A Proof Theoretic Study of Soft Concurrent Constraint Programming

ELAINE PIMENTEL
Universidade Federal do Rio Grande do Norte, Natal, Brazil
(*e-mail:* elaine.pimentel@gmail.com)

CARLOS OLARTE
Pontificia Universidad Javeriana Cali, Colombia
Universidade Federal do Rio Grande do Norte, Natal, Brazil
(*e-mail:* carlos.olarte@gmail.com)

VIVEK NIGAM
Universidade Federal da Paraíba, João Pessoa, Brazil
(*e-mail:* vivek.nigam@gmail.com)

submitted 14 February 2014; revised 25 March 2014; accepted 6 May 2014

Abstract

Concurrent Constraint Programming (CCP) is a simple and powerful model for concurrency where agents interact by telling and asking constraints. Since their inception, CCP-languages have been designed for having a strong connection to logic. In fact, the underlying constraint system can be built from a suitable fragment of intuitionistic (linear) logic –ILL– and processes can be interpreted as formulas in ILL. Constraints as ILL formulas fail to represent accurately situations where "preferences" (called soft constraints) such as probabilities, uncertainty or fuzziness are present. In order to circumvent this problem, c-semirings have been proposed as algebraic structures for defining constraint systems where agents are allowed to tell and ask soft constraints. Nevertheless, in this case, the tight connection to logic and proof theory is lost. In this work, we give a proof theoretical meaning to soft constraints: they can be defined as formulas in a suitable fragment of ILL with subexponentials (SELL) where subexponentials, ordered in a c-semiring structure, are interpreted as preferences. We hence achieve two goals: (1) obtain a CCP language where agents can tell and ask soft constraints and (2) prove that the language in (1) has a strong connection with logic. Hence we keep a declarative reading of processes as formulas while providing a logical framework for soft-CCP based systems. An interesting side effect of (1) is that one is also able to handle probabilities (and other modalities) in SELL, by restricting the use of the promotion rule for non-idempotent c-semirings. This finer way of controlling subexponentials allows for considering more interesting spaces and restrictions, and it opens the possibility of specifying more challenging computational systems.

KEYWORDS: Concurrent Constraint Programming, Linear Logic, Soft Constraints

1 Introduction

Providing logical and proof theoretic semantics to (fragments of) programming languages not only gives a declarative meaning to these languages, but also normally leads to the development of new features allowing more expressive programming constructions to be used. For example, we investigated recently (Nigam et al. 2013) a proof theoretic specification of the concurrent constraint programming (CCP) (Saraswat et al. 1991) language introduced in (Knight et al. 2012) that mentions epistemic (`eccp`) and spatial (`sccp`) modalities. We used as underlying logical framework linear logic with subexponentials (*SELL*) (Danos et al. 1993; Nigam and Miller 2009), showing that our encodings faithfully specify `eccp` and `sccp`. More interestingly, this study allowed us to develop extensions of `eccp` and `sccp` with features not available in (Knight et al. 2012), such as systems with an unbounded number of agents for `eccp` or spaces for `sccp` and constructs that allow the communication of location names (Olarte et al. 2013). In this paper we turn our attention to computing with *soft constraints*.

Soft concurrent constraint programming (Soft-CCP) (Bistarelli et al. 2006) is an extension of CCP where agents are allowed to tell and ask soft constraints, *i.e.*, constraints with certain level of preference. This allows the modeling of systems with levels of uncertainty, or those mentioning probabilistic or fuzzy entities. However, moving from hard (crisp) constraints to soft constraints was not followed by a corresponding logical/proof theoretic characterization of these systems. This is unfortunate because one of the key motivations of the original CCP was its tight connection to logic and proof theory which enabled the proposal of more advanced systems such as its linear version `lcc` (Fages et al. 2001). The main contribution of this paper is to recover this connection by studying the proof theory of soft constraint systems in the form of *SELL* theories.

A key feature of *SELL* is that subexponentials are organized into a pre-order, specifying the provability relation among subexponentials. In (Nigam et al. 2013), we used a simple partially ordered set of subexponential names. While this was enough for modeling `eccp` and `sccp`, it is not enough to model soft constraints. In this case, we need more sophisticated algebraic structures.

This paper investigates the proof theory of *SELL* with more involved pre-orders and demonstrates that it is possible to characterize *soft constraint systems* by using pre-orders based on semirings. During this investigation, we have also identified variants of soft constraints, namely those based on non-idempotent c-semirings, such as probabilistic soft constraints, that do not seem to have a sensible correspondence in *SELL*. We thus propose a new proof system, called *SELLS*, with a different promotion rule that allows for such a correspondence. We prove that *SELLS* admits cut-elimination. We also point out that due to the tight correspondence of soft constraints and proof theory, it seems possible to extend `eccp` and `sccp` with soft constraints. Although we provide some pointers in this paper, its full development is left as future work.

Organization. Section 2 reviews CCP and soft constraints. Then in Section 3.2, after reviewing *SELL*, we propose a novel soft constraint system based on subexponential signatures proving that it is indeed a sensible CCP constraint system (Theorem 1). Then we also propose an encoding of Soft-CCP into *SELL* proving its adequacy (Theorem 2). Section 4 gives some examples of the use of the novel soft constraint system and we point out some limitations of *SELL* to represent non-idempotent soft constraints (e.g., probabilistic systems). We thus propose *SELLS* and prove that it admits cut-elimination (Theorem 3). Section 5 concludes and presents related works. Some missing proofs and auxiliary results are shown in the Appendix available at arXiv:1312.2552.

2 Concurrent Constraint Programming

Concurrent Constraint Programming (CCP) (Saraswat et al. 1991) (see a survey in (Olarte et al. 2013)) 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.

Processes in CCP *interact* with each other by *telling* and *asking* constraints (pieces of information) in a common store of partial information. The type of constraints is not fixed but parametric in a constraint system (CS). Intuitively, a CS provides a signature from which constraints can be built from basic tokens (e.g., predicate symbols), and two basic operations: conjunction ($\sqcup$) and variable hiding ($\exists$). The CS defines also an *entailment* relation ($\models$) specifying inter-dependencies between constraints: $c \models d$ means that the information d can be deduced from the information c. Such systems can be formalized as a Scott information system as in (Saraswat et al. 1991), or they can be built upon a suitable fragment of logic *e.g.*, as in (Fages et al. 2001). In Section 3, we will specify such systems as formulas in intuitionistic linear logic (ILL (Girard 1987)).

2.1 The language of CCP processes

In the spirit of process calculi, the language of processes in CCP is given by a small number of primitive operators or combinators as described below.

Definition 1 (*Syntax. Indeterminate CCP language (Saraswat et al. 1991)*)
Processes in CCP are built from constraints in the underlying CS and the syntax:

$$P,Q ::= \mathbf{tell}(c) \mid \sum_{i\in I} \mathbf{ask}\ c_i\ \mathbf{then}\ P_i \mid P \parallel Q \mid (\mathbf{local}\,x)\,P \mid p(\overline{x})$$

The process $\mathbf{tell}(c)$ adds c to the current store d producing the new store $c \sqcup d$. Given a non-empty finite set of indexes I, the process $\sum\limits_{i\in I} \mathbf{ask}\ c_i\ \mathbf{then}\ P_i$ non-deterministically chooses P_j for execution if the store entails c_j. The chosen alternative, if any, precludes the others. This provides a powerful synchronization mechanism based on constraint entailment. When I is a singleton, we shall omit the "$\sum$" and we simply write $\mathbf{ask}\ c\ \mathbf{then}\ P$. The process $P \parallel Q$ represents the parallel (interleaved) execution of P and Q. The process $(\mathbf{local}\,x)\,P$ behaves as P and binds the variable x to be local to it. We shall use $fv(P)$ to denote the set of free variables of P. Given a process definition $p(\overline{y}) \overset{\Delta}{=} P$, where all free variables of P are in the set of pairwise distinct variables $\overline{y}$, the process $p(\overline{x})$ evolves into $P[\overline{x}/\overline{y}]$. A CCP program takes the form $\mathscr{D}.P$ where $\mathscr{D}$ is a set of process definitions and P a process.

Structural Operational Semantics (SOS). The SOS of CCP is given by the transition relation $\gamma \longrightarrow \gamma'$ satisfying the rules in Figure 1. Here we follow the formulation in (Fages et al. 2001; Haemmerlé et al. 2007) where the local variables created by the program appear explicitly in the transition system and parallel composition of agents is identified as a multiset of agents. More precisely, a *configuration* γ is a triple of the form $(X;\Gamma;c)$, where c is a constraint representing 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 $P_1 \parallel P_2 \parallel \cdots \parallel P_n$. We shall indistinguishably use both notations to denote parallel composition. Moreover, processes are quotiented by a structural congruence relation $\cong$ satisfying: (STR1) $(\mathbf{local}\,x)\,P \cong (\mathbf{local}\,y)\,P[y/x]$ if $y \notin fv(P)$ (alpha conversion); (STR2) $P \parallel Q \cong Q \parallel P$; (STR3) $P \parallel (Q \parallel R) \cong (P \parallel Q) \parallel R$. We shall write $(X;\Gamma;c) \equiv (X';\Gamma';c')$ whenever $X = X'$, $\Gamma \cong \Gamma'$ and $c \equiv c'$ (i.e., $c \models c'$ and $c' \models c$).

The rules in Figure 1 are straightforward realizing the operational intuitions given above: a tell agent $\mathbf{tell}(c)$ adds c to the current store d (Rule $\mathrm{R}_{\mathrm{TELL}}$); the process $\sum\limits_{i\in I} \mathbf{ask}\ c_i\ \mathbf{then}\ P_i$ executes P_j if its corresponding guard c_j can be entailed from the store (Rule $\mathrm{R}_{\mathrm{SUM}}$); a local process $(\mathbf{local}\,x)\,P$ adds x to the set of hidden variable X when no clashes of variables occur (Rule $\mathrm{R}_{\mathrm{LOC}}$). Observe that Rule $\mathrm{R}_{\mathrm{EQUIV}}$ can be used, for instance, to do alpha conversion if the premise of $\mathrm{R}_{\mathrm{LOC}}$ cannot be satisfied; the call $p(\overline{x})$ executes the body of the process definition (Rule $\mathrm{R}_{\mathrm{CALL}}$).

$$\frac{}{(X;\mathbf{tell}(c),\Gamma;d) \longrightarrow (X;\Gamma;c\sqcup d)}\ \mathrm{R_{TELL}} \qquad \frac{d \models c_j \quad j \in I}{(X;\sum\limits_{i\in I} \mathbf{ask}\ c_i\ \mathbf{then}\ P_i;\Gamma;d) \longrightarrow (X;P_j;d)}\ \mathrm{R_{SUM}}$$

$$\frac{x \notin X \cup fv(d) \cup fv(\Gamma)}{(X;(\mathbf{local}\, x)\, P,\Gamma;d) \longrightarrow (X\cup\{x\};P,\Gamma;d)}\ \mathrm{R_{LOC}} \qquad \frac{p(\overline{x}) \overset{\Delta}{=} P \in \mathscr{D}}{(X;p(\overline{y}),\Gamma;d) \longrightarrow (X;P[\overline{y}/\overline{x}],\Gamma;d)}\ \mathrm{R_{CALL}}$$

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

Fig. 1. Operational semantics for CCP calculi

Definition 2 (*Observable behavior*)
Let $\longrightarrow^*$ be the reflexive and transitive closure of $\longrightarrow$. If $(X;\Gamma;d) \longrightarrow^* (X';\Gamma';d')$ and $\exists X'.d' \models c$ we write $(X;\Gamma;d) \Downarrow_c$. If $X = \emptyset$ and $d = \mathtt{true}$ we simply write $\Gamma \Downarrow_c$.

Intuitively, if P is a process then $P \Downarrow_c$ says that P can reach a store d strong enough to entail c, *i.e.*, we can regard c as an output of P. Note that in the above definition, the variables in X' are hidden since the information about them is not observable.

As processes manipulate the store of constraints, the CS used dictates much of the behavior of the system. For instance, in (Fages et al. 2001) it is shown that by using formulas in a fragment of ILL (Girard 1987) as CS, one obtains a more expressive language where ask processes can *consume* information. The same goal is achieved here, but by demonstrating that soft constraints in CCP can be obtained by allowing subexponentials (Danos et al. 1993) in the CS.

2.2 Soft Constraint in Concurrent Constraint Programming

It is well known that crisp (hard) constraints fail to represent accurately situations where soft constraints, *i.e.*, preferences, probabilities, uncertainty or fuzziness, are present. In constraint programming (Rossi et al. 2006), two general frameworks have been proposed to deal with soft constraints: *semiring* based constraints (Bistarelli et al. 1997) and *valued* constraints (Schiex et al. 1995). Roughly speaking, in both frameworks an algebraic structure defines the operations needed to *combine* soft constraints and choosing when a constraint (or solution) is *better* than another. In (Bistarelli et al. 1999), it is shown that both frameworks are equally expressive and they are general enough to represent different kind of soft constraints including, e.g., fuzzy, probabilistic and weighted constraints.

In the forthcoming sections, we shall build soft constraints from formulas in a suitable fragment of ILL with subexponentials (SELL) where subexponentials are ordered in a semiring structure. Before that, let us recall the framework of semiring based constraints.

Definition 3 (*C-Semiring (Bistarelli et al. 1997)*)
A c-semiring is a tuple $\langle \mathscr{A}, +_{\mathscr{A}}, \times_{\mathscr{A}}, \bot_{\mathscr{A}}, \top_{\mathscr{A}} \rangle$ satisfying: (S1) $\mathscr{A}$ is a set and $\bot_{\mathscr{A}}, \top_{\mathscr{A}} \in \mathscr{A}$; (S2) $+_{\mathscr{A}}$ is a binary, commutative, associative and idempotent operator on $\mathscr{A}$, $\bot_{\mathscr{A}}$ is its unit element and $\top_{\mathscr{A}}$ its absorbing element; (S3) $\times_{\mathscr{A}}$ is a binary, associative and commutative operator on $\mathscr{A}$ with unit element $\top_{\mathscr{A}}$ and absorbing element $\bot_{\mathscr{A}}$. Moreover, $\times_{\mathscr{A}}$ distributes over $+_{\mathscr{A}}$. Let $\leq_{\mathscr{A}}$ be defined as $a \leq_{\mathscr{A}} b$ iff $a +_{\mathscr{A}} b = b$. Then, $\langle \mathscr{A}, \leq_{\mathscr{A}} \rangle$ is a complete lattice where: (S4) $+_{\mathscr{A}}$ and $\times_{\mathscr{A}}$ are monotone on $\leq_{\mathscr{A}}$; (S5) $\times_{\mathscr{A}}$ is intensive on $\leq_{\mathscr{A}}$, *i.e.*, $a \times b \leq_{\mathscr{A}} a$. (S6) $\bot_{\mathscr{A}}$ (resp. $\top_{\mathscr{A}}$) is the bottom (resp. top) of $\mathscr{A}$; (S7) $+_{\mathscr{A}}$ is the *lub* operator. If $\times_{\mathscr{A}}$ is idempotent, then: (S8) $+_{\mathscr{A}}$ distributes over $\times_{\mathscr{A}}$; (S9) $\langle \mathscr{A}, \leq_{\mathscr{A}} \rangle$ is a complete distribute lattice and $\times_{\mathscr{A}}$ is its *glb*. We shall say that a c-semiring is idempotent whenever its $\times_{\mathscr{A}}$ operator is idempotent, and non-idempotent otherwise.

Elements in the set $\mathscr{A}$ (c-semiring values) are used to denote the *upper bound of preference degrees*, or simply *preference level*, where the "preference" could be a probability, cost, etc. The $\times_{\mathscr{A}}$ operator is used to combine values while $+_{\mathscr{A}}$ is used to select which is the "best" value in the sense that $a +_{\mathscr{A}} b = b$ iff $a \leq_{\mathscr{A}} b$ iff b is "better" than a.

Instances of c-semirings. Before giving some instances of c-semirings, an important clarification is in order. In soft constraint logic programming (Bistarelli et al. 1997) and soft concurrent constraint programming (Bistarelli et al. 2006), constraints are usually seen as mappings from variable assignments into elements in the semiring $\mathscr{A}$. For instance, let x and y be integer variables and consider the constraint $\mathtt{leq}(x,y)$ with the usual meaning. Then, using the crisp semiring described below, the constraint $\mathtt{leq}(x,y)$ maps the tuple $\langle 1,2\rangle$ to $\mathtt{true}$ and $\langle 2,1\rangle$ to $\mathtt{false}$. Hence, combining two constraints c_1 and c_2 means that there are *fewer* possible values in the variable domains that can satisfy both constraints (i.e., the variable-assignment problem is "harder" to solve). In this paper we adhere to the tradition of CCP-languages and constraint systems (Saraswat et al. 1991; de Boer et al. 1995) where constraints are seen as tokens of (partial) information. Hence, when the token $\mathtt{leq}(x,y)$ is added to the current store d, we are not interested in *solving* the constraint problem $d \sqcup \mathtt{leq}(x,y)$ (i.e., find the values for x and y that satisfy such constraint). Instead, we see the addition of $\mathtt{leq}(x,y)$ to d as *increasing monotonically* the information we have about x and y in d. For instance, that information can be used to deduce (via the entailment relation) that $\mathtt{leq}(x,y+1)$ also holds. Accordingly, in the context of soft constraints, adding a constraint c with a *preference level* $a \in \mathscr{A}$, denoted as $[c]_a$, will mean that c is *believed* with a probability, preference, costs, etc. a. The higher the value of a the more the information we add to the store.

Let us now give some well-known instances of c-semirings. Let c_1 and c_2 be constraints. The c-semiring $S_c = \langle\{\mathtt{true},\mathtt{false}\},\vee,\wedge,\mathtt{false},\mathtt{true}\rangle$ models **crisp** (hard) constraints. Then, $[c_1]_{\mathtt{false}}$ means that the agent *does not believe* in c_1 and hence, regardless the preference level of c_2, the conjunction of c_1 and c_2 must be also assigned a preference level of $\mathtt{false}$. The **fuzzy** c-semiring $S_F = \langle[0,1], max, min, 0, 1\rangle$ allows for fuzzy constraints that have an associate preference level in the real interval $[0,1]$ where 1 represents the best value. Then, if $[c]_{0.2}$ and $[d]_{0.7}$ are in the store, we can say that d is believed with a "better" (higher) preference level (wrt $+_{\mathscr{A}}$) than c. From that store we can also deduce that the conjunction $c \sqcup d$ is believed with preference level 0.2 (using the $\times_{\mathscr{A}}$ operator to combine 0.7 and 0.2). In a **probabilistic** setting, a constraint c is annotated with its probability of existence where probabilities are supposed to be independent (*i.e.*, no conditional probabilities). This can be modeled with the c-semiring $S_P = \langle[0,1], max, \times, 0, 1\rangle$. Then, if $[c_1]_{0.2}$ and $[c_2]_{0.7}$ are in the store, the probability of deducing $c_1 \sqcup c_2$ is 0.14. In **weighted** constraints there is an accumulate cost that can be computed with the c-semiring $S_w = \langle\mathbb{R}^-, max, +, -\infty, 0\rangle$, where 0 means no cost. Then, from a store containing $[c_1]_{-2}$ and $[c_2]_{-7}$ we can deduce $[c_1 \sqcup c_2]_{-9}$. We note that the first two c-semirings are idempotent (*i.e.*, $\times_{\mathscr{A}}$ idempotent), while the last two are not.

3 Soft-CCP as Theories in Linear Logic with Subexponential

In this section we build soft constraints from formulas in a suitable fragment of intuitionistic linear logic (ILL) with subexponentials (Danos et al. 1993; Nigam and Miller 2009) (*SELL*) where subexponentials are ordered in a c-semiring structure. By doing that, we achieve two goals: (1) obtain a CCP language where agents can tell and ask soft constraints and (2) prove that

$$\frac{\Gamma, F, H \longrightarrow G}{\Gamma, F \otimes H \longrightarrow G}\ \otimes_L \quad \frac{\Gamma_1 \longrightarrow F \quad \Gamma_2 \longrightarrow H}{\Gamma_1, \Gamma_2 \longrightarrow F \otimes H}\ \otimes_R \quad \frac{\Gamma \longrightarrow G}{\Gamma, \mathbf{1} \longrightarrow G}\ \mathbf{1}_L \quad \frac{}{\longrightarrow \mathbf{1}}\ \mathbf{1}_R \quad \frac{}{\Gamma \longrightarrow \top}\ \top_R$$

$$\frac{\Gamma_1 \longrightarrow F \quad \Gamma_2, H \longrightarrow G}{\Gamma_1, \Gamma_2, F \multimap H \longrightarrow G}\ \multimap_L \quad \frac{\Gamma, F \longrightarrow H}{\Gamma \longrightarrow F \multimap H}\ \multimap_R \quad \frac{\Gamma, F_i \longrightarrow G}{\Gamma, F_1 \,\&\, F_2 \longrightarrow G}\ \&_{L_i} \quad \frac{\Gamma \longrightarrow F \quad \Gamma \longrightarrow H}{\Gamma \longrightarrow F \,\&\, H}\ \&_R$$

$$\frac{\Gamma, F[e/x] \longrightarrow G}{\Gamma, \exists x.F \longrightarrow G}\ \exists_L \quad \frac{\Gamma \longrightarrow G[t/x]}{\Gamma \longrightarrow \exists x.G}\ \exists_R \quad \frac{\Gamma, F[t/x] \longrightarrow G}{\Gamma, \forall x.F \longrightarrow G}\ \forall_L \quad \frac{\Gamma \longrightarrow G[e/x]}{\Gamma \longrightarrow \forall x.G}\ \forall_R$$

Fig. 2. A fragment of the LL introduction rules. Here e is a fresh variable and t is a term.

the language in (1) has a strong connection with logic (Section 3.3). We then keep a declarative reading of processes as formulas and provide a logical framework for soft-CCP based systems. This last goal is remarkable. In fact, the beauty of CCP relies on the fact that it is simple, yet powerful, and with a strong connection to logic, hence correct.

3.1 Linear Logic with Subexponentials

SELL shares with intuitionistic linear logic all its connectives except the exponentials: instead of having a single pair of exponentials ! and ?, *SELL* may contain as many *subexponentials* (Danos et al. 1993; Nigam and Miller 2009) as needed.

Figure 2 presents the introduction rules of the fragment of linear logic that will be used in order to build soft constraint system ($\otimes, \exists, \mathbf{1}, \forall, \top$) and to give meaning to processes ($\&, \multimap$). Note that formulas are not always allowed to contract and weaken: this is controlled in linear logic by the use of the *exponentials* ! and ?. In *SELL*, this control is finer since it is possible to specify which subexponentials behave classically or not.

Formally, a *SELL* system is specified by a *subexponential signature* $\Sigma = \langle I, \preceq, U \rangle$, where I is a set of labels, $U \subseteq I$ specifying which subexponentials allow both weakening and contraction, and $\preceq$ is a pre-order among the elements of I. We shall use $a, b, \ldots$ to range over elements in I and we will assume that $\preceq$ is upwardly closed with respect to U, *i.e.*, if $a \in U$ and $a \preceq b$, then $b \in U$. For a given such subexponential signature, *SELL*$_\Sigma$ is the system obtained by substituting the linear logic exponential ! by the subexponential $!^a$ for each $a \in I$, and by adding to the rules in Figure 2 the following inference rules:
- for each $a \in I$ (dereliction and the promotion rules):

$$\frac{\Gamma, F \longrightarrow G}{\Gamma, !^a F \longrightarrow G}\ !^a{}_L \qquad \frac{!^{a_1}F_1, \ldots, !^{a_n}F_n \longrightarrow F}{!^{a_1}F_1, \ldots, !^{a_n}F_n \longrightarrow !^a F}\ !^a{}_R, \text{provided } a \preceq a_i \text{ for } 1 \leq i \leq n.$$

- for each $b \in U$ (structural rules):

$$\frac{\Gamma \longrightarrow G}{\Gamma, !^b F \longrightarrow G}\ W \qquad \frac{\Gamma, !^b F, !^b F \longrightarrow G}{\Gamma, !^b F \longrightarrow G}\ C$$

In this paper we will not use the $?^a$ subexponential, since the specifications will be within the *minimal* setting of *SELL*. We would like to stress out that this choice do not affect the expressiveness of the framework, as pointed out in (Chaudhuri 2010). Observe that provability is preserved *downwards* i.e. the sequent $\Gamma \longrightarrow !^a P$ is provable in *SELL*$_\Sigma$, then so is the sequent $\Gamma \longrightarrow !^b P$ for all $b \preceq a$. We shall elide the signature Σ whenever it is not important or clear from the context.

Subexponentials greatly increase the expressiveness of the system when compared to linear logic. The key difference is that while linear logic has only seven logically distinct prefixes of

! and ? (e.g., $!F$, $!?\,F$, $?!F$, etc) (Danos et al. 1993), *SELL* allows for an unbounded number of such prefixes (e.g., $!^a ?^b F, !^b ?^a F$, etc). In fact, in (Nigam et al. 2013), we showed that by using different prefixes it is possible to interpret subexponentials in more creative ways, such as temporal units or spatial and epistemic modalities.

3.2 C-semiring as Subexponentials Signatures

In (Nigam et al. 2013) we studied the logical meaning of CCP processes as *SELL* formulas. For that, we assumed that the underlying constraint system had a logical structure and we required simple pre-orders as subexponential signatures. Here we go in the opposite direction: assuming that CCP processes can be endowed with a logical meaning, we propose a logical framework for building *soft* constraints, thus recovering the logical reading of Soft CCP systems. This requires a more involving algebraic structure in the subexponential signature, as follows.

Definition 4 (*Soft Constraint System (SCS)*)
Let $S = \langle \mathscr{A}, +_{\mathscr{A}}, \times_{\mathscr{A}}, \bot_{\mathscr{A}}, \top_{\mathscr{A}} \rangle$ be a c-semiring with $\leq_{\mathscr{A}}$ the order induced by $+_{\mathscr{A}}$; $\mathscr{P}$ be a first order signature; $\Sigma = \langle \mathscr{A}, \leq_{\mathscr{A}}, \mathscr{A} \rangle$ be a subexponential signature; and $\mathscr{C}$ be a set of *SELL* formulas built from the syntax:

$$\mathbf{C} \quad ::= \quad \mathbf{1} \mid C \otimes C \mid \exists x.(C) \mid !^a A \mid !^a(!^a A_1 \otimes \cdots \otimes !^a A_n)$$

where $a \in \mathscr{A}$ and A, A_i are atomic formulas (*i.e.*, predicate symbols in $\mathscr{P}$ applied to terms). Elements in $\mathscr{C}$, with typical elements c, d, are called constraints. Let $\Delta = \{\delta_1, ..., \delta_n\}$ be a (possibly empty) set of non-logical axioms of the form $\forall \bar{x}_i.(c_i \multimap d_i)$ where all free variables in c_i and d_i are in $\bar{x}_i$. A soft constraint system SCS is a structure $\langle \mathscr{A}, \mathscr{C}, \models \rangle$ where $d \models c$ iff the sequent $!^{\top_{\mathscr{A}}} \delta_1, ..., !^{\top_{\mathscr{A}}} \delta_n, d \longrightarrow c$ is provable in SELL.

We shall call *pre-constraints* formulas of the shape $A_1 \otimes \cdots \otimes A_n$ or an atom A. As usual in the specification of constraint systems as formulas in a given logic, the previous definition built constraints from the the empty store ($\mathbf{1}$); conjunction of constraints ($\otimes$); and existential quantification of constraints. In our case, additionally, a constraint can be a formula F of the form $!^a A$ or $!^a(!^a A_1 \otimes \cdots \otimes !^a A_n)$ where A, A_i are atomic formulas. Roughly, F means that the pre-constraint A (or $A_1 \otimes \cdots \otimes A_n$) was added to the store with an upper bound preference degree a. Note that a is a c-semiring value and, according to the previous definition, it is a subexponential. Moreover, due to the signature Σ, all the subexponentials are unbounded which means that soft constraints cannot be removed from the store. In what follows, we shall write $[A]_a$ instead of $!^a A$; $[A_1 \otimes \cdots \otimes A_n]_a$ instead of $!^a(!^a A_1 \otimes \cdots \otimes !^a A_n)$; and $[F]$ instead of $[F]_{\top_{\mathscr{A}}}$.

Now we shall show that our construction is indeed an instance of the general definition of CS as cylindric algebras in (Saraswat et al. 1991; de Boer et al. 1995). This guarantees that all the machinery developed for CCP calculi can be used also when considering programs with the SCS in Definition 4. The reader may find the details in the companion Appendix.

Theorem 1 (*Constraint System*)
Let $\mathbb{C} = \langle \mathscr{A}, \mathscr{C}, \models \rangle$ be as in Definition 4. Then, the structure $\langle \mathscr{C}, \leq, \otimes, \mathbf{1}, \mathbf{0}, Var, \exists, D \rangle$ is a cylindric constraint system where $D = \{!^{\top_{\mathscr{A}}}(d_{xy}) \mid x, y \in Var\}$ and $c \leq d$ iff $d \models c$.

3.3 Logical Reading of Processes

In (Nigam et al. 2013) we extended the results in (Fages et al. 2001) and we showed that CCP processes have a strong connection with ILL: operational steps matches *exactly* focused logical steps (Nigam and Miller 2009). We also showed that such characterization extends to various CCP calculi like epistemic, spatial and timed systems. Unlike the results in (Nigam et al. 2013), the encoding here considers non-determinism and we do not need the extension of *SELL* with families or quantification over subexp., as the Soft-CCP systems do not mention nested modalities. It seems possible, however, to include these modalities to obtain Soft-CCP systems that mention spatial or temporal modalities (see Section 5).

Assume a SCS and let $\Sigma' = \langle \mathscr{A} \cup \{\mathfrak{p}, \mathfrak{d}, \mathfrak{u}\}, \preceq, \mathscr{A} \cup \{\mathfrak{u}\}\rangle$ be a subexponential signature where, for any $a, b \in \mathscr{A}$, $a \preceq b$ iff $a \leq_{\mathscr{A}} b$, and $a, \mathfrak{p}, \mathfrak{d}, \mathfrak{u}$ are unrelated wrt $\preceq$. Observe that $a, \mathfrak{u} \in U$ while $\mathfrak{p}, \mathfrak{d} \notin U$. Intuitively, the subexponential $\mathfrak{p}$ is used to mark processes; $\mathfrak{u}$ marks process definitions; and $\mathfrak{d}$ marks calls $p(\bar{x})$ whose definition may be unfolded. We will build the subexponential signature Σ from Σ', as the completion of Σ' to a c-semiring. This is easily achieved by adding two distinguished elements: $\bot_c, \top_c$ such that $\bot_c \leq_{\mathscr{A}} \bot_{\mathscr{A}}$; $\bot_c \leq_{\mathscr{A}} \mathfrak{p}, \mathfrak{d}, \mathfrak{u}$; $\top_{\mathscr{A}} \leq_{\mathscr{A}} \top_c$; and $\mathfrak{p}, \mathfrak{d}, \mathfrak{u} \leq_{\mathscr{A}} \top_c$. Then, for example, $\mathfrak{p} \times_{\mathscr{A}} a = \bot_c$ and $\mathfrak{p} +_{\mathscr{A}} a = \top_c$ for any $a \in \mathscr{A}$.

Definition 5 (*Encoding of processes, non-logical axioms and process definitions*)
For any process P, $\mathscr{P}[\![P]\!]$ is defined recursively as:

- $\mathscr{P}[\![\mathbf{tell}(c)]\!] = !^{\mathfrak{p}} c$
- $\mathscr{P}[\![\sum\limits_{i \in I} \mathbf{ask}\ c_i\ \mathbf{then}\ P_i]\!] = !^{\mathfrak{p}} \mathbin{\&}_{i \in I}(c_i \multimap \mathscr{P}[\![P_i]\!])$
- $\mathscr{P}[\![(\mathbf{local}\,\bar{x})\,P]\!] = !^{\mathfrak{p}}(\exists \bar{x}. \mathscr{P}[\![P]\!])$
- $\mathscr{P}[\![P_1, ..., P_n]\!] = \mathscr{P}[\![P_1]\!] \otimes ... \otimes \mathscr{P}[\![P_n]\!]$
- $\mathscr{P}[\![p(\bar{x})]\!] = !^{\mathfrak{d}} p(\bar{x})$

Recall that non-logical axioms are encoded as formulas of the form $!^{\top_{\mathscr{A}}}(\forall \bar{x}(d \multimap c))$ (see Def. 4). A process definition of the form $p(\bar{x}) \overset{\Delta}{=} P$ is encoded as $!^{\mathfrak{u}}[\forall \bar{x}.(!^{\mathfrak{d}} p(\bar{x}) \multimap \mathscr{P}[\![P]\!])]$.. We shall use $[\![\Psi]\!]$ to denote the set of *SELL* formulas encoding the set of process definitions Ψ.

Theorem 2 (*Adequacy*)
Let P be a process, $(\mathscr{A}, \mathscr{C}, \models)$ be a SCS with a (possible empty) set of non-logical axioms Δ and Ψ be a set of process definitions. Then $P \Downarrow_c$ iff $\Delta, [\![\Psi]\!], \mathscr{P}[\![P]\!] \longrightarrow c \otimes \top$.

4 Computing with Soft Constraints

We distinguish two classes of SCS according to the underlying c-semiring: idempotent and non-idempotent. First, from the pre-order induced by the c-semiring (Definition 4), we can rephrase the side-condition of *SELL*'s promotion rule for SCS as follows:

$$\frac{!^{a_1}F_1, \ldots, !^{a_n}F_n \longrightarrow F}{!^{a_1}F_1, \ldots, !^{a_n}F_n \longrightarrow !^{a}F}\ !^{a}{}_R, \text{ provided } a \leq_{\mathscr{A}} glb(a_1, ..., a_n) \tag{1}$$

4.1 Idempotent soft constraints

It turns out that, in an idempotent c-semiring, $a \times_{\mathscr{A}} b = glb(a, b)$. Hence the side-condition in (1) is equivalent to

$$a \leq_{\mathscr{A}} a_1 \times_{\mathscr{A}} ... \times_{\mathscr{A}} a_n \tag{2}$$

For illustrating better how the promotion rule is used in idempotent systems, consider the fuzzy c-semiring $S_F = \langle [0,1], max, min, 0, 1\rangle$ and its corresponding SCS as in Definition 4. Let c, d be pre-constraints and consider $T = P \parallel Q \parallel R \parallel S$ where:

$$P = \mathbf{tell}([c]_{0.7}) \parallel \mathbf{tell}([d]_{0.2}) \qquad Q = \mathbf{ask}\ [c]_{0.3}\ \mathbf{then}\ Q'$$
$$R = \mathbf{ask}\ [c \otimes d]_{0.5}\ \mathbf{then}\ R' \qquad S = \mathbf{ask}\ [c \otimes d]_{0.2}\ \mathbf{then}\ S'$$

From the initial store $\mathbf{1}$, we observe the following transitions:

$$(\emptyset\ ;\ T\ ;\ \mathbf{1})\ \longrightarrow^*\ (\emptyset\ ;\ Q\ \parallel R \parallel S\ ;\ [c]_{0.7} \otimes [d]_{0.2})\ \longrightarrow^*\ (\emptyset\ ;\ Q' \parallel R \parallel S'\ ;\ [c]_{0.7} \otimes [d]_{0.2})$$

The ask Q can proceed since the sequent $[c]_{0.7} \longrightarrow [c]_{0.3}$ is provable. Furthermore, since the sequent $[c]_{0.7}, [d]_{0.2} \longrightarrow [c \otimes d]_{0.2}$ is also provable, S can evolve into S'. Finally, R remains blocked since the sequent $[c]_{0.7}, [d]_{0.2} \longrightarrow [c \otimes d]_{0.5}$ is not provable: introducing $!^{0.5}$ on the right implies weakening the formula $[d]_{0.2}$ on the left. That is, the process P adds the information that c (resp. d) is *preferred* with a level of 0.7 (resp. 0.2). Hence the pre-constraint $c \otimes d$ can be deduced only with a preference level less or equal to 0.2.

4.2 Non-idempotent soft constraints

It is well known that some of the interesting properties of the c-semiring framework for constraint programming do not hold for non-idempotent c-semirings (see Section 5). In our framework, if $\times_{\mathscr{A}}$ is not idempotent then it may be the case that $a \times_{\mathscr{A}} b <_{\mathscr{A}} glb(a,b)$; hence the side conditions in (1) and (2) are no longer equivalent and therefore the promotion rule in (1) does not seem to be adequate anymore.

For an example, let $S_P = \langle [0,1], max, \times, 0, 1 \rangle$ be the probabilistic c-semiring and T be the process as above. Under this SCS, the sequent $[c]_{0.7}, [d]_{0.2} \longrightarrow [c \otimes d]_{0.2}$ is provable (as in the case of the Fuzzy c-semiring) and then, the process S can proceed. This does not fit to our intuition that from $[c]_{0.7} \otimes [d]_{0.2}$ we can only entail $c \otimes d$ with a probability less or equal to 0.14. This undesired behavior comes with no surprise since the provability relation takes into account the ordering induced by $+_{\mathscr{A}}$ but it does not "combine" information with the $\times_{\mathscr{A}}$ operator.

Fortunately, it is possible to redefine the promotion rule in order to specify the "combination" of c-semiring values when non-idempotent c-semirings are considered. We define the system *SELLS* from *SELL*, replacing the side condition of the promotion rule.

Definition 6 (*SELLS system*)
Let Σ be a subexponential signature as in Definition 4. The *SELLS*$_\Sigma$ system shares with *SELL* all the rules but the promotion rule, which is defined as

$$\frac{!^{a_1}F_1, \ldots, !^{a_n}F_n \longrightarrow F}{!^{a_1}F_1, \ldots, !^{a_n}F_n \longrightarrow !^{a}F}\ !^{a}{}_{R_S}, \text{ provided } a \leq_{\mathscr{A}} a_1 \times_{\mathscr{A}} \ldots \times_{\mathscr{A}} a_n$$

We shall write *SELLS* instead of *SELLS*$_\Sigma$ when Σ can be inferred by the context.

Note that, for an idempotent c-semiring, this condition is the same as in the *SELL* system since $a_1 \times_{\mathscr{A}} \cdots \times_{\mathscr{A}} a_n = glb(a_1, \cdots, a_n)$. In the case of non-idempotent c-semirings, though, this condition is *stronger* since $a_1 \times_{\mathscr{A}} \cdots \times_{\mathscr{A}} a_n \leq glb(a_1, \cdots, a_n)$. The new rule is not at all ad-hoc: while *SELLS* is a smooth extension of ILL, it is a closed subsystem of *SELL* which is strict when non-idempotent c-semirings are considered. Hence *SELLS* inherits all *SELL* good properties such as cut elimination (see the proof in the companion Appendix).

Theorem 3
SELLS admits cut-elimination.

We note that Theorem 2 is also valid for the non-idempotent case as shown in the companion Appendix. Observe also that the rule $!^{a}{}_{R_S}$ above has a strong synchronous flavor: not only it inherits the synchronous behavior of the bang, but it also introduces a strong non-determinism on choosing the formulas on the left-hand-side of the sequent marked with exponentials $a_1, \ldots, a_n$.

Finally, notice that, in *SELLS*, the sequent $[c]_{0.7},[d]_{0.2} \longrightarrow [c\otimes d]_{0.2}$ is no longer provable while $[c]_{0.7},[d]_{0.2} \longrightarrow [c\otimes d]_{a}$ is provable whenever $a \leq 0.14$, as desired. This finer way of controlling subexponentials on the left side of sequents allows considering more interesting spaces as signatures, and it opens the possibility of specifying more challenging computational systems.

4.3 Monotonicity and level of preferences

Let us now explain how the Soft-CCP language here proposed adheres to the elegant properties of its predecessors. In CCP languages, the store grows monotonically, *i.e.*, one can easily verify by induction on the structure of P that if $(X\ ;\ \Gamma\ ;\ c) \longrightarrow^{*} (X'\ ;\ \Gamma'\ ;\ c')$ then $\exists X'.c' \models \exists X.c$. In c-semiring based constraints, when two constraints are combined, one gets a lower value of the c-semiring. In the case of constraint solving and soft concurrent constraint programming as in (Bistarelli et al. 2006), this can be understood as the fact that having more constraints implies that it is "more difficult" to satisfy all of them. Hence we have: (i) more constraints imply a stronger store and then, more information can be deduced from it; and (ii) more constraints imply a lower level of preference in the semiring. How should we interpret these somehow contradictory ideas?

This problem was already addressed in (Bistarelli et al. 2006) where the entailment relation (that is only defined for idempotent c-semirings –see Section 5) is defined as the inverse of the ordering of the semiring. Roughly speaking, C entails c iff $\bigotimes C \sqsubseteq c$ where $\bigotimes C$ denotes the combination ($\times_{\mathscr{A}}$) of constraints in the set C and $\sqsubseteq$ is the ordering induced by $\leq_{\mathscr{A}}$ on constraints.

Now let us explain how (i) and (ii) above coexists in our framework. We note first that the sequent $[c_1]_{a_1},\ldots,[c_n]_{a_n},[c]_a \longrightarrow [c]_b$ is provable for any $b \leq_{\mathscr{A}} a$ This means that, if an agent adds the constraint c with level of preference a, then it is possible to deduce c with a preference level less or equal to a. That is, the store grows monotonically. Now consider the store $[c_1]_{a_1} \otimes \cdots \otimes [c_n]_{a_n} \otimes [c]_a \otimes [c]_b$ where $a <_{\mathscr{A}} b$. In this case, c can be deduced with a preference level less or equal to b. This also matches the monotonic behavior we want in the store: if $[c]_b$ is added first, then the agent adding $[c]_a$ is just adding "irrelevant" information to the store that can be weakened when needed; on the other hand, if $[c]_a$ is added first, then, adding $[c]_b$ means that c is believed with a greater level of preference and the store becomes stronger. Consider the stores $d_1 = [c]_a \otimes [d]_b$ and $d_2 = [c\otimes d]_b$ where $a <_{\mathscr{A}} b$. If $e \leq_{\mathscr{A}} a \times_{\mathscr{A}} b$, it is clear that $d_1 \models [c\otimes d]_e$. Moreover, $d_2 \models d_1$. This shows that believing both c and d with a given preference level b (i.e., $[c\otimes d]_b$) is stronger than believing c with a preference level $a \leq_{\mathscr{A}} b$. Note that the sequent $d_2 \longrightarrow d_1$ is provable only because all atoms in the constraint system are classical – in our example, $d_2 = !^{b}(!^{b}c \otimes !^{b}d)$. Finally, the store is idempotent as in CCP ($c \sqcup c \cong c$). To see that, notice that $[c]_a \otimes [c]_a \equiv [c]_a$ (regardless the idempotency of $\times_{\mathscr{A}}$).

5 Concluding Remark

We have established a tight connection between Soft-CCP systems and linear logic proof systems. In particular, we investigated the use of subexponentials in linear logic with more involved pre-orders as logical foundations for soft constraints. Moreover, we have also proposed a novel proof system, *SELLS*, giving a logical meaning to soft constraints based on non-idempotent semirings, such as probabilistic and weighted soft constraints.

Related Work In (Bistarelli et al. 2006) the first CCP language featuring soft constraints was proposed. There, c-semiring based constraints, seen as functions mapping variable assignments into c-semiring values, are lifted to a higher-order semiring where constraints can be combined and compared. In such formalization, an entailment relation à la Saraswat (Saraswat et al. 1991) can be defined only if the $\times_{\mathscr{A}}$ operator is idempotent (see (Bistarelli et al. 2006, Def. 3.8, Th. 3.9)). In particular, given a set of constraints C, if $\times_{\mathscr{A}}$ is non-idempotent, $C \models d$ does not imply that $C \sqcup d \equiv C$. Note that in our case, if $C \longrightarrow d$ then the equivalence $(\bigotimes C \otimes d) \equiv (\bigotimes C)$ is provable (regardless the idempotency of $\times_{\mathscr{A}}$). Hence, our logical characterization of soft constraints as formulas in *SELL* follows closely the idea of monotonic store in CCP.

The language in (Bistarelli et al. 2006) allows agents to be *guarded* by a semiring value $a \in \mathscr{A}$. Hence, an agent performs an action only if the resulting store is not *weaker* than the cut level a. For instance, $\textbf{tell}(c) \longrightarrow^{a} P$ adds c to the store and then executes P if c in conjunction with the current store has a level of preference greater than a. We could also add to our language such kind of constructs by modifying accordingly the SOS in order to handling a-guarded constructs. Nevertheless, one should be careful since the logical meaning of processes is lost (Theorem 2). The main reason is that such constructs do not have a proof theoretically meaning: it is necessary to check the consistency of the system first, to latter add a formula to the context.

The work in (Bistarelli et al. 2008) combines the notion of time in `tccp` (de Boer et al. 2000) with soft constraints. Due to Theorem 1, a similar extension can be also done with our framework by plugin into `tcc` (Saraswat et al. 1996) or `tccp` the soft constraint system in Definition 4. Moreover, due to the logic inspiration of the constraint system proposed here, it is possible to show also that timed processes manipulating soft constraints can be declaratively characterized as formulas in SELL ((Nigam et al. 2013)).

A model-based (semantic) characterization of soft constraints based on c-semirings is given in (Wilson 2006). To the best of our knowledge, ours is the first proof-theoretic characterization of such systems. However, the use of more involved orders for subexponentials is not completely new. They were used recently in different contexts, such as in Bounded Linear Logic (Ghica and Smith 2013) and in programming languages (Brunel et al. 2014).

Future Work. From the point of view of proof theory, the proof system *SELLS* is novel. We are currently investigating a focused proof system for it, which seems to be a non trivial task: the key problem is how to handle contraction of formulas. In fact, when contracting a formula one is no longer able to prove formulas marked with some subexponential bang. This is different from *SELL*. It seems possible, however, to use the fact that subexponentials are unbounded to come up with a sensible focused proof system for (fragments of) *SELLS*.

The definition we gave for soft constraint systems is general enough to be used in different CCP idioms. In particular, it is possible to define systems with spatial information where agents can *believe* the same information with different levels of preferences. Theorem 2 along with the logical characterization of spatial CCP in (Nigam et al. 2013) may allow us to prove correct such approach. We also foresee systems where agents can update their preferences. For that, we shall need to use quantifiers over subexponentials as defined in (Nigam et al. 2013). Finally, it seems that we can define our subexponentials to be linear in order to have declaratively some forms of *retraction* of soft constraints.

Acknowledgments. We thank Francesco Santini for helpful discussions. Nigam was supported by CNPq and Pimentel was supported by CNPq and CAPES. The work of Olarte has been (partially) supported by Colciencias (Colombia), CNPq and by Digiteo and DGAR (École Polytechnique) funds for visitors.

References

ANDREOLI, J.-M. 1992. Logic programming with focusing proofs in linear logic. *J. Log. Comput. 2,* 3, 297–347.

BISTARELLI, S., GABBRIELLI, M., MEO, M. C., AND SANTINI, F. 2008. Timed soft concurrent constraint programs. In *COORDINATION*, LNCS, vol. 5052. Springer, 50–66.

BISTARELLI, S., MONTANARI, U., AND ROSSI, F. 1997. Semiring-based constraint satisfaction and optimization. *J. ACM 44,* 2, 201–236.

BISTARELLI, S., MONTANARI, U., AND ROSSI, F. 2006. Soft concurrent constraint programming. *ACM Trans. Comput. Log. 7,* 3, 563–589.

BISTARELLI, S., MONTANARI, U., ROSSI, F., SCHIEX, T., VERFAILLIE, G., AND FARGIER, H. 1999. Semiring-based csps and valued csps: Frameworks, properties, and comparison. *Constraints 4,* 3, 199–240.

BRUNEL, A., GABOARDI, M., MAZZA, D., AND ZDANCEWIC, S. 2014. A core quantitative coeffect calculus. In *ESOP*, LNCS, vol. 8410. Springer, 351–370.

CHAUDHURI, K. 2010. Classical and intuitionistic subexponential logics are equally expressive. In *CSL*, LNCS, vol. 6247. Springer, 185–199.

DANOS, V., JOINET, J.-B., AND SCHELLINX, H. 1993. The structure of exponentials: Uncovering the dynamics of linear logic proofs. In *Kurt Gödel Colloq.*, LNCS, vol. 713. Springer, 159–171.

DE BOER, F. S., GABBRIELLI, M., AND MEO, M. C. 2000. A timed concurrent constraint language. *Inf. Comput. 161,* 1, 45–83.

DE BOER, F. S., PIERRO, A. D., AND PALAMIDESSI, C. 1995. Nondeterminism and infinite computations in constraint programming. *Theoretical Computer Science 151,* 1, 37–78.

FAGES, F., RUET, P., AND SOLIMAN, S. 2001. Linear concurrent constraint programming: Operational and phase semantics. *Inf. Comput. 165,* 1, 14–41.

GHICA, D. R. AND SMITH, A. 2013. From bounded affine types to automatic timing analysis. *CoRR abs/1307.2473*.

GIRARD, J.-Y. 1987. Linear logic. *Theor. Comput. Sci. 50*, 1–102.

HAEMMERLÉ, R., FAGES, F., AND SOLIMAN, S. 2007. Closures and modules within linear logic concurrent constraint programming. In *FSTTCS*, LNCS, vol. 4855. Springer, 544–556.

KNIGHT, S., PALAMIDESSI, C., PANANGADEN, P., AND VALENCIA, F. D. 2012. Spatial and epistemic modalities in constraint-based process calculi. In *CONCUR*, LNCS, vol. 7454. Springer, 317–332.

NIGAM, V. AND MILLER, D. 2009. Algorithmic specifications in linear logic with subexponentials. In *PPDP*, ACM, 129–140.

NIGAM, V., OLARTE, C., AND PIMENTEL, E. 2013. A general proof system for modalities in concurrent constraint programming. In *CONCUR*, LNCS, vol. 8052. Springer, 410–424.

OLARTE, C., NIGAM, V., AND PIMENTEL, E. 2013. Dynamic spaces in concurrent constraint programming. In *LSFA'13*. To be published in ENTCS.

OLARTE, C., RUEDA, C., AND VALENCIA, F. D. 2013. Models and emerging trends of concurrent constraint programming. *Constraints 18,* 4, 535–578.

ROSSI, F., VAN BEEK, P., AND WALSH, T., Eds. 2006. *Handbook of Constraint Programming*. Foundations of Artificial Intelligence, vol. 2. Elsevier.

SARASWAT, V. A., JAGADEESAN, R., AND GUPTA, V. 1996. Timed default concurrent constraint programming. *J. Symb. Comput. 22,* 5/6, 475–520.

SARASWAT, V. A., RINARD, M. C., AND PANANGADEN, P. 1991. Semantic foundations of concurrent constraint programming. In *POPL*, ACM Press, 333–352.

SCHIEX, T., FARGIER, H., AND VERFAILLIE, G. 1995. Valued constraint satisfaction problems: Hard and easy problems. In *IJCAI (1)*. Morgan Kaufmann, 631–639.

WILSON, N. 2006. A logic of soft constraints based on partially ordered preferences. *J. Heuristics 12,* 4-5, 241–262.