text{(res)} \quad \frac{P \xrightarrow{\alpha} Q}{P \mid R \xrightarrow{\alpha} Q \mid R} \text{(par)} \quad \frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \text{(com)} \\
\\
\frac{P \xrightarrow{(\nu y)x(y)} P' \quad Q \xrightarrow{x(y)} Q'}{P \mid Q \xrightarrow{\tau} (\nu y)(P' \mid Q')} \text{(close)} \quad \frac{P \xrightarrow{x(y)} Q}{(\nu y)P \xrightarrow{(\nu y)x(y)} Q} \text{(open)} \quad x(y).P \xrightarrow{x(y)} P \text{(out)} \\
\\
x(y).P \xrightarrow{x(z)} P\{z/y\} \text{(in)} \quad !x(y).P \xrightarrow{x(z)} P\{z/y\} \mid !x(y).P \text{(rep)} \quad x.\text{inl}; P \xrightarrow{x.\text{inl}} P \text{(lout)} \\
\\
x.\text{inr}; P \xrightarrow{x.\text{inr}} P \text{(rout)} \quad x.\text{case}(P, Q) \xrightarrow{x.\text{inl}} P \text{(lin)} \quad x.\text{case}(P, Q) \xrightarrow{x.\text{inr}} Q \text{(rin)}
\end{array}$$

Figure 1: π -calculus Labeled Transition System.

Definition 2.4 (Labeled Transition System). *The relation labeled transition ($P \xrightarrow{\alpha} Q$) is defined by the rules in Figure 1, subject to the side conditions: in rule (res), we require $y \notin \text{fn}(\alpha)$; in rule (par), we require $\text{bn}(\alpha) \cap \text{fn}(R) = \emptyset$; in rule (close), we require $y \notin \text{fn}(Q)$. We omit the symmetric versions of rules (par), (com), and (close).*

We recall some basic facts about reduction, structural congruence, and labeled transition, namely: closure of labeled transitions under structural congruence, and coincidence of τ -labeled transition and reduction [24]: (1) if $P \equiv^{\alpha} Q$, then $P \xrightarrow{\alpha} Q$, and (2) $P \rightarrow Q$ if and only if $P \xrightarrow{\tau} Q$. We write $\rho_1 \rho_2$ for relation composition (e.g., $\xrightarrow{\tau} \equiv$).

3 Type System

We first describe our type structure, which coincides with intuitionistic linear logic [14, 3], omitting atomic formulas and the additive constants \top and $\mathbf{0}$.

Definition 3.1 (Types). *Types (A, B, C) are given by*

$$A, B ::= \mathbf{1} \mid !A \mid A \otimes B \mid A \multimap B \mid A \oplus B \mid A \& B$$

Types are assigned to (channel) names, and may be conveniently interpreted as a form of session types; an assignment $x:A$ enforces that the process will use x according to the discipline A . $A \otimes B$ is the type of a session channel that first performs an output (sending a session channel of type A) to its partner before proceeding as specified by B . In a similar way, $A \multimap B$ types a session channel that first performs an input (receiving a session channel of type A) from its partner, before proceeding as specified by B . The type $\mathbf{1}$ means that the session terminated, no further interaction will take place on it. Notice that names of type $\mathbf{1}$ may still be passed around in sessions, as opaque values. $A \oplus B$ types a session that either selects “left” and then proceed as specified by A , or else selects “right”, and then proceeds as specified by B . Dually, $A \& B$ types a session channel that offers its partner a choice between an A typed behavior (“left” choice) and a B typed behavior (“right” choice). The type $!A$ types a non-session (non-linearized, shared) channel (called *standard channel* in [13]), to be used by a server for spawning an arbitrary number of new sessions (possibly none), each one conforming to type A .

A type environment is a collection of type assignments, of the form $x : A$ where x is a name and A a type, the names being pairwise disjoint. Following the insights behind dual intuitionistic linear logic, which goes back to Andreoli’s *dyadic* system for classical linear logic [2], we distinguish two kinds of type environments subject to different structural properties: a *linear* part Δ and an *unrestricted* part Γ , where weakening and contraction principles hold for Γ but not for Δ . A judgment of our system has then the form $\Gamma; \Delta \vdash P :: z:C$ where name declarations in Γ are always propagated unchanged to all premises in the typing rules, while name declarations in Δ are handled multiplicatively or additively, depending on the nature of the type being defined. The domains of Γ, Δ and $z:C$ are required to be pairwise disjoint.

Intuitively, such a judgment asserts: P is ensured to safely provide a usage of name z according to the behavior (session) specified by type C , whenever composed with any process environment providing usages of names according to the behaviors specified by names in $\Gamma; \Delta$. As shown in Section 5, in our case safety ensures that the behavior is

$$\begin{array}{c}
\frac{\Gamma; \Delta \vdash P :: T}{\Gamma; \Delta, x:\mathbf{1} \vdash P :: T} \text{ (T1L)} \quad \frac{}{\Gamma; \cdot \vdash 0 :: x:\mathbf{1}} \text{ (T1R)} \\
\frac{\Gamma; \Delta, y:A, x:B \vdash P :: T}{\Gamma; \Delta, x:A \otimes B \vdash x(y).P :: T} \text{ (T}\otimes\text{L)} \quad \frac{\Gamma; \Delta \vdash P :: y:A \quad \Gamma; \Delta' \vdash Q :: x:B}{\Gamma; \Delta, \Delta' \vdash (\nu y)x\langle y \rangle.(P \mid Q) :: x:A \otimes B} \text{ (T}\otimes\text{R)} \\
\frac{\Gamma; \Delta \vdash P :: y:A \quad \Gamma; \Delta', x:B \vdash Q :: T}{\Gamma; \Delta, \Delta', x:A \multimap B \vdash (\nu y)x\langle y \rangle.(P \mid Q) :: T} \text{ (T}\multimap\text{L)} \quad \frac{\Gamma; \Delta, y:A \vdash P :: x:B}{\Gamma; \Delta \vdash x(y).P :: x:A \multimap B} \text{ (T}\multimap\text{R)} \\
\frac{\Gamma; \Delta \vdash P :: x:A \quad \Gamma; \Delta', x:A \vdash Q :: T}{\Gamma; \Delta, \Delta' \vdash (\nu x)(P \mid Q) :: T} \text{ (Tcut)} \quad \frac{\Gamma; \cdot \vdash P :: y:A \quad \Gamma, u:A; \Delta \vdash Q :: T}{\Gamma; \Delta \vdash (\nu u)(!u(y).P \mid Q) :: T} \text{ (Tcut}^!\text{)} \\
\frac{\Gamma, u:A; \Delta, y:A \vdash P :: T}{\Gamma, u:A; \Delta \vdash (\nu y)u\langle y \rangle.P :: T} \text{ (Tcopy)} \\
\frac{\Gamma, u:A; \Delta \vdash P\{u/x\} :: T}{\Gamma; \Delta, x:!A \vdash P :: T} \text{ (T!L)} \quad \frac{\Gamma; \cdot \vdash Q :: y:A}{\Gamma; \cdot \vdash !x(y).Q :: x:!A} \text{ (T!R)} \\
\frac{\Gamma; \Delta, x:A \vdash P :: T \quad \Gamma; \Delta, x:B \vdash Q :: T}{\Gamma; \Delta, x:A \oplus B \vdash x.\text{case}(P, Q) :: T} \text{ (T}\oplus\text{L)} \quad \frac{\Gamma; \Delta, x:B \vdash P :: T}{\Gamma; \Delta, x:A \& B \vdash x.\text{inr}; P :: T} \text{ (T}\&\text{L}_2\text{)} \\
\frac{\Gamma; \Delta \vdash P :: x:A \quad \Gamma; \Delta \vdash Q :: x:B}{\Gamma; \Delta \vdash x.\text{case}(P, Q) :: x:A \& B} \text{ (T}\&\text{R)} \quad \frac{\Gamma; \Delta, x:A \vdash P :: T}{\Gamma; \Delta, x:A \& B \vdash x.\text{inl}; P :: T} \text{ (T}\&\text{L}_1\text{)} \\
\frac{\Gamma; \Delta \vdash P :: x:A}{\Gamma; \Delta \vdash x.\text{inl}; P :: x:A \oplus B} \text{ (T}\oplus\text{R}_1\text{)} \quad \frac{\Gamma; \Delta \vdash P :: x:B}{\Gamma; \Delta \vdash x.\text{inr}; P :: x:A \oplus B} \text{ (T}\oplus\text{R}_2\text{)}
\end{array}$$

Figure 2: The Type System π_{DILL} .

free of communication errors and deadlock. A pure client Q that just relies on external services, and does not provide any, will be typed as $\Gamma; \Delta \vdash Q :: -:\mathbf{1}$. In general, a process P such that $\Gamma; \Delta \vdash P :: z:C$ represents a system providing behavior C at channel z , building on “services” declared in $\Gamma; \Delta$. Of particular interest is a system typed as $\Gamma; \Delta \vdash R :: z:!A$, representing a shared server. Quite interestingly, the asymmetry induced by the intuitionistic interpretation of $!A$ enforces locality of shared names but not of linear (session names), which exactly corresponds to the intended model of sessions.

We present the rules of our type system π_{DILL} in Fig. 2. We use T, S for right hand side singleton environments (e.g., $z:C$). The interpretation of the various rules should be clear, given the explanation of types given above. Notice that since in $\otimes R$ the sent name is always fresh, our typed calculus conforms to a session-based internal mobility discipline [23, 7], without loss of expressiveness. The composition rules (cut and cut[!]) follow the “composition plus hiding” principle [1], extended to a name passing setting. More familiar linear typing rules for parallel composition (e.g., as in [21]) are derivable (see Section 6). Since we are considering π -calculus terms up to structural congruence, typability is closed under \equiv by definition. π_{DILL} enjoys the usual properties of equivariance, weakening in Γ and contraction in Γ . The coverage property also holds: if $\Gamma; \Delta \vdash P :: z:A$ then $\text{fn}(P) \subseteq \Gamma \cup \Delta \cup \{z\}$. In the presence of type-annotated restrictions $(\nu x:A)P$, as usual in typed π -calculi [24], type-checking is decidable.

We illustrate the type system with a simple example, frequently used to motivate session based interactions (see e.g., [13]). A client may choose between a “buy” operation, in which it indicates a product name and a credit card number to receive a receipt, and a “quote” operation, in which it indicates a product name, to obtain the product price. From the client perspective, the session protocol exposed by the server may be specified by the type

$$\text{ServerProto} \triangleq (N \multimap I \multimap (N \otimes \mathbf{1})) \& (N \multimap (I \otimes \mathbf{1}))$$

We assume that N and I are types representing shareable values (e.g., strings N and integers I). To simplify, we set $N = I = \mathbf{1}$. Assuming s to be the name of the session channel connecting the client and server, consider the code

$$\text{QClntBody}_s \triangleq s.\text{inr}; (\nu tea)s\langle tea \rangle.s(\text{pr}).\mathbf{0}$$

QClntBody_s specifies a client that asks for the price of tea (we simply abstract away from what the client might do

$$\begin{array}{c}
\frac{\Gamma; \Delta \vdash D : C}{\Gamma; \Delta, x : \mathbf{1} \vdash \mathbf{1L} x D : C} \text{ (1L)} \quad \frac{}{\Gamma; \cdot \vdash \mathbf{1R} : \mathbf{1}} \text{ (1R)} \\
\frac{\Gamma; \Delta, y : A, x : B \vdash D : C}{\Gamma; \Delta, x : A \otimes B \vdash \otimes L x (y.x. D) : C} \text{ (\otimes L)} \quad \frac{\Gamma; \Delta \vdash D : A \quad \Gamma; \Delta' \vdash E : B}{\Gamma; \Delta, \Delta' \vdash \otimes R D E : A \otimes B} \text{ (\otimes R)} \\
\frac{\Gamma; \Delta \vdash D : A \quad \Gamma; \Delta', x : B \vdash E : C}{\Gamma; \Delta, \Delta', x : A \multimap B \vdash \multimap L x D (x. E) : C} \text{ (\multimap L)} \quad \frac{\Gamma; \Delta, y : A \vdash D : B}{\Gamma; \Delta \vdash \multimap R (y. D) : A \multimap B} \text{ (\multimap R)} \\
\frac{\Gamma; \Delta \vdash D : A \quad \Gamma; \Delta', x : A \vdash E : C}{\Gamma; \Delta, \Delta' \vdash \text{cut} D (x. E) : C} \text{ (cut)} \quad \frac{\Gamma; \cdot \vdash D : A \quad \Gamma, u : A; \Delta \vdash E : C}{\Gamma; \Delta \vdash \text{cut}^! D (u. E) : C} \text{ (cut}^!\text{)} \\
\frac{\Gamma, u : A; \Delta, y : A \vdash D : C}{\Gamma, u : A; \Delta \vdash \text{copy} u (y. D) : C} \text{ (copy)} \\
\frac{\Gamma; \cdot \vdash D : A}{\Gamma; \cdot \vdash !R D : !A} \text{ (!R)} \quad \frac{\Gamma, u : A; \Delta \vdash D : C}{\Gamma; \Delta, x : !A \vdash !L x (u.D) : C} \text{ (!L)} \\
\frac{\Gamma; \Delta, x : A \vdash D : C}{\Gamma; \Delta, x : A \& B \vdash \& L_1 x (x. D) : C} \text{ (\&L}_1\text{)} \quad \frac{\Gamma; \Delta, x : B \vdash D : C}{\Gamma; \Delta, x : A \& B \vdash \& L_2 x (x. D) : C} \text{ (\&L}_2\text{)} \\
\frac{\Gamma; \Delta \vdash D : A \quad \Gamma; \Delta \vdash E : B}{\Gamma; \Delta \vdash \& R D E : A \& B} \text{ (\&R)} \quad \frac{\Gamma; \Delta x : A \vdash D : C \quad \Gamma; \Delta, x : B \vdash E : C}{\Gamma; \Delta, x : A \oplus B \vdash \oplus L x (x. D) (x. E) : C} \text{ (\oplus)} \\
\frac{\Gamma; \Delta \vdash D : A}{\Gamma; \Delta \vdash \oplus R_1 D : A \oplus B} \text{ (\oplus R}_1\text{)} \quad \frac{\Gamma; \Delta \vdash D : B}{\Gamma; \Delta \vdash \oplus R_2 D : A \oplus B} \text{ (\oplus R}_2\text{)}
\end{array}$$

Figure 3: Dual Intuitionistic Linear Logic DILL.

with the price after reading it). It first selects the quoting operation on the server ($s.\text{inr}$), then sends the id of the product to the server ($s\langle \text{tea} \rangle$), then receives the price $s\langle \text{pr} \rangle$ from the server and finally terminates the session ($\mathbf{0}$). Then

$$\cdot; s : \text{ServerProto} \vdash \text{QClntBody}_s :: - : \mathbf{1}$$

is derivable (by $T\mathbf{1R}$, $T\otimes L$, $T\multimap L$ and $T\&L_2$). Here we wrote $-$ for an anonymous variable that does not appear in QClntBody . This is possible even in a linear type discipline since the inactive process $\mathbf{0}$ is typed by $x:\mathbf{1}$ and does not use x . Concerning the server code, let $\text{SrvBody}_s \triangleq s.\text{case}(s\langle \text{pn} \rangle.s\langle \text{cn} \rangle.(vrc)s\langle \text{rc} \rangle.\mathbf{0}, s\langle \text{pn} \rangle.(vpr)s\langle \text{pr} \rangle.\mathbf{0})$.

Then $\cdot; \vdash \text{SrvBody}_s :: s:\text{ServerProto}$ is derivable, by $T\&R$. By $T\text{cut}$ we obtain for the system $\text{QSimple} \triangleq (v s)(\text{SrvBody}_s \mid \text{QClntBody}_s)$ the typing $\cdot; \vdash \text{QSimple} :: - : \mathbf{1}$. In this example we have only introduced processes interacting in a single session, but clearly the system accommodates all the generality of session types, e.g., a simple process interacting in different sessions is $x:A \multimap \mathbf{1}, y:A \otimes \mathbf{1} \vdash y(w).(vk)x\langle k \rangle.\mathbf{0} :: - : \mathbf{1}$.

4 Dual Intuitionistic Linear Logic

As presented, session type constructors correspond directly to intuitionistic linear logic connectives. Typing judgments directly correspond to sequents in dual intuitionistic linear logic, by erasing processes [3, 10]. In Figure 3 we present the DILL sequent calculus. In our presentation, DILL is conveniently equipped with a faithful proof term assignment, so sequents have the form $\Gamma; \Delta \vdash D : C$ where Γ is the unrestricted context, Δ the linear context, C a formula (= type) and D the proof term that faithfully represents the derivation of $\Gamma; \Delta \vdash C$. Our use of names in the proof system will be consistent with the proof discipline, u, v, w for variables in Γ and x, y, z for variables in Δ . This is consistent with standard usage of names in π -calculus. Given the parallel structure of the two systems, if $\Gamma; \Delta \vdash D : A$ is derivable in DILL then there is a process P and a name z such that $\Gamma; \Delta \vdash P :: z:A$ is derivable in πDILL , and the converse result also holds: if $\Gamma; \Delta \vdash P :: z:A$ is derivable in πDILL there is a derivation D that proves $\Gamma; \Delta \vdash D : A$. This correspondence is made explicit by a translation from faithful proof terms to processes, defined in Fig. 4: for $\Gamma; \Delta \vdash D : C$ we write \hat{D}^z for the translation of D such that $\Gamma; \Delta \vdash \hat{D}^z :: z:C$.

D	$\rightsquigarrow \hat{D}^z$	D	$\rightsquigarrow \hat{D}^z$
1R	$\rightsquigarrow 0$	$\oplus R_1 D$	$\rightsquigarrow z.\text{inl}; \hat{D}^z$
1L $x D$	$\rightsquigarrow \hat{D}^z$	$\oplus R_2 E$	$\rightsquigarrow z.\text{inr}; \hat{E}^z$
$\otimes R D E$	$\rightsquigarrow (\nu y) z \langle y \rangle. (\hat{D}^y \hat{E}^z)$	$\oplus L x (x. D) (x. E)$	$\rightsquigarrow x.\text{case}(\hat{D}^z, \hat{E}^z)$
$\otimes L x (y.x. D)$	$\rightsquigarrow x(y). \hat{D}^z$	cut $D (x. E)$	$\rightsquigarrow (\nu x)(\hat{D}^x \hat{E}^z)$
$\multimap R (y. D)$	$\rightsquigarrow z(y). \hat{D}^z$!R D	$\rightsquigarrow !z(y). \hat{D}^y$
$\multimap L x D (x. E)$	$\rightsquigarrow (\nu y) x \langle y \rangle. (\hat{D}^y \hat{E}^z)$!L $x (u. D)$	$\rightsquigarrow \hat{D}^z \{x/u\}$
$\& R D E$	$\rightsquigarrow z.\text{case}(\hat{D}^z, \hat{E}^z)$	copy $u (y. D)$	$\rightsquigarrow (\nu y) u \langle y \rangle. \hat{D}^z$
$\& L_1 x (x. D)$	$\rightsquigarrow x.\text{inl}; \hat{D}^z$	cut [!] $D (u. E)$	$\rightsquigarrow (\nu u)((!u(y). \hat{D}^y) \hat{E}^z)$
$\& L_2 x (x. E)$	$\rightsquigarrow x.\text{inr}; \hat{E}^z$		

Figure 4: Proof D extracts to process \hat{D}^z .

Definition 4.1 (Typed Extraction). We write $\Gamma; \Delta \vdash D \rightsquigarrow P :: z:A$, meaning “proof D extracts to P ”, whenever $\Gamma; \Delta \vdash D : A$ and $\Gamma; \Delta \vdash P :: z:A$ and $P \equiv \hat{D}^z$.

Typed extraction is unique up to structural congruence, in the sense that if $\Gamma; \Delta \vdash D \rightsquigarrow P :: z:A$ and $\Gamma; \Delta \vdash D \rightsquigarrow Q :: z:A$ then $P \equiv Q$, as a consequence of closure of typing under structural congruence. The system DILL as presented does not admit atomic formulas, and hence has no true initial sequents. However, the correspondence mentioned above yields an explicit identity theorem:

Proposition 4.2. For any type A and distinct names x, y , there is a process $id_A(x, y)$ and a cut-free derivation D such that $x:A \vdash D \rightsquigarrow id_A(x, y) :: y:A$.

The $id_A(x, y)$ process, with exactly the free names x, y , implements a synchronous mediator that bidirectionally plays the protocol specified by A between channels x and y . For example, we analyze the interpretation of the sequent $A \otimes B \vdash B \otimes A$. We have

$$x:A \otimes B \vdash F \rightsquigarrow x(z).(\nu n)y \langle n \rangle. (P | Q) :: y:B \otimes A$$

where $F = \otimes L x (z.x. \otimes R D E)$, $P = id_B(x, n)$ and $Q = id_A(z, y)$. This process is an interactive proxy that coerces a session of type $A \otimes B$ at x to a session of type $B \otimes A$ at y . It first receives a session of type A (bound to z) and after sending on y a session of type B (played by copying the continuation of x to n), it progresses with a session of type A on y (copying the continuation of z to y).

As processes are related by structural and computational rules, namely those involved in the definition of \equiv and \rightarrow , derivations in DILL are related by structural and computational rules, that express certain sound proof transformations that arise in cut-elimination. The reductions (Figure 5) generally take place when a right rule meets a left rule for the same connective, and correspond to reduction steps in the process term assignment. On the left, we show the usual reductions for cuts; on the right, we show the corresponding reductions (if any) of the process terms, modulo structural congruence. Since equivalences depend on variable occurrences, we write D_x if x may occur in D .

The structural conversions in Figure 6 correspond to structural equivalences in the π -calculus, since they just change the order of cuts, e.g., $(\text{cut}/-/\text{cut}_1)$ translates to

$$(\nu x)(\hat{D}^x | (\nu y)(\hat{E}^y | \hat{F}^z)) \equiv (\nu y)((\nu x)(\hat{D}^x | \hat{E}^y) | \hat{F}^z)$$

In addition, we have two special conversions. Among those, $(\text{cut}/\mathbf{1R}/\mathbf{1L})$ is not needed in order to simulate the π -calculus reduction, while $(\text{cut}/\mathbf{!R}/\mathbf{!L})$ is. In cut-elimination procedures, these are always used from left to right. Here, they are listed as equivalences because the corresponding π -calculus terms are structurally congruent. The root cause for this is that the rules **1L** and **!L** are *silent*: the extracted terms in the premise and conclusion are the same, modulo renaming. For **1L**, this is the case because a terminated process, represented by $0 :: - : \mathbf{1}$ silently disappears from a parallel composition by structural congruence. For **!L**, this is the case because the actual replication of a server process is captured in the copy rule which clones $u:A$ to $y:A$, rule rather than **!L**. It is precisely for this reason that the rule commuting a persistent cut ($\text{cut}^!$) over a copy rule (copy) is among the computational conversions.

The structural conversions in Figure 8 propagate $\text{cut}^!$. From the proof theoretic perspective, because $\text{cut}^!$ cuts a persistent variable u , $\text{cut}^!$ may be duplicated or erased. On the π -calculus side, these no longer correspond to structural congruences, but, quite remarkably, to behavioral equivalences, derivable from known properties of typed processes, the (sharpened) Replication Theorems [24]. These hold in our language, due to our interpretation of $!$ types. Our operational correspondence results also depend on six commuting conversions, four in Figure 7 plus two symmetric versions. The commuting conversions push a cut up (or inside) the $!L$ and $!L$ rules. During the usual cut elimination procedures, these are used from left to right. In the correspondence with the sequent calculus, the situation is more complex. Because the $!L$ and $!L$ rules do not affect the extracted term, cuts have to be permuted with these two rules in order to simulate π -calculus reduction. From the process calculus perspective, such conversions correspond to identity. There is a second group of commuting conversions (not shown), not necessary for our current development. Those do not correspond to structural congruence nor to strong bisimilarities on π -calculus, as they may not preserve process behavior in the general untyped setting, since they promote an action prefix from a subexpression to the top level. We conjecture that such equations denote behavioral identities under a natural definition of typed observational congruence for our calculus.

Definition 4.3 (Relations on derivations induced by conversions). (1) \equiv : the least congruence on derivations generated by the structural conversions (I) and the commuting conversions (II); (2) \simeq_s : the least congruence on derivations generated by all structural conversions (I-III). We extend \simeq_s to processes as the congruence generated by the process equations on the right. (3) \Rightarrow : the reduction on derivations obtained by orienting all conversions in the direction shown, from left to right or top to bottom.

As discussed above, \simeq_s is a typed behavioral equivalence on processes.

5 Computational Correspondence, Preservation, and Progress

We now present the results stating the key properties of our type system and logical interpretation. Theorem 5.3 states the existence of a simulation between reductions in the typed π -calculus and proof conversions / reductions, expressing a strong form of subject reduction for our type system. The proof relies on several auxiliary lemmas that relate process reduction with proof reduction at a particular type. The lemmas themselves are all very similar, so we only present the lemmas for \otimes and $!$. The remaining lemmas, and their proofs, are detailed in Appendix A.

Lemma 5.1. *Assume*

1. $\Gamma; \Delta_1 \vdash D_1 \rightsquigarrow P_1 :: x:C_1 \otimes C_2$ with $P_1 \xrightarrow{(\nu y)x\langle y \rangle} P'_1$;
2. $\Gamma; \Delta_2, x:C_1 \otimes C_2 \vdash D_2 \rightsquigarrow Q_1 :: z:C$ with $Q_1 \xrightarrow{x\langle y \rangle} Q'_1$.

Then

1. $\text{cut } D_1(x.D_2) \equiv \Rightarrow \equiv D$ for some D ;
2. $\Gamma; \Delta_1, \Delta_2 \vdash D \rightsquigarrow Q_2 :: z : C$ for some $Q_2 \equiv (\nu x)(P'_1 \mid Q'_1)$.

Proof. See Appendix A.6. □

Lemma 5.2. *Assume*

1. $\Gamma; \Delta_1 \vdash D_1 \rightsquigarrow P_1 :: x:!A$ with $P_1 \xrightarrow{x\langle y \rangle} P'_1$;
2. $\Gamma; \Delta_2, x:!A \vdash D_2 \rightsquigarrow Q_1 :: z:C$ with $Q_1 \xrightarrow{(\nu y)x\langle y \rangle} Q'_1$.

Then

1. $\text{cut } D_1(x.D_2) \equiv \Rightarrow \equiv D$ for some D ;

$$\begin{array}{lcl}
\text{cut } (\otimes R D_1 D_2) (x. \otimes L x (y.x. E_{xy})) & \rightsquigarrow & (\nu x)((\nu y) x \langle y \rangle. (\hat{D}_1^y \mid \hat{D}_2^x)) \mid x \langle y \rangle. \hat{E}^z \\
\Rightarrow & \rightarrow & \\
\text{cut } D_1 (y. \text{cut } D_2 (x. E_{xy})) & \rightsquigarrow & (\nu x)(\nu y)(\hat{D}_1^y \mid \hat{D}_2^x \mid \hat{E}^z) \\
\text{cut } (\multimap R (y. D_y)) (x. \multimap L x E_1 (x. E_{2x})) & \rightsquigarrow & (\nu x)((x \langle y \rangle. \hat{D}^x) \mid (\nu y) x \langle y \rangle. (\hat{E}_1^y \mid \hat{E}_2^z)) \\
\Rightarrow & \rightarrow & \\
\text{cut } (\text{cut } E_1 (y. D_y)) (x. E_{2x}) & \rightsquigarrow & (\nu x)(\nu y)(\hat{D}^x \mid \hat{E}_1^y \mid \hat{E}_2^z) \\
\text{cut } (\& R D_1 D_2) (x. \& L_i x (x. E_x)) & \rightsquigarrow & (\nu x)(x. \text{case}(\hat{D}_1^x, \hat{D}_2^x) \mid x. \text{inl}; \hat{E}^z) \\
\Rightarrow & \rightarrow & \\
\text{cut } D_i (x. E_x) & \rightsquigarrow & (\nu x)(\hat{D}_i^x \mid \hat{E}^z) \\
\text{cut } (\oplus R_i D) (x. \oplus L x (x. E_{1x}) (x. E_{2x})) & \rightsquigarrow & (\nu x)(x. \text{inl}; \hat{D}^x \mid x. \text{case}(\hat{E}_1^z, \hat{E}_2^z)) \\
\Rightarrow & \rightarrow & \\
\text{cut } D (x. E_{ix}) & \rightsquigarrow & (\nu x)(\hat{D}^x \mid \hat{E}_i^z) \\
\text{cut}^! D (u. \text{copy } u (y. E_{uy})) & \rightsquigarrow & (\nu u)((!u \langle y \rangle. \hat{D}^y) \mid (\nu y) u \langle y \rangle. \hat{E}^z) \\
\Rightarrow & \rightarrow & \\
\text{cut } D (y. \text{cut}^! D (u. E_{uy})) & \rightsquigarrow & (\nu y)(\hat{D}^y \mid (\nu u)((!u \langle y \rangle. \hat{D}^y) \mid \hat{E}^z))
\end{array}$$

Figure 5: Computational Conversions

$$\begin{array}{lcl}
(\text{cut}/-/ \text{cut}_1) & \text{cut } D (x. \text{cut } E_x (y. F_y)) & \equiv \text{cut } (\text{cut } D (x. E_x)) (y. F_y) \\
(\text{cut}/-/ \text{cut}_2) & \text{cut } D (x. \text{cut } E (y. F_{xy})) & \equiv \text{cut } E (y. \text{cut } D (x. F_{xy})) \\
(\text{cut}/\text{cut}^!/-) & \text{cut } (\text{cut}^! D (u. E_u)) (x. F_x) & \equiv \text{cut}^! D (u. \text{cut } E_u (x. F_x)) \\
(\text{cut}/-/ \text{cut}^!) & \text{cut } D (x. \text{cut}^! E (u. F_{xu})) & \equiv (\text{cut}^! E (u. \text{cut } D (x. F_{xu})) \\
(\text{cut}/!R/!L) & \text{cut } !R (x. !L x D) & \equiv D \\
(\text{cut}/!R/!L) & \text{cut } (!R D) (x. !L x (u. E)) & \equiv \text{cut}^! D (u. E)
\end{array}$$

Figure 6: Structural Conversions (I): Cut Conversions

$$\begin{array}{lcl}
(\text{cut}/!L/-) & \text{cut } (!L y D) (x. F_x) & \equiv !L y (\text{cut } D (x. F_x)) \\
(\text{cut}/!L/-) & \text{cut } (!L y (u. D_u)) (x. F_x) & \equiv !L y (u. \text{cut } D_u (x. F_x)) \\
(\text{cut}^!/-/!L) & \text{cut}^! D (u. !L y E_u) & \equiv !L y (\text{cut}^! D (u. E_u)) \\
(\text{cut}^!/-/!L) & \text{cut}^! D (u. !L y (v. E_{uv})) & \equiv !L y (v. \text{cut}^! D (u. E_{uv}))
\end{array}$$

Figure 7: Structural Conversions (II): Commuting Conversions

$$\begin{array}{lcl}
\text{cut}^! D (u. \text{cut } E_u (y. F_{uy})) & \rightsquigarrow & (\nu u)(!u \langle y \rangle. \hat{D}^y \mid (\nu y)(\hat{E}^y \mid \hat{F}^z)) \\
\cong & \cong & \\
\text{cut } (\text{cut}^! D (u. E_u)) (y. \text{cut}^! D (u. F_{uy})) & \rightsquigarrow & (\nu y)((\nu u)(!u \langle y \rangle. \hat{D}^y \mid \hat{E}^y) \mid \\
& & (\nu u)(!u \langle y \rangle. \hat{D}^y \mid \hat{F}^z)) \\
\text{cut}^! D (u. \text{cut}^! E_u (v. F_{uv})) & \rightsquigarrow & (\nu u)(!u \langle y \rangle. \hat{D}^y \mid (\nu v)(!v \langle y \rangle. \hat{E}^y \mid \hat{F}^z)) \\
\cong & \cong & \\
\text{cut}^! (\text{cut}^! D (u. E_u)) (v. \text{cut}^! D (u. F_{uv})) & \rightsquigarrow & (\nu v)((!v \langle y \rangle. (\nu u)(!u \langle y \rangle. \hat{D}^y \mid \hat{E}^y)) \mid \\
& & (\nu u)(!u \langle y \rangle. \hat{D}^y \mid \hat{F}^z)) \\
\text{cut}^! (\text{cut}^! D (u. E_u)) (v. F_v) & \rightsquigarrow & (\nu v)(!v \langle y \rangle. (\nu u)(!u \langle y \rangle. \hat{D}^y \mid \hat{E}^y)) \mid \hat{F}^z \\
\cong & \cong & \\
\text{cut}^! D (u. \text{cut}^! E_u (v. F_v)) & \rightsquigarrow & (\nu u)(!u \langle y \rangle. \hat{D}^y \mid (\nu v)(!v \langle y \rangle. \hat{E}^y \mid \hat{F}^z)) \\
\text{cut}^! D (u. E) & \rightsquigarrow & (\nu u)(!u \langle y \rangle. \hat{D}^y \mid \hat{E}^z) \\
\cong & \cong & \\
E & \rightsquigarrow & \hat{E}^z \quad (\text{for } u \notin FN(\hat{E}^z))
\end{array}$$

Figure 8: Structural Conversions (III): Cut! Conversions

2. $\Gamma; \Delta_1, \Delta_2 \vdash D \rightsquigarrow Q_2 :: z : C$ for some $Q_2 \equiv (\nu x)(\nu y)(P'_1 \mid Q'_1)$.

Proof. See Appendix A.8. □

The key idea of the lemmas above is that by relating process reduction with proof reduction at a given type we can conclude a strong form of type preservation, as follows.

Theorem 5.3. *Let $\Gamma; \Delta \vdash D \rightsquigarrow P :: z : A$ and $P \rightarrow Q$. Then there is E such that $D \equiv \Rightarrow \equiv E$ and $\Gamma; \Delta \vdash E \rightsquigarrow Q :: z : A$*

Proof. See Appendix A.11. □

Theorems 5.4 and 5.5 state that any proof reduction or conversion also corresponds to either a process equivalence or to a reduction step on the π -calculus.

Theorem 5.4. *Let $\Gamma; \Delta \vdash D \rightsquigarrow P :: z : A$ and $D \simeq_s E$. Then there is Q where $P \simeq_s Q$ and $\Gamma; \Delta \vdash E \rightsquigarrow Q :: z : A$.*

Proof. Following the commuting squares relating \equiv , \rightsquigarrow and \simeq in Figures 6, 7 and 8. □

Theorem 5.5. *Let $\Gamma; \Delta \vdash D \rightsquigarrow P :: z : A$ and $D \Rightarrow E$. Then there is Q such that $P \rightarrow Q$ and $\Gamma; \Delta \vdash E \rightsquigarrow Q :: z : A$.*

Proof. Following the commuting squares relating \Rightarrow , \rightsquigarrow and \rightarrow in Figure 5. □

Notice that the simulation of π -calculus reductions by proof term conversions provided by Theorem 5.3, and from which subject reduction follows, is very tight indeed, as reduction is simulated up to structural congruence, which is a very fine equivalence on processes. To that end, structural conversions need to be applied symmetrically (as equations), unlike in a standard proof of cut-elimination, where they are usually considered as directed computational steps. Under the assumptions of Theorem 5.3, we can also prove that there is an E such that $D \Rightarrow E$ and $\Gamma; \Delta \vdash E \rightsquigarrow R :: z : A$, for $Q \simeq_s R$. Thus, even if one considers the proof conversions as directed reduction rules (\Rightarrow), we still obtain a sound simulation up to typed strong behavioral congruence.

We now state type preservation and progress results for our type system. The subject reduction property (Theorem 5.6) directly follows from Theorem 5.3.

Theorem 5.6 (Subject Reduction). *If $\Gamma; \Delta \vdash P :: z : A$ and $P \rightarrow Q$ then $\Gamma; \Delta \vdash Q :: z : A$.*

Together with direct consequences of linear typing, Theorem 5.6 ensures session fidelity. Our type discipline also enforces a global progress property. For any P , define

$$\text{live}(P) \text{ iff } P \equiv (\nu \bar{n})(\pi.Q \mid R) \text{ for some } \pi.Q, R, \bar{n}$$

where $\pi.Q$ is a *non-replicated* guarded process. We first establish the following contextual progress property, from which Theorem 5.8 follows as a corollary. Lemma 5.7 relies on an inversion lemma that relates types with action labels (Lemma A.1) and on a lemma that characterizes the typing of non-live processes (Lemma A.12).

Lemma 5.7. *Let $\Gamma; \Delta \vdash D \rightsquigarrow P :: z : C$. If $\text{live}(P)$ then there is Q such that either*

1. $P \rightarrow Q$, or
2. $P \xrightarrow{\alpha} Q$ for α where $s(\alpha) \in (z, \Gamma, \Delta)$. More: if $C = !A$ for some A , then $s(\alpha) \neq z$.

Proof. See Appendix A.13 □

Theorem 5.8 (Progress). *If $;\cdot \vdash D \rightsquigarrow P :: x : 1$ and $\text{live}(P)$ then exists Q st. $P \rightarrow Q$.*

Proof. See Appendix A.14 □

6 Discussion and Further Examples

We further compare our linear type system for (finite) session types with more familiar session type systems [21, 18, 13]. An immediate observation is that in our case types are freely generated, while traditionally there is a stratification of types in “session” and “standard types” (the later corresponding to our $!A$ types, typing session initiation channels). In our interpretation, a session may either terminate ($\mathbf{1}$), or become a replicated server ($!A$), which is more general and uniform, and a natural consequence of the logical interpretation. Concerning parallel composition, usually two rules can be found, one corresponding to the cancellation of two dual session endpoints (a name restriction rule), and another corresponding to independent parallel composition, also present in most linear type systems for mobile processes. In our case, cut combines both principles, and the following rule is derivable:

$$\frac{\Gamma; \Delta \vdash P :: -:\mathbf{1} \quad \Gamma; \Delta' \vdash Q :: T}{\Gamma; \Delta, \Delta' \vdash P \mid Q :: T} \text{ (comp)}$$

A consequence of the logical composition rules cut and cut¹ is that typing intrinsically enforces global progress, unlike with traditional session type systems [18, 13], which do not ensure progress in the presence of multiple open sessions, as we do here. Techniques to ensure progress in sessions, but building on extraneous devices such as well-founded orderings on events, have been proposed [20, 12]. It would be interesting to further compare the various approaches, as far as process typability is concerned.

Channel “polarities” are captured in our system by the left-right distinction of sequents, rather than by annotations on channels (cf. x^+ , x^-). Session and linear type systems [21, 18, 13] also include a typing rule for output of the form

$$\frac{\Gamma; \Delta \vdash P :: x:C}{\Gamma; \Delta, y:A \vdash x\langle y \rangle.P :: x:A \otimes C}$$

In our case, an analogous rule may be derived by $\otimes R$ and the copycat construction, where a “proxy” for the free name y , bidirectionally copying behavior A , is linked to z .

$$\frac{\Gamma; \Delta \vdash P :: x:C}{\Gamma; \Delta, y:A \vdash (\nu z)x\langle z \rangle.(id_A(y, z) \mid P) :: x:A \otimes C}$$

The copycat $id_A(y, z)$ plays the role of the “link” processes of [23, 7]. Notice that in our case the definition of the “link” is obtained for free by the interpretation of identity axioms (Proposition 4.2). The two processes can be shown to be behaviorally equivalent, under an adequate notion of observational equivalence, as in [7].

We now elaborate on the example of Section 3, in order to illustrate sharing and session initiation. Consider now a different client, that picks the “buy” rather than the “quote” operation, and the corresponding composed system.

$$\begin{aligned} BCIntBody_s &\triangleq s.in1; (\nu cof)s\langle cof \rangle.(\nu pin)s\langle pin \rangle.s(rc)\mathbf{0} \\ BSimple &\triangleq (\nu s)(SrvBody_s \mid BCIntBody_s) \end{aligned}$$

We have the typings $\cdot; s:ServerProto \vdash BCIntBody_s :: -:\mathbf{1}$ and $\cdot \vdash BSimple :: -:\mathbf{1}$.

In these examples, there is a single installed pair client-server, where the session is already initiated, and only known to the two partners. To illustrate sharing, we now consider a replicated server. Such a replicated server is able to spawn a fresh session instance for each initial invocation, each one conforming to the general behavior specified by $ServerProto$, and can be typed by $!ServerProto$. Correspondingly, clients must initially invoke the replicated server to instantiate a new session (cf. the Tcopy rule).

$$\begin{aligned} QClient &\triangleq (\nu s)c\langle s \rangle.QCIntBody_s & BClient &\triangleq (\nu s)c\langle s \rangle.BCIntBody_s \\ Server &\triangleq !c(s).SrvBody_s & SharSys &\triangleq (\nu c)(Server \mid BClient \mid QClient) \end{aligned}$$

For the shared server, by T!R, we type $\cdot; \cdot \vdash Server :: c:!ServerProto$. We also have, for the clients, by Tcopy the typings $c:ServerProto; \cdot \vdash BClient :: -:\mathbf{1}$ and $c:ServerProto; \cdot \vdash QClient :: -:\mathbf{1}$. By (comp), T!L, and Tcut we obtain the intended typing for the whole system: $\cdot; \cdot \vdash SharSys :: -:\mathbf{1}$. Notice how the session instantiation protocol is naturally explained by the logical interpretation of the $!$ operator.

7 Related Work and Conclusions

We have established a tight correspondence between a session-based type discipline for the π -calculus and intuitionistic linear logic: typing rules correspond to dual intuitionistic linear sequent calculus proof rules, moreover process reduction may be simulated in a type preserving way by proof conversions and reductions, and *vice versa*. As a result, we obtain the subject reduction property, from which session fidelity follows. Our basic typing discipline intrinsically ensures global progress, beyond the restricted “progress on a single session” property obtained in pure session type systems.

Other works have investigated π -calculus models of linear logic proofs. Bellin and Scott [5] establish a mapping from linear logic proofs to a variant of the π -calculus and some connections between proof reduction and π -calculus reduction. However, this mapping results in complex encodings, so that their system could hardly be considered a type assignment system for processes, which has been achieved in this work. Moreover, no relation between behavioral descriptions and logical propositions was identified, as put by the authors: “[our encodings] have less to do with logic than one might think, they are essentially only about the abstract pluggings in proof structures”. A realizability interpretation for a linear logic augmented with temporal modalities (cf. Hennessy-Milner) was proposed in [4], also based on a π -calculus variant. A recent related development is [17], where a correspondence between (independently formulated) proof nets and an IO-typed π -calculus is established. In our case, the type system and the logic proof system are exactly the same, and we reveal a direct connection between pure linear logic propositions and behavioral types on π -calculus, that covers all (both shared and linear) features of finite session types. A development of session types as linear process types (in the sense of [21]) is presented in [15], where linearity and sharing are expressed by special annotations, unrelated to a linear logic interpretation.

We have also analyzed the relation between our type discipline and (finite, deadlock-free) session types. It is important to notice that our interpretation does not require locality for session (linear) channels (under which only the output capability of names could be transmitted), which seems required in other works on linearity for π -calculi (e.g., [26]). On the other hand, our intuitionistic discipline enforces locality of shared channels, which, quite interestingly, seems to be the sensible choice for distributed implementations of sessions. Interesting related topics would be the accommodation of recursive types, logical relations [8], and the characterization of observational equivalences under our typing discipline. In particular, we expect that all conversions (including commuting conversions) between DILL derivations correspond to observational equivalences on our typed π -calculus.

One important motivation for choosing a purely logical approach to typing is that it often suggests uniform and expressive generalizations. In ongoing work, we have also established an explicit relationship between session-based concurrency and functional computation where in both cases determinacy (no races) and progress (deadlock-freedom) are expected features. In particular, we have been investigating new encodings of λ -calculi into the π -calculus that arise from translations from DILL natural deduction into sequent calculus. We also believe that dependent generalizations of our system of simple linear types, perhaps along the lines of LLF [9] or CLF [25], may be able to capture many additional properties of communication behavior in a purely logical manner. Already, some systems of session types have dependent character, such as [6] that, among other properties, integrates correspondence assertions into session types.

Acknowledgments. To FCT/MCTES (INTERFACES NGN44), and the ICTI at Carnegie-Mellon. Thanks also to Bernardo Toninho, Nobuko Yoshida, and Andre Platzer.

References

- [1] S. Abramsky. Computational Interpretations of Linear Logic. *Theor. Comp. Sci.*, 111(1&2), 1993.
- [2] J.-M. Andreoli. Logic Programming with Focusing Proofs in Linear Logic. *Journal of Logic and Computation*, 2(3):197–347, 1992.
- [3] A. Barber and G. Plotkin. Dual Intuitionistic Linear Logic. Technical Report LFCS-96-347, Univ. of Edinburgh, 1997.
- [4] E. Beffara. A Concurrent Model for Linear Logic. *ENTCS*, 155:147–168, 2006.

- [5] G. Bellin and P. Scott. On the π -Calculus and Linear Logic. *Theor. Comp. Sci.*, 135:11–65, 1994.
- [6] E. Bonelli, A. Compagnoni, and E. L. Gunter. Correspondence Assertions for Process Synchronization in Concurrent Communications. *J. of Func. Prog.*, 15(2):219–247, 2005.
- [7] M. Boreale. On the Expressiveness of Internal Mobility in Name-Passing Calculi. *Theoretical Computer Science*, 195(2):205–226, 1998.
- [8] L. Caires. Logical Semantics of Types for Concurrency. In *International Conference on Algebra and Coalgebra in Computer Science*, CALCO'07, pages 16–35. Springer LNCS 4624, 2007.
- [9] I. Cervesato and F. Pfenning. A linear logical framework. *Inf. & Comput.*, 179(1), 2002.
- [10] B.-Y. E. Chang, K. Chaudhuri, and F. Pfenning. A Judgmental Analysis of Linear Logic. Technical Report CMU-CS-03-131R, Carnegie Mellon University, 2003.
- [11] M. Dezani-Ciancaglini and U. de' Liguoro. Sessions and Session Types: an Overview. In *6th Intl Workshop on Web Services and Formal Methods WS-FM'09*, Lecture Notes in Computer Science. Springer-Verlag, 2010.
- [12] M. Dezani-Ciancaglini, U. de' Liguoro, and N. Yoshida. On Progress for Structured Communications. In G. Barthe and C. Fournet, editors, *3th Symposium Trustworthy Global Computing, TGC 2007*, pages 257–275. Springer LNCS 4912, 2008.
- [13] S. Gay and M. Hole. Subtyping for Session Types in the Pi Calculus. *Acta Informatica*, 42(2-3):191–225, 2005.
- [14] J.-Y. Girard and Y. Lafont. Linear Logic and Lazy Computation. In H. Ehrig, R. A. Kowalski, G. Levi, and U. Montanari, editors, *Theory and Practice of Software Development, TAPSOFT'87*, pages 52–66. Springer LNCS 250, 1987.
- [15] M. Giunti and V. T. Vasconcelos. A Linear Account of Session Types in the Pi-Calculus. In *21st International Conference on Concurrency Theory*, CONCUR'10, pages 432–446. Springer LNCS 6269, 2010.
- [16] K. Honda. Types for dyadic interaction. In *4th International Conference on Concurrency Theory*, CONCUR'93, pages 509–523. Springer LNCS 715, 1993.
- [17] K. Honda and O. Laurent. An exact correspondence between a typed pi-calculus and polarised proof-nets. *Theor. Comp. Sci.*, 411:2223–2238, 2010.
- [18] K. Honda, V. T. Vasconcelos, and M. Kubo. Language primitives and type discipline for structured communication-based programming. In *ESOP'98*, pages 122–138. Springer LNCS 1381, 1998.
- [19] J. M. E. Hyland and C.-H. L. Ong. Pi-Calculus, Dialogue Games and PCF. In *WG2.8 Conference on Functional Programming Languages*, pages 96–107, 1995.
- [20] N. Kobayashi. A Partially Deadlock-Free Typed Process Calculus. *ACM Tr. Progr. Lang. Sys.*, 20(2):436–482, 1998.
- [21] N. Kobayashi, B. C. Pierce, and D. N. Turner. Linearity and the pi-calculus. In *23rd Symposium on Principles of Programming Languages*, POPL'96, pages 358–371. ACM, 1996.
- [22] R. Milner. Functions as processes. *Math. Struct. in Comp. Sci.*, 2(2):119–141, 1992.
- [23] D. Sangiorgi. Pi-Calculus, Internal Mobility, and Agent Passing Calculi. *Theoretical Computer Science*, 167(1&2):235–274, 1996.
- [24] D. Sangiorgi and D. Walker. *The π -calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

- [25] K. Watkins, I. Cervesato, F. Pfenning, and D. Walker. Specifying properties of concurrent computations in CLF. In C.Schürmann, editor, *4th Intl. Workshop on Logical Frameworks and Meta-Languages (LFM'04)*, Cork, Ireland, July 2004. ENTCS, vol 199.
- [26] N. Yoshida, K. Honda, and M. Berger. Linearity and Bisimulation. *J. Logic and Algebraic Programming*, 72(2):207–238, 2007.

A Proofs

A.1 Inversion Lemmas

Lemma A.1. *Let $\Gamma; \Delta \vdash D \rightsquigarrow P :: x : C$.*

1. *If $P \xrightarrow{\alpha} Q$ and $C = \mathbf{1}$ then $s(\alpha) \neq x$.*
2. *If $P \xrightarrow{\alpha} Q$ and $y : \mathbf{1} \in \Delta$ then $s(\alpha) \neq y$.*
3. *If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \otimes B$ then $\alpha = \overline{(\nu y)x\langle y \rangle}$.*
4. *If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \otimes B \in \Delta$ then $\alpha = y(z)$.*
5. *If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \multimap B$ then $\alpha = x(y)$.*
6. *If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \multimap B \in \Delta$ then $\alpha = \overline{(\nu z)y\langle z \rangle}$.*
7. *If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \& B$ then $\alpha = x.\text{inl}$ or $\alpha = x.\text{inr}$.*
8. *If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \& B \in \Delta$ then $\alpha = \overline{y.\text{inl}}$ or $\alpha = \overline{y.\text{inr}}$.*
9. *If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \oplus B$ then $\alpha = \overline{x.\text{inl}}$ or $\alpha = \overline{x.\text{inr}}$.*
10. *If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \oplus B \in \Delta$ then $\alpha = y.\text{inl}$ or $\alpha = y.\text{inr}$.*
11. *If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = !A$ then $\alpha = x(y)$.*
12. *If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : !A$ or $y \in \Gamma$ then $\alpha = \overline{(\nu z)y\langle z \rangle}$.*

Proof. By induction on the structure of D .

1. If $P \xrightarrow{\alpha} Q$ and $C = \mathbf{1}$ then $s(\alpha) \neq x$.

Case: copy, all left rules except 1L and !L

$$s(\alpha) \neq x$$

by the definition of the l.t.s.

Case: 1L or !L

$$s(\alpha) \neq x$$

by i.h.

Case: cut $D_1 (y. D_2)$

$$D_1 \rightsquigarrow P_1 \text{ and } D_2 \rightsquigarrow P_2$$

Subcase: $P_1 \xrightarrow{\alpha} Q_1$

$$s(\alpha) \neq x$$

trivial, since $x \notin \text{fn}(P_1)$ by typing

Subcase: $P_2 \xrightarrow{\alpha} Q_2$

$$s(\alpha) \neq x$$

by i.h. on D_2

Case: cut[!] $D_1 (u. D_2)$

$D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $P_1 \xrightarrow{\alpha} Q_1$

$s(\alpha) \neq x$

trivial, since $x \notin fn(P_1)$ by typing

Subcase: $P_2 \xrightarrow{\alpha} Q_2$

$s(\alpha) \neq x$

by i.h. on D_2

Case: All other rules do not offer $\xrightarrow{\alpha}$ or $C \neq \mathbf{1}$

2. If $P \xrightarrow{\alpha} Q$ and $y : \mathbf{1} \in \Delta$ then $s(\alpha) \neq y$.

Case: copy, all left rules except 1L and !L

$s(\alpha) \neq y$

by the definition of the l.t.s.

Case: 1L or !L

$s(\alpha) \neq y$

by i.h.

Case: $\Gamma; \Delta_1, \Delta_2 \vdash \text{cut } D_1 (z. D_2) \rightsquigarrow P :: x : C$, with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $y : \mathbf{1} \in \Delta_1$

$s(\alpha) \neq y$

by i.h. and $y \notin fn(P_2)$

Subcase: $y : \mathbf{1} \in \Delta_2$ with $z \neq y$

$s(\alpha) \neq y$

by i.h. and $y \notin fn(P_1)$

Case: $\text{cut}^! D_1 (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $P_1 \xrightarrow{\alpha} Q_1$

$s(\alpha) \neq y$

trivial, since linear ctxt. is empty for D_1 and $y : \mathbf{1}$

Subcase: $P_2 \xrightarrow{\alpha} Q_2$

$s(\alpha) \neq y$

by i.h.

3. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \otimes B$ then $\alpha = \overline{(\nu y)x\langle y \rangle}$

Case: 1L or !L

$\alpha = \overline{(\nu y)x\langle y \rangle}$

by i.h.

Case: $\otimes R$

$\alpha = \overline{(\nu y)x\langle y \rangle}$

by the l.t.s

Case: $\text{cut } D_1 (y. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$\alpha = \overline{(\nu y)x\langle y \rangle}$

by i.h. on D_2 and $x \notin fn(P_1)$

Case: $\text{cut}^! D_1 (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$\alpha = \overline{(\nu y)x\langle y \rangle}$

by i.h. on D_2

Case: All other rules do not have $s(\alpha) = x$ and $C = A \otimes B$

4. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \otimes B \in \Delta$ then $\alpha = y(z)$

Case: 1L or !L

$\alpha = y(z)$

by i.h.

Case: $\otimes L$

$\alpha = y(z)$

by the l.t.s

Case: $\Gamma; \Delta_1, \Delta_2 \vdash \text{cut } D_1 (z. D_2) \rightsquigarrow P :: x : C$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $y \in \Delta_1$

$$\alpha = y(z)$$

by i.h. on D_1 and $y \notin \text{fn}(P_2)$

Subcase: $y \in \Delta_2$

$$\alpha = y(z)$$

by i.h. on D_2 and $y \notin \text{fn}(P_1)$

Case: $\text{cut}^! D_1 (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$$\alpha = y(z)$$

by i.h. on D_2

5. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \multimap B$ then $\alpha = x(y)$

Case: 1L or !L

$$\alpha = x(y)$$

by i.h.

Case: $\multimap R$

$$\alpha = x(y)$$

by the l.t.s

Case: $\text{cut } D_1 (y. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$$\alpha = x(y)$$

by i.h. on D_2 and $x \notin \text{fn}(P_1)$

Case: $\text{cut}^! D_1 (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$$\alpha = x(y)$$

by i.h. on D_2

Case: All other rules do not have $s(\alpha) = x$ and $C = A \multimap B$

6. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \multimap B \in \Delta$ then $\alpha = \overline{(\nu z)y\langle z \rangle}$

Case: 1L or !L

$$\alpha = \overline{(\nu z)y\langle z \rangle}$$

by i.h.

Case: $\multimap L$

$$\alpha = \overline{(\nu z)y\langle z \rangle}$$

by the l.t.s

Case: $\Gamma; \Delta_1, \Delta_2 \vdash \text{cut } D_1 (w. D_2) \rightsquigarrow P :: x : C$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $y \in \Delta_1$

$$\alpha = \overline{(\nu z)y\langle z \rangle}$$

by i.h. on D_1 and $y \notin \text{fn}(P_2)$

Subcase: $y \in \Delta_2$

$$\alpha = \overline{(\nu z)y\langle z \rangle}$$

by i.h. on D_2 and $y \notin \text{fn}(P_1)$

Case: $\text{cut}^! D_1 (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$$\alpha = \overline{(\nu z)y\langle z \rangle}$$

by i.h. on D_2

7. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \& B$ then $\alpha = x.\text{inl}$ or $\alpha = x.\text{inr}$

Case: 1L or !L

$$\alpha = x.\text{inl} \text{ or } \alpha = x.\text{inr}$$

by i.h.

Case: $\& R$

$$\alpha = x.\text{inl} \text{ or } \alpha = x.\text{inr}$$

by the l.t.s

Case: cut $D_1 (y. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$$\alpha = x(y)$$

by i.h. on D_2 and $x \notin fn(P_1)$

Case: cut[!] $D_1 (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$$\alpha = x(y)$$

by i.h. on D_2

Case: All other rules do not have $s(\alpha) = x$ and $C = A \& B$

8. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \& B \in \Delta$ then $\alpha = \overline{y.inl}$ or $\alpha = \overline{y.inr}$

Case: 1L or !L

$$\alpha = \overline{y.inl} \text{ or } \alpha = \overline{y.inr}$$

by i.h.

Case: $\&L_1$

$$\alpha = \overline{y.inl}$$

by the l.t.s

Case: $\&L_2$

$$\alpha = \overline{y.inr}$$

by the l.t.s

Case: $\Gamma; \Delta_1, \Delta_2 \vdash \text{cut } D_1 (w. D_2) \rightsquigarrow P :: x : C$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

Subcase: $y \in \Delta_1$

$$\alpha = \overline{y.inl} \text{ or } \alpha = \overline{y.inr}$$

by i.h. on D_1 and $y \notin fn(P_2)$

Subcase: $y \in \Delta_2$

$$\alpha = \overline{y.inl} \text{ or } \alpha = \overline{y.inr}$$

by i.h. on D_2 and $y \notin fn(P_1)$

Case: cut[!] $D_1 (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$$\alpha = \overline{y.inl} \text{ or } \alpha = \overline{y.inr}$$

by i.h. on D_2

9. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \oplus B$ then $\alpha = \overline{x.inl}$ or $\alpha = \overline{x.inr}$

Case: 1L or !L

$$\alpha = \overline{x.inl} \text{ or } \alpha = \overline{x.inr}$$

by i.h.

Case: $\oplus R_1$

$$\alpha = \overline{x.inl}$$

by the l.t.s

Case: $\oplus R_2$

$$\alpha = \overline{x.inr}$$

by the l.t.s

Case: cut $D_1 (y. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$$\alpha = \overline{x.inl} \text{ or } \alpha = \overline{x.inr}$$

by i.h. on D_2 and $x \notin fn(P_1)$

Case: cut[!] $D_1 (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$$\alpha = \overline{x.inl} \text{ or } \alpha = \overline{x.inr}$$

by i.h. on D_2

Case: All other rules do not have $s(\alpha) = x$ and $C = A \oplus B$

10. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : A \oplus B \in \Delta$ then $\alpha = y.inl$ or $\alpha = y.inr$

Case: 1L or !L

$$\alpha = \overline{y.inl} \text{ or } \alpha = \overline{y.inr}$$

by i.h.

Case: $\oplus L$

$$\alpha = \overline{y.inl} \text{ or } \alpha = \overline{y.inr} \quad \text{by the l.t.s}$$

Case: $\Gamma; \Delta_1, \Delta_2 \vdash \text{cut } D_1 (w. D_2) \rightsquigarrow P :: x : C \text{ with } D_1 \rightsquigarrow P_1 \text{ and } D_2 \rightsquigarrow P_2$

Subcase: $y \in \Delta_1$
 $\alpha = \overline{y.inl} \text{ or } \alpha = \overline{y.inr}$ by i.h. on D_1 and $y \notin fn(P_2)$

Subcase: $y \in \Delta_2$
 $\alpha = \overline{y.inl} \text{ or } \alpha = \overline{y.inr}$ by i.h. on D_2 and $y \notin fn(P_1)$

Case: $\text{cut}^! D_1 (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$$\alpha = \overline{y.inl} \text{ or } \alpha = \overline{y.inr} \quad \text{by i.h. on } D_2$$

11. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = !A$ then $\alpha = x(y)$

Case: 1L or !L

$$\alpha = x(y) \quad \text{by i.h.}$$

Case: !R

$$\alpha = x(y) \quad \text{by the l.t.s}$$

Case: $\text{cut } D_1 (z. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$$\alpha = x(y) \quad \text{by i.h. on } D_2 \text{ and } x \notin fn(P_1)$$

Case: $\text{cut}^! D_1 (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$$\alpha = x(y) \quad \text{by i.h. on } D_2$$

Case: All other rules do not have $s(\alpha) = x$ and $C = !A$

12. If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y : !A$ or $y \in \Gamma$ then $\alpha = \overline{(\nu z)y\langle z \rangle}$.

Case: 1L or !L

$$\alpha = \overline{(\nu z)y\langle z \rangle} \quad \text{by i.h.}$$

Case: copy

$$\alpha = \overline{(\nu z)y\langle z \rangle} \quad \text{by the l.t.s}$$

Case: $\Gamma; \Delta_1, \Delta_2 \vdash \text{cut } D_1 (w. D_2) \rightsquigarrow P :: x : C \text{ with } D_1 \rightsquigarrow P_1 \text{ and } D_2 \rightsquigarrow P_2$

Subcase: $y : !A$ and $y \in \Delta_1$
 $\alpha = \overline{(\nu z)y\langle z \rangle}$ by i.h. on D_1 and $y \notin fn(P_2)$

Subcase: $y : !A$ and $y \in \Delta_2$
 $\alpha = \overline{(\nu z)y\langle z \rangle}$ by i.h. on D_2 and $y \notin fn(P_1)$

Subcase: $y \in \Gamma$ and $P \xrightarrow{\alpha} Q$ from $P_1 \xrightarrow{\alpha} Q_1$
 $\alpha = \overline{(\nu z)y\langle z \rangle}$ by i.h. on D_1

Subcase: $y \in \Gamma$ and $P \xrightarrow{\alpha} Q$ from $P_2 \xrightarrow{\alpha} Q_2$
 $\alpha = \overline{(\nu z)y\langle z \rangle}$ by i.h. on D_2

Case: $\text{cut}^! D_1 (u. D_2)$ with $D_1 \rightsquigarrow P_1$ and $D_2 \rightsquigarrow P_2$

$$\begin{array}{l} \text{Subcase: } y : !A \\ \alpha = \overline{(\nu z)y\langle z \rangle} \\ \text{Subcase: } y \in \Gamma \\ \alpha = \overline{(\nu z)y\langle z \rangle} \end{array}$$

by i.h. on D_2

by i.h. on D_2

□

A.2 Preservation Lemmas

Lemma A.2. *Assume*

$$(a) \Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x:C_1 \& C_2 \text{ with } P \xrightarrow{x.\text{inl}} P';$$

$$(b) \Gamma; \Delta_2, x:C_1 \& C_2 \vdash E \rightsquigarrow Q :: z:C \text{ with } Q \xrightarrow{x.\text{inl}} Q'.$$

Then

$$(c) \text{cut } D(x.E) \equiv \equiv F \text{ for some } F;$$

$$(d) \Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z : C \text{ for some } R \equiv (\nu x)(P' \mid Q').$$

Proof. By simultaneous induction on D and E . That is, in each appeal to the induction hypothesis either D becomes smaller and E remains the same, or D remains the same and E becomes smaller.

The possible cases for D are $\&R$, $1L$, $!L$, cut , and cut' . In all other cases P^x cannot offer $x.\text{inl}$. The possible cases for E are $\&L_1$, $1L$, $!L$, cut , and cut' .

Case: $D = \&R D_1 D_2$ and $E = \&L_1 x(x.E_1)$.

$$\text{cut } D(x.E) = \text{cut } (\&R D_1 D_2) (\&L_1 x(x.E_1)) = F \quad \text{[a]}$$

$$F \rightsquigarrow (\nu x)(x.\text{case}(P_1^x, P_2^x) \mid x.\text{inl}; Q_1^z) = (\nu x)(P^x \mid Q^z) \quad \text{[b]}$$

$$F \Rightarrow \text{cut } D_1(x.E_1) = F' \quad \text{By reduction (cut/\&R/\&L_1) [c]}$$

$$F' \rightsquigarrow (\nu x)(P_1^x \mid Q_1^z) \equiv R' \text{ with } P_1^x = P'^x \text{ and } Q_1^z = Q'^z \quad \text{[d]}$$

Case: $D = 1L y D_1$ and E arbitrary.

$$\text{cut } D(x.E) = \text{cut } (1L y D_1) (x.E)$$

$$\equiv 1L y (\text{cut } D_1(x.E)) = F$$

By rule (cut/1L/-) [a]

$$F \rightsquigarrow (\nu x)(P^x \mid Q^z) = R$$

since $D_1 \rightsquigarrow P^x$ [b]

$$\text{cut } D_1(x.E) \equiv F_1 \text{ for some } F_1;$$

$$F_1 \rightsquigarrow R_1^z \equiv (\nu x)(P^x \mid Q^z);$$

$$F_1 \Rightarrow F'_1;$$

$$F'_1 \rightsquigarrow R_1'^z \equiv (\nu x)(P'^x \mid Q'^z)$$

By i.h. on D_1, E

$$F = 1L y F_1 \Rightarrow 1L y F'_1 = F'$$

By congruence [c]

$$F' \rightsquigarrow R_1^z \equiv (\nu x)(P'^x \mid Q'^z) \quad \text{[d]}$$

Case: $D = !L y (u.D_1)$ and E arbitrary.

$$\text{cut } D(x.E) = \text{cut } (!L y (u.D_1)) (x.E)$$

$$\equiv !L y (u.\text{cut } D_1(x.E)) = F$$

By rule (cut!/L/-) [a]

$$F \rightsquigarrow (\nu x)(P^x \mid Q^z) = R$$

since $D_1 \rightsquigarrow P^x\{u/y\}$ [b]

$$\begin{aligned}
& \text{cut } D_1 (x. E) \equiv F_1 \text{ for some } F_1; \\
& F_1 \rightsquigarrow R_1^z \equiv (\nu x)(P^x \mid Q^z); \\
& F_1 \Rightarrow F_1'; \\
& F_1' \rightsquigarrow R_1'^z \equiv (\nu x)(P'^x \mid Q'^z) \qquad \text{By i.h. on } D_1, E \\
& F = !L y (u. F_1) \Rightarrow !L y (u. F_1') = F' \qquad \text{By congruence [c]} \\
& F' \rightsquigarrow R_1^z \equiv (\nu x)(P'^x \mid Q'^z) \qquad \text{[d]}
\end{aligned}$$

Case: $D = \text{cut } D_1 (y. D_2)$ and E arbitrary.

$$\begin{aligned}
& \text{cut } D (x. E) = \text{cut} (\text{cut } D_1 (y. D_2)) (x. E) \\
& \equiv \text{cut } D_1 (y. \text{cut } D_2 (x. E)) = F \qquad \text{By rule (cut/-/cut}_1\text{) [a]} \\
& F \rightsquigarrow (\nu y)(P_1^y \mid ((\nu x)(P_2^x \mid Q^z))) \equiv (\nu x)((\nu y)(P_1^y \mid P_2^x) \mid Q^z) = R \qquad \text{[b]} \\
& \text{cut } D_2 (x. E) \equiv F_2 \text{ with} \\
& F_2 \rightsquigarrow R_2 \text{ and } R_2 \equiv (\nu x)(P_2^x \mid Q^z) \text{ and} \\
& F_2 \Rightarrow F_2' \text{ with} \\
& F_2' \rightsquigarrow R_2' \equiv (\nu x)(P_2'^x \mid Q'^z) \qquad \text{By i.h. on } D_2, E \\
& F \Rightarrow \text{cut } D_1 (y. F_2') = F' \qquad \text{By congruence [c]} \\
& F' \rightsquigarrow (\nu y)(P_1^y \mid (\nu x)(P_2'^x \mid Q'^z)) \equiv (\nu x)((\nu y)(P_1^y \mid P_2'^x) \mid Q'^z) \\
& = (\nu x)(P'^x \mid Q'^z) \qquad \text{[d]}
\end{aligned}$$

Case: $D = \text{cut}^! D_1 (u. D_2)$ and E arbitrary.

$$\begin{aligned}
& \text{cut } D (x. E) = \text{cut} (\text{cut}^! D_1 (u. D_2)) (x. E) \\
& \equiv \text{cut}^! D_1 (u. \text{cut } D_2 (x. E)) = F \qquad \text{By rule (cut/cut}^!\text{-) [a]} \\
& F \rightsquigarrow (\nu u)((!u(y).P_1^y) \mid (\nu x)(P_2^x \mid Q^z)) \\
& \equiv (\nu x)((\nu u)((!u(y).P_1^y) \mid P_2^x) \mid Q^z) = R \qquad \text{By struct. cong. [b]} \\
& \text{cut } D_2 (x. E) \equiv F_2 \text{ with} \\
& F_2 \rightsquigarrow R_2 \text{ and } R_2 \equiv (\nu x)(P_2^x \mid Q^z) \text{ and} \\
& F_2 \Rightarrow F_2' \text{ with} \\
& F_2' \rightsquigarrow R_2' \equiv (\nu x)(P_2'^x \mid Q'^z) \qquad \text{By i.h. on } D_2, E \\
& F \Rightarrow \text{cut}^! D_1 (u. F_2') = F' \qquad \text{By congruence [c]} \\
& F' \rightsquigarrow (\nu u)((!u(y).P_1^y) \mid (\nu x)(P_2'^x \mid Q'^z)) \\
& \equiv (\nu x)((\nu u)((!u(y).P_1^y) \mid P_2'^x) \mid Q'^z) = (\nu x)(P'^x \mid Q'^z) \qquad \text{[d]}
\end{aligned}$$

Case: $E = !L y E_1$ and D arbitrary.

$$\begin{aligned}
& \text{cut } D (x. E) = \text{cut } D (x. !L y E_1) \\
& \equiv !L y (\text{cut } D (x. E_1)) = F \qquad \text{By rule (cut/-/!L) [a]} \\
& F \rightsquigarrow (\nu x)(P^x \mid Q^z) = R \qquad \text{[b]} \\
& \text{cut } D (x. E_1) \equiv F_1 \text{ for some } F_1; \\
& F_1 \rightsquigarrow R_1^z \equiv (\nu x)(P^x \mid Q^z); \\
& F_1 \Rightarrow F_1'; \\
& F_1' \rightsquigarrow R_1' \equiv (\nu x)(P'^x \mid Q'^z) \qquad \text{By i.h. on } D, E_1 \\
& F = !L y F_1 \Rightarrow !L y F_1' = F' \qquad \text{By congruence [c]} \\
& F' \rightsquigarrow R_1^z \equiv (\nu x)(P'^x \mid Q'^z) \qquad \text{[d]}
\end{aligned}$$

Case: $E = !L y (u. E_1)$ and D arbitrary.

$$\begin{aligned}
\text{cut } D(x.E) &= \text{cut } D(x.!\mathbb{L} y(u.E_1)) \\
&\equiv !\mathbb{L} y(u.\text{cut } D(x.E_1)) = F && \text{By rule (cut/-/!L) [a]} \\
F &\rightsquigarrow (\nu x)(P^x | Q^z) = R && \text{since } E_1 \rightsquigarrow Q^z\{u/y\} \text{ [b]} \\
\text{cut } D(x.E_1) &\equiv F_1 \text{ for some } F_1; \\
F_1 &\rightsquigarrow R_1^z \equiv (\nu x)(P^x | Q^z); \\
F_1 &\Rightarrow F_1'; \\
F_1' &\rightsquigarrow R_1' \equiv (\nu x)(P'^x | Q'^z) && \text{By i.h. on } D, E_1 \\
F &= !\mathbb{L} y(u.F_1) \Rightarrow \mathbb{1}\mathbb{L} y(u.F_1') = F' && \text{By congruence [c]} \\
F' &\rightsquigarrow R_1^z \equiv (\nu x)(P'^x | Q'^z) && \text{[d]}
\end{aligned}$$

Case: $E = \text{cut } E_1(y.E_2)$ with $x \in FV(E_1)$ and D arbitrary.

$$\begin{aligned}
\text{cut } D(x.E) &= \text{cut } D(x.\text{cut } E_1(y.E_2)) \\
&\equiv \text{cut } (\text{cut } D(x.E_1))(y.E_2) = F && \text{By reduction (cut/-/cut}_1\text{) [a]} \\
F &\rightsquigarrow (\nu y)((\nu x)(P^x | Q_1^y) | Q_2^z) \equiv (\nu x)(P^x | (\nu y)(Q_1^y | Q_2^z)) = R && \text{[b]} \\
\text{cut } D(x.E_1) &\equiv F_1 \text{ with} \\
F_1 &\rightsquigarrow R_1 \equiv (\nu x)(P^x | Q_1^y) \text{ and} \\
F_1 &\Rightarrow F_1' \text{ with} \\
F_1' &\rightsquigarrow R_1' \equiv (\nu x)(P'^x | Q_1'^y) && \text{By i.h. on } D, E_1 \\
F &\Rightarrow \text{cut } F_1'(y.E_2) = F' && \text{By congruence [c]} \\
F' &\rightsquigarrow (\nu y)((\nu x)(P'^x | Q_1'^y) | Q_2^z) \equiv (\nu x)(P'^x | (\nu y)(Q_1'^y | Q_2^z)) \\
&= (\nu x)(P'^x | Q'^z) && \text{[d]}
\end{aligned}$$

Case: $E = \text{cut } E_1(y.E_2)$ with $x \in FV(E_2)$ and D arbitrary.

$$\begin{aligned}
\text{cut } D(x.E) &= \text{cut } D(x.\text{cut } E_1(y.E_2)) \\
&\equiv \text{cut } E_1(y.\text{cut } D(x.E_2)) = F && \text{By reduction (cut/-/cut}_2\text{) [a]} \\
F &\rightsquigarrow (\nu y)(Q_1^y | (\nu x)(P^x | Q_2^z)) \equiv (\nu x)(P^x | (\nu y)(Q_1^y | Q_2^z)) = R && \text{[b]} \\
\text{cut } D(x.E_2) &\equiv F_2 \text{ with} \\
F_2 &\rightsquigarrow R_2 \equiv (\nu x)(P^x | Q_2^z) \text{ and} \\
F_2 &\Rightarrow F_2' \text{ with} \\
F_2' &\rightsquigarrow R_2' \equiv (\nu x)(P'^x | Q_2'^z) && \text{By i.h. on } D, E_2 \\
F &\Rightarrow \text{cut } E_1(y.F_2') = F' && \text{By congruence [c]} \\
F' &\rightsquigarrow (\nu y)(Q_1^y | (\nu x)(P'^x | Q_2'^z)) \equiv (\nu x)(P'^x | (\nu y)(Q_1^y | Q_2'^z)) \\
&= (\nu x)(P'^x | Q'^z) && \text{[d]}
\end{aligned}$$

Case: $E = \text{cut}^! E_1(u.E_2)$ with $x \in FV(E_1)$ and D arbitrary. This case is impossible because E_1 contains no free linear variables.

Case: $E = \text{cut}^! E_1(u.E_2)$ with $x \in FV(E_2)$ and D arbitrary.

$$\begin{aligned}
\text{cut } D(x.E) &= \text{cut } D(x.\text{cut}^! E_1(u.E_2)) \\
&\equiv \text{cut}^! E_1(u.\text{cut } D(x.E_2)) = F && \text{By reduction (cut/-/cut}^!\text{) [a]} \\
F &\rightsquigarrow (\nu u)((!u(y).Q_1^y) | (\nu x)(P^x | Q_2^z)) \equiv \\
&(\nu x)(P^x | (\nu u)((!u(y).Q_1^y) | Q_2^z)) = R && \text{[b]} \\
\text{cut } D(x.E_2) &\equiv F_2 \text{ with} \\
F_2 &\rightsquigarrow R_2 \equiv (\nu x)(P^x | Q_2^z) \text{ and} \\
F_2 &\Rightarrow F_2' \text{ with}
\end{aligned}$$

$$\begin{aligned}
F'_2 \rightsquigarrow R'_2 &\equiv (\nu x)(P'^x \mid Q'^z) && \text{By i.h. on } D, E_2 \\
F &\Rightarrow \text{cut}^! E_1 (u. F'_2) = F' && \text{By congruence [c]} \\
F' &\rightsquigarrow (\nu u)((!u(y).Q_1^y) \mid (\nu x)(P'^x \mid Q'^z)) \\
&\equiv (\nu x)(P'^x \mid (\nu u)((!u(y).Q_1^y) \mid Q'^z)) = (\nu x)(P'^x \mid Q'^z) && \text{[d]}
\end{aligned}$$

□

Lemma A.3. *Assume*

- (a) $\Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x:C_1 \& C_2$ with $P \xrightarrow{x.\text{inr}} P'$;
- (b) $\Gamma; \Delta_2, x:C_1 \& C_2 \vdash E \rightsquigarrow Q :: z:C$ with $Q \xrightarrow{x.\text{inr}} Q'$.

Then

- (c) $\text{cut } D(x. E) \equiv \Rightarrow \equiv F$ for some F ;
- (d) $\Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z : C$ for some $R \equiv (\nu x)(P' \mid Q')$.

Proof. Completely symmetric to the previous lemma.

□

Lemma A.4. *Assume*

- (a) $\Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x:C_1 \oplus C_2$ with $P \xrightarrow{x.\text{inl}} P'$;
- (b) $\Gamma; \Delta_2, x:C_1 \oplus C_2 \vdash E \rightsquigarrow Q :: z:C$ with $Q \xrightarrow{x.\text{inl}} Q'$.

Then

- (c) $\text{cut } D(x. E) \equiv \Rightarrow \equiv F$ for some F ;
- (d) $\Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z : C$ for some $R \equiv (\nu x)(P' \mid Q')$.

Proof. By simultaneous induction on D and E . That is, in each appeal to the induction hypothesis either D becomes smaller and E remains the same, or D remains the same and E becomes smaller.

The possible cases for D are $\oplus R_1$, $\mathbf{1L}$, $!L$, cut , and $\text{cut}^!$. In all other cases P^x cannot offer $\overline{x.\text{inl}}$. The possible cases for E are $\oplus L$, $\mathbf{1L}$, $!L$, cut , and $\text{cut}^!$.

Case: $D = \oplus R_1 D_1$ and $E = \oplus L x (x. E_1) (x. E_2)$.

$$\begin{aligned}
\text{cut } D (x. E) &= \text{cut } (\oplus R_1 D_1) (\oplus L x (x. E_1) (x. E_2)) = F && \text{[a]} \\
F &\rightsquigarrow (\nu x)(x.\text{inl}; P_1^x \mid x.\text{case}(Q_1^z, Q_2^z)) = (\nu x)(P^x \mid Q^z) && \text{[b]} \\
F &\Rightarrow \text{cut } D_1 (x. E_1) = F' && \text{By reduction (cut}/\oplus R_1/\oplus L) \text{ [c]} \\
F' &\rightsquigarrow (\nu x)(P_1^x \mid Q_1^z) \equiv R' \text{ with } P_1^x = P'^x \text{ and } Q_1^z = Q'^z && \text{[d]}
\end{aligned}$$

Case: $D = \mathbf{1L} y D_1$ and E arbitrary.

$$\begin{aligned}
\text{cut } D (x. E) &= \text{cut } (\mathbf{1L} y D_1) (x. E) \\
&\equiv \mathbf{1L} y (\text{cut } D_1 (x. E)) = F && \text{By rule (cut}/\mathbf{1L}/-) \text{ [a]} \\
F &\rightsquigarrow (\nu x)(P^x \mid Q^z) = R && \text{since } D_1 \rightsquigarrow P^x \text{ [b]} \\
\text{cut } D_1 (x. E) &\equiv F_1 \text{ for some } F_1; \\
F_1 &\rightsquigarrow R_1^z \equiv (\nu x)(P^x \mid Q^z); \\
F_1 &\Rightarrow F'_1; \\
F'_1 &\rightsquigarrow R'^z \equiv (\nu x)(P'^x \mid Q'^z) && \text{By i.h. on } D_1, E \\
F &= \mathbf{1L} y F_1 \Rightarrow \mathbf{1L} y F'_1 = F' && \text{By congruence [c]} \\
F' &\rightsquigarrow R_1^z \equiv (\nu x)(P'^x \mid Q'^z) && \text{[d]}
\end{aligned}$$

Case: $D = !L y (u. D_1)$ and E arbitrary.

$$\begin{aligned} \text{cut } D (x. E) &= \text{cut } (!L y (u. D_1)) (x. E) \\ &\equiv !L y (u. \text{cut } D_1 (x. E)) = F \end{aligned}$$

By rule (cut/!L/-) [a]

$$F \rightsquigarrow (\nu x)(P^x | Q^z) = R$$

since $D_1 \rightsquigarrow P^x\{u/y\}$ [b]

$$\text{cut } D_1 (x. E) \equiv F_1 \text{ for some } F_1;$$

$$F_1 \rightsquigarrow R_1^z \equiv (\nu x)(P^x | Q^z);$$

$$F_1 \Rightarrow F'_1;$$

$$F'_1 \rightsquigarrow R_1'^z \equiv (\nu x)(P'^x | Q'^z)$$

By i.h. on D_1, E

$$F = !L y (u. F_1) \Rightarrow !L y (u. F'_1) = F'$$

By congruence [c]

$$F' \rightsquigarrow R_1^z \equiv (\nu x)(P'^x | Q'^z)$$

[d]

Case: $D = \text{cut } D_1 (y. D_2)$ and E arbitrary.

$$\begin{aligned} \text{cut } D (x. E) &= \text{cut } (\text{cut } D_1 (y. D_2)) (x. E) \\ &\equiv \text{cut } D_1 (y. \text{cut } D_2 (x. E)) = F \end{aligned}$$

By rule (cut/-/cut₁) [a]

$$F \rightsquigarrow (\nu y)(P_1^y | ((\nu x)(P_2^x | Q^z))) \equiv (\nu x)((\nu y)(P_1^y | P_2^x) | Q^z) = R$$

[b]

$$\text{cut } D_2 (x. E) \equiv F_2 \text{ with}$$

$$F_2 \rightsquigarrow R_2 \text{ and } R_2 \equiv (\nu x)(P_2^x | Q^z) \text{ and}$$

$$F_2 \Rightarrow F'_2 \text{ with}$$

$$F'_2 \rightsquigarrow R_2' \equiv (\nu x)(P_2'^x | Q'^z)$$

By i.h. on D_2, E

$$F \Rightarrow \text{cut } D_1 (y. F_2) = F'$$

By congruence [c]

$$F' \rightsquigarrow (\nu y)(P_1^y | ((\nu x)(P_2'^x | Q'^z))) \equiv (\nu x)((\nu y)(P_1^y | P_2'^x) | Q'^z)$$

$$= (\nu x)(P'^x | Q'^z)$$

[d]

Case: $D = \text{cut}^1 D_1 (u. D_2)$ and E arbitrary.

$$\begin{aligned} \text{cut } D (x. E) &= \text{cut } (\text{cut}^1 D_1 (u. D_2)) (x. E) \\ &\equiv \text{cut}^1 D_1 (u. \text{cut } D_2 (x. E)) = F \end{aligned}$$

By rule (cut/cut¹/-) [a]

$$F \rightsquigarrow (\nu u)((!u(y).P_1^y) | (\nu x)(P_2^x | Q^z))$$

$$\equiv (\nu x)((\nu u)((!u(y).P_1^y) | P_2^x) | Q^z) = R$$

By struct. cong. [b]

$$\text{cut } D_2 (x. E) \equiv F_2 \text{ with}$$

$$F_2 \rightsquigarrow R_2 \text{ and } R_2 \equiv (\nu x)(P_2^x | Q^z) \text{ and}$$

$$F_2 \Rightarrow F'_2 \text{ with}$$

$$F'_2 \rightsquigarrow R_2' \equiv (\nu x)(P_2'^x | Q'^z)$$

By i.h. on D_2, E

$$F \Rightarrow \text{cut}^1 D_1 (u. F_2) = F'$$

By congruence [c]

$$F' \rightsquigarrow (\nu u)((!u(y).P_1^y) | (\nu x)(P_2'^x | Q'^z))$$

$$\equiv (\nu x)((\nu u)((!u(y).P_1^y) | P_2'^x) | Q'^z) = (\nu x)(P'^x | Q'^z)$$

[d]

Case: $E = 1L y E_1$ and D arbitrary.

$$\begin{aligned} \text{cut } D (x. E) &= \text{cut } D (x. 1L y E_1) \\ &\equiv 1L y (\text{cut } D (x. E_1)) = F \end{aligned}$$

By rule (cut/-/1L) [a]

$$F \rightsquigarrow (\nu x)(P^x | Q^z) = R$$

[b]

$$\text{cut } D (x. E_1) \equiv F_1 \text{ for some } F_1;$$

$$F_1 \rightsquigarrow R_1^z \equiv (\nu x)(P^x | Q^z);$$

$$F_1 \Rightarrow F'_1;$$

$$\begin{aligned}
F'_1 \rightsquigarrow R'_1 &\equiv (\nu x)(P'^x \mid Q'^z) && \text{By i.h. on } D, E_1 \\
F = \mathbf{!L} \ y \ F_1 &\Rightarrow \mathbf{!L} \ y \ F'_1 = F' && \text{By congruence [c]} \\
F' \rightsquigarrow R'_1 &\equiv (\nu x)(P'^x \mid Q'^z) && \text{[d]}
\end{aligned}$$

Case: $E = \mathbf{!L} \ y \ (u. E_1)$ and D arbitrary.

$$\begin{aligned}
\text{cut } D \ (x. E) &= \text{cut } D \ (x. \mathbf{!L} \ y \ (u. E_1)) \\
&\equiv \mathbf{!L} \ y \ (u. \text{cut } D \ (x. E_1)) = F && \text{By rule (cut/-/!L) [a]} \\
F &\rightsquigarrow (\nu x)(P^x \mid Q^z) = R && \text{since } E_1 \rightsquigarrow Q^z \{u/y\} \text{ [b]} \\
\text{cut } D \ (x. E_1) &\equiv F_1 \text{ for some } F_1; \\
F_1 \rightsquigarrow R'_1 &\equiv (\nu x)(P^x \mid Q^z); \\
F_1 &\Rightarrow F'_1; \\
F'_1 \rightsquigarrow R'_1 &\equiv (\nu x)(P'^x \mid Q'^z) && \text{By i.h. on } D, E_1 \\
F = \mathbf{!L} \ y \ (u. F_1) &\Rightarrow \mathbf{!L} \ y \ (u. F'_1) = F' && \text{By congruence [c]} \\
F' \rightsquigarrow R'_1 &\equiv (\nu x)(P'^x \mid Q'^z) && \text{[d]}
\end{aligned}$$

Case: $E = \text{cut } E_1 \ (y. E_2)$ with $x \in FV(E_1)$ and D arbitrary.

$$\begin{aligned}
\text{cut } D \ (x. E) &= \text{cut } D \ (x. \text{cut } E_1 \ (y. E_2)) \\
&\equiv \text{cut} \ (\text{cut } D \ (x. E_1)) \ (y. E_2) = F && \text{By reduction (cut/-/cut}_1\text{) [a]} \\
F &\rightsquigarrow (\nu y)((\nu x)(P^x \mid Q_1^y) \mid Q_2^z) \equiv (\nu x)(P^x \mid (\nu y)(Q_1^y \mid Q_2^z)) = R && \text{[b]} \\
\text{cut } D \ (x. E_1) &\equiv F_1 \text{ with} \\
F_1 \rightsquigarrow R_1 &\equiv (\nu x)(P^x \mid Q_1^y) \text{ and} \\
F_1 &\Rightarrow F'_1 \text{ with} \\
F'_1 \rightsquigarrow R'_1 &\equiv (\nu x)(P'^x \mid Q_1'^y) && \text{By i.h. on } D, E_1 \\
F &\Rightarrow \text{cut } F'_1 \ (y. E_2) = F' && \text{By congruence [c]} \\
F' \rightsquigarrow (\nu y)((\nu x)(P'^x \mid Q_1'^y) \mid Q_2^z) &\equiv (\nu x)(P'^x \mid (\nu y)(Q_1'^y \mid Q_2^z)) \\
&= (\nu x)(P'^x \mid Q'^z) && \text{[d]}
\end{aligned}$$

Case: $E = \text{cut } E_1 \ (y. E_2)$ with $x \in FV(E_2)$ and D arbitrary.

$$\begin{aligned}
\text{cut } D \ (x. E) &= \text{cut } D \ (x. \text{cut } E_1 \ (y. E_2)) \\
&\equiv \text{cut } E_1 \ (y. \text{cut } D \ (x. E_2)) = F && \text{By reduction (cut/-/cut}_2\text{) [a]} \\
F &\rightsquigarrow (\nu y)(Q_1^y \mid (\nu x)(P^x \mid Q_2^z)) \equiv (\nu x)(P^x \mid (\nu y)(Q_1^y \mid Q_2^z)) = R && \text{[b]} \\
\text{cut } D \ (x. E_2) &\equiv F_2 \text{ with} \\
F_2 \rightsquigarrow R_2 &\equiv (\nu x)(P^x \mid Q_2^z) \text{ and} \\
F_2 &\Rightarrow F'_2 \text{ with} \\
F'_2 \rightsquigarrow R'_2 &\equiv (\nu x)(P'^x \mid Q_2'^z) && \text{By i.h. on } D, E_2 \\
F &\Rightarrow \text{cut } E_1 \ (y. F'_2) = F' && \text{By congruence [c]} \\
F' \rightsquigarrow (\nu y)(Q_1^y \mid (\nu x)(P'^x \mid Q_2'^z)) &\equiv (\nu x)(P'^x \mid (\nu y)(Q_1^y \mid Q_2'^z)) \\
&= (\nu x)(P'^x \mid Q'^z) && \text{[d]}
\end{aligned}$$

Case: $E = \text{cut}^1 \ E_1 \ (u. E_2)$ with $x \in FV(E_1)$ and D arbitrary. This case is impossible because E_1 contains no free linear variables.

Case: $E = \text{cut}^1 \ E_1 \ (u. E_2)$ with $x \in FV(E_2)$ and D arbitrary.

$$\begin{aligned} \text{cut } D(x.E) &= \text{cut } D(x.\text{cut}^! E_1(u.E_2)) \\ &\equiv \text{cut}^! E_1(u.\text{cut } D(x.E_2)) = F \end{aligned}$$

By reduction (cut/–/cut[!]) [a]

$$\begin{aligned} F &\rightsquigarrow (\nu u)((!u(y).Q_1^y) \mid (\nu x)(P^x \mid Q_2^z)) \equiv \\ &(\nu x)(P^x \mid (\nu u)((!u(y).Q_1^y) \mid Q_2^z)) = R \end{aligned}$$

[b]

$$\begin{aligned} \text{cut } D(x.E_2) &\equiv F_2 \text{ with} \\ F_2 &\rightsquigarrow R_2 \equiv (\nu x)(P^x \mid Q_2^z) \text{ and} \\ F_2 &\Rightarrow F'_2 \text{ with} \\ F'_2 &\rightsquigarrow R'_2 \equiv (\nu x)(P'^x \mid Q_2'^z) \end{aligned}$$

By i.h. on D, E_2

$$F \Rightarrow \text{cut}^! E_1(u.F'_2) = F'$$

By congruence [c]

$$F' \rightsquigarrow (\nu u)((!u(y).Q_1^y) \mid (\nu x)(P'^x \mid Q_2'^z))$$

$$\equiv (\nu x)(P'^x \mid (\nu u)((!u(y).Q_1^y) \mid Q_2'^z)) = (\nu x)(P'^x \mid Q'^z)$$

[d]

□

Lemma A.5. *Assume*

$$(a) \Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x:C_1 \oplus C_2 \text{ with } P \xrightarrow{x.\text{inr}} P';$$

$$(b) \Gamma; \Delta_2, x:C_1 \oplus C_2 \vdash E \rightsquigarrow Q :: z:C \text{ with } Q \xrightarrow{x.\text{inr}} Q'.$$

Then

$$(c) \text{cut } D(x.E) \equiv \Rightarrow \equiv F \text{ for some } F;$$

$$(d) \Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z:C \text{ for some } R \equiv (\nu x)(P' \mid Q').$$

Proof. Symmetric to the previous lemma.

□

Lemma A.6. *Assume*

$$(a) \Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x:C_1 \otimes C_2 \text{ with } P \xrightarrow{(\nu y)x\langle y \rangle} P';$$

$$(b) \Gamma; \Delta_2, x:C_1 \otimes C_2 \vdash E \rightsquigarrow Q :: z:C \text{ with } Q \xrightarrow{x\langle y \rangle} Q'.$$

Then

$$(c) \text{cut } D(x.E) \equiv \Rightarrow \equiv F \text{ for some } F;$$

$$(d) \Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z:C \text{ for some } R \equiv (\nu x)(P' \mid Q').$$

Proof. By simultaneous induction on D and E . The possible cases for D are $\otimes R$, $1L$, $!L$, cut and $\text{cut}^!$. The possible cases for E are $\otimes L$, $1L$, $!L$, cut , and $\text{cut}^!$.

Case: $D = \otimes R D_1 D_2$ and $E = \otimes L x(y.x.E')$.

$$D_1 \rightsquigarrow P_1^y \text{ and } D_2 \rightsquigarrow P_2^x \text{ where}$$

$$P^x = (\nu y)(x\langle y \rangle.(P_1^y \mid P_2^x)) \xrightarrow{(\nu y)x\langle y \rangle} (\nu y)(P_1^y \mid P_2^x) = P'^x$$

By inversion

$$E' \rightsquigarrow Q'^z \text{ where } Q^z = x(y).Q'^z \xrightarrow{x\langle y \rangle} Q'^z$$

By inversion

$$\begin{aligned} \text{cut } D(x.E) &= \text{cut } (\otimes R D_1 D_2) (\otimes L x(y.x.E')) \\ &\Rightarrow \text{cut } D_1(y.\text{cut } D_2(x.E')) = F \end{aligned}$$

By rule (cut/ $\otimes R$ / $\otimes L$)

$$F \rightsquigarrow (\nu y)(P_1^y \mid (\nu x)(P_2^x \mid Q'^z)) \equiv (\nu x)((\nu y)(P_1^y \mid P_2^x) \mid Q'^z) = (\nu x)(P'^x \mid Q'^z)$$

Case: $D = \mathbf{1L} \ n \ D'$ and E arbitrary.

$$D' \rightsquigarrow P$$

By inversion

$$\text{cut } D \ (x. E) = \text{cut } (\mathbf{1L} \ n \ D') \ (x. E)$$

$$\equiv \mathbf{1L} \ n \ (\text{cut } D' \ (x. E))$$

By rule (cut/ $\mathbf{1L}/-$)

$$\Rightarrow \mathbf{1L} \ n \ F' = F \text{ for some } F' \rightsquigarrow (\nu x)(P' \mid Q')$$

By i.h. on D' and E

$$F \rightsquigarrow (\nu x)(P' \mid Q')$$

Case: $D = \mathbf{!L} \ n \ (u. D')$ and E arbitrary.

$$D' \rightsquigarrow P$$

By inversion

$$\text{cut } D \ (x. E) = \text{cut } (\mathbf{!L} \ n \ (u. D')) \ (x. E)$$

$$\equiv \mathbf{!L} \ n \ (u. \text{cut } D' \ (x. E))$$

By rule (cut/ $\mathbf{!L}/-$)

$$\Rightarrow \mathbf{!L} \ n \ (u. F') = F \text{ for some } F' \rightsquigarrow (\nu x)(P' \mid Q')$$

By i.h. on D' and E

$$F \rightsquigarrow (\nu x)(P' \mid Q')$$

Case: $D = \text{cut } D_1 \ (n. D_2)$ and E arbitrary.

$$D_1 \rightsquigarrow P_1^n \text{ and } D_2 \rightsquigarrow P_2^x \text{ and } P_2^x \xrightarrow{(\nu y)x(y)} P_2'^x \text{ where}$$

$$P = (\nu n)(P_1^n \mid P_2^x) \xrightarrow{(\nu y)x(y)} (\nu n)(P_1^n \mid P_2'^x) = P'^x$$

By inversion

$$\text{cut } D \ (x. E) = \text{cut } (\text{cut } D_1 \ (n. D_2)) \ (x. E)$$

$$\equiv \text{cut } D_1 \ (n. \text{cut } D_2 \ (x. E))$$

By rule (cut/cut/ $-$)

$$\Rightarrow \text{cut } D_1 \ (n. F_2) = F \text{ where } F_2 \rightsquigarrow (\nu x)(P_2'^x \mid Q'^z)$$

By i.h. on D_2 and E

$$F \rightsquigarrow (\nu n)(P_1^n \mid (\nu x)(P_2'^x \mid Q'^z))$$

$$\equiv (\nu x)((\nu n)(P_1^n \mid P_2^x) \mid Q'^z)$$

$$= (\nu x)(P'^x \mid Q'^z)$$

Case: $D = \text{cut}^! \ D_1 \ (u. D_2)$ and E arbitrary.

$$D_1 \rightsquigarrow (!u(n).P_1^n) \text{ and } D_2 \rightsquigarrow P_2^x \text{ and } P_2^x \xrightarrow{(\nu y)x(y)} P_2'^x \text{ where}$$

$$P = (\nu u)((!u(n).P_1^n) \mid P_2^x) \xrightarrow{(\nu y)x(y)} (\nu u)((!u(n).P_1^n) \mid P_2'^x) = P'^x$$

By inversion

$$\text{cut } D \ (x. E) = \text{cut } (\text{cut}^! \ D_1 \ (u. D_2)) \ (x. E)$$

$$\equiv \text{cut}^! \ D_1 \ (u. \text{cut } D_2 \ (x. E))$$

By rule (cut/ $\text{cut}^!/-$)

$$\Rightarrow \text{cut } D_1 \ (u. F_2) = F \text{ where } F_2 \rightsquigarrow (\nu x)(P_2'^x \mid Q'^z)$$

By i.h. on D_2 and E

$$F \rightsquigarrow (\nu u)((!u(n).P_1^n) \mid (\nu x)(P_2'^x \mid Q'^z))$$

$$\equiv (\nu x)((\nu u)((!u(n).P_1^n) \mid P_2^x) \mid Q'^z)$$

$$= (\nu x)(P'^x \mid Q'^z)$$

Case: $E = \mathbf{1L} \ n \ E'$ and D arbitrary.

$$E' \rightsquigarrow Q^z$$

By inversion

$$\text{cut } D \ (x. E) = \text{cut } D \ (x. \mathbf{1L} \ n \ E')$$

$$\equiv \mathbf{1L} \ n \ (\text{cut } D \ (x. E'))$$

By rule (cut/ $-/\mathbf{1L}$)

$$\Rightarrow \mathbf{1L} \ n \ F' = F \text{ for some } F' \rightsquigarrow (\nu x)(P'^x \mid Q'^z)$$

By i.h. on D and E'

$$F \rightsquigarrow (\nu x)(P'^x \mid Q'^z)$$

Case: $E = \mathbf{!L} \ n \ (u. E')$ and D arbitrary.

$$\begin{aligned}
E' &\rightsquigarrow Q^z && \text{By inversion} \\
\text{cut } D(x.E) &= \text{cut } D(x.!\text{L } n(u.E')) \\
&\equiv !\text{L } n(u.\text{cut } D(x.E')) && \text{By rule (cut/-/!L)} \\
&\Rightarrow !\text{L } n(u.F') = F \text{ for some } F' \rightsquigarrow (\nu x)(P'^x \mid Q'^z) \\
F &\rightsquigarrow (\nu x)(P'^x \mid Q'^z) && \text{By i.h. on } D \text{ and } E'
\end{aligned}$$

Case: $E = \text{cut } E_1(n.E_2)$ and $x \in FV(E_1)$.

$$\begin{aligned}
E_1 &\rightsquigarrow Q_1^n \text{ and } E_2 \rightsquigarrow Q_2^z \text{ for } Q_1^n \xrightarrow{x(y)} Q_1^m \text{ where} \\
Q &= (\nu n)(Q_1^n \mid Q_2^z) \xrightarrow{x(y)} (\nu n)(Q_1^m \mid Q_2^z) && \text{By inversion} \\
\text{cut } D(x.E) &= \text{cut } D(x.\text{cut } E_1(n.E_2)) \\
&= \text{cut } (\text{cut } D(x.E_1))(n.E_2) && \text{By rule (cut/cut/-) and } x \notin FV(E_2) \\
&\Rightarrow \text{cut } F_1(n.E_2) = F \text{ for some } F_1 \rightsquigarrow (\nu x)(P'^x \mid Q_1^m) && \text{By i.h. on } D \text{ and } E_1 \\
F &\rightsquigarrow (\nu n)((\nu x)(P'^x \mid Q_1^m) \mid Q_2^z) \\
&\equiv (\nu x)(P'^x \mid (\nu n)(Q_1^m \mid Q_2^z)) \\
&= (\nu x)(P'^x \mid Q'^z)
\end{aligned}$$

Case: $E = \text{cut } E_1(n.E_2)$ and $x \in FV(E_2)$.

$$\begin{aligned}
E_1 &\rightsquigarrow Q_1^n \text{ and } E_2 \rightsquigarrow Q_2^z \text{ for } Q_2^z \xrightarrow{x(y)} Q_2'^z \text{ where} \\
Q &= (\nu n)(Q_1^n \mid Q_2^z) \xrightarrow{x(y)} (\nu n)(Q_1^n \mid Q_2'^z) && \text{By inversion} \\
\text{cut } D(x.E) &= \text{cut } D(x.\text{cut } E_1(n.E_2)) \\
&= \text{cut } E_1(n.\text{cut } D(x.E_2)) && \text{By rule (cut/-/cut) and } x \notin FV(E_1) \\
&\Rightarrow \text{cut } E_1(n.F_2) = F \text{ for some } F_2 \rightsquigarrow (\nu x)(P'^x \mid Q_2'^z) && \text{By i.h. on } D \text{ and } E_2 \\
F &\rightsquigarrow (\nu n)(Q_1^n \mid (\nu x)(P'^x \mid Q_2'^z)) \\
&\equiv (\nu x)(P'^x \mid (\nu n)(Q_1^n \mid Q_2'^z)) \\
&= (\nu x)(P'^x \mid Q'^z)
\end{aligned}$$

Case: $E = \text{cut}^! E_1(u.E_2)$

$$\begin{aligned}
x &\in FV(E_2) && \text{By inversion} \\
E_1 &\rightsquigarrow (!u(n).Q_1^n) \text{ and } E_2 \rightsquigarrow Q_2^z \text{ for } Q_2^z \xrightarrow{x(y)} Q_2'^z \text{ where} \\
Q &= (\nu u)((!u(n).Q_1^n) \mid Q_2^z) \xrightarrow{x(y)} (\nu u)((!u(n).Q_1^n) \mid Q_2'^z) && \text{By inversion} \\
\text{cut } D(x.E) &= \text{cut } D(x.\text{cut}^! E_1(u.E_2)) \\
&= \text{cut}^! E_1(u.\text{cut } D(x.E_2)) && \text{By rule (cut/-/cut}^!) \\
&\Rightarrow \text{cut}^! E_1(u.F_2) = F \text{ for some } F_2 \rightsquigarrow (\nu x)(P'^x \mid Q_2'^z) && \text{By i.h. on } D \text{ and } E_2 \\
F &\rightsquigarrow (\nu u)((!u(n).Q_1^n) \mid (\nu x)(P'^x \mid Q_2'^z)) \\
&\equiv (\nu x)(P'^x \mid (\nu u)((!u(n).Q_1^n) \mid Q_2'^z)) \\
&= (\nu x)(P'^x \mid Q'^z)
\end{aligned}$$

□

Lemma A.7. *Assume*

- (a) $\Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x:C_1 \multimap C_2$ with $P \xrightarrow{x(y)} P'$;
- (b) $\Gamma; \Delta_2, x:C_1 \multimap C_2 \vdash E \rightsquigarrow Q :: z:C$ with $Q \xrightarrow{(\nu y)x(y)} Q'$.

Then

(c) $\text{cut } D(x. E) \equiv \equiv F$ for some F ;

(d) $\Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z : C$ for some $R \equiv (\nu x)(\nu y)(P' \mid Q')$.

Proof. By simultaneous induction on D and E . The possible cases for D are $\multimap R$, $1L$, $!L$, cut and $\text{cut}^!$. The possible cases for E are $\multimap L$, $1L$, $!L$, cut , and $\text{cut}^!$.

Case: $D = \multimap R (x. D_1)$ and $E = \multimap L x E_1 (x. E_2)$.

$$\begin{aligned}
& E_1 \rightsquigarrow Q_1^y \text{ and } E_2 \rightsquigarrow Q_2^z \text{ where} \\
& Q^z = (\nu y)(x \langle y \rangle. (Q_1^y \mid Q_2^z)) \xrightarrow{(\nu y)x \langle y \rangle} (\nu y)(Q_1^y \mid Q_2^z) = Q'^x && \text{By inversion} \\
& D_1 \rightsquigarrow P'^x \text{ where } P^x = x(y). P'^x \xrightarrow{x \langle y \rangle} P'^x && \text{By inversion} \\
& \text{cut } D (x. E) = \text{cut } (\multimap R (x. D_1)) (\multimap L x E_1 (x. E_2)) \\
& \Rightarrow \text{cut } E_1 (y. \text{cut } D_1 (x. E_2)) = F && \text{By rule (cut}/\multimap R/\multimap L) \\
& F \rightsquigarrow (\nu y)(Q_1^y \mid (\nu x)(P_1^x \mid Q_2^z)) \equiv (\nu x)((\nu y)P_1^x \mid (Q_1^y \mid Q_2^z)) = (\nu x)(P'^x \mid Q'^z)
\end{aligned}$$

Case: $D = 1L n D'$ and E arbitrary.

$$\begin{aligned}
& D' \rightsquigarrow P && \text{By inversion} \\
& \text{cut } D (x. E) = \text{cut } (1L n D') (x. E) \\
& \equiv 1L n (\text{cut } D' (x. E)) && \text{By rule (cut}/1L/-) \\
& \Rightarrow 1L n F' = F \text{ for some } F' \rightsquigarrow (\nu x)(P' \mid Q') && \text{By i.h. on } D' \text{ and } E \\
& F \rightsquigarrow (\nu x)(P' \mid Q')
\end{aligned}$$

Case: $D = !L n (u. D')$ and E arbitrary.

$$\begin{aligned}
& D' \rightsquigarrow P && \text{By inversion} \\
& \text{cut } D (x. E) = \text{cut } (!L n (u. D')) (x. E) \\
& \equiv !L n (u. \text{cut } D' (x. E)) && \text{By rule (cut}/!L/-) \\
& \Rightarrow !L n (u. F') = F \text{ for some } F' \rightsquigarrow (\nu x)(P' \mid Q') && \text{By i.h. on } D' \text{ and } E \\
& F \rightsquigarrow (\nu x)(P' \mid Q')
\end{aligned}$$

Case: $D = \text{cut } D_1 (n. D_2)$ and E arbitrary.

$$\begin{aligned}
& D_1 \rightsquigarrow P_1^n \text{ and } D_2 \rightsquigarrow P_2^x \text{ and } P_2^x \xrightarrow{(\nu y)x \langle y \rangle} P_2'^x \text{ where} \\
& P = (\nu n)(P_1^n \mid P_2^x) \xrightarrow{(\nu y)x \langle y \rangle} (\nu n)(P_1^n \mid P_2'^x) = P'^x && \text{By inversion} \\
& \text{cut } D (x. E) = \text{cut } (\text{cut } D_1 (n. D_2)) (x. E) \\
& \equiv \text{cut } D_1 (n. \text{cut } D_2 (x. E)) && \text{By rule (cut}/\text{cut}/-) \\
& \Rightarrow \text{cut } D_1 (n. F_2) = F \text{ where } F_2 \rightsquigarrow (\nu x)(P_2'^x \mid Q'^z) && \text{By i.h. on } D_2 \text{ and } E \\
& F \rightsquigarrow (\nu n)(P_1^n \mid (\nu x)(P_2'^x \mid Q'^z)) \\
& \equiv (\nu x)((\nu n)(P_1^n \mid P_2^x) \mid Q'^z) \\
& = (\nu x)(P'^x \mid Q'^z)
\end{aligned}$$

Case: $D = \text{cut}^! D_1 (u. D_2)$ and E arbitrary.

$$\begin{aligned}
& D_1 \rightsquigarrow (!u(n). P_1^n) \text{ and } D_2 \rightsquigarrow P_2^x \text{ and } P_2^x \xrightarrow{(\nu y)x \langle y \rangle} P_2'^x \text{ where} \\
& P = (\nu u)((!u(n). P_1^n) \mid P_2^x) \xrightarrow{(\nu y)x \langle y \rangle} (\nu u)((!u(n). P_1^n) \mid P_2'^x) = P'^x && \text{By inversion}
\end{aligned}$$

$$\begin{aligned}
& \text{cut } D(x.E) = \text{cut}(\text{cut}^! D_1(u.D_2))(x.E) \\
& \equiv \text{cut}^! D_1(u.\text{cut } D_2(x.E)) \\
& \Rightarrow \text{cut } D_1(u.F_2) = F \text{ where } F_2 \rightsquigarrow (\nu x)(P_2^{'x} \mid Q_2^{'z}) \\
& F \rightsquigarrow (\nu u)((!u(n).P_1^n) \mid (\nu x)(P_2^{'x} \mid Q_2^{'z})) \\
& \equiv (\nu x)((\nu u)((!u(n).P_1^n) \mid P_2^{'x}) \mid Q_2^{'z}) \\
& = (\nu x)(P_1^{'x} \mid Q_2^{'z})
\end{aligned}$$

By rule (cut/cut[!]/-)
By i.h. on D_2 and E

Case: $E = \mathbf{1L} \ n \ E'$ and D arbitrary.

$$\begin{aligned}
& E' \rightsquigarrow Q^z \\
& \text{cut } D(x.E) = \text{cut } D(x.\mathbf{1L} \ n \ E') \\
& \equiv \mathbf{1L} \ n \ (\text{cut } D(x.E')) \\
& \Rightarrow \mathbf{1L} \ n \ F' = F \text{ for some } F' \rightsquigarrow (\nu x)(P^{'x} \mid Q^{'z}) \\
& F \rightsquigarrow (\nu x)(P^{'x} \mid Q^{'z})
\end{aligned}$$

By inversion

By rule (cut/-/1L)
By i.h. on D and E'

Case: $E = \mathbf{!L} \ n \ (u.E')$ and D arbitrary.

$$\begin{aligned}
& E' \rightsquigarrow Q^z \\
& \text{cut } D(x.E) = \text{cut } D(x.\mathbf{!L} \ n \ (u.E')) \\
& \equiv \mathbf{!L} \ n \ (u.\text{cut } D(x.E')) \\
& \Rightarrow \mathbf{!L} \ n \ (u.F') = F \text{ for some } F' \rightsquigarrow (\nu x)(P^{'x} \mid Q^{'z}) \\
& F \rightsquigarrow (\nu x)(P^{'x} \mid Q^{'z})
\end{aligned}$$

By inversion

By rule (cut/-/!L)
By i.h. on D and E'

Case: $E = \text{cut } E_1(n.E_2)$ and $x \in FV(E_1)$.

$$\begin{aligned}
& E_1 \rightsquigarrow Q_1^n \text{ and } E_2 \rightsquigarrow Q_2^z \text{ for } Q_1^n \xrightarrow{x(y)} Q_1^m \text{ where} \\
& Q = (\nu n)(Q_1^n \mid Q_2^z) \xrightarrow{x(y)} (\nu n)(Q_1^m \mid Q_2^z) \\
& \text{cut } D(x.E) = \text{cut } D(x.\text{cut } E_1(n.E_2)) \\
& = \text{cut}(\text{cut } D(x.E_1))(n.E_2) \\
& \Rightarrow \text{cut } F_1(n.E_2) = F \text{ for some } F_1 \rightsquigarrow (\nu x)(P_1^{'x} \mid Q_1^m) \\
& F \rightsquigarrow (\nu n)((\nu x)(P_1^{'x} \mid Q_1^m) \mid Q_2^z) \\
& \equiv (\nu x)(P_1^{'x} \mid (\nu n)(Q_1^m \mid Q_2^z)) \\
& = (\nu x)(P_1^{'x} \mid Q_2^z)
\end{aligned}$$

By inversion

By rule (cut/cut/-) and $x \notin FV(E_2)$
By i.h. on D and E_1

Case: $E = \text{cut } E_1(n.E_2)$ and $x \in FV(E_2)$.

$$\begin{aligned}
& E_1 \rightsquigarrow Q_1^n \text{ and } E_2 \rightsquigarrow Q_2^z \text{ for } Q_2^z \xrightarrow{x(y)} Q_2^{'z} \text{ where} \\
& Q = (\nu n)(Q_1^n \mid Q_2^z) \xrightarrow{x(y)} (\nu n)(Q_1^n \mid Q_2^{'z}) \\
& \text{cut } D(x.E) = \text{cut } D(x.\text{cut } E_1(n.E_2)) \\
& = \text{cut } E_1(n.\text{cut } D(x.E_2)) \\
& \Rightarrow \text{cut } E_1(n.F_2) = F \text{ for some } F_2 \rightsquigarrow (\nu x)(P^{'x} \mid Q_2^{'z}) \\
& F \rightsquigarrow (\nu n)(Q_1^n \mid (\nu x)(P^{'x} \mid Q_2^{'z})) \\
& \equiv (\nu x)(P^{'x} \mid (\nu n)(Q_1^n \mid Q_2^{'z})) \\
& = (\nu x)(P^{'x} \mid Q_2^{'z})
\end{aligned}$$

By inversion

By rule (cut/-/cut) and $x \notin FV(E_1)$
By i.h. on D and E_2

Case: $E = \text{cut}^! E_1(u.E_2)$

$$\begin{aligned}
& x \in FV(E_2) \\
& E_1 \rightsquigarrow (!u(n).Q_1^n) \text{ and } E_2 \rightsquigarrow Q_2^z \text{ for } Q_2^z \xrightarrow{x(y)} Q_2^{'z} \text{ where}
\end{aligned}$$

By inversion

$$\begin{aligned}
Q &= (\nu u)((!u(n).Q_1^n \mid Q_2^z) \xrightarrow{x(y)} (\nu u)((!u(n).Q_1^n \mid Q_2'^z)) && \text{By inversion} \\
\text{cut } D(x.E) &= \text{cut } D(x.\text{cut}^! E_1(u.E_2)) \\
&= \text{cut}^! E_1(u.\text{cut } D(x.E_2)) && \text{By rule (cut/-/cut}^!) \\
&\Rightarrow \text{cut}^! E_1(u.F_2) = F \text{ for some } F_2 \rightsquigarrow (\nu x)(P'^x \mid Q_2'^z) && \text{By i.h. on } D \text{ and } E_2 \\
F &\rightsquigarrow (\nu u)((!u(n).Q_1^n \mid (\nu x)(P'^x \mid Q_2'^z)) \\
&\equiv (\nu x)(P'^x \mid (\nu u)((!u(n).Q_1^n \mid Q_2'^z)) \\
&= (\nu x)(P'^x \mid Q'^z)
\end{aligned}$$

□

Lemma A.8. *Assume*

$$(a) \Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x:A \text{ with } P \xrightarrow{x(y)} P';$$

$$(b) \Gamma; \Delta_2, x:A \vdash E \rightsquigarrow Q :: z:C \text{ with } Q \xrightarrow{(\nu y)x(y)} Q'.$$

Then

$$(c) \text{cut } D(x.E) \equiv \Rightarrow \equiv F \text{ for some } F;$$

$$(d) \Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z : C \text{ for some } R \equiv (\nu x)(\nu y)(P' \mid Q').$$

Proof. By simultaneous induction on the structure of D, E . There are only five possible cases for D : $!R D'_1$, $!L n D'_1$, $!L n(u.D_1)$, $\text{cut } D'_1(n.D'_1)$, and $\text{cut}^! D'_1(u.D'_1)$. In all other cases P cannot offer $x(y)$, which follows by analysis of the typed extraction rules and the definition of $\xrightarrow{\alpha}$. There are only four possible cases for E : $!L n(u.E'_2)$, $!L n E'_2$, $\text{cut } E'_2(n.E''_2)$, and $\text{cut}^! E'_2(u.E''_2)$. In all other cases Q cannot offer $(\nu y)x(y)$, which follows by analysis of the typed extraction rules and the definition of $\xrightarrow{\alpha}$.

Case: $D = !R D'_1, E = !L x(u.E'_2)$.

$$\begin{aligned}
\Delta_1 &= (\cdot) \\
\Gamma; \cdot \vdash D'_1 \rightsquigarrow R_1 :: u:A \text{ where } P &= !x(u).R_1 \\
\Gamma, u : A; \Delta_2 \vdash E' \rightsquigarrow Q_1 :: z:C \text{ where } Q &= (\nu y)x(y).Q'_1 && \text{by inversion} \\
\text{cut } D(x.E) &= && \text{this case} \\
\text{cut } (!R D'_1)(x.!L x(u.E'_2)) &= && \\
\equiv \text{cut}^! D'_1(u.E'_2) &= && \text{by (cut!/R!/L)} \\
\text{cut}^! D'_1(u.E'_2) \equiv \Rightarrow \equiv \text{cut}^! D'_1(u.E^*) \text{ for some } E^* &= && \\
\Gamma, u : A; \Delta_2 \vdash E^* \rightsquigarrow R' :: z:C \text{ with } R' \equiv (\nu y)(R_1\{y/u\} \mid Q'_1) &= && \text{by Lemma !Red} \\
\text{Pick } F = \text{cut}^! D'_1(u.E^*) &= && \text{[satisfying (c)]} \\
\Gamma; \Delta \vdash F \rightsquigarrow R :: z:C &= && \text{by cut} \\
\text{with } R = (\nu x)(!x(u).R_1 \mid (\nu y)(R_1\{y/u\} \mid Q'_1)) &= && \\
\equiv (\nu y)(\nu x)(!x(u).P_1 \mid R_1\{y/u\} \mid Q'_1) &= && \\
\equiv (\nu y)(\nu x)(P'_1 \mid Q'_1) \text{ since } P'_1 \equiv !x(u).R_1 \mid R_1\{y/u\} &= && \text{[satisfying (d)]}
\end{aligned}$$

Case: D arbitrary, $E = !L y(u.E'_2)$.

$$\begin{aligned}
\Delta_2 &= (\Delta^*, y : !B) \\
\Gamma, u : B; \Delta^*, x : !A \vdash E'_2 \rightsquigarrow Q_1 :: z:C &= && \text{by inversion} \\
\text{cut } D(x.E'_2) \equiv \Rightarrow \equiv E^* \text{ for some } E^* &= && \\
\Gamma, u : B; \Delta^* \vdash E^* \rightsquigarrow R' :: z:C \text{ with } R' \equiv (\nu y)(\nu x)(P'_1 \mid Q'_1) &= && \text{by i.h.} \\
\text{cut } D(x.E) = \text{cut } D(x.!L y(u.E'_2)) &= && \\
\equiv !L y(u.\text{cut } D(x.E'_2)) &= && \text{by (cut/ - /!L)}
\end{aligned}$$

$\equiv \Rightarrow \equiv !L y (u. E^*)$	by congruence
$\text{Pick } F = !L y (u. E^*)$	[satisfying (c)]
$\Gamma; \Delta \vdash F \rightsquigarrow R :: z:C$	by !L
with $R = R'$	[satisfying (d)]

Case: $D = !L y (u. D'_1)$, E arbitrary.

$\Delta_1 = (\Delta^*, y : !B)$	
$\Gamma, u : B; \Delta^* \vdash D'_1 \rightsquigarrow P_1 :: x:A$	by inversion
$\text{cut } D'_1 (x. E) \equiv \Rightarrow \equiv D^*$ for some D^*	
$\Gamma, u : B; \Delta^* \vdash D^* \rightsquigarrow R' :: z:C$ with $R' \equiv (\nu y)(\nu x)(P'_1 \mid Q'_1)$	by i.h.
$\text{cut } D (x. E) = \text{cut } (!L y (u. D'_1)) (x. E)$	
$\equiv !L y (u. \text{cut } D'_1 (x. E))$	by (cut/!L/-)
$\equiv \Rightarrow \equiv !L y (u. D^*)$	by congruence
$\text{Pick } F = !L y (u. D^*)$	[satisfying (c)]
$\Gamma; \Delta \vdash F \rightsquigarrow R :: z:C$	by !L
with $R = R'$	[satisfying (d)]

Case: D arbitrary, $E = 1L n E'_2$.

$\Delta_2 = (\Delta^*, n : 1)$	
$\Gamma, x : A; \Delta^* \vdash E'_2 \rightsquigarrow Q_1 :: z:C$	by inversion
$\text{cut } D (x. E'_2) \equiv \Rightarrow \equiv E^*$ for some E^*	
$\Gamma; \Delta^* \vdash E^* \rightsquigarrow R' :: z:C$ with $R' \equiv (\nu y)(\nu x)(P'_1 \mid Q'_1)$	by i.h.
$\text{cut } D (x. E) = \text{cut } D (x. 1L n E'_2)$	
$\equiv 1L n (\text{cut } D (x. E'_2))$	by (cut/-/1L)
$\equiv \Rightarrow \equiv 1L n E^*$	by congruence
$\text{Pick } F = 1L n E^*$	[satisfying (c)]
$\Gamma; \Delta \vdash F \rightsquigarrow R :: z:C$	by 1L
with $R = R'$	[satisfying (d)]

Case: $D = 1L n D'_1$, E arbitrary.

$\Delta_1 = (\Delta^*, n : 1)$	
$\Gamma; \Delta^* \vdash D'_1 \rightsquigarrow P_1 :: x:A$	by inversion
$\text{cut } D'_1 (x. E) \equiv \Rightarrow \equiv D^*$ for some D^*	
$\Gamma; \Delta^* \vdash D^* \rightsquigarrow R' :: z:C$ with $R' \equiv (\nu y)(\nu x)(P'_1 \mid Q'_1)$	by i.h.
$\text{cut } D (x. E) = \text{cut } (1L n. D'_1) (x. E)$	
$\equiv 1L n. (\text{cut } D'_1 (x. E))$	by (cut/1L/-)
$\equiv \Rightarrow \equiv 1L n D^*$	by congruence
$\text{Pick } F = 1L n D^*$	[satisfying (c)]
$\Gamma; \Delta \vdash F \rightsquigarrow R :: z:C$	by 1L
with $R = R'$	[satisfying (d)]

Case: D arbitrary, $E = \text{cut } E'_2 (n. E''_2)$.

$\Delta_2 = (\Delta'_2, \Delta''_2)$	
$Q_1 = (\nu n)(R_1 \mid R_2)$	
$\Gamma, x : A; \Delta'_2 \vdash E'_2 \rightsquigarrow R_1 :: n:B$	
$\Gamma, x : A; \Delta''_2, n : B \vdash E''_2 \rightsquigarrow R_2 :: z:C$	by inversion
Subcase: $R_1 \xrightarrow{(\nu y)x(y)} R'_1$ and $Q'_1 = (\nu n)(R'_1 \mid R_2)$	

$\text{cut } D (x. E'_2) \equiv \equiv D^*$ for some D^*
 $\Gamma; \Delta'_2 \vdash D^* \rightsquigarrow R' :: n:B$ with $R' \equiv (\nu y)(\nu x)(P'_1 \mid R'_1)$ by i.h.
 $\text{cut } D (x. E) = \text{cut } D (x. \text{cut } E'_2 (n. E''_2))$
 $\equiv \text{cut } (\text{cut } D (x. E'_2)) (n. E''_2)$ by (cut/ – /cut₁)
 $\equiv \equiv \text{cut } D^* (n. E''_2)$ by congruence
Pick $F = \text{cut } D^* (n. E''_2)$ [satisfying (c)]
 $\Gamma; \Delta \vdash F \rightsquigarrow R :: z:C$ by cut
with $R = (\nu n)(R' \mid R_2)$
 $\equiv (\nu n)((\nu y)(\nu x)(P'_1 \mid R'_1) \mid R_2)$
 $\equiv (\nu n)(\nu y)(\nu x)(P'_1 \mid R'_1 \mid R_2)$
 $\equiv (\nu y)(\nu x)(P'_1 \mid (\nu n)(R'_1 \mid R_2))$
 $\equiv (\nu y)(\nu x)(P'_1 \mid Q'_1)$ [satisfying (d)]

Subcase: $R_2 \xrightarrow{(\nu y)x(y)} R'_2$ and $Q'_1 = (\nu n)(R_1 \mid R'_2)$
 $\text{cut } D (x. E''_2) \equiv \equiv D^*$ for some D^*
 $\Gamma; \Delta'_2, n : B \vdash D^* \rightsquigarrow R' :: z:C$ with $R' \equiv (\nu y)(\nu x)(P'_1 \mid R'_2)$ by i.h.
 $\text{cut } D_1 (x. D_2) = \text{cut } D (x. \text{cut } E'_2 (n. E''_2))$
 $\equiv \text{cut } E'_2 (n. \text{cut } D (x. E''_2))$ by (cut/ – /cut₂)
 $\equiv \equiv \text{cut } E'_2 (n. D^*)$ by congruence
Pick $F = \text{cut } E'_2 (n. D^*)$ [satisfying (c)]
 $\Gamma; \Delta \vdash F \rightsquigarrow R :: z:C$ by cut
with $R = (\nu n)(R_1 \mid R')$
 $\equiv (\nu n)(R_1 \mid (\nu y)(\nu x)(P'_1 \mid R'_2))$
 $\equiv (\nu y)(\nu x)(\nu n)(R_1 \mid P'_1 \mid R'_2)$
 $\equiv (\nu y)(\nu x)(\nu n)(P'_1 \mid R_1 \mid R'_2)$
 $\equiv (\nu y)(\nu x)(P'_1 \mid (\nu n)(R_1 \mid R'_2))$
 $\equiv (\nu y)(\nu x)(P'_1 \mid Q'_1)$ [satisfying (d)]

Case: $D = \text{cut } D'_1 (n. D''_1)$, E arbitrary.

$\Delta_1 = (\Delta'_1, \Delta''_1)$
 $P_1 = (\nu n)(R_1 \mid R_2)$
 $\Gamma; \Delta'_1 \vdash D'_1 \rightsquigarrow R_1 :: n:B$
 $\Gamma; \Delta''_1, n:B \vdash D''_1 \rightsquigarrow R_2 :: x:A$ by inversion
 $R_2 \xrightarrow{x(y)} R'_2$ and $P'_1 = (\nu n)(R_1 \mid R'_2)$
 $\text{cut } D''_1 (x. E) \equiv \equiv D^*$ for some D^*
 $\Gamma; \Delta'_1, n:B, \Delta_2 \vdash D^* \rightsquigarrow R' :: z:C$ with $R' \equiv (\nu y)(\nu x)(R'_2 \mid Q'_1)$ by i.h.
 $\text{cut } D (x. E) = \text{cut } (\text{cut } D'_1 (n. D''_1)) (x. E)$
 $\equiv \text{cut } D'_1 (n. \text{cut } D''_1 (x. E))$ by (cut/ – /cut₁)
 $\equiv \equiv \text{cut } D'_1 (n. D^*)$ by congruence
Pick $F = \text{cut } D'_1 (n. D^*)$ [satisfying (c)]
 $\Gamma; \Delta \vdash F \rightsquigarrow R :: z:C$ by cut
with $R = (\nu n)(R_1 \mid R')$
 $\equiv (\nu n)(R_1 \mid (\nu y)(\nu x)(R'_2 \mid Q'_1))$
 $\equiv (\nu y)(\nu x)(\nu n)(R_1 \mid R'_2 \mid Q'_1)$
 $\equiv (\nu y)(\nu x)(\nu n)(R_1 \mid R'_2) \mid Q_1)$
 $\equiv (\nu y)(\nu x)(P'_1 \mid Q'_1)$ [satisfying (d)]

Case: D arbitrary, $E = \text{cut}^! E'_2 (u. E''_2)$.

$Q_1 = (\nu u)(!u(w).R_1 \mid R_2)$

$$\begin{array}{l}
\Gamma; \cdot \vdash E'_2 \rightsquigarrow R_1 :: w:B \\
\Gamma, u : B; \Delta_2, x : !A \vdash E''_2 \rightsquigarrow R_2 :: z:C \quad \text{by inversion} \\
R_2 \xrightarrow{(\nu y)x\langle y \rangle} R'_2 \text{ and } Q'_1 = (\nu u)(!u(w).R_1 \mid R'_2) \\
\text{cut } D(x.E''_2) \equiv \equiv D^* \text{ for some } D^* \\
\Gamma, u : B; \Delta_2 \vdash D^* \rightsquigarrow R' :: z:C \text{ with } R' \equiv (\nu y)(\nu x)(P'_1 \mid R'_2) \quad \text{by i.h.} \\
\text{cut } D(x.E) = \text{cut } D(x.\text{cut}^! E'_2(u.E''_2)) \\
\equiv \text{cut}^! E'_2(u.\text{cut } D(x.E''_2)) \quad \text{by (cut/ - /cut}^!) \\
\equiv \equiv \text{cut}^! E'_2(u.D^*) \quad \text{by congruence} \\
\text{Pick } F = \text{cut}^! E'_2(u.D^*) \quad \text{[satisfying (c)]} \\
\Gamma; \Delta \vdash F \rightsquigarrow R :: z:C \quad \text{by cut}^! \\
\text{with } R = (\nu u)(!u(w).R_1 \mid R') \\
\equiv (\nu u)(!u(w).R_1 \mid (\nu y)(\nu x)(P'_1 \mid R'_2)) \\
\equiv (\nu y)(\nu x)(\nu u)(!u(w).R_1 \mid P'_1 \mid R'_2) \\
\equiv (\nu y)(\nu x)(\nu u)(P'_1 \mid !u(w).R_1 \mid R'_2) \\
\equiv (\nu y)(\nu x)(P'_1 \mid (\nu u)(!u(w).R_1 \mid R'_2)) \\
\equiv (\nu y)(\nu x)(P'_1 \mid Q'_1) \quad \text{[satisfying (d)]}
\end{array}$$

Case: $D_1 = \text{cut}^! D'_1(u.D''_1)$, E arbitrary.

$$\begin{array}{l}
P_1 = (\nu u)(!u(w).R_1 \mid R_2) \\
\Gamma \vdash D'_1 \rightsquigarrow R_1 :: w:B \\
\Gamma, u : B; \Delta_1 \vdash D''_1 \rightsquigarrow R_2 :: x:!A \quad \text{by inversion} \\
R_2 \xrightarrow{x\langle y \rangle} R'_2 \text{ and } P'_1 = (\nu u)(!u(w).R_1 \mid R'_2) \\
\text{cut } D''_1(x.E) \equiv \equiv D^* \text{ for some } D^* \\
\Gamma, u : B; \Delta \vdash D^* \rightsquigarrow R' :: z:C \text{ with } R' \equiv (\nu y)(\nu x)(R'_2 \mid Q'_1) \quad \text{by i.h.} \\
\text{cut } D_1(x.E) = \text{cut}(\text{cut}^! D'_1(u.D''_1))(x.E) \\
\equiv \text{cut}^! D'_1(u.\text{cut } D''_1(x.E)) \quad \text{by (cut/cut}^!/-) \\
\equiv \equiv \text{cut}^! D'_1(u.D^*) \quad \text{by congruence} \\
\text{Pick } F = \text{cut}^! D'_1(u.D^*) \quad \text{[satisfying (c)]} \\
\Gamma; \Delta \vdash F \rightsquigarrow R :: z:C \quad \text{by cut}^! \\
\text{with } R = (\nu u)(!u(w).R_1 \mid (\nu y)(\nu x)(R'_2 \mid Q'_1)) \\
\equiv (\nu y)(\nu x)(\nu u)(!u(w).R_1 \mid R'_2 \mid Q'_1) \\
\equiv (\nu y)(\nu x)((\nu u)(!u(w).R_1 \mid R'_2) \mid Q'_1) \\
\equiv (\nu y)(\nu x)(P'_1 \mid Q'_1) \quad \text{[satisfying (d)]}
\end{array}$$

□

Lemma A.9. *Assume*

- (a) $\Gamma; \cdot \vdash D \rightsquigarrow P :: u:A$ and
- (b) $\Gamma, u:A; \Delta \vdash E \rightsquigarrow Q :: z:C$ with $Q \xrightarrow{(\nu y)u\langle y \rangle} Q'$.

Then

- (c) $\text{cut}^! D_1(u.D_2) \equiv \equiv \text{cut}^! D_1(u.F)$ for some F ;
- (d) $\Gamma, u:A; \Delta \vdash F \rightsquigarrow R :: z:C$ with $R \equiv (\nu y)(P\{y/u\} \mid Q')$.

Proof. By induction on the structure of E . There are only five possible cases for E : $\mathbf{1L} \ n \ E'_2$, copy $u(x.E'_2)$, $\mathbf{!L} \ n \ (u.E'_2)$, $\text{cut } E'_2(n.E''_2)$, and $\text{cut}^! E'_2(v.E''_2)$. In all other cases Q_1 cannot offer $(\nu y)u\langle y \rangle$, which follows by analysis of the typed extraction rules and the definition of $\xrightarrow{\alpha}$.

Case: $E = \text{copy } u (x. E'_2)$.

$$\begin{array}{l}
Q_1 \equiv (\nu y)u(y).Q'_1 \\
\Gamma, u : A; \Delta, y : A \vdash E'_2 \rightsquigarrow Q'_1 :: z : C \\
\text{cut}^! D (u. (\text{copy } u (y. E'_2))) \\
\Rightarrow \text{cut} D (y. \text{cut}^! D (u. E'_2)) \\
\equiv \text{cut}^! D (u. \text{cut} D (y. E'_2)) \\
\text{Pick } F = \text{cut} D (y. E'_2) \\
\Gamma, u : A; \Delta \vdash F \rightsquigarrow R :: z : C \text{ with } R \equiv (\nu y)(P_1\{y/u\} \mid Q'_1)
\end{array}
\begin{array}{l}
\text{by inversion} \\
\text{by } (\text{cut}^! / - / \text{copy}) \\
\text{by } (\text{cut} / - / \text{cut}^!) \\
[\text{satisfying (c)}] \\
[\text{satisfying (d)}]
\end{array}$$

Case: $E = \mathbf{1L} \ n \ E'_2$.

$$\begin{array}{l}
\Delta = (\Delta^*, n : \mathbf{1}) \\
\Gamma, u : A; \Delta^* \vdash E'_2 \rightsquigarrow Q_1 :: z : C \text{ with } Q_1 \xrightarrow{(\nu y)u(y)} Q'_1 \\
\text{cut}^! D (u. E'_2) \equiv \equiv \text{cut}^! D (u. D^*) \text{ for some } D^* \\
\Gamma; \Delta^* \vdash D^* \rightsquigarrow R' :: z : C \text{ with } R' \equiv (\nu y)(P_1\{y/u\} \mid Q'_1) \\
\text{cut}^! D (u. \mathbf{1L} \ n \ E'_2) \\
\equiv \mathbf{1L} \ n \ (\text{cut}^! D (u. E'_2)) \\
\equiv \equiv \mathbf{1L} \ n \ (\text{cut}^! D (u. D^*)) \\
\equiv \text{cut}^! D (u. \mathbf{1L} \ n \ D^*) \\
\text{Pick } F = \mathbf{1L} \ n \ D^* \\
\text{Pick } R = R' \\
\Gamma, u : A; \Delta \vdash F \rightsquigarrow R :: z : C \text{ with } R \equiv (\nu y)(P_1\{y/u\} \mid Q'_1)
\end{array}
\begin{array}{l}
\text{By inversion} \\
\text{By i.h.} \\
\text{by } (\text{cut}^! / - / \mathbf{1L}) \\
\text{By congruence} \\
\text{by } (\text{cut}^! / - / \mathbf{1L}) \\
[\text{satisfying (c)}] \\
\text{By } \mathbf{1L} \text{ [satisfying (d)}]
\end{array}$$

Case: $E = \mathbf{!L} \ x \ (v. E'_2)$.

$$\begin{array}{l}
\Delta = (\Delta^*, x : \mathbf{!B}) \\
\Gamma, v : B, u : A; \Delta^* \vdash E'_2 \rightsquigarrow Q_1 :: z : C \text{ with } Q_1 \xrightarrow{(\nu y)u(y)} Q'_1 \\
\text{cut}^! D (u. E'_2) \equiv \equiv \text{cut}^! D (u. D^*) \text{ for some } D^* \\
\Gamma, v : B; \Delta^* \vdash D^* \rightsquigarrow R' :: z : C \text{ with } R' \equiv (\nu y)(P_1\{y/u\} \mid Q'_1) \\
\text{cut}^! D (u. (\mathbf{!L} \ x \ (v. E'_2))) \\
\equiv \mathbf{!L} \ x \ (v. \text{cut}^! D (u. E'_2)) \\
\equiv \equiv \mathbf{!L} \ x \ (v. \text{cut}^! D (u. D^*)) \\
\equiv \text{cut}^! D (u. \mathbf{!L} \ x \ (v. D^*)) \\
\text{Pick } F = \mathbf{!L} \ x \ (v. D^*) \\
\text{Pick } R = R' \\
\Gamma, u : A; \Delta \vdash F \rightsquigarrow R :: z : C \text{ with } R \equiv (\nu y)(P_1\{y/u\} \mid Q'_1)
\end{array}
\begin{array}{l}
\text{By inversion} \\
\text{By i.h.} \\
\text{by } (\text{cut}^! / - / \mathbf{!L}) \\
\text{By congruence} \\
\text{by } (\text{cut}^! / - / \mathbf{!L}) \\
[\text{satisfying (a)}] \\
\text{By } \mathbf{!L} \text{ [satisfying (b)}]
\end{array}$$

Case: $E = \text{cut } E'_2 (n. E''_2)$.

$$\begin{array}{l}
\Delta = (\Delta_1, \Delta_2) \\
\Gamma, x : A; \Delta_1 \vdash E'_2 \rightsquigarrow R_1 :: n : B \\
\Gamma, x : A; \Delta_2, n : B \vdash E''_2 \rightsquigarrow R_2 :: z : C \\
Q_1 = (\nu n)(R_1 \mid R_2)
\end{array}
\begin{array}{l}
\text{By inversion}
\end{array}$$

Either $R_1 \xrightarrow{(\nu y)x\langle y \rangle} R'_1$ and $Q'_1 \equiv (\nu n)(R'_1 \mid R_2)$, or
 $R_2 \xrightarrow{(\nu y)x\langle y \rangle} R'_2$ and $Q'_1 \equiv (\nu n)(R_1 \mid R'_2)$

Subcase: $R_1 \xrightarrow{(\nu y)x\langle y \rangle} R'_1$

$\text{cut}^! D(u, E'_2) \equiv \equiv \text{cut}^! D(u, D^*)$ for some D^*

$\Gamma, u : A; \Delta_1 \vdash D^* \rightsquigarrow S :: n : B$

with $S \equiv (\nu y)(P_1\{y/u\} \mid R'_1)$

By i.h.

$\text{cut}^! D(u, \text{cut } E'_2(n, E''_2))$

$\equiv \text{cut}(\text{cut}^! D(u, E'_2))(n, \text{cut}^! D(u, E''_2))$

$\equiv \text{cut}(\text{cut}^! D(u, D^*))(n, \text{cut}^! D(u, E''_2))$

$\equiv \text{cut}^! D(u, \text{cut } D^*(n, E''_2))$

Pick $D = \text{cut } D^*(n, D''_2)$

Pick $R = (\nu n)(S \mid R_2)$

$\Gamma, u : A; \Delta \vdash D \rightsquigarrow R :: z : C$

with $R = (\nu n)((\nu y)(P_1\{y/u\} \mid R'_1) \mid R_2)$

$\equiv (\nu y)(P_1\{y/u\} \mid (\nu n)(R'_1 \mid R_2))$

$\equiv (\nu y)(P_1\{y/u\} \mid Q'_1)$

by (cut[!]/ – /cut)

By congruence

by (cut[!]/ – /cut)

[satisfying (c)]

By cut

[satisfying (d)]

Subcase: $R_2 \xrightarrow{(\nu y)x\langle y \rangle} R'_2$

$\text{cut}^! D(u, E''_2) \equiv \equiv \text{cut}^! D(u, D^*)$ for some D^*

$\Gamma, u : A; \Delta_2, n : B \vdash D^* \rightsquigarrow S :: z : C$

with $S \equiv (\nu y)(P_1\{y/u\} \mid R'_2)$

By i.h.

$\text{cut}^! D(u, \text{cut } E'_2(n, E''_2))$

$\equiv \text{cut}(\text{cut}^! D(u, E'_2))(n, \text{cut}^! D(u, E''_2))$

$\equiv \text{cut}(\text{cut}^! D(u, E''_2))(n, \text{cut}^! D(u, D^*))$

$\equiv \text{cut}^! D(u, \text{cut } E'_2(n, D^*))$

Pick $F = \text{cut } E'_2(n, D^*)$

Pick $R = (\nu n)(R_1 \mid S)$

$\Gamma, u : A; \Delta \vdash D \rightsquigarrow R :: z : C$

with $R = (\nu n)(R_1 \mid (\nu y)(P_1\{y/u\} \mid R'_2))$

$\equiv (\nu y)(P_1\{y/u\} \mid (\nu n)(R_1 \mid R'_2))$

$\equiv (\nu y)(P_1\{y/u\} \mid Q'_1)$

By (cut[!]/ – /cut)

By congruence

by (cut[!]/ – /cut)

[satisfying (c)]

By cut

[satisfying (d)]

Case: $E = \text{cut}^! E'_2(v, E''_2)$.

$\Gamma, u : A; \vdash E'_2 \rightsquigarrow R_1 :: w : B$

$\Gamma, v : B, u : A; \Delta_2, \vdash E''_2 \rightsquigarrow R_2 :: z : C$

By inversion

$Q_1 = (\nu v)(!v(w).R_1 \mid R_2)$

$R_2 \xrightarrow{(\nu y)x\langle y \rangle} R'_2$ and $Q'_1 \equiv (\nu v)(!v(w).R_1 \mid R'_2)$

$\text{cut}^! D(v, E''_2) \equiv \equiv \text{cut}^! D(v, D^*)$ for some D^*

$\Gamma, v : B, u : A; \Delta_2 \vdash D^* \rightsquigarrow S :: z : C$

with $S \equiv (\nu y)(P_1\{y/u\} \mid R'_2)$

By i.h.

$\text{cut}^! D(u, \text{cut}^! E'_2(v, E''_2))$

$\equiv \text{cut}^! (\text{cut}^! D(u, E'_2))(v, \text{cut}^! D(u, E''_2))$

$\equiv \text{cut}^! (\text{cut}^! D(u, E''_2))(v, \text{cut}^! D(u, D^*))$

$\equiv \text{cut}^! D(u, \text{cut}^! E'_2(v, D^*))$

by (cut[!]/ – /cut[!])

By congruence

by (cut[!]/ – /cut[!])

[satisfying (c)]

Pick $F = \text{cut}^! E'_2(v, D^*)$

$$\begin{aligned}
& \Gamma, u : A; \Delta \vdash F \rightsquigarrow R :: z:C && \text{By cut} \\
& \text{with } R = (\nu v)(!v(w).R_1 \mid S) \\
& \equiv (\nu v)(!v(w).R_1 \mid (\nu y)(P_1\{y/u\} \mid R'_2)) \\
& \equiv (\nu y)(\nu v)(!v(w).R_1 \mid P_1\{y/u\} \mid R'_2) \\
& \equiv (\nu y)(\nu v)(P_1\{y/u\} \mid !v(w).R_1 \mid R'_2) \\
& \equiv (\nu y)(P_1\{y/u\} \mid (\nu v)(!v(w).R_1 \mid R'_2)) \\
& \equiv (\nu y)(P_1\{y/u\} \mid Q'_1)
\end{aligned}$$

[satisfying (d)]

□

Lemma A.10. *Assume*

(a) $\Gamma; \cdot \vdash D \rightsquigarrow P :: u:A$ and

(b) $\Gamma, u:A; \Delta_2 \vdash E \rightsquigarrow Q :: z:C$ with $Q \xrightarrow{(\nu y)u(y)} Q'$.

Then

(c) $\text{cut}^! D (u. E) \equiv \equiv F$ for some F and

(d) $\Gamma; \Delta \vdash F \rightsquigarrow R :: z : C$ for some $R \equiv (\nu u)(!u(x).P \mid (\nu y)(P\{y/u\} \mid Q'))$.

Proof. Directly from Lemma A.9 and the typed extraction rule for $\text{cut}^!$. □

Theorem A.11. *Assume* $\Gamma; \Delta \vdash D \rightsquigarrow P :: z:A$ and $P \rightarrow Q$.

Then there is E such that $D \equiv \equiv E$ and $\Gamma; \Delta \vdash E \rightsquigarrow Q :: z:A$

Proof. By induction on the structure of D . The possible cases for D are $1L$, $!L$, cut , and $\text{cut}^!$. In all other cases P cannot offer τ .

Case: $D = 1L n D'$.

$$\begin{aligned}
& \Delta = (\Delta^*, n : 1) \\
& \Gamma; \Delta^* \vdash D' \rightsquigarrow P :: z:A && \text{by inversion} \\
& \Gamma; \Delta^* \vdash E' \rightsquigarrow Q :: z:A \text{ for some } E' \text{ with } D' \equiv \equiv E' && \text{by i.h.} \\
& \text{Pick } E = 1L n E'. \\
& D \equiv \equiv E && \text{by congruence} \\
& \Gamma; \Delta \vdash E \rightsquigarrow Q :: z:A && \text{by } 1L
\end{aligned}$$

Case: $D = !L x (u. D')$.

$$\begin{aligned}
& \Delta = (\Delta^*, x : !B) \\
& \Gamma, u : B; \Delta^* \vdash D' \rightsquigarrow P :: z:A && \text{by inversion} \\
& \Gamma, u : B; \Delta^* \vdash E' \rightsquigarrow Q :: z:A \text{ for some } E' \text{ with } D' \equiv \equiv E' && \text{by i.h.} \\
& \text{Pick } E = !L x (u. E'). \\
& D \equiv \equiv E && \text{by congruence} \\
& \Gamma : \Delta \vdash E \rightsquigarrow Q :: z:A && \text{by } 1L
\end{aligned}$$

Case: $D = \text{cut}^! D_1 (u. D_2)$.

$$\begin{aligned}
& P \equiv (\nu u)(!u(w).P_1 \mid P_2) \\
& \Gamma; \vdash D_1 \rightsquigarrow P_1 :: u:C \\
& \Gamma, u : C; \Delta \vdash D_2 \rightsquigarrow P_2 :: z:A && \text{by inversion} \\
& \text{From } P \rightarrow Q \text{ either}
\end{aligned}$$

(1) $P_2 \rightarrow Q_2$ and $Q = (\nu u)(!u(w).P_1 \mid Q_2)$

(2) $P_2 \xrightarrow{(\nu y)u\langle y \rangle} Q_2$ and $Q = (\nu u)(!u(w).P_1 \mid (\nu y)(P_1\{y/u\} \mid Q_2))$

Subcase (1):

$\Gamma, u : C; \Delta \vdash D \rightsquigarrow Q_2 :: z:A$ for some E' with $D_2 \equiv \Rightarrow \equiv E$ by i.h.

$\text{cut}^! D_1 (u. D_2) \equiv \Rightarrow \equiv \text{cut}^! D_1 (u. E')$

Pick $E = \text{cut}^! D_1 (u. E')$

$\Gamma; \Delta \vdash E \rightsquigarrow Q :: z:A$ by cut[!]

Subcase (2):

$\text{cut}^! D_1 (u. D_2) \equiv \Rightarrow \equiv E$ for some E

$\Gamma; \Delta \vdash E \rightsquigarrow R :: z:A$ with $R \equiv Q$ by Corollary A.10

Case: $D = \text{cut } D_1 (x. D_2)$.

$P \equiv (\nu x)(P_1 \mid P_2)$

$\Delta = (\Delta_1, \Delta_2)$

$\Gamma; \Delta_1 \vdash D_1 \rightsquigarrow P_1 :: x:C$

$\Gamma; \Delta_2, x : C \vdash D_2 \rightsquigarrow P_1 :: z:A$ by inversion

Since $P \rightarrow Q$ there are four subcases:

(1) $P_1 \rightarrow Q_1$ and $Q = (\nu x)(Q_1 \mid P_2)$

(2) $P_2 \rightarrow Q_2$ and $Q = (\nu x)(P_1 \mid Q_2)$

(3) $P_1 \xrightarrow{\alpha} Q_1$ and $P_2 \xrightarrow{\bar{\alpha}} Q_2$

(4) $P_1 \xrightarrow{\bar{\alpha}} Q_1$ and $P_2 \xrightarrow{\alpha} Q_2$

Subcase (1): $P_1 \rightarrow Q_1$

$D_1 \equiv \Rightarrow \equiv E_1$ for some E_1

$\Gamma; \Delta_1 \vdash E_1 \rightsquigarrow Q_1 :: x:C$ by i.h.

$D = \text{cut } D_1 (x. D_2)$

$\equiv \Rightarrow \equiv \text{cut } E_1 (x. D_2)$ by congruence

Pick $E = \text{cut } E_1 (x. D_2)$

$\Gamma; \Delta \vdash E \rightsquigarrow Q :: z:A$

Subcase (2): $P_2 \rightarrow Q_2$

Symmetric to Subcase (1).

Subcase (3): $P_1 \xrightarrow{\alpha} Q_1$ and $P_2 \xrightarrow{\bar{\alpha}} Q_2$

Subsubcase: $C = 1$

not possible

Subsubcase: $C = C_1 \& C_2$

$\alpha = x.\text{inl}$ or $\alpha = x.\text{inr}$ By Lemma A.1

$\text{cut } D_1 (x. D_2) \equiv \Rightarrow \equiv D$ for some D

$\Gamma; \Delta \vdash D \rightsquigarrow R :: z : C$

with $R \equiv (\nu x)(Q_1 \mid Q_2) = Q$ by Lemmas A.2 and A.3

Subsubcase: $C = C_1 \oplus C_2$

not possible

Subsubcase: $C = C_1 \otimes C_2$

not possible

Subsubcase: $C = C_1 \multimap C_2$

$\alpha = x(y)$ and $\bar{\alpha} = \overline{(\nu y)x\langle y \rangle}$ By Lemma A.1

$\text{cut } D_1 (x. D_2) \equiv \Rightarrow \equiv D$ for some D

$\Gamma; \Delta \vdash D \rightsquigarrow Q :: z : C$

with $R \equiv (\nu x)(\nu y)(Q_1 \mid Q_2) = Q$ by Lemma A.7

Subsubcase: $C = !C_1$

$\alpha = x(y)$ and $\bar{\alpha} = \overline{(\nu y)x(y)}$

cut $D_1 (x. D_2) \equiv \equiv D$ for some D

$\Gamma; \Delta \vdash D \rightsquigarrow Q :: z : C$

with $R \equiv (\nu x)(\nu y)(Q_1 \mid Q_2) = Q$

Subcase (4): $P_1 \xrightarrow{\bar{\alpha}} Q_1$ and $P_2 \xrightarrow{\alpha} Q_2$

Subsubcase: $C = \mathbf{1}$

not possible

Subsubcase: $C = C_1 \& C_2$

not possible

Subsubcase: $C = C_1 \oplus C_2$

$\bar{\alpha} = x.\text{inl}$ or $\bar{\alpha} = x.\text{inr}$

cut $D_1 (x. D_2) \equiv \equiv D$ for some D

$\Gamma; \Delta \vdash D \rightsquigarrow R :: z : C$

with $R \equiv (\nu x)(Q_1 \mid Q_2) = Q$

Subsubcase: $C = C_1 \otimes C_2$

$\bar{\alpha} = \overline{(\nu y)x(y)}$ and $\alpha = x(y)$

cut $D_1 (x. D_2) \equiv \equiv D$ for some D

$\Gamma; \Delta \vdash D \rightsquigarrow Q :: z : C$

with $R \equiv (\nu x)(\nu y)(Q_1 \mid Q_2) = Q$

Subsubcase: $C = C_1 \multimap C_2$

not possible

Subsubcase: $C = !C_1$

not possible

By Lemma A.1

by Lemma A.8

By Lemma A.1

by Lemmas A.4 and A.5

By Lemma A.1

□

A.3 Progress Lemmas

Lemma A.12. Assume $\Gamma, \Delta \vdash D \rightsquigarrow P :: z : C$ and not *live*(P); then

1. $C = \mathbf{1}$ or $C = !C'$ for some C' .
2. $(x_i : A_i) \in \Delta$ implies $A_i = \mathbf{1}$ or there is B_j with $A_i = !B_j$;
3. $C = !C'$ implies $P \equiv (\nu \bar{x})(!z(y).R \mid R')$.

Proof. By structural induction on the structure of D . The only possible cases for D are $\mathbf{1R}$, $\mathbf{1L}$ D' , $!\mathbf{L}$ $x (u. D')$, cut $D' (x. D'')$, $\mathbf{R}!$ D' , and cut[!] $D' (x. D'')$, which follows by analysis of the typing rules.

Case: $D = \text{cut } D' (x. D'')$

$\Delta = (\Delta_1, D_2)$

$P \equiv (\nu x)(P_1 \mid P_2)$

$\Gamma; \Delta_1 \vdash D' \rightsquigarrow P_1 :: x:A$

$\Gamma; \Delta_2, x : A \vdash D'' \rightsquigarrow P_2 :: z:C$

not *live*(P_1) and not *live*(P_2)

$C = \mathbf{1}$ or $C = !C'$ for some C'

$(x_i : A_i) \in (\Delta_2, x : A)$ implies $A_i = \mathbf{1}$ or there is B_j with $A_i = !B_j$

$C = !C'$ implies $P_2 \equiv (\nu \bar{n})(!z(y).P'_2 \mid P''_2)$

$A = \mathbf{1}$ or $A = !A'$ for some A'

by inversion
Since not *live*(P_2)

[satisfying (1)]

by i.h.

$(x_i : A_i) \in \Delta_1$ implies $A_i = \mathbf{1}$ or there is B_j with $A_i = !B_j$
 $A = !A'$ implies $P_1 \equiv (\nu \bar{m})(!x(y).P'_1 \mid P''_1)$ by i.h.

$(x_i : A_i) \in (\Delta_1, \Delta_2) = \Delta$ implies $A_i = \mathbf{1}$
or there is B_j with $A_i = !B_j$ [satisfying (2)]

$C = !C'$ implies $P \equiv (\nu x)(P_1 \mid P_2)$
 $\equiv (\nu x)((\nu \bar{m})(!x(y).P'_1 \mid P''_1) \mid (\nu \bar{n})(!z(y).P'_2 \mid P''_2))$
 $\equiv (\nu x)(\nu \bar{m})(\nu \bar{n})(!x(y).P'_1 \mid P''_1 \mid !z(y).P'_2 \mid P''_2)$
 $\equiv (\nu x)(\nu \bar{m})(\nu \bar{n})(z(y).R \mid R')$
with $R = P'_2$ and $R' = !x(y).P'_1 \mid P''_1 \mid !z(y).P'_2 \mid P''_2$ [satisfying (3)]

Case: $D = \text{cut}^! D' (u. D'')$

$P \equiv (\nu x)(!x(y).P_1 \mid P_2)$
 $\Gamma; \vdash D' \rightsquigarrow P_1 :: y:A$
 $\Gamma, u : A; \Delta \vdash D'' \rightsquigarrow P_2 :: z:C$ by inversion
not *live*(P_2) Since not *live*(P_2)

$C = \mathbf{1}$ or $C = !C'$ for some C' [satisfying (1)]
 $(x_i : A_i) \in \Delta$ implies $A_i = \mathbf{1}$ or there is B_j with $A_i = !B_j$ [satisfying (2)]
 $C = !C'$ implies $P_2 \equiv (\nu \bar{n})(!z(y).P'_2 \mid P''_2)$ by i.h.

$C = !C'$ implies $P \equiv (\nu x)(!x(y).P_1 \mid P_2)$
 $\equiv (\nu x)(!x(y).P_1 \mid (\nu \bar{n})(!z(y).P'_2 \mid P''_2))$
 $\equiv (\nu x)(\nu \bar{n})(!x(y).P_1 \mid !z(y).P'_2 \mid P''_2)$
 $\equiv (\nu x)(\nu \bar{n})(z(y).R \mid R')$
with $R = P'_2$ and $R' = !x(y).P_1 \mid P_2$ [satisfying (3)]

□

Lemma A.13. Let $\Gamma; \Delta \vdash D \rightsquigarrow P :: z : C$. If *live*(P) then there is Q such that one of the following holds:

- (a) $P \rightarrow Q$,
- (b) $P \xrightarrow{\alpha} Q$ for some α where $s(\alpha) \in z, \Gamma, \Delta$ and $s(\alpha) \in \Gamma, \Delta$ if $C = !A$.

Proof. By induction on the structure of D . All cases are possible for D except $\mathbf{1R}$ and $\mathbf{!R}$ D' , which follows by analysis of the typing rules.

Case: $D = \mathbf{1L} n D'$

$\Delta = (\Delta^*, n : \mathbf{1})$
 $\Gamma; \Delta^* \vdash D' \rightsquigarrow P :: z:C$ by inversion
There is Q such that either $P \rightarrow Q$, or $P \xrightarrow{\alpha} Q$
for some α with $s(\alpha) \in z, \Gamma, \Delta^*$ and $s(\alpha) \in \Gamma, \Delta^*$ if $C = !A$. by i.h.
There is Q such that either $P \rightarrow Q$, or $P \xrightarrow{\alpha} Q$
for some α with $s(\alpha) \in z, \Gamma, \Delta$ and $s(\alpha) \in \Gamma, \Delta$ if $C = !A$.

Case: $D = \mathbf{!L} n (u. D')$

$\Delta = (n : !A, \Delta^*)$
 $\Gamma, u : A; \Delta^* \vdash D' \rightsquigarrow P :: z:C$ by inversion
There is Q such that either $P \rightarrow Q$, or $P \xrightarrow{\alpha} Q$

for some α with $s(\alpha) \in z, \Gamma, u, \Delta^*$ and $s(\alpha) \in \Gamma, u, \Delta^*$ if $C = !A$. by i.h.
 There is Q such that either $P \rightarrow Q$, or $P \xrightarrow{\alpha} Q$
 for some α with $s(\alpha) \in z, \Gamma, \Delta$ and $s(\alpha) \in \Gamma, \Delta$ if $C = !A$.

Case: $D = \otimes R D_1 D_2$

$\Delta = (\Delta_1, \Delta_2), C = C_1 \otimes C_2$.
 $\Gamma; \Delta_1 \vdash D_1 \rightsquigarrow Q :: y:C_1$
 $\Gamma; \Delta_2 \vdash D_2' \rightsquigarrow R :: z:C_2$
 $P \equiv (\nu y)z\langle y \rangle.(Q \mid R)$ by inversion
 $P \xrightarrow{(\nu y)z\langle y \rangle} Q$ with $z \in z, \Gamma, \Delta$ and $C \neq !A$.

Case: $D = \otimes L (y. D_1)$

$\Delta = (D^*, x : C_1 \otimes C_2)$
 $\Gamma; \Delta^*, y : C_1, x : C_2 \vdash D_1 \rightsquigarrow Q :: z:C$
 $P \equiv x(y).Q$ by inversion
 $P \xrightarrow{x(y)} Q$ with $x \in \Gamma, \Delta$

Case: $D = \multimap R D_1$

$C = C_1 \multimap C_2$.
 $\Gamma; \Delta, y : C_1 \vdash D_1 \rightsquigarrow Q :: z:C_2$
 $P \equiv z(y).Q$ by inversion
 $P \xrightarrow{z(y)} Q$ with $z \in z, \Gamma, \Delta$ and $C \neq !A$.

Case: $D = \multimap L D_1 D_2$

$\Delta = (\Delta_1, \Delta_2, x : C_1 \multimap C_2)$
 $\Gamma; \Delta_1 \vdash D_1 \rightsquigarrow Q :: y:C_1$
 $\Gamma; \Delta_2, x : C_2 \vdash D_2' \rightsquigarrow R :: z:C_2$
 $P \equiv (\nu y)x\langle y \rangle.(Q \mid R)$ by inversion
 $P \xrightarrow{(\nu y)x\langle y \rangle} Q$ with $x \in \Gamma, \Delta$

Case: $D = \text{cut } D_1 (x. D_2)$

$\Delta = (\Delta_1, \Delta_2)$
 $\Gamma; \Delta_1 \vdash D_1 \rightsquigarrow P_1 :: x:A$
 $\Gamma; \Delta_2, x : A \vdash D_2 \rightsquigarrow P_2 :: z:C$
 $P \equiv (\nu x)(P_1 \mid P_2)$ by inversion
 $\text{live}(P_1)$ or $\text{live}(P_2)$ since $\text{live}(P)$

Case (1): $\text{live}(P_1)$ and $\text{live}(P_2)$.

There is P_1' such that either $P_1 \rightarrow P_1'$, or $P_1 \xrightarrow{\alpha_1} P_1'$
 for some α_1 with $s(\alpha_1) \in x, \Gamma, \Delta_1$ and $s(\alpha_1) \in \Gamma, \Delta_1$ if $A = !B_1$.

There is P_2' such that either $P_2 \rightarrow P_2'$, or $P_2 \xrightarrow{\alpha_2} P_2'$
 for some α_2 with $s(\alpha_2) \in x, \Gamma, \Delta_2, z$ and $s(\alpha_2) \in x, \Gamma, \Delta_2$ if $C = !B_2$. by i.h.

Subcase (0.1): $P_1 \rightarrow P_1'$ or $P_2 \rightarrow P_2'$

$P \rightarrow Q$ [satisfying (a)]

Subcase (1.1): $s(\alpha_1) \neq x$

$P \overset{\alpha_1}{\rightarrow} Q \equiv (\nu x)(P'_1 \mid P_2)$ with $\alpha_1 \in \Gamma, \Delta$ [satisfying (b)]

Subcase (1.2): $s(\alpha_2) \neq x$

$P \overset{\alpha_2}{\rightarrow} Q \equiv (\nu x)(P_1 \mid P'_2)$ with $\alpha_2 \in z, \Gamma, \Delta$ [satisfying (b)]

Subcase (1.3): $s(\alpha_1) = s(\alpha_2) = x$

$\alpha_2 = \overline{\alpha_1}$

By Lemma A.1

$P \rightarrow Q$ with $Q \equiv (\nu x)(\nu y)(P'_1 \mid P'_2)$ or $Q \equiv (\nu x)(P'_1 \mid P'_2)$

[satisfying (a)]

Case (2): not $live(P_1)$ and $live(P_2)$

There is P'_2 such that either $P_2 \rightarrow P'_2$, or $P_2 \overset{\alpha_2}{\rightarrow} P'_2$

for some α_2 with $s(\alpha_2) \in x, \Gamma, \Delta_2, z$ and $s(\alpha_2) \in x, \Gamma, \Delta_2$ if $C = !B_2$

by i.h.

Subcase (2.1): $P_2 \rightarrow P'_2$

$P \rightarrow Q$ with $Q \equiv (\nu x)(P_1 \mid P'_2)$

[satisfying (a)]

Subcase (2.2): $P_2 \overset{\alpha_2}{\rightarrow} P'_2$

Subcase (2.2.1): $s(\alpha_2) \neq x$

$P \overset{\alpha_2}{\rightarrow} Q$ with $Q \equiv (\nu x)(P_1 \mid P'_2)$

[satisfying (b)]

Subcase (2.2.2): $s(\alpha_2) = x$

$A \neq \mathbf{1}$

By Lemma A.1

$P_1 \equiv (\nu \bar{y})(!x(w).R'_1 \mid R''_1)$

$A = !B$

By Lemma A.12

$P_1 \overset{x(y)}{\rightarrow} P_1$

$\alpha_2 = (\nu y)x\langle w \rangle$

By Lemma A.1

$P \rightarrow Q$ with $Q \equiv (\nu x)(\nu y)(P_1 \mid P'_2)$

[satisfying (a)]

Case (3): $live(P_1)$ and not $live(P_2)$

There is P'_1 such that either $P_1 \rightarrow P'_1$, or $P_1 \overset{\alpha_1}{\rightarrow} P'_1$

for some α_1 with $s(\alpha_1) \in \Gamma, \Delta_1, x$ and $s(\alpha_1) \in \Gamma, \Delta_1$ if $C = !B_2$

by i.h.

Subcase (3.1): $P_1 \rightarrow P'_1$

$P \rightarrow Q$ with $Q \equiv (\nu x)(P'_1 \mid P_2)$

[satisfying (a)]

Subcase (3.1): $P_1 \overset{\alpha_1}{\rightarrow} P'_1$

for some α_1 with $s(\alpha_1) \in \Gamma, \Delta_1, x$ and $s(\alpha_1) \in \Gamma, \Delta_1$ if $A = !B$ for some B

Subcase (3.1.1) $s(\alpha_1) = x$

$A = \mathbf{1}$ or $A = !B$ for some B

By Lemma A.12

Subcase (3.1.1.1) $A = \mathbf{1}$

Impossible, since $P_1 \overset{\alpha_1}{\rightarrow} P'_1$ and $s(\alpha_1) = x$

By Lemma A.1

Subcase (3.1.1.2) $A = !B$ for some B

Impossible, since $s(\alpha_1) = x$ contradicts $x \in \Gamma, \Delta_1$.

Case: $D = \text{cut}^! D_1 (u. D_2)$

$\Gamma; \vdash D_1 \rightsquigarrow P_1 :: y:A$

$\Gamma, u : A; \Delta \vdash D_2 \rightsquigarrow P_2 :: z:C$

$P \equiv (\nu u)(!u(y).P_1 \mid P_2)$

by inversion

$live(P_2)$

since $live(P)$

There is P'_2 such that either $P_2 \rightarrow P'_2$, or $P_2 \overset{\alpha_2}{\rightarrow} P'_2$

for some α_2 with $s(\alpha_2) \in u, \Gamma, \Delta, z$ and $s(\alpha_2) \in u, \Gamma, \Delta$ if $C = !B$.

Subcase (1): $P_2 \rightarrow P'_2$

$P \rightarrow Q$ with $Q \equiv (\nu u)(P_1 \mid P'_2)$

[satisfying (a)]

Subcase (2): $P_2 \overset{\alpha_2}{\rightarrow} P'_2$

Subcase (2.1): $s(\alpha_2) \neq u$

$P \xrightarrow{\alpha_2} Q$ with $Q \equiv (\nu u)(P_1 \mid P'_2)$ [satisfying (b)]
 where $s(\alpha_2) \in \Gamma, \Delta, z$ and $s(\alpha_1) \in \Gamma, \Delta$ if $C = !B$
 Subcase (2.2): $s(\alpha_2) = u$
 $P_2 \xrightarrow{(\nu y)u(y)} P'_2$ By Lemma A.1
 $!u(y).P_1 \xrightarrow{u(y)} (P_1 \mid !u(y).P_1)$
 $P \rightarrow Q$ with $Q \equiv (\nu x)(\nu y)(P_1 \mid !u(y).P_1 \mid P'_2)$ [satisfying (a)]

□

Theorem A.14. *Let $\cdot; \cdot \vdash D \rightsquigarrow P :: x : \mathbf{1}$. If $\text{live}(P)$ then there is Q such that $P \rightarrow Q$.*

Proof. (a) $P \rightarrow Q$, or (b) $P \xrightarrow{\alpha} Q$ for some α where $s(\alpha) = x$, by Lemma A.13. But $P \xrightarrow{\alpha} Q$ with $s(\alpha) = x$ is not possible since $x : \mathbf{1}$, by Lemma A.1. So $P \rightarrow Q$. □