A Subexponential View of Domains in Session Types

Daniele Nantes
MAT-UnB
Brasilia, Brazil
dnantes@unb.br

Carlos Olarte
ECT-UFRN
Natal, Brazil
LIPN, Université Sorbonne Paris Nord
Villetaneuse, France
olarte@lipn.univ-paris13.fr

Daniel Ventura
INF-UFG
Goiânia, Brazil
ventura@ufg.br

Linear logic (LL) has inspired the design of many computational systems, offering reasoning techniques built on top of its meta-theory. Since its inception, several connections between concurrent systems and LL have emerged from different perspectives. In the last decade, the seminal work of Caires and Pfenning showed that formulas in LL can be interpreted as session types and processes in the π-calculus as proof terms. This leads to a Curry-Howard interpretation where proof reductions in the cut-elimination procedure correspond to process reductions/interactions. The subexponentials in LL have also played an important role in concurrent systems since they can be interpreted in different ways, including timed, spatial and even epistemic modalities in distributed systems. In this paper we address the question: What is the meaning of the subexponentials from the point of view of a session type interpretation? Our answer is a π-like process calculus where agents reside in locations/sites and they make it explicit how the communication among the different sites should happen. The design of this language relies completely on the proof theory of the subexponentials in LL, thus extending the Caires-Pfenning interpretation in an elegant way.

1 Introduction

One of the most profound connections between proof theory and computation is the Curry-Howard isomorphism (also known as *formulas-as-types*) where (functional) programs correspond to proofs in intuitionistic logic, and reductions/computations in one system correspond to the process of normalization in the other [34]. Several correspondences with formal systems in a Curry-Howard fashion has been considered ever since, including classical logic [19, 31, 22], intuitionistic modal logic [14, 26] and intuitionistic linear logic [1, 4], to name a few.

Inspired by the *proofs-as-processes* paradigm [2], session types were proposed as a type foundation of communicating processes [20] with a wide range of applications [17, 21]. The seminal work of Caires and Pfenning [8, 9] establishes a correspondence between Girard's intuitionistic linear logic [18] and the session-typed π-calculus, a process calculus with mobility [25]. This correspondence was recently extended in [7], dealing with a notion of domain/location not featured in session type systems. The present work proposes an alternative interpretation to obtain a domain-aware session type system. For that, we build on a different logical system and, we design a language of processes with explicit annotations for locations. Let us elaborate more on these two distinguished characteristics of our approach.

Linear logic and variants. Linear logic (LL) [18] is a substructural logic that can be seen as a refinement of classical and intuitionistic logics, integrating the dualities of the former with many of the constructive properties of the latter. In this logic, formulas are seen as resources that are consumed when used. For instance, the linear implication $A \multimap B$ specifies that, in order to produce B, one copy of A must be consumed. In order to specify that a formula A can be used unboundedly many times, as in classical logic, the formula A must be marked with the connective !, called exponential (or bang).

Apart from substructural behaviors (being linear or unbounded), it is not natural to specify some other modal behaviors directly in LL [23]. For this reason, recent developments in LL have brought new systems to specify multimodalities in it. One of such systems is Hybrid Linear Logic (HyLL) [15, 11] where LL formulas are attached to worlds, representing constraints or modalities. We shall elaborate more on HyLL in Section 5, since the domain-aware type system in [7] is based on it.

In this paper, the base logic is Subexponential Linear Logic (*SELL*) [13, 28]. In *SELL*, instead of having a unique exponential !, we may have infinitely many *colored* subexponentials of the form $!^a$. This kind of modalities have been extensively used to interpret different concurrent languages featuring behaviors as spatial, epistemic and temporal modalities [30]. For that, following the *computation-as-proof-search* interpretation [24], it has been shown that (concurrent/modal) programs correspond to *SELL* formulas and their execution to proof search. However, so far, we do not have yet a *formulas-as-types* interpretation for *SELL*, and this is one of the main contributions of this paper.

Session typed π-calculi. Session types is a formalism to reason about processes, providing a typing discipline for concurrent programming based on process algebras such as the π-calculus [35, 20]. They have been deeply investigated in several contexts, as, for instance: detection of subtle errors in implementations; designing and analysis of choreographies between distributed systems as in multiparty session types; expressiveness power to model communication properties such as deadlock-freedom, abortable computations, non-determinism, security, etc (see [17, 10, 12, 27]). The work of Caires, Pfenning and Toninho [9] proposes a connection between a type system for the synchronous π-calculus and intuitionistic linear logic that provides a correspondence between cut elimination steps and process reductions, thus extending the Curry-Howard isomorphism to linear propositions as session types. Since then, other extensions based on variants of the π-calculus were proposed, e.g., for the asynchronous π-calculus [16] and for the π-calculus with constructs for non-determinism and abortable behavior [6].

Contributions. Inspired by a recent connection between session typed processes and HyLL [7], we propose a novel interpretation of *SELL* via a session typed π-calculus with explicit domains. The interest of this extension is justified by the expressive power of linear logic and its two extensions *SELL* and HyLL [11]: whereas HyLL can be encoded in (vanilla) linear logic, the subexponentials in *SELL* cannot be directly encoded in LL in its standard presentation [23].

Differently from [7], our processes are decorated with their current locations, and at each step, the domain/location of each end-point of the communication is explicit. Interaction may happen from different locations and, if needed, a process can request to migrate to a common secure location. This gives an extra expressiveness to the language, allowing for reasoning about domains, migration and accessibility at the level of processes. We thus propose a session type discipline for a domain-aware synchronous π-calculus with mobility and choice, that provides yet another extension of the Curry-Howard interpretation. As we shall see, we interpret subexponentials as domains, which allows us to express and enforce where the processes reside and where communication must happen, respecting the accessibility constraints embedded in *SELL*'s subexponential preorder. We prove that cut-reductions correspond to process reductions (Th. 4.3). Moreover, the proof of the identity expansion theorem (Th. 4.6) reveals the existence of process mediators between the different locations of the system. As a corollary of the cut elimination theorem, we prove other properties of the system as type preservation and global progress.

Organization. Section 2 presents the basic notions regarding LL and *SELL* that are necessary for understanding this work. In Section 3 we introduce the language of processes that is adopted, as well as the associated structural congruence identities and operational semantics. Section 4 presents the proposed Curry-Howard interpretation with subexponentials as domains. We prove relevant properties as cut-elimination and identity expansion, and provide examples to illustrate our framework. Section 5 concludes with a discussion about the results presented and compares our approach with the one in [7].

$$\dfrac{\Gamma, F, H \vdash G}{\Gamma, F \otimes H \vdash G} \otimes_L \quad \dfrac{\Gamma_1 \vdash F \quad \Gamma_2 \vdash H}{\Gamma_1, \Gamma_2 \vdash F \otimes H} \otimes_R \quad \dfrac{\Gamma, F_i \vdash G}{\Gamma, F_1 \& F_2 \vdash G} \&_{L_i} \quad \dfrac{\Gamma \vdash F \quad \Gamma \vdash H}{\Gamma \vdash F \& H} \&_R$$

$$\dfrac{\Gamma_1 \vdash F \quad \Gamma_2, H \vdash G}{\Gamma_1, \Gamma_2, F \multimap H \vdash G} \multimap_L \quad \dfrac{\Gamma, F \vdash H}{\Gamma \vdash F \multimap H} \multimap_R \quad \dfrac{\Gamma, F \vdash G \quad \Gamma, H \vdash G}{\Gamma, F \oplus H \vdash G} \oplus_L \quad \dfrac{\Gamma \vdash F_i}{\Gamma \vdash F_1 \oplus F_2} \oplus_{R_i}$$

$$\dfrac{\Gamma \vdash G}{\Gamma, 1 \vdash G} 1_L \quad \dfrac{}{\cdot \vdash 1} 1_R \quad \dfrac{}{\Gamma, \mathbf{0} \vdash G} \mathbf{0}_L \quad \dfrac{}{\Gamma \vdash \top} \top_R \quad \dfrac{}{\mathsf{p} \vdash \mathsf{p}} I$$

Figure 1: Propositional fragment of iMALL (intuitionistic linear logic without exponentials).

2   Preliminaries

In this section we review some of the basic proof theory of linear logic (see [37] for more details) and its extension with subexponentials [13].

Intuitionistic Linear logic (ILL) [18] is a resource conscious logic, in the sense that formulas are consumed when used during proofs, unless they are marked with the exponential !. Formulas marked with ! behave classically, i.e., they can be contracted (duplicated) and weakened (erased) during proofs.

Formulas in ILL (without exponentials) are built from the grammar below:

$$F ::= \mathsf{p} \mid \mathbf{0} \mid 1 \mid \top \mid F_1 \otimes F_2 \mid F_1 \oplus F_2 \mid F_1 \multimap F_2 \mid F_1 \& F_2$$

where p is an atomic proposition. The connectives are: the *multiplicative* versions of true (1), conjunction $\otimes$ and implication $\multimap$; and the *additive* versions of conjunction &, disjunction $\oplus$, false $\mathbf{0}$ and true $\top$.

The proof rules are in Figure 1. Note that the multiplicative rules split the context ($\otimes_R$, $\multimap_L$) while the additive rules share the context in the premises ($\&_R$, $\oplus_L$).

Contraction and weakening of formulas are controlled by the connective !, called bang, whose inference rules are:

$$\dfrac{\Gamma, F \vdash G}{\Gamma, !F \vdash G} !_L \quad \dfrac{!\Gamma \vdash G}{!\Gamma \vdash !G} !_R \quad \dfrac{\Gamma \vdash G}{\Gamma, !F \vdash G} W \quad \dfrac{\Gamma, !F, !F \vdash G}{\Gamma, !F \vdash G} C$$

Notice that one is allowed to introduce a ! on the right ($!_R$) only if all formulas in the context are marked with a bang ($!\Gamma = \{!F \mid F \in \Gamma\}$). On the other side, the left rule does not impose any restriction. The rule $!_R$ is commonly called *promotion*, while the rule $!_L$ is called *dereliction*.

2.1   Subexponentials

Intuitionistic linear logic with subexponentials (*SELL*) shares with ILL all connectives except the exponentials: Instead of having a single exponential !, *SELL* may contain as many colored/labelled bangs (e.g., $!^a$) as needed. Such labelled bangs are called *subexponentials* [13, 28].

The proof system for intuitionistic *SELL* is parametric on a *subexponential signature* $\Sigma = \langle I, \preceq, U \rangle$, where I is a set of labels, $U \subseteq I$ is a set specifying the *unbounded* subexponentials, i.e. those that allow weakening and contraction (the subexponentials in $I \setminus U$ are called *linear*), and $\preceq$ is a pre-order among the elements of I. For cut-elimination [28], it is assumed that U is upwardly closed w.r.t. $\preceq$, *i.e.*, if $u \in U$ and $u \preceq u'$, then $u' \in U$. We shall use $s, a, b, c \cdots$ to range over elements in I and $u, u', w, w', \cdots$ to denote unbounded subexponentials.

The system *SELL* is constructed by extending the rules of ILL in Fig. 1 with the following ones. For each $a \in I$, we add the rules corresponding to dereliction and promotion:

$$\dfrac{\Gamma, F \vdash G}{\Gamma, !^a F \vdash G} !^a_L \quad \dfrac{!^{a_1} F_1, \ldots !^{a_n} F_n \vdash G \quad a \preceq a_i}{!^{a_1} F_1, \ldots !^{a_n} F_n \vdash !^a G} !^a_R$$

Note the side condition in $!^a{}_R$: One can only introduce a $!^a$ if the formulas in the left-context are all marked with indices greater or equal than a. Hence, if b is accessible from a ($a \preceq b$), we can show that $!^b F \vdash !^a F$. However, if a and b are not related, it is not possible to prove $!^a F \equiv !^b F$ for an arbitrary formula F, where $F \equiv G$ denotes logical equivalence: $(F \multimap G) \& (G \multimap F)$. This means that the subexponentials are not canonical [13, 28] and the subexponential signature determines the provability relation in *SELL*. This fact has been used to interpret the subexponentials in different ways, ranging from (nested) spaces in distributed systems, time-units, the epistemic state of agents and preferences [30, 32, 29].

Dyadic System. In this paper we shall use the so-called dyadic system [3] that results from the incorporation of the structural rules of weakening and contraction into the introduction rules. The key observation is that, on the left of the sequent, a formula $!^u F$ can be contracted and weakened whenever u is unbounded (i.e., $u \in U$). This is reflected in the syntax of dyadic sequents of the form $\Gamma : \Psi \vdash G$ which have two contexts, namely, a subexponential context containing formulas marked with a bang ($\Gamma = \{!^{a_1} F_1, \ldots, !^{a_n} F_n\}$) and the linear context (Ψ). As an example, consider the following rules: [1]

$$\frac{\Gamma : \Psi, F \vdash G}{\Gamma, !^a F : \Psi \vdash G} \; !^a{}_L \qquad \frac{\Gamma, !^u F : \Psi, F \vdash G}{\Gamma, !^u F : \Psi \vdash G} \; copy \qquad \frac{\Gamma, !^a F : \Psi \vdash G}{\Gamma : \Psi, !^a F \vdash G} \; store \qquad \frac{\Gamma^{\succeq a} : \cdot \vdash G}{\Gamma : \cdot \vdash !^a G} \; !_R$$

In $!^a{}_L$, $a \notin U$ (i.e., a is a linear subexponential); in *copy*, $u \in U$ (and F is contracted); The *store* rule moves marked formulas to the subexponential context. The promotion rule $!_R$ has the proviso that there are no formulas marked with $!^c$ if $c \notin U$ or $a \npreceq c$. Note that this must be the case since linear subexponentials cannot be weakened (nor contracted). The notation $\Gamma^{\succeq a}$ will be used to denote the following subset of Γ: $\{!^b F \in \Gamma \mid a \preceq b\}$ and it is defined if for all linear c, $!^c F \in \Gamma$ implies $a \preceq c$. Therefore, in $\Gamma^{\succeq a}$, all the unbounded subexponentials not related with a are weakened, and the remaining subexponentials must be greater than a, thus reflecting the behavior of the promotion rule.

2.2 Subexponential Quantifiers

The universal ($\Cap$) and existential ($\Cup$) quantifiers for subexponentials [29] require a typing information to guarantee that the cut rule is admissible (Theorem 2.1 below). More precisely, given a subexponential signature $\Sigma = \langle I, \preceq, U \rangle$, the judgment $b : a$ is true whenever $b \preceq a$. Hence, we obtain the following set of typed *subexponential constants*: $\mathscr{A}_\Sigma = \{b : a \mid a, b \in I, b \preceq a\}$. Similar to the universal quantifier $\forall$, that introduces *eigenvariables* to the signature, the rule for universal quantification on the right (and the rule for existential on the left) introduces a (fresh) *subexponential variable* $\alpha : a$, where a is a subexponential constant, *i.e.*, $a \in I$. Thus, SELL$^\Cap$ sequents have the form $\mathscr{A} : \Gamma : \Psi \vdash G$, where $\mathscr{A} = \mathscr{A}_\Sigma \cup \{\alpha_1 : a_1, \ldots, \alpha_n : a_n\}$, $\{\alpha_1, \ldots, \alpha_n\}$ is a set of subexponential variables and $\{a_1, \ldots, a_n\} \subseteq I$ is a set of subexponential constants. We shall use $I(X)$ to denote the set of constants and subexponential variables.

Formulas in SELL$^\Cap$ are as before with the addition of $\Cap \alpha : s.F$ and $\Cup \alpha : s.F$ where $\alpha : s$ is a (typed) subexponential variable, and s is a subexponential index (i.e., $s \in I(X)$).

The introduction rules for the subexponential quantifiers look similar to those introducing the first-order quantifiers (see Fig. 2). Intuitively, subexponential variables play a similar role as eigenvariables. The generic variable $\alpha : a$ represents *any subexponential – constant or variable –* that is in the ideal of a (i.e., in the set $\{c \mid c \preceq a\}$). Hence it can be substituted by any subexponential s of type b, with $b \preceq a$.

The other rules of the (dyadic) system for SELL$^\Cap$ in Figure 2 are similar to those in Fig. 1. Note that the context with unbounded subexponentials (Υ) is duplicated while those with linear subexponentials

---

[1]The reader familiar with dyadic systems for linear logic may wonder why we keep the bang in the classical context. The reason is to simplify the notation of the type system introduced in the next section.

$$\dfrac{\Gamma \,:\, \Psi, F, F' \vdash G}{\Gamma \,:\, \Psi, F \otimes F' \vdash G} \;\otimes_L \qquad \dfrac{\Upsilon, \Delta \,:\, \Psi \vdash F \quad \Upsilon, \Delta' \,:\, \Psi' \vdash F'}{\Upsilon, \Delta, \Delta' \,:\, \Psi, \Psi' \vdash F \otimes F'} \;\otimes_R \qquad \dfrac{\Gamma \,:\, \Psi, F_i \vdash G}{\Gamma \,:\, \Psi, F_1 \,\&\, F_2 \vdash G} \;\&_{L_i} \qquad \dfrac{\Gamma \,:\, \Psi \vdash F \quad \Gamma \,:\, \Psi \vdash F'}{\Gamma \,:\, \Psi \vdash F \,\&\, F'} \;\&_R$$

$$\dfrac{\Upsilon, \Delta \,:\, \Psi \vdash F \quad \Upsilon, \Delta' \,:\, \Psi', F' \vdash G}{\Upsilon, \Delta, \Delta' \,:\, \Psi, \Psi', F \multimap F' \vdash G} \;\multimap_L \qquad \dfrac{\Gamma \,:\, \Psi, F \vdash F'}{\Gamma \,:\, \Psi \vdash F \multimap F'} \;\multimap_R \qquad \dfrac{\Gamma \,:\, \Psi, F \vdash G \quad \Gamma \,:\, \Psi, F' \vdash G}{\Gamma \,:\, \Psi, F \oplus F' \vdash G} \;\oplus_L \qquad \dfrac{\Gamma \,:\, \Psi \vdash F_i}{\Gamma \,:\, \Psi \vdash F_1 \oplus F_2} \;\oplus_{R_i}$$

$$\dfrac{\Gamma \,:\, \Psi, F[l/\alpha] \vdash G}{\Gamma \,:\, \Psi, \Cap \alpha : a.F \vdash G} \;\Cap_L \qquad \dfrac{(\mathscr{A}, \alpha_e : a) \,:\, \Gamma \,:\, \Psi \vdash F[\alpha_e/\alpha]}{\mathscr{A} \,:\, \Gamma \,:\, \Psi \vdash \Cap \alpha : a.F} \;\Cap_R \qquad \dfrac{(\mathscr{A}, \alpha_e : a) \,:\, \Gamma \,:\, \Psi, F[\alpha_e/\alpha] \vdash G}{\mathscr{A} \,:\, \Gamma \,:\, \Psi, \Cup \alpha : a.F \vdash G} \;\Cup_L \qquad \dfrac{\Gamma \,:\, \Psi \vdash G[l/\alpha]}{\Gamma \,:\, \Psi \vdash \Cup \alpha : a.G} \;\Cup_R$$

$$\dfrac{\Gamma \,:\, \Psi, F \vdash G}{\Gamma, !^a F \,:\, \Psi \vdash G} \;!_L \qquad \dfrac{\Gamma, !^u F \,:\, \Psi, F \vdash G}{\Gamma, !^u F \,:\, \Psi \vdash G} \;copy \qquad \dfrac{\Gamma, !^a F \,:\, \Psi \vdash G}{\Gamma \,:\, \Psi, !^a F \vdash G} \;store \qquad \dfrac{\Gamma^{\succeq a} \,:\, \cdot \vdash G}{\Gamma \,:\, \cdot \vdash !^a G} \;!_R$$

$$\dfrac{\Gamma \,:\, \Psi \vdash G}{\Gamma \,:\, \Psi, 1 \vdash G} \;1_L \qquad \dfrac{}{\Upsilon \,:\, \cdot \vdash 1} \;1_R \qquad \dfrac{}{\Gamma \,:\, \Psi, \mathbf{0} \vdash G} \;\mathbf{0}_L \qquad \dfrac{}{\Gamma \,:\, \Psi \vdash \top} \;\top_R \qquad \dfrac{}{\Upsilon \,:\, \mathrm{p} \vdash \mathrm{p}} \;I$$

Figure 2: Dyadic system for SELL$^{\Cap}$. Υ denotes a context with formulas of the form $!^u F$ where $u \in U$ (unbounded subexp.). Δ and Δ' contain only formulas of the form $!^a F$ where $a \notin U$ (linear subexp.). Γ denotes a context that may contain both, linear and unbounded subpexponentials. Ψ is a multiset of formulas. In $!_L$, a is linear and u is unbounded in *copy*. In the rules for $\Cap$ and $\Cup$, $l : b \in \mathscr{A}$, $b \preceq a$ and α_e is fresh. Since the context $\mathscr{A}$ is only modified in rules $\Cap_R$ and $\Cup_L$, it is omitted in the other rules.

(Δ) are split in the premises of multiplicative rules. When the context $\mathscr{A}$ is unimportant or it can be inferred from the context, we will omit it.

Theorem 2.1 (Cut-elimination [29])**.** *The cut-rule below is admissible in SELL$^{\Cap}$ for any (upwardly closed w.r.t. U) signature.*

$$\dfrac{\mathscr{A} \,:\, \Upsilon, \Delta \,:\, \Psi \vdash F \quad \mathscr{A} \,:\, \Upsilon, \Delta' \,:\, \Psi', F \vdash G}{\mathscr{A} \,:\, \Upsilon, \Delta, \Delta' \,:\, \Psi, \Psi' \vdash G} \;Cut$$

3 The Language of Processes

We introduce the syntax and operational semantics of the synchronous π-calculus extended with binary choice [9] and explicit domain location, migration and communication, similarly to [7]. Concrete domains/locations range over a, b, c, domain/location variables over α, α', β and channel names over x, y, z. Domain identifiers are attached to processes ($P, Q, \ldots$), to explicitly identify their current location. Processes are defined in terms of pre-processes as below.

Definition 3.1 (Processes)**.** *Given countably infinite (disjoint) sets, $\mathscr{N}$ of names (x, y, z, u, v), I of concrete domains (a, b, c, d) and $\mathscr{V}$ of domain variables (α, α', β), processes are built from:*

Pre-process	p, q	$::= \;\; \mathbf{0} \mid (p \parallel q) \mid \bar{x}\langle y \rangle.p \mid x(y).p \mid !x(y).p \mid (\nu\, x)p$
		$x.\mathtt{case}(p, q) \mid x.\mathtt{inl}.p \mid x.\mathtt{inr}.p$
		$\bar{x}.\mathtt{move}\langle y, b \rangle.p \mid x.\mathtt{move}(y, b).p$
		$x\langle b \rangle.p \mid x(\alpha).p \mid x \overset{b}{\underset{a}{\longleftrightarrow}} y$
Processes	P, Q	$::= \;\; [p]_a \mid \mathbf{0} \mid (P \parallel Q) \mid (\nu\, x)P$

Process $P = [p]_a$ denotes a pre-process p that is currently in location a. Moreover,

- Process $[\bar{x}\langle y \rangle.p]_b$ denotes a process in location b that outputs y through channel x and continues as p. The dual process $[x(y).p]_a$ denotes a process in location a that expects to receive y via x and proceeds as p.

- Process $[x.\texttt{case}(p,q)]_b$ in location b offers the choice of continuing as p or q to another process via session x. The dual process $[x.\texttt{inl}.p]_a$ (resp. $[x.\texttt{inr}.p]_a$) in location a chooses the process on the left (resp. right) offered on session x and continues as p.

- Process $[\bar{x}.\texttt{move}\langle y,c\rangle.p]_b$ in location b denotes the pre-process that signals to another process the location c to which the pre-process p should be migrated to. After the migration, the communication will continue on the fresh session channel y. The dual process $[x.\texttt{move}(y,c).p]_a$ in location a is prepared to migrate the communication actions in p to session y on a location c.

- Processes $[x\langle b\rangle.p]_b$ and $[x(\alpha').p]_a$ are dual and denote output and input, respectively, of locations via session x.

- Pre-process $x \overset{b}{\underset{a}{\longleftrightarrow}} y$ is a copycat or mediator (see §4.6 below) that identifies/connects the channel x on location b with the channel y on location a.

- Pre-process $!x(y).p$ denotes a replicated/persistent input, useful for defining services that can be used as many times as needed.

- As usual, $\mathbf{0}$, $p \parallel q$ and $(\nu\, x)P$ denote, resp., inaction, parallel composition and name restriction.

As in [33, 5], the objects in outputs are fresh names. Hence, in $\bar{x}\langle y\rangle.p$ and $\bar{x}.\texttt{move}\langle y,b\rangle.p$, the output y is a fresh name. Moreover, input names are bound in the body of a pre-process, i.e. y in p is bound in all $x(y).p$, $!x(y).p$, $x.\texttt{move}(y,b).p$ (and α in p in $x(\alpha).p$). Finally, name x in process P is bound in $(\nu\, x)P$.

Example 3.2 (Locations). *The following process illustrates the parallel composition of processes in different locations, one in sv (server location) and the other in c (client location) that aim to communicate with each other via a restricted channel x: $P = (\nu\, x)([x(y).\bar{x}\langle z\rangle.y(k).\mathbf{0}]_{sv} \parallel [\bar{x}\langle y\rangle.(\bar{y}\langle k\rangle.\mathbf{0} \parallel x(z).\mathbf{0})]_c)$ Notice that the process on the left aims to use the name y received first in a later communication, after communicating z on channel x.*

Communication will rely on a location sensitive extension of the structural congruence and the operational semantics. The structural congruence identities are defined at the level of processes and pre-processes and the operational semantics is defined by the reduction $\rightarrow$ (closed under $\equiv$) in Fig. 3, where $q\{y/x\}$ denotes the implicit substitution of y for x in q (also for location substitution in rule $[\texttt{LocOut}]$).

Our semantics is similar to the standard operational semantics of the π-calculus. The difference here is that the location of each process is explicit and one process in one location can communicate with a process in another location. For instance, rule $[\texttt{Comm}]$ describes the communication of a process in location a that sends a fresh name y to a process in location b through channel x. After the communication, the two processes share the restricted channel y. Note that both processes continue in their original location after the interaction. The rule $[\texttt{Move}]$ allows processes to migrate to a *known a-priori* location c and the communication continues through the fresh channel y. Using the rule $[\texttt{LocOut}]$, it is possible to communicate a location before migrating to it. The other rules are standard.

Example 3.3. *Consider the process P in Example 3.2. Note that $P \rightarrow^* (\nu\, x)\mathbf{0} \equiv \mathbf{0}$, since:*
$$[x(y).\bar{x}\langle z\rangle.y(k).\mathbf{0}]_{sv} \parallel [\bar{x}\langle y\rangle.(\bar{y}\langle k\rangle.\mathbf{0} \parallel x(z).\mathbf{0})]_c \rightarrow (\nu\, y)([\bar{x}\langle z\rangle.y(k).\mathbf{0}]_{sv} \parallel [(\bar{y}\langle k\rangle.\mathbf{0} \parallel x(z).\mathbf{0})]_c)$$
$$\rightarrow (\nu\, y,z)([y(k).\mathbf{0}]_{sv} \parallel [\bar{y}\langle k\rangle.\mathbf{0}]_c) \rightarrow (\nu\, y,z,k)([\mathbf{0}]_{sv} \parallel [\mathbf{0}]_c) \equiv \mathbf{0}$$
The intended communications are performed: first the pre-process on the left receives name y through x, then it communicates z through the same channel, and finally it uses the name y received initially. Observe that the left pre-process has "stored" the name y for future use.

Structural Congruence for Pre-Processes

$$p \parallel \mathbf{0} \equiv p \qquad\qquad p \parallel q \equiv q \parallel p \qquad\qquad p \parallel (q \parallel r) \equiv (p \parallel q) \parallel r$$

$$(\nu\, x)\mathbf{0} \equiv \mathbf{0} \qquad\qquad (\nu\, x)(\nu\, y)p \equiv (\nu\, y)(\nu\, x)p \qquad\qquad p \equiv_\alpha q \Rightarrow p \equiv q$$

$$p \parallel (\nu\, y)q \equiv (\nu\, y)(p \parallel q), \text{ if } y \notin fn(p)$$

Structural Congruence for Processes

$$[\mathbf{0}]_a \equiv \mathbf{0} \qquad\qquad [p \parallel q]_a \equiv [p]_a \parallel [q]_a \qquad\qquad [(\nu\, y)p]_a \equiv (\nu\, y)[p]_a$$

$$P \parallel \mathbf{0} \equiv P \qquad\qquad (\nu\, x)\mathbf{0} \equiv \mathbf{0} \qquad\qquad [p]_a \parallel [(\nu\, y)q]_a \equiv (\nu\, y)[p \parallel q]_a, \text{ if } x \notin fn(p)$$

$$P \parallel Q \equiv Q \parallel P \qquad\qquad (\nu\, x)(\nu\, y)P \equiv (\nu\, y)(\nu\, x)P \qquad\qquad P \equiv_\alpha Q \Rightarrow P \equiv Q$$

$$P \parallel (Q \parallel R) \equiv (P \parallel Q) \parallel R \qquad P \parallel (\nu\, x)Q \equiv (\nu\, x)(P \parallel Q), \text{ if } x \notin fn(P) \qquad [x \overset{b}{\underset{a}{\leftrightarrow}} y]_a \equiv [y \overset{a}{\underset{b}{\leftrightarrow}} x]_b$$

[Comm]	$[\overline{x}\langle y\rangle.p]_a \parallel [x(z).q]_b$	$\rightarrow \quad (\nu\, y)([p]_a \parallel [q\{y/z\}]_b)$
[!Comm]	$[\overline{x}\langle y\rangle.p]_a \parallel [!x(z).q]_b$	$\rightarrow \quad (\nu\, y)([p]_a \parallel [q\{y/z\}]_b) \parallel [!x(z).q]_b$
[LocOut]	$[x\langle c\rangle.p]_a \parallel [x(\alpha).q]_b$	$\rightarrow \quad [p]_a \parallel [q\{c/\alpha\}]_b$
[Move]	$[\overline{x}.\mathsf{move}\langle y,c\rangle.p]_a \parallel [x.\mathsf{move}\langle y',c\rangle.q]_b$	$\rightarrow \quad (\nu\, y)[p \parallel q\{y/y'\}]_c$
[LCase]	$[x.\mathtt{inl}.p]_a \parallel [x.\mathtt{case}(q_1,q_2)]_b$	$\rightarrow \quad [p]_a \parallel [q_1]_b$
[RCase]	$[x.\mathtt{inr}.p]_a \parallel [x.\mathtt{case}(q_1,q_2)]_b$	$\rightarrow \quad [p]_a \parallel [q_2]_b$
[Id]	$(\nu\, x)([x \overset{b}{\underset{a}{\leftrightarrow}} y]_a \parallel P)$	$\rightarrow \quad P\{y/x\}$
[Res]	$\text{if } P \rightarrow Q \text{ then} \quad (\nu y)P$	$\rightarrow \quad (\nu y)Q$
[Par]	$\text{if } Q \rightarrow Q' \text{ then} \quad P \parallel Q$	$\rightarrow \quad P \parallel Q'$

Figure 3: Structural congruence and operational semantics

4 Subexponentials as Domains: Curry-Howard Interpretation

In this section we introduce a type system for the domain-aware process calculus presented in the previous section. We show that operational reductions correspond to proof reductions in the process of cut-elimination. Also, the proof of the identity expansion theorem (§4.6) reveals the existence of process mediators among the different locations of the system. We start with the definition of types.

Definition 4.1 (Types)**.** *Types are built from formulas in* $SELL^{\cap}$ *as follows:*

Pre-types	f,g	$::= 1 \mid \mathtt{p} \mid f \otimes g \mid f \multimap g \mid f \oplus g \mid f \,\&\, g \mid !^s f \mid \cap l_x : a.f \mid \cup l_x : a.f$
Location-sequences	w,w'	$::= s \mid (w,w') \mid w.w' \mid \lambda x.w$
Types	A,B	$::= \blacktriangledown[w](f)$

where s is a subexponential (constant or variable) and $\blacktriangledown[w](f)$ *is defined as:*

$$\blacktriangledown[s](1) = !^s 1 \qquad\qquad \blacktriangledown[s](\mathtt{p}) = !^s\mathtt{p}$$

$$\blacktriangledown[a.(w,w')](f \star g) = !^a(\blacktriangledown[w](f) \star \blacktriangledown[w'](g)) \qquad \star \in \{\otimes, \multimap, \oplus, \&\}$$

$$\blacktriangledown[a.w](!^c f) = !^a\blacktriangledown[c.w](f)$$

$$\blacktriangledown[a.(\lambda y.w)](\cap l_x : b.f) = !^a\cap l_x : b.\blacktriangledown[w\{l_x/y\}](f) \qquad \blacktriangledown[a.(\lambda y.w)](\cup l_x : b.f) = !^a\cup l_x : b.\blacktriangledown[w\{l_x/y\}](f)$$

Note that in valid types, all the sub-formulas (including the formula itself) are preceded with a bang. For instance, the (valid) types $A = !^a(!^b 1 \otimes !^c 1)$ and $B = !^a(!^b(!^c 1 \otimes !^d 1) \multimap !^e 1)$ can be written, respectively, as $A = \blacktriangledown[a.(b,c)](1 \otimes 1)$ and $B = \blacktriangledown[a.(b.(c,d),e)]((1 \otimes 1) \multimap 1)$. Such a definition keeps track of all the spaces (subexponentials) needed to adequately locate where a service is provided/offered and when processes need to migrate to continue the communication.

The case for quantifiers requires to bind variables occurring in w with the binding variable in $⋓$ or $⋒$. We write $\lambda y.w$ to bind occurrences of a variable y in w and $w\{l_x/y\}$ is the result of the application of $\lambda y.w$ to l_x. For instance, $A = !^a(⋒l_x : b.!^{l_x}(!^s(1) \otimes !^s(1))$ is written as $A = \blacktriangledown[a.\lambda y.(y.(s,s)))](⋒l_x : b(1 \otimes 1))$.

We shall write $\bigtriangledown_a f$ to represent the type $\blacktriangledown[a.w](f)$ where w can be inferred from the context (or from a typing derivation). As we already know, all types are marked with a bang and hence, with $\bigtriangledown_a f$, we signal the outermost bang and leave implicit the inner bangs, simplifying the notation.

Following [9], the multiplicative connectives $\otimes, \multimap$ correspond to input/output and the additive ones $\&, \oplus$ correspond to selection/choice. Here, from SELL$^⋒$, $\bigtriangledown_a$ (a formula preceded with $!^a$) represents a service offered from a site/location a. Moreover, bangs will allow us to migrate services to other locations. The quantifiers $⋒, ⋓$ will be used for the communication of locations.

Since all types (and subformulas) are marked with a subexponential, the linear context will be always empty. Hence, different from the system in Fig. 2, type judgments will take the form $\mathscr{A} : \Delta \vdash P :: T$ where: Δ is a set of *typed channels* of the form $x_a^b : \bigtriangledown_a(A)$ meaning "the service A is available from location a on channel x and it can be used/consumed from location b"; P is a process; and T is a typed channel. We read $P :: x_a^b : \bigtriangledown_a(A)$ as "P implements the service A on channel x at location a to be used by a processes located at b". Processes are considered modulo structural congruence hence, as in [7], typability is closed under $\equiv$ by definition. Channel names appear at most once in the sequent (either on the right or on the left). As we shall see, a process in a location a can only *implement* services on that location but it can consume/use services from any other location accessible from a. Similarly as we did in Fig. 2, we shall omit $\mathscr{A}$ when it is unimportant or it is clear from the context.

Subexponential signature. We shall gradually introduce the system rules, with its intended interpretation. The complete set of rules is in Fig. 4, App. B. We assume a set of linear subexponentials I_l with a preorder $\preceq_l$. Those indices represent the different locations of the system. We build the subexponential signature $\Sigma = (I, \preceq, U)$ where: $I = I_l \cup \{a^+ \mid a \in I_l\} \cup \{a \sqcap b \mid a, b \in I_l\}$ and $U = \{a^+ \mid a \in I_l\}$

The preorder $\preceq$ satisfies: for linear subexponentials, $a \preceq b$ iff $a \preceq_l b$. Moreover, $a \preceq a^+$; $a^+ \preceq b^+$ iff $a \preceq b$; $a \sqcap b = b \sqcap a$; $a \sqcap b \preceq a$; and $a = a \sqcap a$.

Intuitively, we start with a set of (linear) locations. Given one of such location a, the unbounded version of it (where replicated services in that location reside) is denoted as a^+ and $a \preceq a^+$. For each two linear locations a, b, the linear subexponential $a \sqcap b$ denotes a location that is below a and b.

In the following, we shall use Υ to denote a context containing only channel types with unbounded subexponentials, i.e., $x_{a^+}^b : \bigtriangledown_{a^+}(A)$ for some (linear) a. We shall use Δ, Δ' to denote a context with only linear subexponentials and Γ to denote a context with linear and unbounded subexponentials.

Unit. The rules for 1 are the following:

$$\frac{\Gamma \vdash P :: T}{\Gamma, x_a^b : \bigtriangledown_a(1) \vdash P :: T} \; 1_L \qquad \frac{}{\Upsilon \vdash [\mathbf{0}]_b :: x_b^a : \bigtriangledown_b(1)} \; 1_R$$

Note that in rule 1_R, all the formulas in Υ can be weakened. In both cases, $\bigtriangledown_a 1$ is intended to be $\blacktriangledown[a](1) = !^a 1$ (since the bang is already marking an atomic formula). Intuitively, the formula $!^b 1$ is the type of an ending process ($\mathbf{0}$) located at b.

Multiplicative Rules. The rules for $\otimes$ are:

$$\frac{\Gamma, x_a^b : \bigtriangledown_a(B), y_a^b : \bigtriangledown_a(A) \vdash [p]_b :: T}{\Gamma, x_a^b : \bigtriangledown_a(A \otimes B) \vdash [x(y).p]_b :: T} \; \otimes_L \qquad \frac{\Upsilon^{\succeq b}, \Delta^{\succeq b} \vdash [p]_b :: y_b^a : \bigtriangledown_b(A) \quad \Upsilon^{\succeq b}, \Delta'^{\succeq b} \vdash [q]_b :: x_b^a : \bigtriangledown_b(B)}{\Upsilon, \Delta, \Delta' \vdash [\bar{x}\langle y\rangle.(p \parallel q)]_b :: x_b^a : \bigtriangledown_b(A \otimes B)} \; \otimes_R$$

In the rule $\otimes_L$, the process $x(y).p$ is waiting for an input of type A from the location a. Hence, assuming that such a service A is available at the fresh channel y, the remaining process p should be

able to continue by consuming the service B at x (and A at y). In the rule $\otimes_R$, the output process located at b implements the service $A \otimes B$ if p is able to type the output y with the appropriate type A and the continuation q is able to implement the service B at x. Note that no service located in c s.t. $b \not\preceq c$ can be stored in the context (otherwise, $\Delta^{\succeq b}$ is not defined[2]). This means that a process cannot offer a service if it has not still processed all services requested from other (non-related) locations (see Ex. 4.7 below). This is an invariant and a design principle in our system: a pre-process p located at b can only type channels on the same location (right rules). For that, the context cannot have any unrelated resource (due to the promotion rule). On the other hand, left rules do not have any restriction on the context (similar to dereliction in linear logic). However, note that a process located at b is only able to consume a service of the form $x_a^b : \nabla_a(A)$ (offered in a).

We can see the rules of the type system as *macro rules* built on top of rules in Fig. 2 (when the term P and the labels for channels are erased):

$$\cfrac{\cfrac{\cfrac{\Gamma, \nabla_a A, \nabla_a B \;:\; \cdot \vdash G}{\Gamma \;:\; \nabla_a A, \nabla_a B \vdash G}\; store}{\Gamma \;:\; \blacktriangledown[a.w](A) \otimes \blacktriangledown[a.w'](B) \vdash G}\; \otimes_L}{\Gamma, \blacktriangledown[a.(a.w,a.w')](A \otimes B) \;:\; \cdot \vdash G}\;!_L \qquad \cfrac{\cfrac{\Upsilon^{\succeq b}, \Delta^{\succeq b} \;:\; \cdot \vdash \nabla_b A \quad \Upsilon^{\succeq b}, \Delta'^{\succeq b} \;:\; \cdot \vdash \nabla_b B}{\Upsilon^{\succeq b}, \Delta^{\succeq b}, \Delta'^{\succeq b} \;:\; \cdot \vdash \blacktriangledown[b.w](A) \otimes \blacktriangledown[b.w'](B)}\; \otimes_R}{\Upsilon, \Delta, \Delta' \;:\; \cdot \vdash \blacktriangledown[b.(b.w,b.w')](A \otimes B)}\;!_R$$

The rules for linear implication are dual to those for multiplicative conjunction. This means that an output process consumes types of the form $A \multimap B$ while input processes offer services of the same shape:

$$\cfrac{\Upsilon, \Delta \vdash [p]_b \;::\; y_b^a : \nabla_b(A) \quad \Upsilon, \Delta', x_a^b : \nabla_a(B) \vdash [q]_b \;::\; T}{\Upsilon, \Delta, \Delta', x_a^b : \nabla_a(A \multimap B) \vdash [\bar{x}\langle y \rangle.(p \parallel q)]_b \;::\; T}\; \multimap_L \qquad \cfrac{\Gamma^{\succeq b}, y_a^b : \nabla_a(A) \vdash [p]_b \;::\; x_b^a : \nabla_b(B)}{\Gamma \vdash [x(y).p]_b \;::\; x_b^a : \nabla_b(A \multimap B)}\; \multimap_R$$

From the rule $\multimap_L$, it is clear that $\nabla_a(A \multimap B)$ is a formula of the shape $\blacktriangledown[a.(b.w,a.w')](A \multimap B)$. In the rule $\multimap_R$, $\nabla_b(A \multimap B)$ is intended to mean a formula of the shape $\blacktriangledown[b.(a.w,b.w')](A \multimap B)$. The correspondence with the rules in Fig. 2 can be obtained similarly as we did for the rules of $\otimes$. It is worth noticing the annotations on channels. For instance, in $\multimap_R$, the input located at b receives on y some information coming from a. That justifies the annotation y_a^b. Dually, in $\multimap_L$, the output y must be typed by the process p at b, to be used by some process at location a (annotation y_b^a).

Additive rules. Similar to the first systems for session types based on linear logic [9], the additive rules give meaning to processes that offer and select different execution alternatives:

$$\cfrac{\Gamma, x_a^b : \nabla_a(A) \vdash [p]_b \;::\; T \quad \Gamma, x_a^b : \nabla_a(B) \vdash [q]_b \;::\; T}{\Gamma, x_a^b : \nabla_a(A \oplus B) \vdash [x.\mathtt{case}(p,q)]_b \;::\; T}\; \oplus_L$$

$$\cfrac{\Gamma^{\succeq b} \vdash [p]_b \;::\; x_b^a : \nabla_b(A)}{\Gamma \vdash [x.\mathtt{inl}.p]_b \;::\; x_b^a : \nabla_b(A \oplus B)}\; \oplus_{R1} \qquad \cfrac{\Gamma^{\succeq b} \vdash [p]_b \;::\; x_b^a : \nabla_b(B)}{\Gamma \vdash [x.\mathtt{inr}.p]_b \;::\; x_b^a : \nabla_b(A \oplus B)}\; \oplus_{R2}$$

$$\cfrac{\Gamma, x_a^b : \nabla_a(A) \vdash [p]_b \;::\; T}{\Gamma, x_a^b : \nabla_a(A \,\&\, B) \vdash [x.\mathtt{inl}.p]_b \;::\; T}\; \&_{L1} \qquad \cfrac{\Gamma, x_a^b : \nabla_a(B) \vdash [p]_b \;::\; T}{\Gamma, x_a^b : \nabla_a(A \,\&\, B) \vdash [x.\mathtt{inr}.p]_b \;::\; T}\; \&_{L2}$$

$$\cfrac{\Gamma^{\succeq b} \vdash [p]_b \;::\; x_b^a : \nabla_b(A) \quad \Gamma^{\succeq b} \vdash [q]_b \;::\; x_b^a : \nabla_b(B)}{\Gamma \vdash [x.\mathtt{case}(p,q)]_b \;::\; x_b^a : \nabla_b(A \,\&\, B)}\; \&_R$$

---

[2]In fact, if Δ contains only linear subexponentials and $\Delta^{\succeq b}$ is defined, then $\Delta = \Delta^{\succeq b}$ (this fact is important for the proof of Th. 4.3). We have decided to use $\Delta^{\succeq b}$ instead of Δ in order to make explicit the restriction imposed by the promotion rule.

The process $[x.\texttt{case}(p,q)]_b$ can make use of a behavior $A \oplus B$ offered at x on location a, if p is able to consume A and q is able to consume B. On the other side, the process $[x.\texttt{inl}.p]_b$ implements the behavior $A \oplus B$ by selecting A (which is implemented by p at b). The reasoning behind & is dual.

The subexponential rules. The following rules govern the migration of processes to other locations.

$$\frac{\Gamma, y_c^c : \bigtriangledown_c(A) \vdash [p]_c \;::\; T}{\Gamma, x_a^b : \bigtriangledown_a(!^c A) \vdash [x.\texttt{move}(y,c).p]_b \;::\; T} \; !_L \qquad \frac{\Gamma^{\succeq b} \vdash [p]_c \;::\; y_c^c : \bigtriangledown_c(A)}{\Gamma \vdash [\bar{x}.\texttt{move}\langle y,c\rangle.p]_b \;::\; x_b^a : \bigtriangledown_b(!^c A)} \; !_R$$

In both rules, the processes migrate to the location c and the communication continues on the fresh channel y. Note that y is "local" to location c: it is implemented and consumed inside the same location (annotation y_c^c). The corresponding derivations with the system in Fig. 2 are:

$$\frac{\dfrac{\dfrac{\Gamma, \bigtriangledown_c A \;:\; \cdot \vdash G}{\Gamma \;:\; \blacktriangledown[c.w](A) \vdash G} \; store}{\Gamma, !^a \blacktriangledown[c.w](A) \;:\; \cdot \vdash G} \; !_L}{\Gamma, \blacktriangledown[a.w](!^c A) \;:\; \cdot \vdash G} \; def \qquad \frac{\dfrac{\dfrac{\Gamma^{\succeq b} \;:\; \cdot \vdash \bigtriangledown_c A}{\Gamma \;:\; \cdot \vdash !^b \blacktriangledown[c \cdot w](A)} \; !_R}{\Gamma \;:\; \cdot \vdash \blacktriangledown[b.w](!^c A)} \; def}{}$$

The subexponential quantifiers. The quantifiers will be used to communicate locations/domains.

$$\frac{\mathscr{A}, \alpha \preceq a \sqcap b \;:\; \Gamma, x_a^b : \bigtriangledown_a(A) \vdash [p]_b \;::\; T}{\mathscr{A} \;:\; \Gamma, x_a^b : \bigtriangledown_a(\mathbb{U}_{\alpha:a\sqcap b}.A) \vdash [x(\alpha).p]_b \;::\; T} \; \mathbb{U}_L \qquad \frac{\Gamma^{\succeq b} \vdash [p]_b \;::\; x_b^a : \bigtriangledown_b(A\{l/\alpha\})}{\Gamma \vdash [x\langle l\rangle.p]_b \;::\; x_b^a : \bigtriangledown_b(\mathbb{U}_{\alpha:a\sqcap b}.A)} \; \mathbb{U}_R$$

In $\mathbb{U}_L$, α is a fresh subexponential of type $a \sqcap b$. Note that this rule extends the signature with the typing information $\alpha : a \sqcap b$. In $\mathbb{U}_R$, l must be of type $a \sqcap b$ (i.e., $l \preceq a \sqcap b$). Hence, the process $[x\langle l\rangle.p]_b$ implements the behavior of sending a specific location l, which is reachable from both a and b.

The rules for the universal quantifier are dual and omitted (see Fig. 4, App. B).

The unbounded subexponential rules. In [9], the LL exponential ! is used to specify replicated servers. In our setting, the type for a replicated service at location a will be marked with the subexponential $!^{a^+}$ (remember that $a^+ \in U$ is unbounded). A replicated service allows for using as many copies of the service as needed. This is reflected by the rules below:

$$\frac{\Gamma, x_{a^+}^b : \bigtriangledown_{a^+}(!^a A), y_a^b : \bigtriangledown_a(A) \vdash [p]_b \;::\; T}{\Gamma, x_{a^+}^b : \bigtriangledown_{a^+}(!^a A) \vdash [\bar{x}\langle y\rangle.p]_b \;::\; T} \; copy \qquad \frac{\Upsilon^{\succeq b^+} \vdash [p]_b \;::\; y_b^a : \bigtriangledown_b(A)}{\Upsilon \vdash [!x(y).p]_b \;::\; x_{b^+}^a : \bigtriangledown_{b^+}(!^b A)} \; !_R^+$$

In *copy*, an instance of A is replicated and offered on the fresh channel y (located at a). In the right rule, only unbounded resources ($\Upsilon^{\succeq b^+}$) can be used to show that p implements the behavior A at b.

As shown for $\otimes$ and !, the rules of the type system can be seen as macro rules of the system in Fig. 2:

Proposition 4.2. *The typing rules above are admissible in* SELL$^\sqcap$.

4.1 Process Reduction, Cut-Elimination and Identity Expansion

Consider the cut-rule below:

$$\frac{\Upsilon, \Delta \vdash P \;::\; x_a^b : \bigtriangledown_a(A) \qquad \Upsilon, \Delta', x_a^b : \bigtriangledown_a(A) \vdash Q \;::\; T}{\Upsilon, \Delta, \Delta' \vdash (\nu\, x)(P \parallel Q) \;::\; T} \; Cut$$

The following theorem shows that this rule is admissible and, more interestingly, it shows that proof reductions correspond to reductions in processes.

Theorem 4.3 (Cut-elimination). *The cut rule above is admissible.*

Proof. Let us present some of the principal cases.

$$
\cfrac{
\cfrac{
\cfrac{\Pi}{\Upsilon^{\succeq a},\Delta^{\succeq a},y_b^a : \bigtriangledown_b(A) \vdash [p]_a :: x_a^b : \bigtriangledown_a(B)}
}{\Upsilon,\Delta \vdash [x(y).p]_a :: x_a^b : \bigtriangledown_a(A \multimap B)} {\scriptstyle \multimap_R}
\qquad
\cfrac{\Upsilon,\Delta_1 \vdash [q]_b :: w_b^a : \bigtriangledown_b(A) \quad \Upsilon,\Delta_2,x_a^b : \bigtriangledown_a(B) \vdash [q']_b :: T}{\Upsilon,\Delta_1,\Delta_2,x_a^b : \bigtriangledown_a(A \multimap B) \vdash [\bar{x}\langle w\rangle.(q \parallel q')]_b :: T} {\scriptstyle \multimap_L}
}{\Upsilon,\Delta,\Delta_1,\Delta_2 \vdash (\nu\, x)([x(y).p]_a \parallel [\bar{x}\langle w\rangle.(q \parallel q')]_b) :: T} {\scriptstyle Cut}
$$

$$\rightsquigarrow$$

$$
\cfrac{
\Upsilon,\Delta_1 \vdash [q]_b :: w_b^a : \bigtriangledown_b(A)
\qquad
\cfrac{
\cfrac{\Pi'}{\Upsilon,\Delta,w_b^a : \bigtriangledown_b(A) \vdash [p\{w/y\}]_a :: x_a^b : \bigtriangledown_a(B)} \quad \Upsilon,\Delta_2,x_a^b : \bigtriangledown_a(B) \vdash [q']_b :: T
}{\Upsilon,\Delta,\Delta_2,w_b^a : \bigtriangledown_b(A) \vdash (\nu\, x)([p\{w/y\}]_a \parallel [q']_b) :: T} {\scriptstyle Cut}
}{\Upsilon,\Delta,\Delta_1,\Delta_2 \vdash (\nu\, w)([q]_b \parallel (\nu\, x)([p\{w/y\}]_a \parallel [q']_b)) :: T} {\scriptstyle Cut}
$$

Proof Π' can be obtained from Π by using weakening on Υ (and noticing that $\Delta = \Delta^{\succeq a}$ since $\Delta^{\succeq a}$ is defined) and a Substitution Lemma, showing that $\Pi\{w/y\}$ is indeed a proof. The case for $\otimes$ is analogous. Let us present also the case for migration:

$$
\cfrac{
\cfrac{
\cfrac{\Pi_p}{\Upsilon^{\succeq a},\Delta^{\succeq a} \vdash [p]_c :: w_c^c : \bigtriangledown_c(A)}
}{\Upsilon,\Delta \vdash [\bar{x}.\mathtt{move}\langle w,c\rangle.p]_a :: x_a^b : \bigtriangledown_a(!^c A)} {\scriptstyle !_R}
\qquad
\cfrac{
\cfrac{\Pi_q}{\Upsilon,\Delta',y_c^c : \bigtriangledown_c(A) \vdash [q]_c :: T}
}{\Upsilon,\Delta',x_a^b : \bigtriangledown_a(!^c A) \vdash [x.\mathtt{move}(y,c).q]_b :: T} {\scriptstyle !_L}
}{\Upsilon,\Delta,\Delta' \vdash (\nu\, x)([\bar{x}.\mathtt{move}\langle w,c\rangle.p]_a \parallel [x.\mathtt{move}(y,c).q]_b) :: T} {\scriptstyle Cut}
$$

$$\rightsquigarrow$$

$$
\cfrac{
\cfrac{\Pi'_p}{\Upsilon,\Delta \vdash [p]_c :: w_c^c : \bigtriangledown_c(A)}
\qquad
\cfrac{\Pi'_q}{\Upsilon,\Delta',w_c^c : \bigtriangledown_c(A) \vdash [q\{w/y\}]_c :: T}
}{\Upsilon,\Delta,\Delta' \vdash (\nu\, w)([p]_c \parallel [q\{w/y\}]_c) :: T} {\scriptstyle Cut}
$$

where Π'_p is obtained from Π_p by weakening and Π'_q from Π_q by substitution. $\qquad\square$

Corollary 4.4 (Type Preservation). *If $\mathscr{A} : \Gamma \vdash P :: T$ and $P \to Q$ then $\mathscr{A} : \Gamma \vdash Q :: T$.*

Proof. By induction on $P \to Q$. Since P is typed, the unique case is for P of the form $(\nu\, x)(P_1 \parallel P_2)$ where $P_1 \parallel P_2 \to_r Q'$, $r \in \{[\mathtt{Comm}],[\mathtt{LocOut}],[\mathtt{Move}],[\mathtt{LCase}],[\mathtt{RCase}]\}$ and $Q = (\nu\, x)Q'$. Therefore, the last step in a typing derivation of $\mathscr{A} : \Gamma \vdash P :: T$ is the (Cut) rule and the result holds by the Cut Elimination Theorem 4.3. $\qquad\square$

Another corollary from Theorem 4.1 is a global progress property.

Corollary 4.5 (Global Progress). *If $\mathscr{A} : \cdot \vdash P :: x_c^a : \bigtriangledown_c(1)$ then either $P \equiv [\mathbf{0}]_c$ or exists Q s.t. $P \to Q$.*

Note that, if $P \not\equiv [\mathbf{0}]_c$ then, by a simple inspection of the typing rules (see Fig. 4, App. B), (Cut) is the only possibility as the last rule application holding such a typing.

4.2   Identity Expansion

The copycat rule extends the one in [36] with locations:

$$\frac{}{\Gamma, x_b^a : \bigtriangledown_b(A) \vdash [x \overset{b}{\underset{a}{\leftrightarrow}} y]_a \; :: \; y_a^b : \bigtriangledown_a(A)} \; Id \; .$$

As shown in [9], the identity expansion theorem reveals the existence of synchronous mediators of the form $x \leftrightarrow y$, a copycat process that plays the role of a link between two channels [33]. In our system, the same procedure reveals the existence of mediators between different locations. This means that such processes implement the communication between the endpoints located at different sites. Remember that $x \overset{b}{\underset{a}{\leftrightarrow}} y$ is a process that links the channel x on location b with the channel y on location a.

Theorem 4.6 (Identity expansion)**.** *For all type A and locations $a \preceq b$, there exists a process $id(x,y)$ and a cut-free derivation of the judgment $\mathscr{A} \; : \; x_b^a : \bigtriangledown_b(A) \vdash [id(x,y)]_a \; :: \; y_a^b : \bigtriangledown_a(A)$*

Proof. By induction on the structure of A. For $A = 1$, $id(x,y) = 0$ and for an atomic proposition, $id(x,y) = x \overset{b}{\underset{a}{\leftrightarrow}} y$. Consider the case $A \multimap B$:

$$\frac{z_b^a : \bigtriangledown_b(A) \vdash [id(z,w)]_a \; :: \; w_a^b : \bigtriangledown_a(A) \quad x_b^a : \bigtriangledown_b(B) \vdash [id(x,y)]_a \; :: \; y_a^b : \bigtriangledown_a(B)}{\dfrac{x_b^a : \bigtriangledown_b(A \multimap B), z_b^a : \bigtriangledown_b(A) \vdash [\bar{x}\langle w \rangle.P]_a \; :: \; y_a^b : \bigtriangledown_a(B)}{x_b^a : \bigtriangledown_b(A \multimap B) \vdash [y(z).\bar{x}\langle w \rangle.P]_a \; :: \; y_a^b : \bigtriangledown_a(A \multimap B)} \; \multimap R} \; \multimap L$$

Here, $P = (id(z,w) \parallel id(x,y))$. What we observe is that the channels x and z located at b are linked, respectively, to the channels y and w located at a. The other cases can be obtained similarly.   □

Applications. We conclude this section by describing some interesting behaviors that can be specified and verified with our system.

Example 4.7 (Storing Values)**.** *Process P from Examples 3.2 and 3.3 is not typable whenever $c \not\succeq sv$. Let $A = 1 \otimes 1$, then a type derivation for P is of the form*

$$\frac{\overset{\Pi}{\Upsilon, \Delta \vdash [x(y).\bar{x}\langle z \rangle.y(k).\mathbf{0}]_{sv} \; :: \; x_{sv}^c : \bigtriangledown_{sv}(A \multimap A)} \quad \overset{\Psi}{\Upsilon, x_{sv}^c : \bigtriangledown_{sv}(A \multimap A) \vdash [\bar{x}\langle y \rangle.(\bar{y}\langle k \rangle.\mathbf{0} \parallel x(z).\mathbf{0})]_c \; :: \; T}}{\Upsilon, \Delta \vdash (\nu\, x)([x(y).\bar{x}\langle z \rangle.y(k).\mathbf{0}]_{sv} \parallel [\bar{x}\langle y \rangle.(\bar{y}\langle k \rangle.\mathbf{0} \parallel x(z).\mathbf{0})]_c) \; :: \; T} \; Cut$$

where Π is of the form

$$\frac{\Upsilon^{\succeq sv}, \Delta^{\succeq sv} \vdash [\mathbf{0}]_{sv} \; :: \; z_{sv}^c : \bigtriangledown_{sv}(1) \quad \dfrac{\dfrac{\dfrac{\dfrac{\Upsilon^{\succeq sv} \vdash [\mathbf{0}]_{sv} \; :: \; x_{sv}^c : \bigtriangledown_{sv}(1)}{\Upsilon^{\succeq sv}, y_c^{sv} : \bigtriangledown_c(1) \vdash [\mathbf{0}]_{sv} \; :: \; x_{sv}^c : \bigtriangledown_{sv}(1)} \; 1R}{\Upsilon^{\succeq sv}, y_c^{sv} : \bigtriangledown_c(1), k_c^{sv} : \bigtriangledown_c(1) \vdash [\mathbf{0}]_{sv} \; :: \; x_{sv}^c : \bigtriangledown_{sv}(1)} \; 1L}{\Upsilon^{\succeq sv}, y_c^{sv} : \bigtriangledown_c(A) \vdash [y(k).\mathbf{0}]_{sv} \; :: \; x_{sv}^c : \bigtriangledown_{sv}(1)} \; 1L}{\Upsilon^{\succeq sv}, \Delta^{\succeq sv}, y_c^{sv} : \bigtriangledown_c(A) \vdash [\bar{x}\langle z \rangle.y(k).\mathbf{0}]_{sv} \; :: \; x_{sv}^c : \bigtriangledown_{sv}(A)} \; \otimes R}{\Upsilon, \Delta \vdash [x(y).\bar{x}\langle z \rangle.y(k).\mathbf{0}]_{sv} \; :: \; x_{sv}^c : \bigtriangledown_{sv}(A \multimap A)} \; \multimap R$$

Note that the application of $\otimes R$ is valid only if $c \succeq sv$. Intuitively, the permission of a value sent by the client to be stored and "used later" by the server –in this example, the channel name y used after the output $x\langle z \rangle$– needs to be explicit in the subexponential signature.

In general, communicated values are either atomic data, of type 1 and its consumption represented by the application of rule 1_L; or data to be used in further interactions. In the first case, the use of 1_L does not impose any constraint on either location. In the second case, the subexponential signature must indicate that such further interactions are allowed, as illustrated above.

Example 4.8. *We present an adaptation of the example introduced in [7], representing a client-server communication defining a buying protocol where sensitive information, such as a credit card number, is communicated only after an agreed secure channel is established. In other words, we want to type* $Sys := (\nu\, x)([W]_{ws} \parallel [C]_c)$ *such that*

$$\frac{\overset{\Pi}{\Upsilon \vdash [W]_{ws} \; :: \; x_{ws}^c : \bigtriangledown_{ws}(WT)} \qquad \overset{\Psi}{\Upsilon, x_{ws}^c : \bigtriangledown_{ws}(WT) \vdash [C]_c \; :: \; z_c^{ws} : \bigtriangledown_c(1)}}{\Upsilon \vdash (\nu\, x)([W]_{ws} \parallel [C]_c) \; :: \; z_c^{ws} : \bigtriangledown_c(1)} \; Cut$$

for (pre-)types $WT \; := \; addC \multimap (\mathbb{U}_{\alpha:sec}.!^{\alpha}Pay)\,\&\,1 \qquad Pay \; := \; CCNum \multimap (\cap_{\beta:\alpha}.!^{\beta}Rcpt) \oplus 1$

Here, addC denotes an item information added by the client in the online chart and CCNum denotes the credit card number sent by the client. The intended meaning of WT is that, once the item information is processed, the client can choose to either close the session or go to payment. The latter depends on the migration of both, client and server, to a secure location (l_s below) chosen by the server but agreed by both sides of the communication. Note that the credit card number would be sent only if the (channel) migration succeeds. In the end, after receiving CCNum, the processes migrate to yet another location (l_b) to complete the payment. Hence, we consider the following set of locations: $I = \{c \prec ws, sec \prec ws, sec \prec c, l_s \prec sec, l_b \prec l_s\}$ and

$$
\begin{aligned}
C &:= \bar{x}\langle cat\rangle.x.\mathtt{inl}.x(\alpha).C_1 & W &:= x(cat).x.\mathtt{case}(x\langle l_s\rangle.W_1, \mathbf{0}) \\
C_1 &:= x.\mathtt{move}(x', \alpha).\overline{x'}\langle n\rangle.C_2 & W_1 &:= \bar{x}.\mathtt{move}\langle x', l_s\rangle.x'(n).W_2 \\
C_2 &:= x'.\mathtt{case}(x'\langle l_b\rangle.x'.\mathtt{move}(y, l_b).\mathbf{0}, \mathbf{0}) & W_2 &:= x'.\mathtt{inl}.x'(\beta).\overline{x'}.\mathtt{move}\langle y, \beta\rangle.\mathbf{0}
\end{aligned}
$$

Here, sec *is used as the type of locations known by both client and server since* $sec \prec c, sec \prec ws$. l_s *is the location of type* sec ($l_s \prec sec$) *to be used by the server. Moreover,* l_b *(of type l_s) is the location chosen by the client to complete the payment.*

Derivation Π takes the form

$$\cfrac{\cfrac{\cfrac{\cfrac{\Upsilon^{\succeq ws} \vdash [W_1']_{l_s} \; :: \; x_{l_s}^{\prime l_s} : \bigtriangledown_{l_s}(Pay\{l_s/\alpha\})}{\Upsilon^{\succeq ws} \vdash [W_1]_{ws} \; :: \; x_{ws}^c : \bigtriangledown_{ws}(!^{l_s}Pay\{l_s/\alpha\})} \; !_R}{\Upsilon^{\succeq ws} \vdash [x\langle l_s\rangle.W_1]_{ws} \; :: \; x_{ws}^c : \bigtriangledown_{ws}(\mathbb{U}_{\alpha:sec}.!^{\alpha}Pay)} \; \mathbb{U}_R \qquad \cfrac{\Upsilon^{\succeq ws} \vdash [\mathbf{0}]_{ws} \; :: \; x_{ws}^c : \bigtriangledown_{ws}(1)}{} \; 1_R}{\cfrac{\cfrac{\Upsilon^{\succeq ws} \vdash [W']_{ws} \; :: \; x_{ws}^c : \bigtriangledown_{ws}(WT')}{\Upsilon^{\succeq ws}, cat_c^{ws} : \bigtriangledown_c(addC) \vdash [W']_{ws} \; :: \; x_{ws}^c : \bigtriangledown_{ws}(WT')} \; 1_L}{\Upsilon \vdash [W]_{ws} \; :: \; x_{ws}^c : \bigtriangledown_{ws}(WT)} \; \multimap_R} \; \&_R}$$

Note that $cat_c^{ws} : \bigtriangledown_c(addC)$ *needs to be consumed before the choice (rule $\&_R$).*

Derivation Π' is as follows:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\beta \preceq l_s \; : \; (\Upsilon^{\succeq ws})^{\succeq l_s} \vdash [\mathbf{0}]_{\beta} \; :: \; y_{\beta}^{\beta} : \bigtriangledown_{\beta}(Rcpt)}{\beta \preceq l_s \; : \; (\Upsilon^{\succeq ws})^{\succeq l_s}, n_{l_s}^{l_s} : \bigtriangledown_{l_s}(CCNum) \vdash [\mathbf{0}]_{\beta} \; :: \; y_{\beta}^{\beta} : \bigtriangledown_{\beta}(Rcpt)} \; 1_L} \; 1_R}{\beta \preceq l_s \; : \; (\Upsilon^{\succeq ws})^{\succeq l_s}, n_{l_s}^{l_s} : \bigtriangledown_{l_s}(CCNum) \vdash [\overline{x'}.\mathtt{move}\langle y, \beta\rangle.\mathbf{0}]_{l_s} \; :: \; x_{l_s}^{\prime l_s} : \bigtriangledown_{l_s}(!^{\beta}Rcpt)} \; !_R}{\cfrac{(\Upsilon^{\succeq ws})^{\succeq l_s}, n_{l_s}^{l_s} : \bigtriangledown_{l_s}(CCNum) \vdash [W_2']_{l_s} \; :: \; x_{l_s}^{\prime l_s} : \bigtriangledown_{l_s}(\cap_{\beta:l_s}.!^{\beta}Rcpt)}{(\Upsilon^{\succeq ws})^{\succeq l_s}, n_{l_s}^{l_s} : \bigtriangledown_{l_s}(CCNum) \vdash [W_2]_{l_s} \; :: \; x_{l_s}^{\prime l_s} : \bigtriangledown_{l_s}(Pay'\{l_s/\alpha\})} \; \oplus_{R1}} \; \cap_R}{\Upsilon^{\succeq ws} \vdash [W_1']_{l_s} \; :: \; x_{l_s}^{\prime l_s} : \bigtriangledown_{l_s}(Pay\{l_s/\alpha\})} \; \multimap_R$$

5   Related Work and Concluding Remarks

This paper originates from the work of Caires et al. [7] where Hybrid Linear Logic (HyLL) was proposed as the foundation for domain aware session types. Let us compare the two systems from the point of view of the underlying logic and the resulting process calculi.

Logical foundation.   Hybrid Linear Logic (HyLL) is a conservative extension of intuitionistic LL where the truth judgments are labeled by worlds as in $A@w$, read as "A holds at world w". The reachability relation on worlds, $\prec: W \times W$, determines when a world w is accessible from v ($v \prec w$).

Formulas in HyLL are built from the connectives of intuitionistic linear logic plus the following ones:
$$F ::= ... \mid (A \text{ at } w) \mid \downarrow w.A \mid \forall w.A \mid \exists w.A$$
where the world w is bounded in the formulas $\downarrow w.A$, $\forall w.A$ and $\exists w.A$.

The sequent calculus for HyLL [15] uses dyadic sequents of the form $\Gamma; \Delta \vdash C @ w$ where the unbounded (Γ) and linear (Δ) contexts contain judgments of the form $A @ w$. Let us introduce some of the rules as originally proposed in [15]. The initial rule enforces that atomic propositions must be located at the same world. The other rules are similar to those of intuitionistic LL:

$$\frac{}{\Gamma; \mathtt{p} @ w \vdash \mathtt{p} @ w} \; I \qquad \frac{\Gamma; \Delta \vdash A @ w \quad \Gamma; \Delta' \vdash B @ w}{\Gamma; \Delta, \Delta' \vdash A \otimes B @ w} \; \otimes R \qquad \frac{\Gamma; \Delta, A @ u, B @ u \vdash C @ w}{\Gamma; \Delta, A \otimes B @ u \vdash C @ w} \; \otimes L$$

The interesting part is the rules for the hybrid connectives:

$$\frac{\Gamma; \Delta \vdash A @ u}{\Gamma; \Delta \vdash (A \text{ at } u) @ w} \; \text{at } R \qquad \frac{\Gamma; \Delta, A @ u \vdash C @ v}{\Gamma; \Delta, (A \text{ at } u) @ w \vdash C @ v} \; \text{at } L \qquad \frac{\Gamma; \Delta \vdash A[w/u] @ w}{\Gamma; \Delta \vdash \downarrow u.A @ w} \; \downarrow R \qquad \frac{\Gamma; \Delta, A[w/u] @ w \vdash C @ v}{\Gamma; \Delta, \downarrow u.A @ w \vdash C @ v} \; \downarrow L$$

$$\frac{\Gamma; \Delta \vdash A @ w}{\Gamma; \Delta \vdash \forall \alpha. A @ w} \; \forall R \qquad \frac{\Gamma; \Delta, A[\tau/\alpha] @ u \vdash C @ w}{\Gamma; \Delta, \forall \alpha. A @ u \vdash C @ w} \; \forall L$$

The formula $(A \text{ at } u)$ is a *mobile* proposition: it carries the world at which it is true. The *localization* ($\downarrow$) binds a name for the current world where the proposition is true. Hence, the formula $(\downarrow u.A) @ w$ *fixes* the world w in A by substituting the occurrences of u with w in A. The rules for the universal quantifier are as expected. The rules for existential quantification on worlds are dual and omitted.

Note that the rules for the hybrid connectives are not constrained by the context $\Gamma; \Delta$. More precisely, the rule $\mathtt{at}_R$ is able to move the formula A from w to u without checking if u is reachable from the worlds in the context $\Gamma; \Delta$ (recall that $\text{SELL}^{\Cap}$ checks such condition in the promotion rule). Similarly for the quantifiers: $\forall_L$ substitutes α with *any* τ (related or not with u, w, etc). This observation was fundamental in the expressiveness study in [11], where it is shown that the inferences rules of HyLL can be adequately encoded in (vanilla) linear logic. Moreover, a second encoding of HyLL into $\text{SELL}^{\Cap}$ showed that a flat subexponential structure is enough to encode any world structure in HyLL. This is partially explained by the fact that the reachability relation on worlds does not play any role in HyLL's inference rules.

With the aim of controlling migration of formulas between worlds, [7] adds some side conditions to the system for HyLL. For instance, the rules for the connective $\mathtt{at}$ are extended as follows:

- $\mathtt{at}R$. Two side conditions are added: $w \prec u$ and $u \prec^* \Delta$. The second conditions uses the reflexive and transitive closure of $\prec$ and tells us that for any formula $G @ x$ in the context Δ, it must be the case that $w \prec^* x$.

- $\mathtt{at}L$. No side condition is added but, bottom up, the rule extends $\prec$ with the new entry $w \prec u$.

These side conditions (and others imposed in the rest of the rules) guarantee that the system produces only sequents of the form $\Gamma; \Delta \vdash A @ w$ where all the worlds in Δ are reachable from w. Note also that,

due to the new restriction on $\mathtt{at}R$, the movement of formulas on the right side of the sequent can take place only to an *immediate successor* world ($w \prec u$).

Processes. The language of processes considered in [7] does not make explicit the location of an agent. In fact, what we call a pre-process here, is a process in [7]. This implies that, when a process migrates, the information about its location cannot be determined by simply inspecting the terms/processes. (We are not aware whether such information can be recovered somehow in the system in [7]). For instance, consider the following derivations in both systems:

$$
\begin{array}{lllll}
\text{System in [7]:} & x.\overline{\mathtt{move}}\langle y, w \rangle.p & \parallel & x.\mathtt{move}(z, w).q & \longrightarrow & p \parallel q\{y/z\} \\
\text{Our system:} & [\overline{x}.\mathtt{move}\langle y, w \rangle.p]_a & \parallel & [x.\mathtt{move}(z, w).q]_b & \longrightarrow & [p \parallel q\{y/z\}]_w
\end{array}
$$

Type System. Modifying the rules of the system for HyLL allows for controlling the context when formulas are moved between worlds. However, extending the signature in $\mathtt{at}L$ is problematic. The cut-elimination procedure reveals that such extension is not necessary but the identity-expansion theorem requires it. In Appendix A, we elaborate more about this potential problem in the interpretation proposed in [7] of locations as worlds in HyLL.

Our type system is founded on a logic where the promotion rule controls the resources that can be used during a proof. Moreover, consuming resources from the context, i.e., performing dereliction, is not subject to any restriction. The act of communicating locations in our proposal seems also to be more natural: a communicated location is accessible from both endpoints of the communication (notation $a \sqcap b$). As shown in Section 4.2, our type system allows for controlling the information that can be stored in a "non-trusted" (i.e., non-related) site and also, to correctly reproduce the example in [7]. Finally, as stated before, the syntax of processes reflects more precisely the intention of the modeler.

As shown in different works (see e.g., [11, 29, 30, 32]), the subexponentials in *SELL* can be interpreted in different ways. For instance, they can be used to denote time-units, spaces of computation, the epistemic state of agents, preferences, costs, etc. It would be interesting to see if the interpretation proposed here can be seen through the lenses of those modalities. For instance, instead of interpreting $[p]_a$ as "p located at a", we may interpret a as a deadline for the execution of p (or "p can use a resource in the following a time-units"). This may lead to a system with session duration/expiration with strong foundation in logical methods.

References

[1] Samson Abramsky (1993): *Computational Interpretations of Linear Logic*. Theor. Comput. Sci. 111(1&2), pp. 3–57, doi:10.1016/0304-3975(93)90181-R.

[2] Samson Abramsky (1994): *Proofs as Processes*. Theor. Comput. Sci. 135(1), pp. 5–9, doi:10.1016/0304-3975(94)00103-0.

[3] Jean-Marc Andreoli (1992): *Logic Programming with Focusing Proofs in Linear Logic*. J. Log. Comput. 2(3), pp. 297–347, doi:10.1093/logcom/2.3.297.

[4] P. N. Benton, Gavin M. Bierman, Valeria de Paiva & Martin Hyland (1993): *A Term Calculus for Intuitionistic Linear Logic*. In: TLCA, Lecture Notes in Computer Science 664, Springer, pp. 75–90, doi:10.1007/BFb0037099.

[5] Michele Boreale (1998): *On the Expressiveness of Internal Mobility in Name-Passing Calculi*. Theor. Comput. Sci. 195(2), pp. 205–226, doi:10.1016/S0304-3975(97)00220-X.

[6] Luís Caires & Jorge A. Pérez (2017): *Linearity, Control Effects, and Behavioral Types*. In: *ESOP*, *Lecture Notes in Computer Science* 10201, Springer, pp. 229–259, doi:10.1007/978-3-662-54434-1_9.

[7] Luís Caires, Jorge A. Pérez, Frank Pfenning & Bernardo Toninho (2019): *Domain-Aware Session Types*. In: *CONCUR*, *LIPIcs* 140, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 39:1–39:17, doi:10.4230/LIPIcs.CONCUR.2019.39.

[8] Luís Caires & Frank Pfenning (2010): *Session Types as Intuitionistic Linear Propositions*. In Paul Gastin & François Laroussinie, editors: *CONCUR 2010*, *LNCS* 6269, Springer, pp. 222–236, doi:10.1007/978-3-642-15375-4_16.

[9] Luís Caires, Frank Pfenning & Bernardo Toninho (2016): *Linear logic propositions as session types*. *Math. Struct. Comput. Sci.* 26(3), pp. 367–423, doi:10.1017/S0960129514000218.

[10] Sara Capecchi, Ilaria Castellani & Mariangiola Dezani-Ciancaglini (2016): *Information flow safety in multiparty sessions*. *Math. Struct. Comput. Sci.* 26(8), pp. 1352–1394, doi:10.1017/S0960129514000619.

[11] Kaustuv Chaudhuri, Joëlle Despeyroux, Carlos Olarte & Elaine Pimentel (2019): *Hybrid linear logic, revisited*. *Math. Struct. Comput. Sci.* 29(8), pp. 1151–1176, doi:10.1017/S0960129518000439.

[12] Mario Coppo, Mariangiola Dezani-Ciancaglini, Luca Padovani & Nobuko Yoshida (2015): *A Gentle Introduction to Multiparty Asynchronous Session Types*. In: *SFM*, *LNCS* 9104, Springer, pp. 146–178, doi:10.1007/978-3-319-18941-3_4.

[13] Vincent Danos, Jean-Baptiste Joinet & Harold Schellinx (1993): *The Structure of Exponentials: Uncovering the Dynamics of Linear Logic Proofs*. In: *Kurt Gödel Colloquium*, *Lecture Notes in Computer Science* 713, Springer, pp. 159–171, doi:10.1007/BFb0022564.

[14] Rowan Davies & Frank Pfenning (2001): *A modal analysis of staged computation*. *J. ACM* 48(3), pp. 555–604, doi:10.1145/382780.382785.

[15] Joëlle Despeyroux & Kaustuv Chaudhuri (2013): *A Hybrid Linear Logic for Constrained Transition Systems*. In: *TYPES*, *LIPIcs* 26, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 150–168, doi:10.4230/LIPIcs.TYPES.2013.150.

[16] Henry DeYoung, Luís Caires, Frank Pfenning & Bernardo Toninho (2012): *Cut Reduction in Linear Logic as Asynchronous Session-Typed Communication*. In: *CSL*, *LIPIcs* 16, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 228–242, doi:10.4230/LIPIcs.CSL.2012.228.

[17] Mariangiola Dezani-Ciancaglini & Ugo de'Liguoro (2009): *Sessions and Session Types: An Overview*. In: *WS-FM*, *Lecture Notes in Computer Science* 6194, Springer, pp. 1–28, doi:10.1007/978-3-642-14458-5_1.

[18] Jean-Yves Girard (1987): *Linear Logic*. *Theoretical Computer Science* 50, pp. 1–102, doi:10.1016/0304-3975(87)90045-4.

[19] Timothy Griffin (1990): *A Formulae-as-Types Notion of Control*. In: *POPL*, ACM Press, pp. 47–58, doi:10.1145/96709.96714.

[20] Kohei Honda, Vasco Thudichum Vasconcelos & Makoto Kubo (1998): *Language Primitives and Type Discipline for Structured Communication-Based Programming*. In: *ESOP*, *Lecture Notes in Computer Science* 1381, Springer, pp. 122–138, doi:10.1007/BFb0053567.

[21] Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luís Caires, Marco Carbone, Pierre-Malo Deniélou, Dimitris Mostrous, Luca Padovani, António Ravara, Emilio Tuosto, Hugo Torres Vieira & Gianluigi Zavattaro (2016): *Foundations of Session Types and Behavioural Contracts*. *ACM Comput. Surv.* 49(1), pp. 3:1–3:36, doi:10.1145/2873052.

[22] Delia Kesner & Pierre Vial (2020): *Non-idempotent types for classical calculi in natural deduction style*. *Log. Methods Comput. Sci.* 16(1), doi:10.23638/LMCS-16(1:3)2020.

[23] Björn Lellmann, Carlos Olarte & Elaine Pimentel (2017): *A uniform framework for substructural logics with modalities*. In: *LPAR*, *EPiC Series in Computing* 46, EasyChair, pp. 435–455, doi:10.29007/93qg.

[24] Dale Miller, Gopalan Nadathur & Andre Scedrov (1987): *Hereditary Harrop Formulas and Uniform Proof Systems*. In: *LICS*, IEEE Computer Society, pp. 98–105.

[25] Robin Milner, Joachim Parrow & David Walker (1992): *A Calculus of Mobile Processes, I*. *Inf. Comput.* 100(1), pp. 1–40, doi:10.1016/0890-5401(92)90008-4.

[26] Aleksandar Nanevski, Frank Pfenning & Brigitte Pientka (2008): *Contextual modal type theory*. *ACM Trans. Comput. Log.* 9(3), pp. 23:1–23:49, doi:10.1145/1352582.1352591.

[27] Daniele Nantes & Jorge A. Pérez (2018): *Relating Process Languages for Security and Communication Correctness (Extended Abstract)*. In: *FORTE, Lecture Notes in Computer Science* 10854, Springer, pp. 79–100, doi:10.1007/978-3-319-92612-4_5.

[28] Vivek Nigam & Dale Miller (2009): *Algorithmic specifications in linear logic with subexponentials*. In: *PPDP*, ACM, pp. 129–140, doi:10.1145/1599410.1599427.

[29] Vivek Nigam, Carlos Olarte & Elaine Pimentel (2017): *On subexponentials, focusing and modalities in concurrent systems*. *Theor. Comput. Sci.* 693, pp. 35–58, doi:10.1016/j.tcs.2017.06.009.

[30] Carlos Olarte, Elaine Pimentel & Vivek Nigam (2015): *Subexponential concurrent constraint programming*. *Theor. Comput. Sci.* 606, pp. 98–120, doi:10.1016/j.tcs.2015.06.031.

[31] Michel Parigot (1992): *Lambda-Mu-Calculus: An Algorithmic Interpretation of Classical Natural Deduction*. In: *LPAR, Lecture Notes in Computer Science* 624, Springer, pp. 190–201, doi:10.1007/BFb0013061.

[32] Elaine Pimentel, Carlos Olarte & Vivek Nigam (2014): *A Proof Theoretic Study of Soft Concurrent Constraint Programming*. *Theory Pract. Log. Program.* 14(4-5), pp. 649–663, doi:10.1017/S147106841400026X.

[33] Davide Sangiorgi (1996): *pi-Calculus, Internal Mobility, and Agent-Passing Calculi*. *Theor. Comput. Sci.* 167(1&2), pp. 235–274, doi:10.1016/0304-3975(96)00075-8.

[34] Morten Heine B. Sørensen & Pawel Urzyczyn (2006): *Lectures on the Curry-Howard Isomorphism*. *Studies in Logic and the Foundations of Mathematics* 149, Elsevier Science.

[35] Kaku Takeuchi, Kohei Honda & Makoto Kubo (1994): *An Interaction-based Language and its Typing System*. In: *PARLE, LNCS* 817, Springer, pp. 398–413, doi:10.1007/3-540-58184-7_118.

[36] Bernardo Toninho, Luís Caires & Frank Pfenning (2012): *Functions as Session-Typed Processes*. In Lars Birkedal, editor: *FOSSACS 2012, LNCS* 7213, Springer, pp. 346–360, doi:10.1007/978-3-642-28729-9_23.

[37] Anne S. Troelstra (1992): *Lectures on Linear Logic*. CSLI Lecture Notes 29, Center for the Study of Language and Information, Stanford, California.

A   Locations as HyLL worlds

Let us consider the following sequent $\Gamma; x : (A \text{ at } \text{Sec})@w \vdash C :: z : T@c$ in [7, Section 3], where w means *web store* and c stands for *client*. It is assumed in [7] that the world signature, at this point, only contains the entry $c \prec w$. Since $c \not\prec^* \text{Sec}$, first the server has to migrate to the secure location Sec (using $\text{at}L$) and, only after that, the client is able to use a service from that location. According to op. cite, "*this ensures, e.g., that a client cannot exploit the payment platform of the web store by accessing the trusted domain in unforeseen ways*". This behavior is not reflected by the cut-elimination procedure as the following proposition shows:

Proposition A.1. *Let* $C = x.\text{move}(y, \text{Sec}).\mathbf{0}$, $W = x.\overline{move}\langle y, \text{Sec}\rangle.\mathbf{0}$ *and* $S = (\nu\, x)(W \parallel C)$. *Hence, the sequent* $\cdot; \cdot \vdash S :: - : 1@c$ *is provable iff* $c \prec w$, $c \prec \text{Sec}$ *and* $w \prec \text{Sec}$.

Proof. We shall prove something more general. Since the only way of proceeding in the typing of S is to use the rule cut (as proposed in [7]), we shall leave some meta-variables (V_i) in such a derivation to find the restrictions needed to complete the proof. Also, we shall use Ω to denote the context proving statements of the form $v \prec w$:

$$
\dfrac{
\dfrac{\Omega; \cdot; \cdot \vdash W' :: y : T'_w @\, \text{Sec}}{\Omega; \cdot; \cdot \vdash W :: x : (T'_w \text{ at } \text{Sec})@V_2} \text{at}R
\quad
\dfrac{\Omega, V_2 \prec \text{Sec}; \cdot; y : T'_w @\, \text{Sec} \vdash C' :: - : 1@V_1}{\Omega; \cdot; x : (T'_w \text{at } \text{Sec})@V_2 \vdash C :: - : 1@V_1} \text{at}L
}{
\Omega; \cdot; \cdot \vdash (\nu\, x)(W \parallel C) :: - : 1@V_1
} \text{Cut}
$$

According to the rules in [7], the following must hold:

$$\Omega \vdash V_1 \prec^* V_2 \qquad\qquad \text{due to the cut-rule}$$
$$\Omega \vdash V_2 \prec \text{Sec} \qquad\qquad \text{due to } \text{at}R$$

Hence, the addition of "$V_2 \prec \text{Sec}$" in $\text{at}L$ is irrelevant (since it must be already known in Ω).

Without that extension, the initial-expansion theorem does not work (omitting the terms/processes),

$$
\dfrac{
\dfrac{\Omega, V_1 \prec V_2; \cdot; A@V_2 \vdash A@V_2}{\Omega, V_1 \prec V_2; \cdot; A@V_2 \vdash (A\text{at}V_2)@V_1} \text{at}R
}{
\Omega; \cdot; (A\text{at}V_2)@V_1 \vdash (A\text{at}V_2)@V_1
} \text{at}L
$$

since $\text{at}R$ requires that $V_1 \prec V_2$. $\square$

This proposition is telling us that: (1) there is no need to extend the world signature in $\text{at}L$ since, the "new" relation $w \prec \text{Sec}$ must be already known (otherwise, the process S is not typable). And (2), adding $w \prec \text{Sec}$ is not enough to guarantee that the client can move to Sec: from $c \prec w$ and $w \prec \text{Sec}$ we conclude $c \prec^* \text{Sec}$ but, the side condition in $\text{at}R$ requires $c \prec \text{Sec}$. This explains the signature Ω ($c \prec w$, $c \prec \text{Sec}$ and $w \prec \text{Sec}$) in the above proposition.

According to [7], the extension of the signature in $\text{at}L$ was added in order to prove the identity expansion theorem (as shown above). However, the price to be paid is that the cut-elimination procedure does not reflect any more the intended meaning in the example above.

B Type system

In Fig. 4 we show the complete typing system.

$$\frac{}{\Gamma, x_b^a : \nabla_b(A) \vdash [x \overset{b}{\leftrightarrow} y]_a \; :: \; y_a^b : \nabla_a(A)} \; Id \qquad \frac{\Gamma \vdash P \; :: \; T}{\Gamma, x_a^b : \nabla_a(1) \vdash P \; :: \; T} \; 1_L \qquad \frac{}{\Upsilon \vdash [\mathbf{0}]_b \; :: \; x_b^a : \nabla_b(1)} \; 1_R$$

$$\frac{\Gamma, x_a^b : \nabla_a(B), y_a^b : \nabla_a(A) \vdash [p]_b \; :: \; T}{\Gamma, x_a^b : \nabla_a(A \otimes B) \vdash [x(y).p]_b \; :: \; T} \; \otimes_L \qquad \frac{\Upsilon^{\succeq b}, \Delta^{\succeq b} \vdash [p]_b \; :: \; y_b^a : \nabla_b(A) \quad \Upsilon^{\succeq b}, \Delta'^{\succeq b} \vdash [q]_b \; :: \; x_b^a : \nabla_b(B)}{\Upsilon, \Delta, \Delta' \vdash [\bar{x}\langle y\rangle.(p \parallel q)]_b \; :: \; x_b^a : \nabla_b(A \otimes B)} \; \otimes_R$$

$$\frac{\Upsilon, \Delta \vdash [p]_b \; :: \; y_b^a : \nabla_b(A) \quad \Upsilon, \Delta', x_a^b : \nabla_a(B) \vdash [q]_b \; :: \; T}{\Upsilon, \Delta, \Delta', x_a^b : \nabla_a(A \multimap B) \vdash [\bar{x}\langle y\rangle.(p \parallel q)]_b \; :: \; T} \; \multimap_L \qquad \frac{\Gamma^{\succeq b}, y_a^b : \nabla_a(A) \vdash [p]_b \; :: \; x_b^a : \nabla_b(B)}{\Gamma \vdash [x(y).p]_b \; :: \; x_b^a : \nabla_b(A \multimap B)} \; \multimap_R$$

$$\frac{\Gamma, x_a^b : \nabla_a(A) \vdash [p]_b \; :: \; T \quad \Gamma, x_a^b : \nabla_a(B) \vdash [q]_b \; :: \; T}{\Gamma, x_a^b : \nabla_a(A \oplus B) \vdash [x.\mathtt{case}(p,q)]_b \; :: \; T} \; \oplus_L$$

$$\frac{\Gamma^{\succeq b} \vdash [p]_b \; :: \; x_b^a : \nabla_b(A)}{\Gamma \vdash [x.\mathtt{inl}.p]_b \; :: \; x_b^a : \nabla_b(A \oplus B)} \; \oplus_{R1} \qquad \frac{\Gamma^{\succeq b} \vdash [p]_b \; :: \; x_b^a : \nabla_b(B)}{\Gamma \vdash [x.\mathtt{inr}.p]_b \; :: \; x_b^a : \nabla_b(A \oplus B)} \; \oplus_{R2}$$

$$\frac{\Gamma, x_a^b : \nabla_a(A) \vdash [p]_b \; :: \; T}{\Gamma, x_a^b : \nabla_a(A \& B) \vdash [x.\mathtt{inl}.p]_b \; :: \; T} \; \&_{L1} \qquad \frac{\Gamma, x_a^b : \nabla_a(B) \vdash [p]_b \; :: \; T}{\Gamma, x_a^b : \nabla_a(A \& B) \vdash [x.\mathtt{inr}.p]_b \; :: \; T} \; \&_{L2}$$

$$\frac{\Gamma^{\succeq b} \vdash [p]_b \; :: \; x_b^a : \nabla_b(A) \quad \Gamma^{\succeq b} \vdash [q]_b \; :: \; x_b^a : \nabla_b(B)}{\Gamma \vdash [x.\mathtt{case}(p,q)]_b \; :: \; x_b^a : \nabla_b(A \& B)} \; \&_R$$

$$\frac{\Gamma, y_c^c : \nabla_c(A) \vdash [p]_c \; :: \; T}{\Gamma, x_a^b : \nabla_a(!^c A) \vdash [x.\mathtt{move}(y,c).p]_b \; :: \; T} \; !_L \qquad \frac{\Gamma^{\succeq b} \vdash [p]_c \; :: \; y_c^c : \nabla_c(A)}{\Gamma \vdash [\bar{x}.\mathtt{move}\langle y,c\rangle.p]_b \; :: \; x_b^a : \nabla_b(!^c A)} \; !_R$$

$$\frac{\mathscr{A}, \alpha \preceq a \sqcap b \; : \; \Gamma, x_a^b : \nabla_a(A) \vdash [p]_b \; :: \; T}{\mathscr{A} \; : \; \Gamma, x_a^b : \nabla_a(\mathbb{U}_{\alpha : a \sqcap b}.A) \vdash [x(\alpha).p]_b \; :: \; T} \; \mathbb{U}_L \qquad \frac{\Gamma^{\succeq b} \vdash [p]_b \; :: \; x_b^a : \nabla_b(A\{l/\alpha\})}{\Gamma \vdash [x\langle l\rangle.p]_b \; :: \; x_b^a : \nabla_b(\mathbb{U}_{\alpha : a \sqcap b}.A)} \; \mathbb{U}_R$$

$$\frac{\Gamma, x_a^b : \nabla_a(A\{l/\alpha\}) \vdash [p]_b \; :: \; T}{\Gamma, x_a^b : \nabla_a(\mathbb{m}_{\alpha : a \sqcap b}.A) \vdash [x\langle l\rangle.p]_b \; :: \; T} \; \mathbb{m}_L \qquad \frac{\mathscr{A}, \alpha \preceq a \sqcap b \; : \; \Gamma^{\succeq b} \vdash [p]_b \; :: \; x_b^a : \nabla_b(A)}{\mathscr{A} \; : \; \Gamma \vdash [x(\alpha).p]_b \; :: \; x_b^a : \nabla_b(\mathbb{m}_{\alpha : a \sqcap b}.A)} \; \mathbb{m}_R$$

$$\frac{\Gamma, x_{a^+}^b : \nabla_{a^+}(!^a A), y_a^b : \nabla_a(A) \vdash [p]_b \; :: \; T}{\Gamma, x_{a^+}^b : \nabla_{a^+}(!^a A) \vdash [\bar{x}\langle y\rangle.p]_b \; :: \; T} \; copy \qquad \frac{\Upsilon^{\succeq b^+} \vdash [p]_b \; :: \; y_b^a : \nabla_b(A)}{\Upsilon \vdash [!x(y).p]_b \; :: \; x_{b^+}^a : \nabla_{b^+}(!^b A)} \; !_R^+$$

$$\frac{\Upsilon, \Delta \vdash P \; :: \; x_a^b : \nabla_a(A) \quad \Upsilon, \Delta', x_a^b : \nabla_a(A) \vdash Q \; :: \; T}{\Upsilon, \Delta, \Delta' \vdash (\nu \, x)(P \parallel Q) \; :: \; T} \; Cut$$

Figure 4: Typing System. Since the context $\mathscr{A}$ is only modified in rules $\mathbb{U}_L$ and $\mathbb{m}_R$, we omit it in the other rules.