A General Proof System for Modalities in Concurrent Constraint Programming

Vivek Nigam[1], Carlos Olarte[2], and Elaine Pimentel[3]

[1] Universidade Federal da Paraíba, Brazil.
[2] Pontificia Universidad Javeriana-Cali, Colombia.
[3] Universidade Federal de Minas Gerais, Brazil.

Abstract. The combination of timed, spatial, and epistemic information is often needed in the specification of modern concurrent systems. We propose the proof system $\mathrm{SELL}^{\Cap}$, which extends linear logic with subexponentials with quantifiers over subexponentials, therefore allowing for an arbitrary number of modalities. We then show how a proper structure of the subexponential signature in $\mathrm{SELL}^{\Cap}$ allows for the specification of concurrent systems with timed, spatial, and epistemic modalities. In the context of Concurrent Constraint Programming (CCP), a declarative model of concurrency, we illustrate how the view of subexponentials as specific modalities is general enough to modularly encode into $\mathrm{SELL}^{\Cap}$ variants of CCP with these three modalities, thus providing a proof-theoretic foundations for those calculi.

1 Introduction

To specify the behavior of distributed agents or the policies governing a distributed system, it is often necessary to reason by using different types of modalities, such as time, space, or even the epistemic state of agents. For instance, the access-control policies of a building might allow Bob to have access only in some pre-defined time, such as its opening hours. Another policy might also allow Bob to ask Alice who has higher credentials to grant him access to the building, or even specify that Bob has only access to some specific rooms of the building. Following this need, many formalisms have been proposed to specify, program and reason about such policies, *e.g.*, Ambient Calculus, Concurrent Constraint Programming, Authorization Logics, just to name a few.

Logic and proof theory have often inspired the design of many of these formalisms. For example, Saraswat *et al.* proposed Concurrent Constraint Programming (CCP), a model for concurrency that combines the traditional operational view of process calculi with a declarative view based on logic [16, 15] (see [13] for a survey). Agents in CCP *interact* with each other by *telling* and *asking* information represented as *constraints* to a global store. Later, Fages *et al.* in [4] proposed Linear Concurrent Constraint (`lcc`), inspired by linear logic [6], to allow the use of linear constraints, that is, tokens of information that once used by an agent are removed from the global store.

In order to capture the behavior of distributed systems which take into account spatial, temporal and/or epistemic properties, new formalisms have been proposed. For instance, Saraswat *et al.* proposed `tcc` [17], which extends CCP with time modalities.

More recently, Knight *et al.* [7] proposed a CCP-based language with spatial and epistemic modalities. Some of these developments have also been followed by a similar development in proof theory. For instance, Nigam proposed a framework for linear authorization logics [9], which allow the specification of access control policies that may mention the affirmations, possessions and knowledge of principals and demonstrated that a wide range of linear authorization policies can be specified in linear logic with subexponentials (*SELL*) [2, 10].

This paper shows that timed, spatial, and epistemic modalities can be *uniformly* specified in a single logical framework called SELL$^{\Cap}$. Our first contribution is the introduction of the proof system SELL$^{\Cap}$, which extends *SELL* with universal ($\Cap$) and existential ($\Cup$) quantifiers over subexponentials. It turns out that SELL$^{\Cap}$ has good proof-theoretic properties: it admits cut-elimination and also has a complete focusing discipline [1].

For our second contribution, we show that subexponentials can be interpreted as spatial, epistemic and temporal modalities, thus providing a framework for specifying concurrent systems with these modalities. This is accomplished by encoding different CCP languages, for which the proposed quantifiers play an important role. For instance, they enable the use of an *arbitrary number of subexponentials*, required to model the unbounded nesting of modalities, which is a common feature in epistemic and spatial systems. This do not seem possible in existing logical frameworks such as [18] that do not contain subexponentials nor its quantifiers.

Another important feature of subexponentials is that they can be organized into a pre-order, which specifies the provability relation among them. By coupling subexponential quantifiers with a suitable pre-order, it is possible to specify *declaratively* the rules in which agents can manipulate information. For example, an agent cannot see the information contained in a space that she does not have access to. The boundaries are naturally implied by the pre-order of subexponentials.

This work opens a number of possibilities for specifying the behavior of distributed systems. For instance, unlike [7], it seems possible in our framework to handle an infinite number of agents. Moreover, we discuss how linearity of constraints can be straightforwardly included to these systems to represent, *e.g.*, agents that can *update/change* the content of the distributed spaces. Also, by changing the underlying subexponential structure, different modalities can be put in the hands of the modelers and programmers. Finally, all the linear logic meta-theory becomes available for reasoning about distributed systems featuring modalities.

Organization. In Section 2 we review the proof theory of *SELL*, identify its limitations, and propose an extension (SELL$^{\Cap}$) allowing for the quantification of subexponentials ($\Cap$ and $\Cup$). We prove that SELL$^{\Cap}$ admits cut-elimination. Section 3 reviews some background on CCP, for which we provide a sound and faithful encoding in SELL$^{\Cap}$. As we shall show, our encoding is modular enough to extend it so to specify new constructs involving modalities, namely, constructs for epistemic (Section 4.2), spatial (Section 4.3) and temporal modalities (Section 4.4). In Section 5 we identify a number of future work directions that we are currently working on. The detailed proofs and the focused presentation of SELL$^{\Cap}$ appear in the Appendix.

2 Linear Logic and Subexponential Quantifiers

Linear logic with subexponentials (*SELL*) shares with linear logic all connectives except the exponentials: the multiplicative and additive conjunctions, $\otimes$ and $\&$, linear implication, $\multimap$, additive disjunction $\oplus$, units $1, \bot, 0, \top$, and its universal and existential quantifiers $\forall$ and $\exists$. Their proof rules are the same as in standard linear logic [6]. However, instead of having a single pair of exponentials ! and ?, *SELL* may contain as many *labeled* exponentials ($!^l$ and $?^l$) as needed, called *subexponentials* [2, 10].

Formally, the proof system for intuitionistic *SELL* is specified by a *subexponential signature* $\Sigma = \langle I, \preceq, U \rangle$, where I is a set of labels, $U \subseteq I$ is a set specifying which subexponentials allow weakening and contraction, and $\preceq$ is a pre-order among the elements of I. We assume that U is closed wrt $\preceq$, *i.e.*, if $a \in U$ and $a \preceq b$, then $b \in U$. The system *SELL* is constructed by adding all the rules for the linear logic connectives as usual, except for the exponentials, whose right introduction rules are as follows. For each $a \in I$, we add the introduction rules corresponding to dereliction and promotion, where we state explicitly the first-order signature $\mathcal{L}$ of the terms of the language:

$$\frac{\mathcal{L}; \Gamma, F \longrightarrow G}{\mathcal{L}; \Gamma, !^a F \longrightarrow G}\ !^a{}_L \quad \text{and} \quad \frac{\mathcal{L}; !^{x_1} F_1, \ldots !^{x_n} F_n \longrightarrow G}{\mathcal{L}; !^{x_1} F_1, \ldots !^{x_n} F_n \longrightarrow !^a G}\ !^a{}_R$$

The rules for $?^a$ are dual. Here, the rule $!^a{}_R$ (and $?^a{}_L$) have the side condition that $a \preceq x_i$ for all i. That is, one can only introduce a $!^a$ on the right (or a $?^a$ on the left) if all other formulas in the sequent are marked with indices that are greater or equal than a. Furthermore, for all $a \in U$, we add the structural rules:

$$\frac{\mathcal{L}; \Gamma, !^a F, !^a F \longrightarrow G}{\mathcal{L}; \Gamma, !^a F \longrightarrow G}\ C \quad \text{and} \quad \frac{\mathcal{L}; \Gamma \longrightarrow G}{\mathcal{L}; \Gamma, !^a F \longrightarrow G}\ W$$

That is, we are also free to specify which indices are *unbounded* (those appearing in the set U), and which indices are *linear* or *bounded*.

It is known that subexponentials greatly increase the expressiveness of the system when compared to linear logic. For instance, they can be used to represent contexts of proof systems [12], to mark the epistemic state of agents [9], or to specify locations in sequential computations [10].

The key difference to standard presentations of linear logic is that while linear logic has only seven logically distinct prefixes of bangs and question-marks, *SELL* allows for an unbounded number of such prefixes, *e.g.*, $!^i$, or $!^i ?^j$. As we show later, by using different prefixes (written generically as ∇), we will also be able to interpret subexponentials in more creative ways, such as temporal units or spatial and epistemic modalities.

However, *SELL* has a serious limitation: it does not have any sort of quantification over subexponentials. Therefore, given the interpretation above for subexponentials, it is not feasible in *SELL* to specify properties that are valid for all locations or for all agents. Another way of visualizing this limitation is that any sequent in any derivation in *SELL* has the same subexponential signature Σ. For instance, it is not possible to encode in *SELL* none of the encodings of the CCP languages discussed in Section 4.

2.1 Subexponential Quantifiers

In the following we introduce the system SELL$^{\Cap}$, containing two novel connectives: universal ($\Cap$) and existential ($\Cup$) quantifiers over *subexponentials*.

Subexponential Constants and Variables Recall that given a pre-order $(I, \preceq)$, the *ideal* of an element $a \in I$ in $\preceq$, written $\downarrow a$, is the set $\{x \mid x \preceq a\}$. The subexponential signature of SELL$^{\Cap}$ is of the form $\Sigma = \langle I, \preceq, F, U\rangle$, where I is a set of subexponential constants and $\preceq$ is a pre-order among these constants. The new component $F = \{\mathfrak{f}_1, \ldots, \mathfrak{f}_n\}$ specifies families of subexponentials indices. In particular, a family $\mathfrak{f} \in F$ takes an element of $a \in I$ and returns a subexponential index $\mathfrak{f}(a)$. As it will be clear below, families allow us to specify disjoint pre-orders based on $\langle I, \preceq\rangle$. The set of unbounded subexponentials $U \subseteq \{\mathfrak{f}(a) \mid a \in I, \mathfrak{f} \in F\}$, as before, is upwardly closed wrt $\preceq$: if $a \preceq b$, where $a, b \in I$, and $\mathfrak{f}(a) \in U$ then $\mathfrak{f}(b) \in U$. Notice that the SELL$^{\Cap}$ system obtained from the signature $\langle I, \preceq, \{id\}, U\rangle$ conservatively extends the *SELL* system obtained from $\langle I, \preceq, U\rangle$.

For our subexponential quantification, we will be interested in determining whether a subexponential b belongs to the ideal $\downarrow a$ of a given subexponential a. This is formally achieved by adding a typing information to subexponentials. Given the signature $\Sigma = \langle I, \preceq, F, U\rangle$, the judgment $s : a$ is true whenever $s \preceq a$. Thus we obtain the set $\mathcal{A}_\Sigma = \{s : a \mid s, a \in I, s \preceq a\}$ of typed *subexponential constants*. We shall simply write $!^{\mathfrak{f}(l)}$ instead of $!^{\mathfrak{f}(l\,:\,a)}$ when the type "a" can be inferred from the context. Similarly for "?."

As with the universal quantifier $\forall$, which introduces *eigenvariables* to the signature, the universal quantification for subexponentials $\Cap$ introduces *subexponential variables* of the shape $l : a$, where a is a subexponential constant, *i.e.*, $a \in I$. Thus, SELL$^{\Cap}$ sequents have the form $\mathcal{A}; \mathcal{L}; \Gamma \longrightarrow G$, where $\mathcal{A} = \mathcal{A}_\Sigma \cup \{l_1 : a_1, \ldots, l_n : a_n\}$, and $\{l_1, \ldots, l_n\}$ is a disjoint set of subexponential variables and $\{a_1, \ldots, a_n\} \subseteq I$ are subexponential constants. Formally, only these subexponential constants and variables may appear free as an index of subexponential bangs and question marks.

The introduction rules for the subexponential quantifiers look similar to those introducing the first-order quantifiers, but instead of manipulating the context $\mathcal{L}$, they manipulate the context $\mathcal{A}$:

$$\frac{\mathcal{A}; \mathcal{L}; \Gamma, P[l/x] \longrightarrow G}{\mathcal{A}; \mathcal{L}; \Gamma, \Cap x : a.P \longrightarrow G}\ \Cap_L \qquad \frac{\mathcal{A}, l_e : a; \mathcal{L}; \Gamma \longrightarrow G[l_e/x]}{\mathcal{A}; \mathcal{L}; \Gamma \longrightarrow \Cap x : a.G}\ \Cap_R$$

$$\frac{\mathcal{A}, l_e : a; \mathcal{L}; \Gamma, P[l_e/x] \longrightarrow G}{\mathcal{A}; \mathcal{L}; \Gamma, \Cup x : a.P \longrightarrow G}\ \Cup_L \quad \frac{\mathcal{A}; \mathcal{L}; \Gamma \longrightarrow G[l/x]}{\mathcal{A}; \mathcal{L}; \Gamma \longrightarrow \Cup x : a.G}\ \Cup_R$$

In these rules, $l : a \in \mathcal{A}$ and l_e is fresh, *i.e.*, it does not appear in $\mathcal{A}$ nor $\mathcal{L}$. Intuitively, subexponential variables play a similar role as eigenvariables. The generic variable $l_i : a_i$ represents *any subexponential constant* that is in the ideal of the subexponential constant a. This is formalized by constructing from a given sequent, $\mathcal{A}; \mathcal{L}; \Gamma \longrightarrow G$, a pre-order, called *sequent pre-order*, written $\preceq_{\mathcal{A}}$. This pre-order is formally used in the side condition of the promotion rule and is defined on subexponentials obtained from applying a family $\mathfrak{f}_i \in F$ to an element of I. Formally, it is the *transitive* and *reflexive closure* of the following sets:

$$\begin{array}{l}\{\mathfrak{f}(s_i : a) \preceq_{\mathcal{A}} \mathfrak{f}(s_j : b) \mid \mathfrak{f} \in F, s_i, s_j \in I \text{ and } s_i \preceq s_j\} \cup \\ \{\mathfrak{f}(l : a) \preceq_{\mathcal{A}} \mathfrak{f}(s : b) \mid \mathfrak{f} \in F, l \notin I, s \in I \text{ and } a \preceq s\}\end{array}$$

The first component of this set specifies that families preserve the pre-order $\preceq$ in Σ only involving subexponential constants; thus $\preceq_{\mathcal{A}}$ is a conservative extension of $\preceq$. The second component is the interesting one, which relates subexponential obtained from variables and subexponentials obtained from constants: $l : a$ means that l belongs to the ideal of a and if $a \leq s$, then $\mathfrak{f}(l) \leq_{\mathcal{A}} \mathfrak{f}(s)$. Notice that $\mathfrak{f}(l_1)$ and $\mathfrak{f}(l_2)$ are unrelated for two different subexponentials variables l_1 and l_2.

The pre-order $\leq_{\mathcal{A}}$ is used in the right-introduction of bangs and the left-introduction of question-marks in a similar way as before in *SELL*

$$\frac{\mathcal{A}; \mathcal{L}; !^{\mathfrak{f}(l_1:\, a_1)}F_1, \ldots !^{\mathfrak{f}(l_n:\, a_n)}F_n \longrightarrow G}{\mathcal{A}; \mathcal{L}; !^{\mathfrak{f}(l_1:\, a_1)}F_1, \ldots, !^{\mathfrak{f}(l_n:\, a_n)}F_n \longrightarrow !^{\mathfrak{f}(l\,:\,a)}G}\ {!^{\mathfrak{f}(l:a)}}_R$$

$$\frac{\mathcal{A}; \mathcal{L}; !^{\mathfrak{f}(l_1:\, a_1)}F_1, \ldots !^{\mathfrak{f}(l_n:\, a_n)}F_n, P \longrightarrow ?^{\mathfrak{f}(l_{n+1}:\, a_{n+1})}G}{\mathcal{A}; \mathcal{L}; !^{\mathfrak{f}(l_1:\, a_1)}F_1, \ldots, !^{\mathfrak{f}(l_n:\, a_n)}F_n, ?^{\mathfrak{f}(l\,:\,a)}P \longrightarrow ?^{\mathfrak{f}(l_{n+1}:\, a_{n+1})}G}\ {?^{\mathfrak{f}(l:a)}}_L$$

with the side condition that for all $1 \leq i \leq n+1$, $\mathfrak{f}(l : a) \leq_{\mathcal{A}} \mathfrak{f}(l_i : a_i)$.

Notice that bangs and question marks use families, while quantifiers use only constants and variables. This interplay allows us to bind formulas with different families, such as in the formula $\Cap l : a.(!^{\mathfrak{f}(l:a)}F \otimes !^{\mathfrak{g}(l:a)}F')$.

As pointed out in [2], for cut-elimination, one needs to be careful with the structural properties of subexponentials. For subexponential variables, we define $\mathfrak{f}(l_i : a)$ to be always bounded, while for subexponential constants, it is similar as before: if $\mathfrak{f}(s : a) \in U$, then structural rules can be applied. We can now state our desired result.

Theorem 1. *For any signature Σ, the proof system SELL$^{\Cap}$ admits cut-elimination.*

3 CCP calculi

Concurrent Constraint Programming (CCP) [15, 16] is a model for concurrency that combines the traditional operational view of process calculi with a *declarative* view based on logic. This allows CCP to benefit from the large set of reasoning techniques of both process calculi and logic. In CCP, processes *interact* with each other by *telling* and *asking* constraints (pieces of information) in a common store of partial information. The type of constraints processes may act on is not fixed but parametric in a constraint system (CS for short). Such systems can be formalized as a Scott information system as in [15], or they can be built upon a suitable fragment of logic *e.g.*, as in [8]. Here we specify constraints as formulas in a fragment of intuitionistic first-order logic (LJ [5]).

Definition 1 (Constraint System [4]). *A constraint system is a tuple $(\mathcal{C}, \vdash_\Delta)$ where $\mathcal{C}$ is a set of formulas (constraints) built from a first-order signature and the grammar*

$$F := 1 \mid A \mid F \wedge F \mid \exists \overline{x}.F$$

where A is an atomic formula. We shall use c, c', d, d', etc, to denote elements of $\mathcal{C}$. Moreover, let Δ be a set of non-logical axioms of the form $\forall \overline{x}.(c \supset c')$ where all free variables in c and c' are in $\overline{x}$. We say that d entails *d', written as $d \vdash_\Delta d'$, iff the sequent $\Delta, d \longrightarrow d'$ is probable in LJ [5].*

The language of determinate CCP processes is built from constraints in the underlying constraint system as follows:

$$P, Q ::= \mathbf{tell}(c) \mid \mathbf{ask}\ c\ \mathbf{then}\ P \mid P \parallel Q \mid (\mathbf{local}\ \overline{x})\,P \mid p(\overline{x})$$

where variables in $\overline{x}$ are pairwise distinct. A way to introduce non-determinism in CCP is by adding the usual non-deterministic choice operator $P{+}P'$. However, as the systems considered here are all determinate, we shall add this operator only when needed.

$$\frac{(X;\Gamma;c) \equiv (X';\Gamma';c') \longrightarrow (Y';\Delta';d') \equiv (Y;\Delta;d)}{(X;\Gamma;c) \rightarrow (Y;\Delta;d)}\ \mathrm{R_{EQUIV}}$$

$$\frac{}{(X;\mathbf{tell}(c),\Gamma;d) \longrightarrow (X;\Gamma;c \wedge d)}\ \mathrm{R_T} \qquad \frac{d \vdash_\Delta c}{(X;\mathbf{ask}\ c\ \mathbf{then}\ P,\Gamma;d) \longrightarrow (X;P,\Gamma;d)}\ \mathrm{R_A}$$

$$\frac{x \notin X \cup fv(d) \cup fv(\Gamma)}{(X;(\mathbf{local}\ x)\,P,\Gamma;d) \longrightarrow (X \cup \{x\};P,\Gamma;d)}\ \mathrm{R_L} \qquad \frac{p(\overline{x}) \overset{\mathtt{def}}{=} P}{(X;p(\overline{y}),\Gamma;d) \longrightarrow (X;P[\overline{y}/\overline{x}],\Gamma;d)}\ \mathrm{R_C}$$

(a) Operational rules for CCP.

$$\frac{(X;P;c) \longrightarrow (X';P';d)}{(X;[P]_i;c) \longrightarrow (X';[P]_i,P';d)}\ \mathrm{R_E} \qquad \frac{(X;P;d^i) \longrightarrow (X';P';d')}{(X;[P]_i;d) \longrightarrow (X';[P']_i;d \wedge s_i(d'))}\ \mathrm{R_S}$$

(b) Operational rules for `eccp` and `sccp`

$$\frac{}{(X;\square P;\Gamma;d) \longrightarrow (X;P,\circ\square P;\Gamma;d)}\ \mathrm{R_\square} \quad \frac{n \geq 0}{(X;\star P,\Gamma;d) \longrightarrow (X;\circ^n P,\Gamma;d)}\ \mathrm{R_\star} \quad \frac{(\emptyset;P;c) \longrightarrow^* (X;\Gamma;d) \not\longrightarrow}{P \xRightarrow{(c,\exists X.d)} (\mathbf{local}\,X)\,F(\Gamma)}\ \mathrm{R_{Obs}}$$

(c) Internal and Observable rules for timed-ccp. $\circ^n$ means $\circ ... \circ$ n-times. $F(\Gamma)$ –the *future* of Γ– is defined as: $F(\mathbf{ask}\ c\ \mathbf{then}\ Q) = \emptyset$, $F(\circ Q) = Q$ and $F(P_1, ..., P_n) = F(P_1) \parallel ... \parallel F(P_n)$

Fig. 1. Operational semantics for CCP calculi

The process $\mathbf{tell}(c)$ adds c to the store d producing the new store $d \wedge c$. The process $\mathbf{ask}\ c\ \mathbf{then}\ P$ evolves into P if the store entails c. Otherwise, it remains blocked until more information is added to the store. This provides a synchronization mechanism based on constraint entailment. The process $(\mathbf{local}\,\overline{x})\,P$ behaves as P and binds the variables in $\overline{x}$ to be local to it. The process $p(\overline{x})$ evolves into $P[\overline{x}/\overline{y}]$ provided the definition $p(\overline{y}) \overset{\mathtt{def}}{=} P$ where all free variables of P are in the pairwise distinct variables $\overline{y}$.

The operational semantics of CCP is given by the transition relation $\gamma \longrightarrow \gamma'$ satisfying the rules on Figure 1(a). These rules are straightforward realizing the operational intuitions given above. Moreover, they will form the core of transitions common to the other systems that we encode later. A *configuration* γ is a triple of the form $(X;\Gamma;c)$, where c is a constraint (a logical formula specifying the store), Γ is a multiset of processes, and X is a set of hidden (local) variables of c and Γ. The multiset $\Gamma = P_1, P_2, \ldots, P_n$ represents the process $P_1 \parallel P_2 ... \parallel P_n$. We shall indistinguishably use both notations to denote parallel composition of processes.

Processes are quotiented by a structural congruence relation $\cong$ satisfying: (1) $P \parallel Q \cong Q \parallel P$; (2) $P \parallel (Q \parallel R) \cong (P \parallel Q) \parallel R$; and (3) $(\mathbf{local}\,x)\,P \cong (\mathbf{local}\,y)\,P[y/x]$ if $y \notin fv(P)$. Furthermore, $\Gamma = \{P_1, ..., P_n\} \cong \{P'_1, ..., P'_n\} = \Gamma'$ iff $P_i \cong P'_i$ for all $1 \leq i \leq n$. Finally, $(X;\Gamma;c) \cong (X';\Gamma';c')$ iff $X = X'$, $\Gamma \cong \Gamma'$ and $c \equiv_\Delta c'$ (*i.e.*, $c \vdash_\Delta c'$ and $c' \vdash_\Delta c$).

Let $\longrightarrow^*$ be the reflexive and transitive closure of $\longrightarrow$. If $(X;\Gamma;d) \longrightarrow^* (X';\Gamma';d')$ and $\exists X'.d' \vdash_\Delta c$ we write $(X;\Gamma;d) \Downarrow_c$. If $X = \emptyset$ and $d = 1$ we simply write $\Gamma \Downarrow_c$. Intuitively, if P is a process then $P \Downarrow_c$ captures the outputs of P (under input 1).

4 Encoding CCP languages as SELL$^⋒$ formulas

This section gives an interpretation of CCP processes as SELL$^⋒$ formulas. The encoding we propose will be used as basis in the subsequent sections to encode CCP calculi that include modalities. For this, we rely on the two following features of SELL$^⋒$. The first one is the subexponential quantifiers $⋒$ and $⋓$, which enable the specification of systems governing an unbounded number of modalities, *e.g.*, spaces or agents. For instance, these quantifiers allow us to specify that process definitions are available to all entities in the system. The second feature is the presence of non-equivalent subexponential prefixes (such as, *e.g.*, $!^i$ or $!^i?^i$) which will be written generically as ∇_i. This is the key for encoding correctly the different modalities, such as spatial, epistemic or temporal. As we mentioned before, while in linear logic there are only seven classes of modalities, SELL$^{\Cap}$ (and even *SELL*) allows for an unbounded number of non-equivalent prefixes.

4.1 Basic Encoding

We shall use a signature $\langle I \cup \{nil, \infty\}, \preceq, \{\mathfrak{c}, \mathfrak{p}, \mathfrak{d}\}, U\rangle$ with three families and two distinguished elements nil (the least element) and ∞ (the greatest element). Moreover, $\mathfrak{c}(a) \in U$ for all $a \in I \cup \{nil, \infty\}$ and $\mathfrak{p}(\infty) \in U$, while $\mathfrak{p}(nil), \mathfrak{d}(nil) \notin U$. Intuitively, the family $\mathfrak{c}$ is used to mark constraints; the family $\mathfrak{p}$ is used to mark processes; and the family $\mathfrak{d}$ is used to mark procedures $p(\overline{x})$ whose definition $p(\overline{y}) \stackrel{\mathtt{def}}{=} P$ may be unfolded. As it will be clear later, the remaining indices in I specify the modalities available in the system, where nil represents no modality. For instance, $\mathfrak{p}(nil)$ will mark a process that is not under any modality. Since process definitions, non-logical axioms and constraints can be used as many times, $\mathfrak{c}(a)$, $\mathfrak{p}(\infty)$ for any $a \in I$ are unbounded; since processes and procedure calls are consumed when executed, $\mathfrak{p}(nil)$ and $\mathfrak{d}(nil)$ are bounded.

Encoding Constraints and Processes Constraints and processes are encoded in SELL$^{\Cap}$ by using two functions: $\mathcal{P}[\![P]\!]_l$ for processes and $\mathcal{C}[\![c]\!]_l$ for constraints. These encodings will depend on the system that we want to encode and they are parametric on an index $l \in I$. Next we define such functions for the set of basic processes and constraints shown in Section 3. Later, we refine these encodings by adding new cases handling the specific constraints of each system. These cases will basically play with the index l.

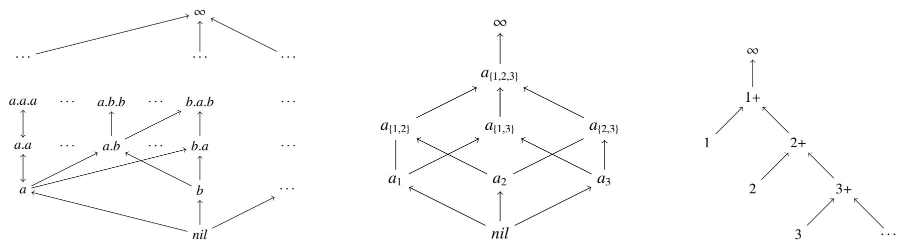
Definition 2 (Encoding of Constraints and Processes). *Let* $\langle I \cup \{nil, \infty\}, \preceq, \{\mathfrak{c}, \mathfrak{p}, \mathfrak{d}\}, U\rangle$ *be a subexponential signature, and let $l \in I$. For any constraint c, atomic formula A and process P we define* $\mathcal{C}[\![c]\!]_l$ *and* $\mathcal{P}[\![P]\!]_l$ *as follows:*

$$
\begin{array}{ll}
\mathcal{C}[\![c_1 \wedge c_2]\!]_l = \mathcal{C}[\![c_1]\!]_l \otimes \mathcal{C}[\![c_2]\!]_l & \mathcal{P}[\![\mathbf{tell}(c)]\!]_l = !^{\mathfrak{p}(l)}[\Cap s : l.(\mathcal{C}[\![c]\!]_s)] \\
\mathcal{C}[\![\exists \overline{x}.c]\!]_l = \exists \overline{x}.\mathcal{C}[\![c]\!]_l & \mathcal{P}[\![\mathbf{ask}\ c\ \mathbf{then}\ P]\!]_l = !^{\mathfrak{p}(l)}[\Cap s : l.(\mathcal{C}[\![c]\!]_s \multimap \mathcal{P}[\![P]\!]_s)] \\
\mathcal{C}[\![A]\!]_l = \nabla_{\mathfrak{c}(l)} A & \mathcal{P}[\![(\mathbf{local}\,\overline{x})\,P]\!]_l = !^{\mathfrak{p}(l)}[\Cap s : l.\exists \overline{x}.(\mathcal{P}[\![P]\!]_s)] \\
\mathcal{C}[\![1]\!]_l = \nabla_{\mathfrak{c}(l)} 1 & \mathcal{P}[\![P_1, ..., P_n]\!]_l = \mathcal{P}[\![P_1]\!]_l \otimes ... \otimes \mathcal{P}[\![P_n]\!]_l \\
 & \mathcal{P}[\![p(\overline{x})]\!]_l = \nabla_{\mathfrak{d}(l)}\, p(\overline{x})
\end{array}
$$

Hence, atomic constraints, processes and procedure calls ($p(\overline{x})$) are marked, respectively, with subexponentials from the $\mathfrak{c}$, $\mathfrak{p}$ and $\mathfrak{d}$ family. The role of the subexponential quantifiers in the encoding will become clear in the following sections. The idea is that they allow choosing in which modality a resulting process should be placed. We note that by using simple logical equivalences, the encoding $\mathcal{C}[\![c]\!]_l$ can be rewritten as $\exists \overline{x}. \left[\nabla_{\mathfrak{c}(l_1)} A_1 \otimes \cdots \otimes \nabla_{\mathfrak{c}(l_n)} A_n\right]$ where each A_i is an atomic formula or the unit 1.

Encoding Non-Logical Axioms and Process Definitions A non-logical axiom of the form $\forall \overline{x}(d \supset c)$ is encoded as:

$$\Cap l : \infty.\forall \overline{x}.(\mathcal{C}[\![c]\!]_l \multimap \mathcal{C}[\![d]\!]_l)$$



(a) Epistemic reasoning. (b) Common knowledge. (c) Timed modalities.

Fig. 2. Subexp. signatures for epistemic and time reasoning. Here $a \rightarrow b$ denotes that $a \preceq b$.

specifying that the non-logical axiom is available to all subexponentials in the ideal of ∞, *i.e.*, all elements in I. Similarly, a process definition is encoded as:

$$\Cap l : \infty.\forall \overline{x}.(\nabla_{\mathrm{b}(l)}\, p(\overline{x}) \multimap \mathcal{P}[\![P]\!]_l)$$

We write $[\![\Delta]\!]$ and $[\![\Psi]\!]$ for the set of formulas encoding the non-logical axioms Δ and the process definitions Ψ. Finally, a configuration $(X; \Gamma; c)$ is encoded as the sequent:

$$\mathcal{A}; \mathcal{L} \cup X; !^{\mathrm{c}(\infty)}[\![\Delta]\!], !^{\mathrm{p}(\infty)}[\![\Psi]\!], \mathcal{P}[\![\Gamma]\!]_{nil}, \mathcal{C}[\![c]\!]_{nil} \longrightarrow G$$

The formula G on the right is the goal to be proved, i.e., the encoding of the constraint we are interested to know whether it can be outputted or not by the system. Finally, as normally done [3], the fresh values X are specified as eigenvariables in the logic.

Since the left introduction rules for $\exists$ and $\otimes$ are invertible [1], we can rewrite the sequent above as follows, where we elide the contexts $\mathcal{A}$ and $\mathcal{L} \cup X$:

$$!^{\mathrm{c}(\infty)}[\![\Delta]\!], !^{\mathrm{p}(\infty)}[\![\Psi]\!], \mathcal{P}[\![\Gamma]\!]_{nil}, \nabla_{\mathrm{c}(l_1)} A_1, \cdots, \nabla_{\mathrm{c}(l_n)} A_n \longrightarrow G$$

It is worth noticing that the store is specified by the atomic formulas it contains (A_i), marked with the prefix, $\nabla_{\mathrm{c}(l_i)}$. Up to now, from Definition 2, we have a unique l_i, namely *nil*. The forthcoming encodings will enable different subexponential indices to be used, illustrating the encoding's modularity. Moreover, by changing the signature's pre-order, we will be able to specify different modalities (see *e.g.*, Figure 2(a)).

The specification of processes, on the other hand, simply manipulates the set of constraints appearing on the left-hand side of sequents. For instance, the encoding of **tell**(c) adds the atomic constraints which compose c to the left-hand side of the sequent, as in rule R_T. Repeating this process we can prove the following adequacy result.

Theorem 2. *Let P be a CCP process, $(\mathcal{C}, \vdash_\Delta)$ be a CS, Ψ be a set of process definitions. Let ∇_l be instantiated to $!^l$. Then $P \Downarrow_c$ iff $!^{\mathrm{c}(\infty)}[\![\Delta]\!], !^{\mathrm{p}(\infty)}[\![\Psi]\!], \mathcal{P}[\![P]\!]_{nil} \longrightarrow \mathcal{C}[\![c]\!]_{nil} \otimes \top$.*

The adequacy that we get is in fact quite strong on the *level of derivations* [11]. It relies on the completeness of the focusing strategy [1] (see the Appendix for details). This means that doing proof search from our encoding corresponds exactly to executing processes in the encoded CCP language. This is much stronger than the encoding of CCP in [4], which is only on the *level of provability*. In fact, all our encodings, except the one for `tcc` (Section 4.4), have that strong level of adequacy.

4.2 Epistemic meaning to subexponentials

Knight *et al.* in [7] proposed Epistemic CCP (`eccp`), a CCP-based calculi where systems of agents are considered for distributed and epistemic reasoning. In `eccp`, the constraint system, seen as an Scott information system as in [15], is extended in order to consider spaces of agents. Roughly, each agent i has a space s_i and $s_i(c)$ means "c holds in the space –store– of agent i."

The following definition gives an instantiation of an epistemic constraint system where basic constraints are built as in Definition 1.

Definition 3 (Epistemic Constraint System (ECS)). *Let A be a countable set of agent names. An ECS $(C_e, \vdash_{\Delta_e})$ is a CS where, for any $i \in A$, $s_i : C_e \longrightarrow C_e$ satisfies:*
1. $s_i(1) = 1$ *(bottom preserving)*
2. $s_i(c \wedge d) = s_i(c) \wedge s_i(d)$ *(lub preserving)*
3. *If $d \vdash_{\Delta_e} c$ then $s_i(d) \vdash_{\Delta_e} s_i(c)$ (monotonicity)*
4. $s_i(c) \vdash_{\Delta_e} c$ *(beliefs are facts –extensiveness–)*
5. $s_i(s_i(c)) = s_i(c)$ *(idempotence)*

CCP processes are extended in eccp with the constructor $[P]_i$ that represents P running in the space of the agent i. The operational rules for $[P]_i$ are specified in Figure 1(b). In epistemic systems, agents are trustful, *i.e.*, if an agent i knows some information c, then c is necessarily true. Furthermore, if j knows that i knows c, then j also knows c. For example, given a hierarchy of agents as in $[[P]_i]_j$, it should be possible to propagate the information produced by P in the space i to the outermost space j. This is captured exactly by the rule $\mathrm{R_E}$, which allows a process P in $[P]_i$ to run also outside the space of agent i, i.e., P can be contracted. The rule $\mathrm{R_S}$, on the other hand, allows us to observe the evolution of processes inside the space of an agent. There, the constraint d^i represents the information the agent i may see or have of d, *i.e.*, $d^i = \bigwedge\{c \mid d \vdash_{\Delta_e} s_i(c)\}$. For instance, i *sees* c from the store $s_i(c) \wedge s_j(c')$.

We now configure the encodings in Section 4 so to encode epistemic modalities, starting by the subexponential signature that we use. Let $A = \{a_1, a_2, ...\}$ be a possible infinite set of agents and let A^* be the set of non-empty strings of elements in A; for example, if $a, b \in A$, then $a, b, a.a, b.a, a.b.a, \ldots \in A^*$. We shall use $\bar{i}, \bar{l}, etc$ to denote elements in A^*. We shall also consider nil to be the empty string, thus the string $\bar{i}.nil.\bar{l}$ is written as $\bar{i}.\bar{l}$. We let $I = A^* \cup \{nil, \infty\}$ and $U = \{\mathfrak{c}(\bar{l}), \mathfrak{d}(\bar{l}), \mathfrak{p}(\bar{l}) \mid \bar{l} \in I\} \setminus \{\mathfrak{d}(nil), \mathfrak{p}(nil)\}$. Intuitively, the connective $!^{\mathfrak{p}(1.2.3)}$ specifies a process in the structure $[[[\cdot]_3]_2]_1$, denoting "agent 1 knows that agent 2 knows that agent 3 knows" expressions. The connective $!^{\mathfrak{c}(1.2.3)}$, on the other hand, specifies a constraint of the form $s_1(s_2(s_3(\cdot)))$. Notice that all $\mathfrak{p}(\cdot)$ and $\mathfrak{d}(\cdot)$ subexponentials except the ones constructed using nil are unbounded. This reflects the fact that both constraints and processes in the space of an agent are unbounded, as specified by rule $\mathrm{R_E}$.

The pre-order $\preceq$ is as depicted in Figure 2(a). More precisely, for every two different agent names a and b in A, the subexponentials a and b are unrelated; Moreover, two sequences in A^* are related $i_1.i_2.\cdots i_m \preceq j_1.j_2.\cdots j_n$ whenever the following sequent is provable $!^{\mathfrak{c}(j_1)}!^{\mathfrak{c}(j_2)}\ldots!^{\mathfrak{c}(j_n)}F \longrightarrow !^{\mathfrak{c}(i_1)}!^{\mathfrak{c}(i_2)}\ldots!^{\mathfrak{c}(i_n)}F$, for any formula F. Alternatively, the pre-order on sequences of agent names could be defined as $a \approx a.\ldots.a$ and $b_1 \ldots b_n \preceq \bar{i}_1.b_1.\bar{i}_2.b_2...\bar{i}_n.b_n$ where each $\bar{i}_i$ is a possible empty string of elements in A.

The shape of the pre-order is key for our encoding. In particular, we are using one subexponential index, *e.g.*, $\mathfrak{p}(i_1.i_2.\cdots i_n)$, to denote a prefix of subexponential bangs $!^{\mathfrak{p}(i_1)}!^{\mathfrak{p}(i_2)}\ldots!^{\mathfrak{p}(i_n)}$. Thus if two subexponentials l, l' are equal in the pre-order ($l \approx l'$), it means that they represent the same equivalence class of prefixes. This way, we are able to quantify over such prefixes (or boxes) by using a single quantifier, for example, as we do for the encoding of the non-logical axioms and procedure calls.

Definition 4 (Epistemic constraints and processes). *We extend $\mathcal{C}[\![\cdot]\!]_l$ in Definition 2 so that $\mathcal{C}[\![s_i(c)]\!]_{\bar{l}} = \mathcal{C}[\![c]\!]_{\bar{l}.i}$ and $\nabla_{\bar{l}}$ is instantiated as $!^{\bar{l}}$. Moreover, we extend $\mathcal{P}[\![\cdot]\!]_l$ in Definition 2 so that $\mathcal{P}[\![[P]_i]\!]_{\bar{l}} = \mathcal{P}[\![P]\!]_{\bar{l}.i}$.*

Observe that, in $\mathcal{P}[\![P]\!]_{\bar{l}}$, $\bar{l}$ is the space-location where P is executed. The role of the quantifier subexponentials in the encoding of processes in Definition 2 is key. For instance, recall that $\mathcal{P}[\![\textbf{ask } c \textbf{ then } P]\!]_l = !^{\mathsf{p}(l)}[\Cap s : l.(\mathcal{C}[\![c]\!]_s \multimap \mathcal{P}[\![P]\!]_s)]$. Here $!^{\mathsf{p}(l)}$ specifies the epistemic state $[]_l$ where the process is. On the other hand, $\Cap s : l.$, specifies that one can move the process anywhere in the ideal of l. From the pre-order shown in Figure 2(a), this means moving the process to anywhere outside the box $[]_l$. This corresponds exactly to the operational rule $\mathrm{R_E}$. Moreover, since $\mathsf{p}(l) \in U$, the process is unbounded, thus the encoding $\mathcal{P}[\![\textbf{ask } c \textbf{ then } P]\!]_l$ is not consumed. In fact, the sequent $\mathcal{P}[\![P]\!]_{\bar{l}.i} \longrightarrow \mathcal{P}[\![P]\!]_{\bar{l}}$ is provable for any process P and indexes $\bar{l}$ and i. That is, any process can move to an outer box (see details in the Appendix).

The following proposition shows that $\mathcal{C}[\![\cdot]\!]_{\bar{l}}$, the proposed translation of constraints to formulas in SELL$^{\Cap}$, represents indeed an epistemic constraint system.

Proposition 1. *Let $(\mathcal{C}_e, \vdash_{\Delta_e})$ be an ECS and $\mathcal{C}[\![\cdot]\!]_l$ be as in Definition 4. Then, for any $\bar{l}$:*
1. $\mathcal{C}[\![1]\!]_{\bar{l}} \equiv 1$ *(bottom preserving);*
2. $\mathcal{C}[\![c \wedge d]\!]_{\bar{l}} \equiv \mathcal{C}[\![c]\!]_{\bar{l}} \otimes \mathcal{C}[\![d]\!]_{\bar{l}}$ *(lub preserving);*
3. *If $d \vdash_{\Delta_e} c$ then $!^{\mathsf{c}(\infty)}[\![\Delta_e]\!], \mathcal{C}[\![d]\!]_{\bar{l}} \longrightarrow \mathcal{C}[\![c]\!]_{\bar{l}}$ (monotonicity);*
4.$\mathcal{C}[\![s_i(c)]\!]_{\bar{l}} \longrightarrow \mathcal{C}[\![c]\!]_{nil}$ *(beliefs are facts);*
5. $\mathcal{C}[\![s_i(s_i(c))]\!]_{\bar{l}} \equiv \mathcal{C}[\![s_i(c)]\!]_{\bar{l}}$ *(idempotence).*

Example 1 (Epistemic Reasoning). Let $P = \textbf{tell}(c)$, $Q = \textbf{ask } c \textbf{ then tell}(d)$ and $R = [P \parallel [Q]_b]_a$. The following sequent is provable $\mathcal{P}[\![R]\!]_{nil} \longrightarrow !^{\mathsf{c}(a)}c \otimes !^{\mathsf{c}(nil)}c \otimes \top$. That is, c is *known* by agent a and the external environment (*i.e.*, c is a *fact*). Also, $\mathcal{P}[\![R]\!]_{nil} \longrightarrow !^{\mathsf{c}(a)}d \otimes \top$ since Q also runs in the scope of a. This intuitively means that a knows that b knows that if c is true, then d is true. Therefore, a knows c and d. Furthermore, the agent b does not know c, *i.e.*, the sequent $\mathcal{P}[\![R]\!]_{nil} \longrightarrow !^{\mathsf{c}(b)}c \otimes \top$ *is not* provable.

Theorem 3 (Adequacy). *Let P be an* `eccp` *process, $(\mathcal{C}_e, \vdash_e)$ be an ECS, Ψ be a set of process definitions and let $\mathcal{C}[\![\cdot]\!]_{\bar{l}}$ and $\mathcal{P}[\![\cdot]\!]_{\bar{l}}$ be as in Definition 4. Then $P \Downarrow_c$ iff $!^{\mathsf{c}(\infty)}[\![\Delta_e]\!], !^{\mathsf{p}(\infty)}[\![\Psi]\!], \mathcal{P}[\![P]\!]_{nil} \longrightarrow \mathcal{C}[\![c]\!]_{nil} \otimes \top$.*

This result, besides giving an interesting interpretation of subexponentials as knowledge spaces, gives a proof system for the verification of `eccp` processes. Note that, because of the "$\top$" connective, we only consider the observables of a process regardless whether the final configuration has suspended ask processes.

So far, we have assumed that knowledge is not shared by agents. Next example shows how to handle common knowledge among agents. The approach is similar to the one given in [7], by introducing *announcements* of constraints among group of agents, but by using our proof theoretic framework.

Example 2 (Common Knowledge). Assume a finite set of agents $A = \{a_1, ..., a_n\}$ and a process definition: $global_P() \stackrel{\text{def}}{=} P \parallel [P \parallel global_P()]_{a_1} \parallel ... \parallel [P \parallel global_P()]_{a_n}$. For instance, $global_{\mathbf{tell}(c)}$ makes c available in all spaces and nested spaces involving agents in A. Instead of computing common knowledge by recursion, we can complement the subexponential signature as in Figure 2(b) where for all $\mathcal{S} \subseteq A$, $\bar{i} \preceq a_{\mathcal{S}}$ for any string $\bar{i} \in \mathcal{S}^*$. Then, the announcement of c on the group of agents $\mathcal{S}$ can be represented by $!^{\mathfrak{c}(a_S)}c$. Notice that the sequent $!^{\mathfrak{c}(a_S)}c \longrightarrow !^{\mathfrak{c}(\bar{i})}c$ can be proved for any $\bar{i} \in \mathcal{S}^*$.

4.3 Spaces and Information Confinement

Inconsistent information in CCP arises when considering theories containing axioms such as $c \wedge d \vdash_\Delta 0$. Notice that agents are not allowed to tell or ask false, *i.e.*, 0 is not a (basic) constraint. Unlike epistemic scenarios, in spatial computations, a space can be locally inconsistent and it does not imply the inconsistency of the other spaces (*i.e.*, $s_i(0)$ does not imply $s_j(0)$). Moreover, the information produced by a process in a space is not propagated to the outermost spaces. In [7], spatial computations are specified in spatial CCP (sccp) by considering processes of the form $[P]_i$ as in the epistemic case, but excluding the rule $\mathrm{R_E}$ in the system shown in Figure 1(b). Furthermore, some additional requirements are imposed on the representation of agents' spaces ($s_i(\cdot)$).

Definition 5 (Spatial Cons. Sys. (SCS)). *Let A be a countable set of agent names. An SCS $(C_s, \vdash_{\Delta_s})$ is a CS where, for any $i \in A$, $s_i : C_s \longrightarrow C_s$ is a mapping satisfying bottom and lub preserving, monotonicity and* false containment *(see Proposition 2).*

The set $I = A^* \cup \{nil, \infty\}$ is the same as in the encoding of the epistemic case but the pre-order is much simpler: we only require that for any $\bar{i} \in A^*$, $\bar{i} \preceq \infty$. That is, two different elements of A^* are unrelated. Since sccp does not contain the $\mathrm{R_E}$ rule, processes in spaces are treated linearly, i.e., we set $U = \{\mathfrak{c}(l) \mid l \in I\} \cup \{\mathfrak{p}(\infty)\}$. Moreover, the confinement of spatial information is captured by a different subexponential prefix, namely, by instantiating ∇_l as the prefix $!^l?^l$.

Definition 6 (Spatial constraints in SELL$^{\Cap}$). *The encoding $\mathcal{C}[\![\cdot]\!]_{\bar{l}}$ maps constraints in a SCS into SELL$^{\Cap}$ formulas and it is defined as in Definition 4. $\mathcal{P}[\![\cdot]\!]_l$ is as in Definition 2 extended with $\mathcal{P}[\![[P]_i]\!]_{\bar{l}} = \mathcal{P}[\![P]\!]_{\bar{l}.i}$. In both cases, however, ∇_l is instantiated as $!^l?^l$.*

Differently from the epistemic case, the encoding of $[P]_i$ runs P only the space of i and not outside it. This is captured by the pre-order above and by instantiating ∇_l as $!^l?^l$. Notice that the ideal of all index l in $I \setminus \{\infty\}$ is the singleton $\{l\}$. This means that the only way of instantiating the subexponential quantifier ($\Cap s : l$) in the encoding of processes is by using the l itself. In this way, we confine the information inside the location of agents as states the following proposition.

Proposition 2 (False confinement). *Let $(C_s, \vdash_{\Delta_s})$ be a SCS and $\mathcal{C}[\![\cdot]\!]_{\bar{l}}$ as in Definition 6. Then, monotonicity, bottom and lub preserving items in Proposition 1 hold. Furthermore, for any $\bar{l} \in A^*$, if we assume that $c \wedge d \vdash_{\Delta_s} 0$:*
1. $\mathcal{C}[\![0]\!]_{\bar{l}} \longrightarrow \mathcal{C}[\![c]\!]_{\bar{l}}$ *(any c can be deduced in $\bar{l}$ if its local store is inconsistent);*
2. $\mathcal{C}[\![0]\!]_{\bar{l}} \longrightarrow \mathcal{C}[\![0]\!]_{\bar{l}'}$ *is not provable (false is confined);*

3. $!^{\mathsf{c}(\infty)}\mathcal{C}[\![\Delta_s]\!], \mathcal{C}[\![c]\!]_{\bar{l}}, \mathcal{C}[\![d]\!]_{\bar{l}} \longrightarrow \mathcal{C}[\![0]\!]_{\bar{l}}$ *($\bar{l}$ becomes inconsistent if it contains c and d);*
4. $!^{\mathsf{c}(\infty)}\mathcal{C}[\![\Delta_s]\!], \mathcal{C}[\![c]\!]_{\bar{l}}, \mathcal{C}[\![d]\!]_{\bar{l}'} \longrightarrow \mathcal{C}[\![0]\!]_{\bar{l}}$ *is not provable*
5. $\mathcal{C}[\![c]\!]_{\bar{l}} \longrightarrow c$ *and* $\mathcal{C}[\![c]\!]_{\bar{l}} \longrightarrow \mathcal{C}[\![c]\!]_{nil}$ *are both not provable (local info. is not global).*

Example 3 (Local stores). Let $P = \mathbf{tell}(c)$ and $Q = \mathbf{ask}\ c\ \mathbf{then}\ \mathbf{tell}(d)$. Let $R = [P]_a \parallel [Q]_b$. Then, Q remains blocked since the information c is only available on the space of a. In our encoding, as $!^{\mathsf{c}(a)}?^{\mathsf{c}(a)}c \longrightarrow\ !^{\mathsf{c}(b)}?^{\mathsf{c}(b)}c$ is not provable, the sequent $\mathcal{P}[\![R]\!]_{nil} \longrightarrow !^{\mathsf{c}(b)}?^{\mathsf{c}(b)}d \otimes \top$ is also not provable. Now let $R = [P]_a \parallel [Q]_a$. The process P adds d in the space of a and then, Q can evolve. Thus, $\mathcal{P}[\![R]\!]_{nil} \longrightarrow !^{\mathsf{c}(a)}?^{\mathsf{c}(a)}d \otimes \top$ is provable. Moreover, c does not propagate outside the scope of agent a, *i.e.*, the sequent $\mathcal{P}[\![R]\!]_{nil} \longrightarrow\ !^{\mathsf{c}(nil)}?^{\mathsf{c}(nil)}c \otimes \top$ is not provable. Finally, consider $R = [[P]_a]_b \parallel [Q]_a$. Since $a \npreceq b.a$ and $b.a \npreceq a$, the sequent $!^{\mathsf{c}(b.a)}?^{\mathsf{c}(b.a)}c \longrightarrow\ !^{\mathsf{c}(a)}?^{\mathsf{c}(a)}c$ is not provable. Thus, the process Q inside the agent a remains blocked, *i.e.*, the sequent $\mathcal{P}[\![R]\!]_{nil} \longrightarrow !^{\mathsf{c}(a)}?^{\mathsf{c}(a)}d \otimes \top$ is not provable. This intuitively means that the space that b confers to a may behave differently (*i.e.*, it contain different information) from the own space of a. The same reasoning applies for the process $R = [[P]_a]_a \parallel [Q]_a$. This means that, in general, the space of a inside a is different from the space a ($a \npreceq a.a$). If we want spaces to be idempotent, we simply need to add the equivalence $a.a \approx a$ to the pre-order.

Theorem 4 (Adequacy). *Let P be an* sccp *process,* $(\mathcal{C}_s, \vdash_s)$ *be an SCS,* Ψ *be a set of process definitions and* $\mathcal{C}[\![\cdot]\!]_{\bar{l}}$ *and* $\mathcal{P}[\![\cdot]\!]_{\bar{l}}$ *be as in Definition 6. Then P* $\Downarrow_c$ *iff*
$!^{\mathsf{c}(\infty)}[\![\Delta_s]\!], !^{\mathsf{p}(\infty)}[\![\Psi]\!], \mathcal{P}[\![P]\!]_{nil} \longrightarrow \mathcal{P}[\![c]\!]_{nil} \otimes \top$.

4.4 Temporal Modalities

Saraswat *et al.* proposed in [17] timed-CCP (tcc), an extension of CCP for the specification of reactive systems. In tcc, time is conceptually divided into *time intervals* (or *time units*). In a particular time interval, a CCP process P gets an input c from the environment, it executes with this input as the initial *store*, and when it reaches its resting point, it *outputs* the resulting store d to the environment. The resting point determines also a residual process Q which is then executed in the next time unit. The resulting store d is not automatically transferred to the next time unit. Hence, computations during a time-unit proceed monotonically (by adding information to the store), but outputs of two different time-units are not supposed to be related to each other. This view of *reactive computation* is akin to synchronous languages such as Esterel, where the system reacts continuously with the environment at a rate controlled by the environment.

The syntax of CCP is extended in tcc by including temporal operators:

$$P, Q ::= \cdots \mid \circ P \mid \Box P$$

The process $\circ P$ delays the execution of P in one time-unit. The replication $\Box P$ means $P \parallel \circ P \parallel \circ \circ P \parallel \ldots$, *i.e.*, unboundly many copies of P, but one at a time.

In tcc, recursive calls are assumed to be guarded by a "$\circ$" process to avoid non-terminating sequences of recursive calls during a time-unit. Recursive procedures can then be encoded via replication (see [8]) and we omit them here. We also distinguish between internal ($\longrightarrow$) and observable ($\Longrightarrow$) transitions. The internal transition of the form $(X; \Gamma; c) \longrightarrow (X'; \Gamma'; c')$ is similar to that of CCP plus the additional rules for the timed constructs (see Figure 1(c)). A process $\Box P$ executes one copy of P in the current time-unit and then, executes again $\Box P$ in the next time-unit (Rule $R_{\Box}$). The seemingly missing rule for $\circ P$ is given by the observable transition relation.

Assume that $(\emptyset; \Gamma; c) \longrightarrow^* (X; \Gamma'; c') \not\longrightarrow$. We say that (the parallel composition in) Γ under input c outputs $\exists X.c'$ and we write $\Gamma \xRightarrow{(c,\exists X.c')} \Upsilon$ where $\Upsilon = (\textbf{local}\, X)\, F(\Gamma')$ corresponds to the *future* of Γ' (see Figure 1(c)). Roughly, the future function drops any ask whose guard cannot be entailed from the final store. Furthermore, it unfolds the processes guarded by "$\circ$". Note that $F(\cdot)$ does not consider the processes $\textbf{tell}(c)$, $\Box P$ and $(\textbf{local}\, x)\, P$ since all of them have an internal transition. Therefore, in a final configuration $(X, \Gamma, c) \not\longrightarrow$ they must occur within the scope of "$\circ$". If, $\Gamma = \Gamma_1 \xRightarrow{(1,c_1)} \Gamma_2 ... \Gamma_n \xRightarrow{(1,c_n)} \Gamma_{n+1}$ and $c_n \vdash_\Delta c$, we say that Γ outputs c and we write $\Gamma \Downarrow_c$.

As before, we use a specific subexponential signature but with only two families $\mathfrak{c}$ and $\mathfrak{p}$ as procedure calls are not required as we explained before:

$$I = \{\infty, nil\} \cup \{i, i{+} \mid i \geq 1\} \qquad U = \{\mathfrak{c}(i), \mid i \in I\} \cup \{\mathfrak{p}(\infty)\}.$$

Notice that only the subexponentials marking constraints, $\mathfrak{c}(\cdot)$, and boxed processes, $\mathfrak{p}(\infty)$, are unbounded, as they can be used as many times as needed. On the other hand, subexponentials processes, $\mathfrak{p}(\cdot)$, are bounded.

The pre-order is depicted in Figure 2(c), where a descending chain is formed with the numbers marked with "+". Intuitively, the subexponential i is used to specify a given time-unit while $i{+}$ is used to store processes valid *from* the time-unit i on. This chain captures the semantics of $\Box P$: if $\Box P$ appears in time i, then P should be available at any future time. Formally, that chain allows us to specify, by using a quantifier $\Cap l : i{+}$, that P can be instantiated anywhere in the ideal of $i{+}$, *i.e.*, in future time units.

Definition 7 (Timed Constraints in SELL$^{\Cap}$). *We instantiate ∇_l as $!^l ?^l$. The interpretation $\mathcal{C}[\![\cdot]\!]_l$ is as in Definition 2, while we modify $\mathcal{P}[\![\cdot]\!]_l$ as follows:*

$$\mathcal{P}[\![\textbf{tell}(c)]\!]_l = !^{\mathfrak{p}(l)}\mathcal{C}[\![c]\!]_l \qquad \mathcal{P}[\![\textbf{ask}\ c\ \textbf{then}\ P]\!]_l = !^{\mathfrak{p}(l)}(\mathcal{C}[\![c]\!]_l \multimap \mathcal{P}[\![P]\!]_l)$$
$$\mathcal{P}[\![(\textbf{local}\, \overline{x})\, P]\!]_l = !^{\mathfrak{p}(l)}(\exists \overline{x}.(\mathcal{P}[\![P]\!]_l)) \qquad \mathcal{P}[\![\circ P]\!]_i = \mathcal{P}[\![P]\!]_{i+1}$$
$$\mathcal{P}[\![\Box P]\!]_i = !^{\mathfrak{p}(\infty)} \Cap l : i{+}(\mathcal{P}[\![P]\!]_l)$$

The encoding of the non-temporal operators are similar as before, just that we do not need the subexponential quantification. While the encoding of $\circ P$ is straightforward, the encoding of $\Box P$ is more interesting. If the process $\Box P$ is executed in the time-unit i, then the encoding of P must be available in subexponentials representing the subsequent time-units. For example, let $P = \Box\, \textbf{ask}\ c\ \textbf{then}\ Q$. The process P must execute Q in all time-units $j \geq i$ whenever c can be deduced in j. We make use of universal quantification over locations to capture this behavior.

We note that the observable transition ($\Longrightarrow$) results from a finite sequence of internal ($\longrightarrow$) transitions (R_{Obs} in Figure 1(c)). Proof theoretically, detecting that a given configuration cannot longer be reduced is problematic in general. In fact, the adequacy theorem below is not on the level of derivations, as our previous theorems, but only at the level of provability [11]: P outputs c iff one can prove that there is a time-unit where c holds. Key for proving this theorem is the use of $!^l ?^l$ prefixes as for the `sccp` case. More precisely, facts are confined to a determinate time unit: any formula derived in a subexponential representing a time unit is not spilled to other subexponentials, unless explicitly specified. We note also that we consider here the monotonic fragment of `tcc` *i.e.*, we do not include the time-out **unless** c $(\circ P)$ that executes P in the next time-unit if c cannot be deduced. This operator lacks of a proper proof theoretic semantics: the reduction to P amounts to showing that there is no proof of c.

Theorem 5 (Adequacy). *Let P be a timed process, (C_t, Δ_t) be a CS and $\mathcal{P}[\![\cdot]\!]_l$ as in Definition 7. Then $P \Downarrow_c$ iff $!^{\mathrm{c}(\infty)}[\![\Delta_t]\!], \mathcal{P}[\![P]\!]_1 \longrightarrow \Cup l : 1{+}.!^{\mathrm{c}(l)}?^{\mathrm{c}(l)}c \otimes \top$.*

5 Concluding Remarks

In this paper, we have introduced quantification over subexponentials in linear logic with subexponentials and proved that cut elimination is admissible for the resulting system (SELL$^{\Cap}$), reflecting a pleasant duality with the standard quantification over terms. We demonstrated that SELL$^{\Cap}$ is, indeed, a powerful tool for specifying concurrent systems involving modalities by proposing novel encodings for CCP-calculi featuring epistemic, spatial and timed modalities, hence providing a proof-theoretic foundation for those calculi.

We believe that there are many directions to follow from this work. For instance, in our encoding, we did not need the generation of *fresh* subexponential variables by using the rules $\Cap_R$ and $\Cup_L$. As done with *eigenvariables* for modeling nonces in security protocol [3], it seems possible to create *new* modalities, such as new spaces or new agents not related to the ones already created as in the Ambient Calculus. This would solve the limitation of `sccp` and `eccp` in [7] where the set of agents is fixed.

Although this paper does not consider non-determinism, some form of it can be easily captured. For instance a non-deterministic choice of the form $P + Q$ can be encoded as the formula $F = \mathcal{P}[\![P]\!]_l \;\&\; \mathcal{P}[\![Q]\!]_l$. In fact, by adding and moving subexponential bangs, it is possible to model precisely don't-care and don't-know choices [10]. Thus, non-determinism (not-considered in [7] for neither `sccp` nor `eccp`) can be also introduced in `sccp`, where processes do not contract. For a second example, consider the `ntcc` calculus [8] which extends `tcc` with guarded non-deterministic choices and asynchrony. For the later, the process $\star P$ represents an arbitrary long, but finite delay for the activation of P; that is, $\star P$ non-deterministically chooses $n \geq 0$ and behaves as $\circ^n P$ (see Rule $\mathrm{R}_\star$ in Figure 1(c)). It seems possible to encode this behavior by extending $\mathcal{P}[\![\cdot]\!]_l$ with the following case: $\mathcal{P}[\![\star P]\!]_i = \Cup l : i{+}.\mathcal{P}[\![P]\!]_l$. Roughly, if $\star P$ is executed in time-unit i, then *there is* a subexponential j such that $j \preceq i{+}$ (*i.e.*, a future time-unit j) and the encoding of P holds using that subexponential.

However, for adequacy, some care has to be taken to avoid undesired interactions between $\square$ and a non-deterministic processes P (containing $\star$ or $+$): $\mathcal{P}[\![\square P]\!]_l$ yields a formula of the form $!^{\mathrm{p}(\infty)}F$. Due to the connective $!^{\mathrm{p}(\infty)}$ that precedes F, by contraction, it is possible to have a derivation with two copies of F representing the process $P \parallel P$ that does not behave as P, thus breaking adequacy.

We think that CCP research can greatly profit from this work. Due to the modularity of our encoding, it seems possible to design variants of CCP by simply configuring the subexponentials differently or by using different prefixes. For instance, by using a mix of linear and unbounded $\mathrm{c}(\cdot)$ subexponentials, it is possible to specify a spatial CCP language that allows constraints to be consumed. It also seems possible to design CCP models that allow for the creation of new spaces or agents. Finally, one could explore the use of these new variants of CCP and their declarative reading as SELL$^{\Cap}$ formulas to reason about other models of concurrency (see e.g., [14] that studies different fragments of the asynchronous π-calculus through the CCP model).

Acknowledgments: We thank Sophia Knight, Frank Valencia and Jorge A. Perez for helpful discussions. Nigam was supported by CNPq. This work has been (partially) supported by Colciencias (Colombia), and by Digiteo and DGAR (École Polytechnique) funds for visitors.

References

1. Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. of Logic and Computation*, 2(3):297–347, 1992.
2. Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. The structure of exponentials: Uncovering the dynamics of linear logic proofs. In *Kurt Gödel Colloquium*, *LNCS*, 1993.
3. Nancy A. Durgin, Patrick Lincoln, John C. Mitchell, and Andre Scedrov. Multiset rewriting and the complexity of bounded security protocols. *JCS*, 12(2):247–311, 2004.
4. François Fages, Paul Ruet, and Sylvain Soliman. Linear concurrent constraint programming: Operational and phase semantics. *Information and Computation*, 165(1):14–41, 2001.
5. Gerhard Gentzen. Investigations into logical deductions. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131. North-Holland, Amsterdam, 1969.
6. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
7. Sophia Knight, Catuscia Palamidessi, Prakash Panangaden, and Frank D. Valencia. Spatial and epistemic modalities in constraint-based process calculi. In *CONCUR*, 2012.
8. Mogens Nielsen, Catuscia Palamidessi, and Frank D. Valencia. Temporal concurrent constraint programming: Denotation, logic and applications. *Nordic Journal of Computing*, 9(1):145–188, 2002.
9. Vivek Nigam. On the complexity of linear authorization logics. In *LICS*, 2012.
10. Vivek Nigam and Dale Miller. Algorithmic specifications in linear logic with subexponentials. In PPDP, 2009.
11. Vivek Nigam and Dale Miller. A framework for proof systems. *J. Autom. Reasoning*, 45(2):157–188, 2010.
12. Vivek Nigam, Elaine Pimentel, and Giselle Reis. Specifying proof systems in linear logic with subexponentials. *Electr. Notes Theor. Comput. Sci.*, 269:109–123, 2011.
13. Carlos Olarte, Camilo Rueda, and Frank D. Valencia. Models and emerging trends of concurrent constraint programming. *Constraints*, 2013.
14. Catuscia Palamidessi, Vijay A. Saraswat, Frank D. Valencia, and Björn Victor. On the expressiveness of linearity vs persistence in the asychronous pi-calculus. In *LICS*, 2006.
15. Vijay A. Saraswat, Martin C. Rinard, Prakash Panangaden. Semantic foundations of concurrent constraint programming. In POPL, 1991.
16. Vijay A. Saraswat. *Concurrent Constraint Programming*. MIT Press, 1993.
17. Vijay A. Saraswat, Radha Jagadeesan, and Vineet Gupta. Timed default concurrent constraint programming. *J. Symb. Comput.*, 22(5/6):475–520, 1996.
18. Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A concurrent logical framework I: Judgments and properties. TR CMU-CS-02-101, CMU, 2003.

A Cut Elimination

In the following we prove Theorem 1: For any signature Σ, the proof system SELL$^{\Cap}$ admits cut-elimination.

Proof. We show only the new principal case that arises from the inclusion of $\Cap, \Cup$. The reduction follows the same idea as for the first-order quantifiers, *i.e.*, the deduction

$$\dfrac{\dfrac{\overset{\Xi}{\mathcal{A}, l_e : a; \mathcal{L}; \Gamma \longrightarrow P[l_e/x]}}{\mathcal{A}; \mathcal{L}; \Gamma \longrightarrow \Cap x : a.P}\ \Cap_R \quad \dfrac{\overset{\Xi'}{\mathcal{A}; \mathcal{L}; \Gamma, P[l/x] \longrightarrow G}}{\mathcal{A}; \mathcal{L}; \Gamma, \Cap x : a.P \longrightarrow G}\ \Cap_L}{\mathcal{A}; \mathcal{L}; \Gamma \longrightarrow G}\ cut$$

is replaced by

$$\dfrac{\overset{\Xi[l/l_e]}{\mathcal{A}; \mathcal{L}; \Gamma \longrightarrow P[l/x]} \quad \overset{\Xi'}{\mathcal{A}; \mathcal{L}; \Gamma, P[l/x] \longrightarrow G}}{\mathcal{A}; \mathcal{L}; \Gamma \longrightarrow G}\ cut$$

We can show by induction that the object $\Xi[l/l_e]$ is indeed a SELL$^{\Cap}$ proof. The only interesting cases are for the right introduction rule for $!^{\dagger(s:a)}$ and the left introduction rule for $?^{\dagger(s:a)}$. We show only the former, as the latter follows similarly. There are two sub-cases to consider, when s is the fresh variable l_e or when s is some existing subexponential b. We only show the former case, as the latter follows similarly. Assume that the formula $!^{\dagger(l_e:a)}F$ is introduced. Then all formulas in the context are of the form $!^{\dagger(s_i:a_i)}Q_i$ (with $s_i \in I$ and with $a \preceq s_i$), or $!^{\dagger(l_e:a)}Q$. As l is in the ideal of a, the formula $!^{\dagger(l:a)}F$ can be introduced and $\Xi[l/l_e]$ is a proof.

B Focused Proof System for SELL$^{\Cap}$

In this section we prove the adequacy results for the epistemic, spatial and timed systems presented in Sections 4.2, 4.3 and 4.4 respectively. Since the proofs rely on the completeness of the focusing discipline [1], we first introduce the focused proof system for SELL$^{\Cap}$, called $SELLF^{\Cap}$.

The focusing proof system for the multiplicative fragment of SELL$^{\Cap}$ with $\top$ is depicted in Figure 3. It is a straightforward generalization of Andreoli's focused proof system, but for intuitionistic linear logic.

Before we introduce the system, we need some more terminology. We classify all formulas as negative the formulas whose main connective is $\multimap, \forall, ?^l$ and the unit $\bot$ as *negative* as well as the negative polarity atomic formulas. The remaining formulas are classified as *positive*.

As in the focused system for classical linear logic with subexponentials [10], we make use of indexed contexts $\mathcal{K}$ that maps a subexponential index to multiset of formulas, *e.g.*, if l is a subexponential index, then $\mathcal{K}[l]$ is a multiset of formulas, where intuitively they are all marked with $!^l$. We also make use of the operations on contexts depicted in Figure 4.

The rules of the system are depicted in Figure 3 and it contains four types of sequents.

- $[\mathcal{K} : \Gamma], \Delta \longrightarrow \mathcal{R}$ is an unfocused sequent, where $\mathcal{R}$ is either a bracketed context $[F]$ or an unbracketed context. Here Γ contains only atomic or negative formulas, while $\mathcal{K}$ is the indexed context containing formulas whose main connective is a $!^l$ for some subexponential index l.
- $[\mathcal{K} : \Gamma] \longrightarrow [F]$ is a sequent representing the end of the negative (or asynchronous) phase.
- $[\mathcal{K} : \Gamma] -_F\!\!\rightarrow$ is a sequent focused on the right.
- $[\mathcal{K} : \Gamma] \overset{F}{\rightarrow} G$ is a sequent focused on the left.

As one can see from inspecting the proof system in Figure 3, proofs are composed of two alternating phases, a negative phase, containing sequent of the first form above and where all the negative non-atomic formulas to the right and all the positive non-atomic formulas to the left are introduced. Atomic or positive formulas to the right and atomic or negative formulas to the left are bracketed by the $[]_l$ and $[]_r$ rules, while formulas whose main connective is a $!^l$ are added to the indexed context $\mathcal{K}$ by rule $!^l{}_L$. The second type of sequent above marks the end of the negative phase. A positive phase starts by using the decide rules to focus either on a formula on the right or on the left, resulting on the third and fourth sequents above. Then one introduces all the positive formulas to the right and the negative formulas to the left, until one is focused either on a negative formula on the right or a positive formula on the left. This point marks the end of the positive phase by using the R_l and R_r rules and starting another negative phase.

The novel rules with respect to the focused proof system for *SELL* are the rules for $\Cap$ and $\Cup$. They behave exactly as the first-order quantifiers: the $\Cap_r$ and $\Cup_l$ are negative becuase they are invertible, while $\Cap_l$ and $\Cup_r$ are positive because they are not-invertible. Notice that in the premise of $\Cap_r$ and $\Cup_l$ rules, the context $\mathcal{K}$ is extended to $\mathcal{K}_{l_e}$ with new indices $\{\mathfrak{f}(l_e) \mid \mathfrak{f} \in F\}$ generated due to the creation of fresh subexponential constant, and since no formulas are yet in these contexts, these are mapped to the empty set.

One can prove the following completeness theorem following the same lines as the proof in Nigam's thesis for the focused proof system for classical linear logic with subexponentials.

Theorem 6. *The sequent* $\longrightarrow G$ *is provable in* $SELL^{\Cap}$ *if and only if the sequent* $[\mathcal{K} : \cdot], \cdot \longrightarrow G$ *is also provable in the focused proof system depicted in Figure 3, where* $\mathcal{K}[l] = \emptyset$ *for all indices l.*

B.1 Adequacy of CCP

We start with the adequacy of CCP. We will keep the proofs general enough so that they can be easily adapted for the encoding of the CCP extensions. Recall that a configuration $(X; \Gamma; c)$ is encoded by a sequent of the form:

$$!^{\mathrm{c}(\infty)}[\![\Delta]\!], !^{\mathrm{p}(\infty)}[\![\Psi]\!], \mathcal{P}[\![\Gamma]\!]_{nil}, \bigvee_{\mathrm{c}(l_1)} A_1, \cdots, \bigvee_{\mathrm{c}(l_n)} A_n \longrightarrow G$$

This was shown by using using the fact that the left introduction rules of $\exists$ and $\otimes$ are invertible (negative). By using the same argument, $\mathcal{P}\llbracket\Gamma\rrbracket_{nil}$ reduces to

$$!^{\mathfrak{p}(l_1)}\mathcal{P}\llbracket P\rrbracket_{l_1},\ldots,!^{\mathfrak{p}(l_n)}\mathcal{P}\llbracket P\rrbracket_{l_n},!^{\mathfrak{d}(l'_1)}p_1(\overline{x}_1),\ldots,!^{\mathfrak{d}(l'_m)}p_m(\overline{x}_m)$$

So in fact, we can re-write the sequent above as follows (abusing here a bit notation where the context $\mathcal{K}$ is split into three contexts):

$$[C,\mathcal{D},\mathcal{P}] \longrightarrow [G]$$

where $C,\mathcal{D}$ and $\mathcal{P}$ are contexts containing all formulas marked, respectively, with bangs of the $\mathfrak{c}$, $\mathfrak{d}$ and $\mathfrak{p}$. For example, if $C[\mathfrak{c}(a)] = \{F,G\}$, then the formulas $!^{\mathfrak{c}(a)}F$ and $!^{\mathfrak{c}(a)}G$ are in the context. Moreover, $C[\mathfrak{p}(a)] = C[\mathfrak{d}(a)] = \emptyset$ for any a, similarly with the contexts $\mathcal{D}$ and $\mathcal{P}$. Finally, the context $\mathcal{D}$ contains only atomic procedure calls of the form $p(\overline{x})$ and $\mathcal{P}$ contains the encoding of processes $\mathcal{P}\llbracket P\rrbracket_l$.

We now show that each focused phase corresponds exaclty to one rule in CCP's operational semantics.

• Case **tell**(c): Suppose $\mathcal{P}\llbracket\mathbf{tell}(c)\rrbracket_a = !^{\mathfrak{p}(a)}\Cap s : a.C\llbracket c\rrbracket_s$ is in the context. Then the focused derivation obtained by focusing on this formula is necessarily as follows:

$$\cfrac{\cfrac{\cfrac{[C',\mathcal{D},\mathcal{P}] \longrightarrow [G]}{[C,\mathcal{D},\mathcal{P}],C\llbracket c\rrbracket_s \longrightarrow [G]}\; n\times\exists_l, m\times\otimes_l, j\times !_l}{[C,\mathcal{D},\mathcal{P}] \xrightarrow{\Cap s:a.C\llbracket c\rrbracket_s} [G]}\; \Cap_L, R_l}{[C,\mathcal{D},\mathcal{P} +_{\mathfrak{p}(a)} \Cap s : a.C\llbracket c\rrbracket_s] \longrightarrow [G]}\; D$$

where the encoding, $C\llbracket c\rrbracket_s$, of the constraint c is completely decomposed into the formulas of the form $\nabla_{\mathfrak{c}(l_1)} A_1,\ldots,\nabla_{\mathfrak{c}(l_n)} A_n$, which are then moved to the context C. Thus, from bottom-up the derivation above corresponds exactly to the operational semantics of **tell**(c), where c is added to the store.

Notice that in the derivation above, we assumed that $\mathfrak{p}(a)$ does not contract. If it can contract, then $\mathcal{P} +_{\mathfrak{p}(a)} C\llbracket c\rrbracket_a$ should persist in the derivation.

• Case **ask**(c): Suppose $\mathcal{P}\llbracket\mathbf{ask}\ c\ \mathbf{then}\ P\rrbracket_a = !^{\mathfrak{p}(a)}\Cap s : a(C\llbracket c\rrbracket_s \multimap \mathcal{P}\llbracket P\rrbracket_s)$ is in the context. Then focusing on this formula:

$$\cfrac{\cfrac{\begin{matrix}\pi_1\\ [C,\mathcal{D},\mathcal{P}]-_{C\llbracket c\rrbracket_s}\!\!\rightarrow\end{matrix} \qquad \cfrac{[C,\mathcal{D},\mathcal{P} +_{\mathfrak{p}(s)} (\mathcal{P}\llbracket P\rrbracket_s)] \longrightarrow [G]}{[C,\mathcal{D},\mathcal{P}] \xrightarrow{(\mathcal{P}\llbracket P\rrbracket_s)} [G]}\; R_l, !^{\mathfrak{p}(s)}{}_l}{[C,\mathcal{D},\mathcal{P}] \xrightarrow{\Cap s:a(C\llbracket c\rrbracket_s\multimap\mathcal{P}\llbracket P\rrbracket_s)} [G]}\; \Cap_l, \multimap_l}{[C,\mathcal{D},\mathcal{P} +_{\mathfrak{p}(a)} \Cap s : a(C\llbracket c\rrbracket_s \multimap \mathcal{P}\llbracket P\rrbracket_s)] \longrightarrow [G]}\; D$$

Since $C\llbracket c\rrbracket_s$ is of the form $\exists\overline{x}.\left[\nabla_{\mathfrak{c}(l_1)} A_1 \otimes\cdots\otimes \nabla_{\mathfrak{c}(l_n)} A_n\right]$, containing only positive formulas, it will be totally decomposed until getting to (possibly many) sequents of the form $[C,\mathcal{D},\mathcal{P}]-_{\nabla_{\mathfrak{c}(l)} A}\!\!\rightarrow$. Since $\nabla_{\mathfrak{c}(l)}$ is of the form $!^{\mathfrak{c}(l)}[?^l]A$, where $[?^l]$ may appear or not, depending on the instantiation of $\nabla_{\mathfrak{c}(l)}$, π_1 will necessarily end with derivations of the form:

$$\cfrac{\begin{matrix}\pi_2\\ [C] \longrightarrow [?^l]A\end{matrix}}{[C,\mathcal{D},\mathcal{P}]-_{!^{\mathfrak{c}(l)}[?^l]A}\!\!\rightarrow}\; !^{\mathfrak{c}(l)}{}_r$$

The important thing to notice is that the contexts $\mathcal{D}$ and $\mathcal{P}$ are necessarily weakened in the premise. This is because $\mathfrak{c}()$ is not related to $\mathfrak{p}()$ or $\mathfrak{d}()$. Hence, since A is atomic, it should be provable from the atomic formulas C_{atom} in C and the theory Δ. That is, $C_{atom} \vdash_{\Delta} A$. Finally, observe that formulas in C_{atom} are constraints, coming from *tells*, as described in the last item. Thus, from bottom-up the derivation above corresponds exactly to the operational semantics of **ask** c **then** P, where c is deduced from the store and only then P can be executed.

- Case **local**(c): Suppose $\mathcal{P}[\![(\mathbf{local}\,\overline{x})\,P]\!]_a = !^{\mathfrak{p}(a)}(\Cap s : a.\exists\overline{x}.(\mathcal{P}[\![P]\!]_s))$ is in the context:

$$\dfrac{\dfrac{\dfrac{[C,\mathcal{D},\mathcal{P} +_{\mathfrak{p}(s)} (\mathcal{P}[\![P[\overline{x}/\overline{y}]]\!]_s)] \longrightarrow [G]}{[C,\mathcal{D},\mathcal{P}], \exists\overline{x}.(\mathcal{P}[\![P]\!]_s \longrightarrow [G]}\; n\times\exists_l, !^{\mathfrak{p}(s)}{}_l}{[C,\mathcal{D},\mathcal{P}] \xrightarrow{(\Cap s:a.\exists\overline{x}.(\mathcal{P}[\![P]\!]_s))} [G]}\; \Cap_l, R_l}{[C,\mathcal{D},\mathcal{P} +_{\mathfrak{p}(a)} (\Cap s : a.\exists\overline{x}.(\mathcal{P}[\![P]\!]_s))] \longrightarrow [G]}\; D$$

Thus, this derivation corresponds exactly to the operational semantics of $(\mathbf{local}\,\overline{x})\,P$, where $P[\overline{x}/\overline{y}]$ can be executed for fresh variables $\overline{y}$.

- Case process definitions: Focusing on the formula $\Cap l : \infty.\forall\overline{x}.(\bigtriangledown_{\mathfrak{d}(l)}\, p(\overline{x}) \multimap \mathcal{P}[\![P]\!]_l)$, we obtain the derivation below. As in the case for ask, $\bigtriangledown_{\mathfrak{c}(l)}$ is of the form $!^{\mathfrak{c}(l)}[?^l]p(\overline{x})$, where $[?^l]$ may appear or not, depending on the instantiation of $\bigtriangledown_{\mathfrak{c}(l)}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{[\mathcal{D}] \longrightarrow [?^a]p(\overline{x})}}{[C,\mathcal{D},\mathcal{P}] \xrightarrow{}_{!^{\mathfrak{c}(a)}[?^a]p(\overline{x})}}\; !^{\mathfrak{d}(a)}{}_l \qquad \dfrac{[C,\mathcal{D},\mathcal{P} +_{\mathfrak{p}(a)} (\mathcal{P}[\![P]\!]_a)] \longrightarrow [G]}{[C,\mathcal{D},\mathcal{P}] \xrightarrow{(\mathcal{P}[\![P]\!]_a)} [G]}\; R_l, !^{\mathfrak{p}(a)}{}_l}{[C,\mathcal{D},\mathcal{P}] \xrightarrow{\Cap l:\infty.\forall\overline{x}.(\bigtriangledown_{\mathfrak{d}(l)}\, p(\overline{x}) \multimap \mathcal{P}[\![P]\!]_l)} [G]}\; \Cap_l, \forall_l, \multimap_l}{[C,\mathcal{D},\mathcal{P} +_{\mathfrak{p}(\infty)} \Cap l : \infty.\forall\overline{x}.(\bigtriangledown_{\mathfrak{d}(l)}\, p(\overline{x}) \multimap \mathcal{P}[\![P]\!]_l)] \longrightarrow [G]}\; D$$

Note that, since $\mathfrak{d}()$ is not related to $\mathfrak{p}()$ or $\mathfrak{c}()$, the contexts $\mathcal{P}$ and C are weakened in the rule $!^{\mathfrak{d}(a)}{}_l$. Hence, since $p(\overline{x})$ is atomic, it should be provable from the formulas in $\mathcal{D}$. But all the formulas in $\mathcal{D}$ are atomic, so it should be the case that $p(\overline{x}) \in \mathcal{D}$ and hence the derivation on the right ends with an initial axiom. Thus, from bottom-up the derivation above corresponds exactly to the operational semantics of processes declarations, where $p(\overline{x})$ is substituted by its defined process P.

B.2 Adequacy of sccp

We now analyze the possible cases for spatial constraint systems. The only difference from the CCP case is the possibility of applying the rule $\mathrm{R_S}$.

Assume that $!^a\mathcal{P}[\![P]\!]_a$ is the context, which corresponds to a process $[P]_i$. By induction on P, we can show that if $P \longrightarrow P'$, once we focus on the encoding $!^a\mathcal{P}[\![P]\!]_a$, then the encoding $!^a\mathcal{P}[\![P']\!]_a$ is also in the context. The cases are similar to the ones shown in the proof of adequacy of CCP. For example, when $P = \mathbf{tell}(c)$, consider the focused derivation below obtained when focusing on its encoding $!^aC[\![C]\!]_a$:

$$\begin{array}{c} [C', \mathcal{D}, \mathcal{P}] \longrightarrow [G] \\ \hline\hline [C, \mathcal{D}, \mathcal{P}], \mathcal{C}[\![c]\!]_a \longrightarrow [G] \\ \hline\hline [C, \mathcal{D}, \mathcal{P}] \xrightarrow{\Cap s:a.\mathcal{C}[\![c]\!]_s} [G] \\ \hline [C, \mathcal{D}, \mathcal{P} +_{\mathfrak{p}(a)} \Cap s : a.\mathcal{C}[\![c]\!]_s] \longrightarrow [G] \end{array} \begin{array}{l} n \times \exists_l, m \times \otimes_l, j \times !_l \\ \Cap_L, R_l \\ D \end{array}$$

As before, the encoding of $\mathcal{C}[\![c]\!]_a$ is of the form $\bigtriangledown_{\mathfrak{c}(a.\bar{l}_1)} A_1, \ldots, \bigtriangledown_{\mathfrak{c}(a.\bar{l}_2)} A_n$, where all constraints are in the box a. The remaining cases are similar. Notice that due to the pre-order, the only instantiation of the quantified subexponential $s : a$ is when $s = a$.

B.3 Adequacy of eccp

For eccp the principle is similar to the spatial case. However, now processes are unbounded and they can be moved outside boxes. This is exactly what happens in the encoding. For example, for the case of **tell**(c) focusing on its encoding is similar as before:

$$\begin{array}{c} [C', \mathcal{D}, \mathcal{P} +_{\mathfrak{p}(a)} \Cap s : a.\mathcal{C}[\![c]\!]_s] \longrightarrow [G] \\ \hline\hline [C, \mathcal{D}, \mathcal{P} +_{\mathfrak{p}(a)} \Cap s : a.\mathcal{C}[\![c]\!]_s], \mathcal{C}[\![c]\!]_s \longrightarrow [G] \\ \hline\hline [C, \mathcal{D}, \mathcal{P} +_{\mathfrak{p}(a)} \Cap s : a.\mathcal{C}[\![c]\!]_s] \xrightarrow{\Cap s:a.\mathcal{C}[\![c]\!]_s} [G] \\ \hline [C, \mathcal{D}, \mathcal{P} +_{\mathfrak{p}(a)} \Cap s : a.\mathcal{C}[\![c]\!]_s] \longrightarrow [G] \end{array} \begin{array}{l} \\ \Cap_L, R_l \\ D \end{array}$$

Notice that differently from before, the process definition is unbounded. Thus the formula $\Cap s : a.\mathcal{C}[\![c]\!]_s$ is contracted. Moreover, there is choice of where to place the result of executing the process, that is, of adding to the store. In particular, $s : a$ can be in the ideal of a, that is, anywhere outside the box represented by a.

In fact, we can prove that $\mathcal{P}[\![P]\!]_{\bar{l}.i} \longrightarrow \mathcal{P}[\![P]\!]_{\bar{l}}$ is provable for any process P and indices $\bar{l}$ and i. That is, a process in a box $[]_{\bar{l}.i}$ can move to the box $[]_{\bar{l}}$. The proof is by induction on P. The case for **ask** c **then** P is shown below:

$$\begin{array}{c} \hline \mathcal{C}[\![c]\!]_s \multimap \mathcal{P}[\![P]\!]_s \longrightarrow \mathcal{C}[\![c]\!]_s \multimap \mathcal{P}[\![P]\!]_s \\ \hline\hline \Cap s : \bar{l}.i.\mathcal{C}[\![c]\!]_s \multimap \mathcal{P}[\![P]\!]_s \longrightarrow \Cap s : \bar{l}.\mathcal{C}[\![c]\!]_s \multimap \mathcal{P}[\![P]\!]_s \\ \hline !^{\mathfrak{p}(\bar{l}.i)} \Cap s : \bar{l}.i.\mathcal{C}[\![c]\!]_s \multimap \mathcal{P}[\![P]\!]_s \longrightarrow \Cap s : \bar{l}.\mathcal{C}[\![c]\!]_s \multimap \mathcal{P}[\![P]\!]_s \\ \hline !^{\mathfrak{p}(\bar{l}.i)} \Cap s : \bar{l}.i.\mathcal{C}[\![c]\!]_s \multimap \mathcal{P}[\![P]\!]_s \longrightarrow !^{\mathfrak{p}(\bar{l})} \Cap s : \bar{l}.\mathcal{C}[\![c]\!]_s \multimap \mathcal{P}[\![P]\!]_s \end{array} \begin{array}{l} I \\ \Cap_r, \Cap_l \\ {!^{\mathfrak{p}(\bar{l}.i)}}_l \\ {!^{\mathfrak{p}(\bar{l})}}_r \end{array}$$

B.4 Timed Behavior

Timed behavior is quite different from the cases analyzed before since there are two notions of barbs: internal and observable.

From the proof theoretical point of view, though, the cases are similar and simpler than the ones described in the CCP case since information is confined to time units due to the use of question marks and the encoding of not boxed processes does not have quantification over subexponentials. The only different cases are focusing on $\mathcal{P}[\![\circ P]\!]_i = \mathcal{P}[\![P]\!]_{i+1}$ and $\mathcal{P}[\![\Box\, P]\!]_i = !^{\mathfrak{p}(\infty)} \Cap l : i + (\mathcal{P}[\![P]\!]_l)$ but these cases are also trivial.

On the other hand, notice that if $P \Downarrow_c$, then, there is a derivation of the form:

$$P \equiv P_1 \xRightarrow{(1,c_1)} P_2 \xRightarrow{(1,c_2)} \cdots P_n \xRightarrow{(1,c_n)} P_{n+1}$$

and $c_n \vdash_{\Delta_t} c$. We shall discharge the proof by showing that the internal (Equation 1 below) and the observable (Equation 2 below) derivations preserve probability.

For the internal derivation, assume that $(X;\Gamma;d) \longrightarrow^* (X \cup X';\Gamma';d \wedge d')$. We shall show that for any $i \geq 1$, and $e \in \mathcal{C}_t$, if $!^{\mathfrak{c}(\infty)}[\![\Delta_t]\!], \mathcal{P}[\![\Gamma]\!]_i, \mathcal{C}[\![d]\!]_i \longrightarrow \mathcal{C}[\![e]\!]_i \otimes \top$ then $!^{\mathfrak{c}(\infty)}[\![\Delta_t]\!], \exists X'.(\mathcal{P}[\![\Gamma']\!]_i \otimes \mathcal{C}[\![d]\!]_i \otimes \mathcal{C}[\![d']\!]_i) \longrightarrow \mathcal{C}[\![e]\!]_i \otimes \top$. The proof proceeds by induction on the length of the derivation with case analysis on the last rule applied. The resulting cases are analogous to those in the proof of CCP adequacy and we only consider the case $\Box P$ (recall that $\circ P$ does not exhibit any internal transition). We know that $(X;\Gamma,\Box P;d) \longrightarrow (X;\Gamma,P,\circ\Box P;d)$. Consider the formulas $F = \mathcal{P}[\![\Box P]\!]_i = !^{\mathfrak{p}(\infty)} \Cap l : i{+}(\mathcal{P}[\![P]\!]_l)$ and $F' = \mathcal{P}[\![P]\!]_i \otimes \mathcal{P}[\![\circ\Box P]\!]_i = \mathcal{P}[\![P]\!]_i \otimes \mathcal{P}[\![\Box P]\!]_{i+1}$. Consider now the sequent

$$!^{\mathfrak{c}(\infty)}[\![\Delta_t]\!], \mathcal{P}[\![\Gamma]\!]_i, F, \mathcal{C}[\![d]\!]_i \longrightarrow \mathcal{C}[\![e]\!]_i \otimes \top$$

We notice that in any proof of such sequent, given that $\mathcal{C}[\![e]\!]_i = !^{\mathfrak{c}(i)}?^{\mathfrak{c}(i)}e$, none of the instances of F of the form $\mathcal{P}[\![P]\!]_j$ with $j > i$ can be used (since $\mathfrak{p}(i)$, $\mathfrak{p}(j)$, $\mathfrak{c}(i)$ and $\mathfrak{c}(j)$ are unrelated). On the other side, due to the connective $!^{\mathfrak{p}(\infty)}$ in F, several instances of the form $\mathcal{P}[\![P]\!]_i$ can be used in the proof of the sequent. Nevertheless, since P is a deterministic process, it is easy to prove by structural induction that for any G, $\mathcal{P}[\![P]\!]_i \longrightarrow G$ iff $\mathcal{P}[\![P,P]\!]_i \longrightarrow G$. Then, the sequent above is provable iff the following one is provable:

$$!^{\mathfrak{c}(\infty)}[\![\Delta_t]\!], \mathcal{P}[\![\Gamma]\!]_i, \mathcal{P}[\![P]\!]_i, \mathcal{C}[\![d]\!]_i \longrightarrow \mathcal{C}[\![e]\!]_i \otimes \top$$

and the result follows.

From here we conclude:

$$\text{if } c_i \vdash_{\Delta_t} e \text{ then } !^{\mathfrak{c}(\infty)}[\![\Delta_t]\!], \mathcal{P}[\![P_i]\!]_i \longrightarrow \mathcal{C}[\![e]\!]_i \otimes \top \qquad (1)$$

As for the observable derivation, assume that $(X;\Gamma,d) \not\longrightarrow$. Note that the process to be executed in the next time-unit corresponds to $(\mathbf{local}\, X)\, F(\Gamma)$ where F is the future function. Let G be a formula of the form $!^{l_j}?^{l_j}G'$ where $i < j$, i.e., G' is an observation of a future time-unit j. We can show that

$$\begin{array}{ll} \text{if} & !^{\mathfrak{c}(\infty)}[\![\Delta_t]\!], \exists X.(\mathcal{P}[\![\Gamma]\!]_i \otimes \mathcal{C}[\![d]\!]_i) \longrightarrow !^{l_j}?^{l_j}G' \\ \text{then} & !^{\mathfrak{c}(\infty)}[\![\Delta_t]\!], \exists X \mathcal{P}[\![F(\Gamma)]\!]_{i+1} \longrightarrow !^{l_j}?^{l_j}G' \end{array} \qquad (2)$$

For that, notice $\mathcal{C}[\![d]\!]_i$ takes the form $!^{\mathfrak{c}(i)}?^{\mathfrak{c}(i)}F$, and then, this formula has to be deleted in a proof for $!^j?^jG'$ (since $l_i \npreceq l_j$). This is, the current store is forgotten and it cannot be used to prove properties in the future. Now we analyze $\mathcal{P}[\![\Gamma]\!]_i$ and $\mathcal{P}[\![F(\Gamma)]\!]_{i+1}$. It is easy to see that $\mathcal{P}[\![\circ P]\!]_i \equiv \mathcal{P}[\![P]\!]_{i+1}$. Notice that $F(\mathbf{ask}\ c\ \mathbf{then}\ P) = \emptyset$. If Γ contains a process $\mathbf{ask}\ c\ \mathbf{then}\ P$, it must be the case that $d \nvdash_{\Delta_t} c$. Then, $\mathcal{P}[\![P]\!]_i$ could not be used in a proof for G.

Negative Phase

$$\frac{}{[\mathcal{K}:\Gamma],\Delta \longrightarrow \top}\ \top_r \qquad \frac{[\mathcal{K}:\Gamma],\Delta,F,G \longrightarrow \mathcal{R}}{[\mathcal{K}:\Gamma],\Delta,F\otimes G \longrightarrow \mathcal{R}}\ \otimes_l \qquad \frac{[\mathcal{K}:\Gamma],\Delta,F \longrightarrow G}{[\mathcal{K}:\Gamma],\Delta \longrightarrow F \multimap G}\ \multimap_r$$

$$\frac{[\mathcal{K}:\Gamma],\Delta \longrightarrow G[c/x]}{[\mathcal{K}:\Gamma],\Delta \longrightarrow \forall x.G}\ \forall_r \qquad \frac{[\mathcal{K}:\Gamma],\Delta,G[c/x] \longrightarrow \mathcal{R}}{[\mathcal{K}:\Gamma],\Delta,\exists x.G \longrightarrow \mathcal{R}}\ \exists_l \qquad \frac{[\mathcal{K} +_{\mathfrak{f}} (l)F:\Gamma],\Delta \longrightarrow \mathcal{R}}{[\mathcal{K}:\Gamma],\Delta,!^{\mathfrak{f}(l)}F \longrightarrow \mathcal{R}}\ {!^{\mathfrak{f}(l)}}_l$$

$$\frac{[\mathcal{K}_{l_e}:\Gamma],\Delta \longrightarrow G[l_e/x]}{[\mathcal{K}:\Gamma],\Delta \longrightarrow \Cap x.G}\ \Cap_r \qquad \frac{[\mathcal{K}_{l_e}:\Gamma],\Delta,G[l_e/x] \longrightarrow \mathcal{R}}{[\mathcal{K}:\Gamma],\Delta,\Cup x.G \longrightarrow \mathcal{R}}\ \Cup_l$$

Positive Phase

$$\frac{[\mathcal{K}_1:\Gamma_1]-_F\!\!\rightarrow \quad [\mathcal{K}_2:\Gamma_2]-_G\!\!\rightarrow}{[\mathcal{K}_1\otimes\mathcal{K}_2:\Gamma_1,\Gamma_2]-_{F\otimes G}\!\!\rightarrow}\ \otimes_r,\text{ where } (\mathcal{K}_1=\mathcal{K}_2)|_U$$

$$\frac{[\mathcal{K}_1:\Gamma_1]-_F\!\!\rightarrow \quad [\mathcal{K}_2:\Gamma_2] \xrightarrow{H} [G]}{[\mathcal{K}_1\otimes\mathcal{K}_2:\Gamma_1,\Gamma_2] \xrightarrow{F\multimap H} [G]}\ \multimap_l\text{ , where } (\mathcal{K}_1=\mathcal{K}_2)|_U$$

$$\frac{[\mathcal{K}:\Gamma]-_{G[t/x]}\!\!\rightarrow}{[\mathcal{K}:\Gamma]-_{\exists x.G}\!\!\rightarrow}\ \exists_r \qquad \frac{[\mathcal{K}:\Gamma] \xrightarrow{F[t/x]} [G]}{[\mathcal{K}:\Gamma] \xrightarrow{\forall x.F} [G]}\ \forall_l \qquad \frac{[\mathcal{K}\leq_{\mathfrak{f}(l)}:\cdot] \longrightarrow F}{[\mathcal{K}:\cdot]-_{!^{\mathfrak{f}(l)}F}\!\!\rightarrow}\ {!^{\mathfrak{f}(l)}}_r\ \star$$

$$\frac{[\mathcal{K}:\Gamma]-_{G[l/x]}\!\!\rightarrow}{[\mathcal{K}:\Gamma]-_{\Cup x.G}\!\!\rightarrow}\ \Cup_r \qquad \frac{[\mathcal{K}:\Gamma] \xrightarrow{F[l/x]} [G]}{[\mathcal{K}:\Gamma] \xrightarrow{\Cap x.F} [G]}\ \Cap_l$$

$$\frac{[\mathcal{K}\leq_l:\cdot],F \longrightarrow [\cdot]}{[\mathcal{K}:\cdot] \xrightarrow{?^{\mathfrak{f}(l)}F} [?^{\mathfrak{f}(k)}G]}\ {?^l}_l\ \star\text{ and } k\in U \wedge \mathfrak{f}(l)\not\preceq_{\mathcal{A}} \mathfrak{f}(k) \qquad \frac{[\mathcal{K}\leq_{\mathfrak{f}}(l):\cdot],F \longrightarrow [?^{\mathfrak{f}(k)}G]}{[\mathcal{K}:\cdot] \xrightarrow{?^{\mathfrak{f}(l)}F} [?^{\mathfrak{f}(k)}G]}\ {?^{\mathfrak{f}(l)}}_l\ \star\text{ and } \mathfrak{f}(l)\preceq_{\mathcal{A}} \mathfrak{f}(k)$$

$$\frac{}{[\mathcal{K}:\Gamma]-_A\!\!\rightarrow}\ I_r\text{ given } A\in(\Gamma\cup\mathcal{K}[\mathcal{I}])\text{ and }(\Gamma\cup\mathcal{K}[\mathcal{I}\setminus U])\subseteq\{A\}$$

Structural Rules

$$\frac{[\mathcal{K}:\Gamma,N_a],\Delta \longrightarrow \mathcal{R}}{[\mathcal{K}:\Gamma],\Delta,N_a \longrightarrow \mathcal{R}}\ []_l \quad \frac{[\mathcal{K}:\Gamma],\Delta \longrightarrow [P_a]}{[\mathcal{K}:\Gamma],\Delta \longrightarrow P_a}\ []_r \quad \frac{[\mathcal{K}:\Gamma],P_a \longrightarrow [F]}{[\mathcal{K}:\Gamma] \xrightarrow{P_a} [F]}\ R_l \quad \frac{[\mathcal{K}:\Gamma] \longrightarrow N}{[\mathcal{K}:\Gamma]-_N\!\!\rightarrow}\ R_r$$

$$\frac{[\mathcal{K}:\Gamma] \xrightarrow{F} [G]}{[\mathcal{K}+_l F:\Gamma] \longrightarrow [G]}\ D_l\text{, provided, } l\notin U \qquad \frac{[\mathcal{K}+_l F:\Gamma] \xrightarrow{F} [G]}{[\mathcal{K}+_l F:\Gamma] \longrightarrow [G]}\ D_l\text{, provided, } l\in U$$

$$\frac{[\mathcal{K}:\Gamma] \xrightarrow{F} [G]}{[\mathcal{K}:\Gamma,F] \longrightarrow [G]}\ D_l \qquad \frac{[\mathcal{K}:\Gamma]-_G\!\!\rightarrow}{[\mathcal{K}:\Gamma] \longrightarrow [G]}\ D_r \qquad \frac{[\mathcal{K}:\Gamma]-_G\!\!\rightarrow}{[\mathcal{K}:\Gamma] \longrightarrow [?^l G]}\ D_r$$

Fig. 3. Focused Proof System for Intuitionistic Linear Logic with Subexponentials. Assume $\Sigma = \langle I,\preceq,F,U\rangle$. We elide the contexts $\mathcal{A}$ and $\mathcal{L}$. We define the set $\mathcal{I} = \{\mathfrak{f}(a) \mid \mathfrak{f}\in F, a\in\mathcal{A}\}$. Here, $\mathcal{R}$ stands for either a bracketed context, $[F]$, or an unbracketed context. A is an atomic formula; P_a is a positive or atomic formula or formulas of the form $?^l F$; N is a negative formula; and N_a is a negative or atomic formula or a formula of the form $!^l F$. In the $?^l$ and $!^l$ rules, $\star$ stands for "given $\mathcal{K}[\{x \mid l \not\preceq_{\mathcal{A}} x \wedge x\notin U\}] = \emptyset$." Finally $\mathcal{K}_{l_e}$ is obtained by extending the domain of $\mathcal{K}$ with $\{\mathfrak{f}(l_e) \mid \mathfrak{f}\in F\}$ and mapping these to the empty set.

- $(\mathcal{K}_1 \otimes \mathcal{K}_2)[i] = \begin{cases} \mathcal{K}_1[i] \cup \mathcal{K}_2[i] & \text{if } i \notin U \\ \mathcal{K}_1[i] & \text{if } i \in U \end{cases}$
- $\mathcal{K}[\mathcal{S}] = \bigcup\{\mathcal{K}[i] \mid i \in \mathcal{S}\}$
- $(\mathcal{K} +_l F)[i] = \begin{cases} \mathcal{K}[i] \cup \{F\} & \text{if } i = l \\ \mathcal{K}[i] & \text{otherwise} \end{cases}$
- $\mathcal{K} \leq_i [l] = \begin{cases} \mathcal{K}[l] & \text{if } i \preceq_{\mathcal{A}} l \\ \emptyset & \text{if } i \not\preceq_{\mathcal{A}} l \end{cases}$
- $(\mathcal{K}_1 \star \mathcal{K}_2) \mid_{\mathcal{S}}$ is true if and only if $(\mathcal{K}_1[j] \star \mathcal{K}_2[j])$ for all $j \in \mathcal{S}$.

Fig. 4. Specification of operations on contexts. Here, $i \in I$, $\mathcal{S} \subseteq I$, and the binary connective $\star \in \{=, \subset, \subseteq\}$. We also assume that the set $\mathcal{A}$ is given from the context.