Soft Subexponentials and Multiplexing

Max Kanovich[1,2], Stepan Kuznetsov[3,2], Vivek Nigam[4,5], and Andre Scedrov[6,2]

[1] University College London, London, UK, m.kanovich@ucl.ac.uk
[2] National Research University Higher School of Economics, Moscow, Russia
[3] Steklov Mathematical Institute, Moscow, Russia, sk@mi.ras.ru
[4] Federal University of Paraíba, Brazil
[5] fortiss GmbH, Germany, nigam@fortiss.org
[6] University of Pennsylvania, USA, scedrov@math.upenn.edu

Abstract. Linear logic and its refinements have been used as a specification language for a number of deductive systems. This has been accomplished by carefully studying the structural restrictions of linear logic modalities. Examples of such refinements are subexponentials, light linear logic, and soft linear logic. We bring together these refinements of linear logic in a non-commutative setting. We introduce a non-commutative substructural system with subexponential modalities controlled by a minimalistic set of rules. Namely, we disallow the contraction and weakening rules for the exponential modality and introduce two primitive subexponentials. One of the subexponentials allows the multiplexing rule in the style of soft linear logic and light linear logic. The second subexponential provides the exchange rule. For this system, we construct a sequent calculus, establish cut elimination, and also provide a complete focused proof system. We illustrate the expressive power of this system by simulating Turing computations and categorial grammar parsing for compound sentences. Using the former, we prove undecidability results. The new system employs Lambek's non-emptiness restriction, which is incompatible with the standard (sub)exponential setting. Lambek's restriction is crucial for applications in linguistics: without this restriction, categorial grammars incorrectly mark some ungrammatical phrases as being correct.

1 Introduction

For the specification of deductive systems, linear logic [4, 5], and a number of refinements of linear logic have been proposed, such as commutative [23, 25] and non-commutative [12, 11] subexponentials, light linear logic [7], soft linear logic [16], and easy linear logic [10]. The key difference between these refinements is their treatment of the linear logic exponentials, $!, ?$. These refinements allow, *e.g.*, a finer control on the structural rules, *i.e.*, weakening, contraction and exchange rules, and how exponentials affect the sequent antecedent. For example [12], we proposed a logical framework with commutative and non-commutative subexponentials, applying it for applications in type-logical grammar. In particular, we demonstrated that this logical framework can be used to "type" correctly sentences that were not able before with previous logical frameworks, such as Lambek calculus.

However, as we have shown recently, our logical framework in [12] is incompatible with the important Lambek's non-emptiness property [13]. This property, which requires all antecedents to be non-empty, is motivated by linguistic applications of Lambek-like calculi. Namely, it prevents the system from recognizing ("typing") incorrectly formed sentences as grammatically correct. We discuss these linguistic issues in detail in Section 3. The lack of Lambek's restriction means that the logical framework proposed in our previous work is too expressive, typing incorrectly sentences.

To address this problem, we propose a new non-commutative proof system, called SLLM, that admits Lambek's non-emptiness condition and at the same time is expressive enough to type correctly sentences in our previous work. This system is also still capable of modelling computational processes, as we show in Section 5.1 on the example of Turing computations.

In particular, SLLM takes inspiration from the following refinements of linear logic: subexponentials, by allowing two types of subexponentials, ! and ∇; soft linear logic, which contributes a version of the multiplexing rule, $!_L$, shown below to the left; and light linear logic, which contributes the two right subexponentials rules, $!_R, \nabla_R$, shown below to the right.

$$\frac{\Gamma, F, \ldots, F, \Delta \to G}{\Gamma, !F, \Delta \to G}\ !_L \qquad \frac{F \to G}{!F \to !G}\ !_R \qquad \frac{F \to G}{\nabla F \to \nabla G}\ \nabla_R$$

In our version of the system, the premise of the rule $L!$ does not allow the zero instances of F. Hence, ! is a relevant subexponential as discussed in [12].

This rule is used to type sentencies correctly, while the rules $!_R$ and ∇_R are used to maintain Lambek's condition. SLLM contains, therefore, soft subexponentials and multiplexing.

Our main contributions are summarized below:

- **Admissibility of Cut Rule**: We introduce the proof system SLLM in Section 2. We also prove that it has basic properties, namely admissibility of Cut Rule and the substitution property. The challenge is to ensure a reasonable balance between the expressive power of systems and complexity of their implementation, and in particular, to circumvent the difficulties caused by linear logic contraction and weakening rules.
- **Lambek's Non-Emptiness Condition:** We demonstrate in Section 3 that SLLM (and thus also SLLMF) admits Lambek's non-emptiness condition. This means that SLLM cannot be used to "type" incorrect sentences.
  We demonstrate this by means of some examples.
- **Focused Proof System:** We introduce in Section 4 a focused proof system (SLLMF) proving that it is sound and complete with respect to SLLM. The focused proof system differs from the focused proof system in our previous work [12] by allowing a subexponential that can contract, but not weaken nor be exchanged. Such subexponentials were not allowed in the proof sys-

$$\frac{}{A \to A}\ I$$

$$\frac{\Phi \to A \quad \Sigma_1, B, \Sigma_2 \to C}{\Sigma_1, \Phi, A \setminus B, \Sigma_2 \to C}\ \setminus_L \qquad \frac{A, \Sigma \to B}{\Sigma \to A \setminus B}\ \setminus_R \ (\Sigma \text{ is not empty})$$

$$\frac{\Phi \to A \quad \Sigma_1, B, \Sigma_2 \to C}{\Sigma_1, B \,/\, A, \Phi, \Sigma_2 \to C}\ /_L \qquad \frac{\Sigma, A \to B}{\Sigma \to B \,/\, A}\ /_R \ (\Sigma \text{ is not empty})$$

$$\frac{\Sigma_1, A, B, \Sigma_2 \to C}{\Sigma_1, A \cdot B, \Sigma_2 \to C}\ \cdot_L \qquad \frac{\Sigma_1 \to A \quad \Sigma_2 \to B}{\Sigma_1, \Sigma_2 \to A \cdot B}\ \cdot_R$$

Table 1. Lambek Calculus: A non-commutative version of ILL

$$\frac{}{A \to A}\ I$$

$$\frac{\Phi \to A \quad \Sigma_1, B, \Sigma_2 \to C}{\Sigma_1, \Phi, A \setminus B, \Sigma_2 \to C}\ \setminus_L \qquad \frac{A, \Sigma \to B}{\Sigma \to A \setminus B}\ \setminus_R \ (\Sigma \text{ is not empty})$$

$$\frac{\Phi \to A \quad \Sigma_1, B, \Sigma_2 \to C}{\Sigma_1, B \,/\, A, \Phi, \Sigma_2 \to C}\ /_L \qquad \frac{\Sigma, A \to B}{\Sigma \to B \,/\, A}\ /_R \ (\Sigma \text{ is not empty})$$

$$\frac{\Sigma_1, A, B, \Sigma_2 \to C}{\Sigma_1, A \cdot B, \Sigma_2 \to C}\ \cdot_L \qquad \frac{\Sigma_1 \to A \quad \Sigma_2 \to B}{\Sigma_1, \Sigma_2 \to A \cdot B}\ \cdot_R$$

$$\frac{\Gamma_1, \overbrace{A, A, \ldots, A}^{k \text{ times}}, \Gamma_2 \to C}{\Gamma_1, !A, \Gamma_2 \to C}\ !_L \ \underline{(k \geq 1)} \qquad \frac{A \to C}{!A \to !C}\ !_R$$

$$\frac{\Gamma_1, A, \Gamma_2 \to C}{\Gamma_1, \nabla A, \Gamma_2 \to C}\ \nabla_L \qquad \frac{A \to C}{\nabla A \to \nabla C}\ \nabla_R$$

$$\frac{\Gamma_1, \Gamma_2, \nabla A, \Gamma_3 \to C}{\Gamma_1, \nabla A, \Gamma_2, \Gamma_3 \to C} \quad \text{and} \quad \frac{\Gamma_1, \nabla A, \Gamma_2, \Gamma_3 \to C}{\Gamma_1, \Gamma_2, \nabla A, \Gamma_3 \to C}\ \nabla_E$$

Table 2. SLLM: Lambek calculus with multiplexing.

tem introduced in [12]. A key insight is to keep track on when a formula necessarily has to be introduced in a branch and when not.

– **Complexity:** We investigate in Section 5 the complexity of SLLM. We first demonstrate that the provability problem for SLLM is undecidable in general and identify a fragment for which it is decidable.

Finally, in Section 6, we conclude by pointing to related and future work.

2 The non-commutative system SLLM with Multiplexing

As the non-commutative source, we take the Lambek calculus [17], Table 1, the well-known fundamental system in linguistic foundations.

The proof system introduced here, SLLM, extends the Lambek calculus by adding two new connectives (subexponentials) ! and ∇ and their rules in Table 2.

Drawing inspiration from commutative logics such as *linear logic* [6], *light linear logic* [7], *soft linear logic* [16], and *easy linear logic* [10], here we introduce our primitive non-commutative modalities $!A$ and ∇A controlled by *a minimalistic set of rules*.
Multiplexing Rule (local):

$$\frac{\Gamma_1, \overbrace{A, A, \ldots, A}^{k \text{ times}}, \Gamma_2 \to C}{\Gamma_1, !A, \Gamma_2 \to C}\ !_L\ \underline{(k \geq 1)} \tag{1}$$

Informally, $!A$ stands for: "*any* *<u>positive number</u>* *of copies of A* *<u>at the same position</u>*"

Remark 1. In contrast to soft linear logic and light linear logic, where Weakening is one of the necessary ingredients, here we exclude the Weakening case: $(k = 0)$.

Unlike Contraction rule that can be recursively reused, and $!A$ keeps the subexponential (it is introduced by Dereliction), our Multiplexing can only be used *once* with all copies provided immediately at the same time in one go, and the subexponentials get removed. Thus, if one wishes to reuse Multiplexing further in the proof, nested subexponentials would be needed (like $!!A$ for two levels of Multiplexing).

The second subexponential, ∇A, provides the exchange rule.
Exchange Rule (non-local):

$$\frac{\Gamma_1, \Gamma_2, \nabla A, \Gamma_3 \to C}{\Gamma_1, \nabla A, \Gamma_2, \Gamma_3 \to C} \quad \text{and} \quad \frac{\Gamma_1, \nabla A, \Gamma_2, \Gamma_3 \to C}{\Gamma_1, \Gamma_2, \nabla A, \Gamma_3 \to C}\ \nabla_E \tag{2}$$

and Dereliction Rule (local):

$$\frac{\Gamma_1, A, \Gamma_2 \to C}{\Gamma_1, \nabla A, \Gamma_2 \to C}\ \nabla_L \tag{3}$$

"∇A *can be thought of as storing a missing candidate A* in a fixed local storage, with the ability to deliver A *<u>to the right place</u>* at the appropriate time."

Remark 2. Notwithstanding that the traditional Promotion rule $\dfrac{\Gamma \to C}{!\Gamma \to !C}$ is accepted in linear logic as well as in soft linear logic, we confine ourselves to the restricted light linear logic promotion $\dfrac{A \to C}{!A \to !C}$, in order to guarantee cut admissibility for the non-commutative SLLM (cf. [11]). E.g., with $\dfrac{\Gamma \to C}{!\Gamma \to !C}$ the sequent

$$!B,\ !(B \backslash C)\ \to (C \cdot C) \tag{4}$$

is derivable by cut

$$\frac{\dfrac{B, B \setminus C \to C}{!B, !(B \setminus C) \to !C} \quad !C \to C \cdot C}{!B, !(B \setminus C) \to C \cdot C}\ Cut$$

but, to finalize a cut-free proof for (4), we need commutativity:

$$\dfrac{\dfrac{\dfrac{B \to B \quad B, C, B \setminus C \to C \cdot C}{B, B, B \setminus C, B \setminus C \to C \cdot C}\;[\text{ok with } C, B, B \setminus C \to C \cdot C]}{B, B, !(B \setminus C) \to C \cdot C}\;!_L}{!B, !(B \setminus C) \to C \cdot C}\;!_L$$

The following theorem states that SLLM enjoys cut admissibility and the substitution property.

Theorem 1.

(a) *The calculus* SLLM *enjoys admissibility of the Cut Rule:*

$$\dfrac{\Pi \to A \quad \Gamma_1, A, \Gamma_2 \to C}{\Gamma_1, \Pi, \Gamma_2 \to C}\;Cut \tag{5}$$

(b) *Given an atomic p, let the sequent $\Gamma(p) \to C(p)$ be derivable in the calculus. Then for any formula B, $\Gamma(B) \to C(B)$ is also derivable in the calculus. Here by $\Gamma(B)$ and $C(B)$ we denote the result of replacing all occurrences of p by B in $\Gamma(p)$ and $C(p)$, resp.*

Proof.

(a) By reductions. E.g.,

$$\dfrac{\dfrac{B \to A}{!B \to !A}\;!_R \qquad \dfrac{\Gamma_1, \overbrace{A, A, \ldots, A}^{k \text{ times}}, \Gamma_2 \to C}{\Gamma_1, !A, \Gamma_2 \to C}\;!_L}{\Gamma_1, !B, \Gamma_2 \to C}\;Cut$$

is reduced to

$$\dfrac{\dfrac{B \to A \quad \Gamma_1, A, A, \ldots, A, \Gamma_2 \to C}{\Gamma_1, B, B, \ldots, B, \Gamma_2 \to C}\;Cut\ (k \text{ times})}{\Gamma_1, !B, \Gamma_2 \to C}\;!_L$$

(b) By induction.

3 Linguistic Motivations

In this Section we illustrate how (and why) our modalities provide parsing complex and compound sentences in a natural language.

We start with standard examples, which go back to Lambek [17]; for in-depth discussion of linguistic matters we refer to standard textbooks [3, 21, 19].

The sentence "*Bob sent the letter yesterday*" is grammatical, and the following "type" specification is provable in Lambek calculus, Table 1.

$$N, (N \setminus S) \,/\, N, N, V \setminus V \to S$$

Here the 'syntactical type' N stands for nouns "the letter" and "Bob", and $((N \backslash S)/N)$, i.e., (V/N), for the transitive verb "sent", and $(V \backslash V)$ for the verb modifier "yesterday", where $V = (N \backslash S)$. The whole sentence is of type S.

Lambek's non-emptiness restriction is important for correctness of Lambek's approach to modelling natural language syntax. Without this restriction Lambek grammars *overgenerate,* that is, recognize ungrammatical phrases as if they were correct. The standard example [19, § 2.5] is as follows: *"very interesting book"* is a correct noun phrase, established by the following derivation:

$$(N\,/\,N)\,/(N\,/\,N), N\,/\,N, N \to N.$$

The sequent above is derivable in the Lambek calculus. Without Lambek's restriction, however, one can also derive

$$(N\,/\,N)\,/(N\,/\,N), N \to N$$

(since $\vdash N\,/\,N$ is derivable with an empty antecedent). This effect is unwanted, since the corresponding phrase *"very book"* is ungrammatical. Thus, Lambek's non-emptiness restriction is a highly desired property for linguistic applications.

Fortunately, SLLM enjoys Lambek's non-emptiness property. That is:

Theorem 2. *The calculus* SLLM *provides Lambek's non-emptiness restriction: If a sequent $\Gamma \to C$ is derivable in the calculus then the list of formulas Γ is not empty.*

Proof. The crucial point is that, in the absence of Weakening, $!A$ never happens to produce the empty list. ■

Theorems 1 and 2 show how our new system SLLM resolves the issues discussed in [13] for the case of the Lambek calculus extended with a full-power exponential in linear logic style. Namely, as we show in that article, no reasonable extension of the Lambek calculus with the exponential modality can have the three properties simultaneously:

– cut elimination;
– substitution;
– Lambek's restriction.

Moreover, as also shown in [13], for the one-variable fragment the same happens to the *relevant* subexponential, which allows contraction and permutation, but not weakening. Our new system overcomes these issues by refining the rules for subexponentials.

Now let us show how one can use subexponentials of SLLM to model more complicated sentences. Our analysis shares much with that of Morrill and Valentín [22]. Unlike ours, the systems in [22] do not enjoy Lambek's restriction.

(1) The noun phrase: "*the letter that Bob sent yesterday,*" is grammatical, so that its "type" specification (6) should be provable in Lambek calculus or alike:

$$N,\ ((N \backslash N)/S'),\ N,\ (V/N),\ (V \backslash V) \to N \qquad (6)$$

Here $((N\backslash N)/S')$ stands for a subordinating connective “that”.
As a type for the whole dependent clause,
“*Bob sent _ yesterday,*”
we take some S', not a full S, because the direct object, “the letter”, is missing.

Our solution refines the approach of [2, 20]. We mark the missing item, the direct object “the letter” of type N, by a specific formula ∇N stored *at a fixed local position* and by means of Exchange Rule (2) and Dereliction Rule (3) deliver the missing N to the right place with providing

$$N,\ (V/N),\ N,\ (V\backslash V)\ \to S,$$

which is the type specification for the sentence “*Bob sent the letter yesterday* ” completed with the direct object, “the letter”.
By taking $S' = (\nabla N\backslash S)$, we can prove (6):

$$\dfrac{\dfrac{\dfrac{N, V\,/\,N, N, V\,\backslash\,V \to S \quad \boxed{\swarrow\ \text{“Bob sent the letter yesterday”}}}{\nabla N, N, V\,/\,N, V\,\backslash\,V \to S}\ \nabla_L, \nabla_E}{N, V\,/\,N, V\,\backslash\,V \to S'} \qquad N, N\,\backslash\,N \to N}{N, (N\,\backslash\,N)\,/\,S', N, V\,/\,N, V\,\backslash\,V \to N}$$

Remark 3. If we allowed Weakening, we would prove the ungrammatical
“*the letter that Bob sent the letter yesterday,*”

(2) The noun phrase: “*the letter that Bob sent without reading*” is grammatical despite two missing items: the direct object to “sent” and the direct object to “reading”, resp. Hence, the corresponding “type” specification (7) should be provable in Lambek calculus or alike:

$$N,\ ((N\backslash N)/S''),\ \Delta_1,\ \Delta_2\ \to N \tag{7}$$

Here N stands for “the letter”, and $((N\backslash N)/S'')$ for “that”, and some Δ_1 for “Bob sent”, and Δ_2 for “without reading”.
As a type for the whole dependent clause,
“*Bob sent _ _ without reading _ _* ”
we have to take some S'', not a full S, to respect the fact that this time two items are missing: the direct objects to “sent” and “reading”, resp.

To justify “*the letter that Bob sent _ without reading _*” with its (7), in addition to the ∇-rules, we invoke Multiplexing Rule (1).
The correlation between the *full* S and S'' *with its multiple holes* is given by:

$$S'' = ((!\nabla N)\backslash S). \tag{8}$$

As compared with the previous case of S' representing one missing item ∇N, the S'' here is dealing with $!\nabla N$ which provides two copies of ∇N to represent two missing items.

Then the proof for (7) is as:

$$\dfrac{\dfrac{\dfrac{\dfrac{\varDelta_1, N, \varDelta_2, N \to S \;\; \boxed{\swarrow \textit{“Bob sent the letter without reading it”}}}{\nabla N, \nabla N, \varDelta_1, \varDelta_2 \to S}\,\nabla_L, \nabla_E}{!\nabla N, \varDelta_1, \varDelta_2 \to S}\,!_L}{\varDelta_1, \varDelta_2 \to S''} \qquad N, N \setminus N \to N}{N, (N \setminus N) \,/\, S'', \varDelta_1, \varDelta_2 \to N}$$

4 Focused Proof System

This section introduces a sound and complete focused proof system SLLMF for SLLM. Focusing [1] is a discipline on proofs that reduces proof search space. We take an intermediate step, by first introducing the proof system SLLM# that handles the non-determinism caused by the multiplexing rule.

4.1 Handling Local Contraction

For bottom-up proof-search, the multiplexing rule

$$\dfrac{\varGamma, F, \ldots, F, \varDelta \to G}{\varGamma, !F, \varDelta \to G}$$

has a great deal of *don't know non-determinism* as one has to decide how many copies of F appears in its premise. This decision affects provability as each formula has to be necessarily be used in the proof, *i.e.*, they cannot be weakened.

To address this problem, we take a lazy approach by using two new connectives $\sharp^*$ and $\sharp^+$. The formula $\sharp^* F$ denotes zero or more local copies of F, and $\sharp^+ F$ one of more copies of F. We construct the proof system SLLM# from SLLM as follows. It contains all rules of SLLM, except the rule $!_L$, which is replaced by the following rules

$$\dfrac{\varGamma, \sharp^+ F, \varDelta}{\varGamma, !F, \varDelta \to G} \quad \dfrac{\varGamma, F, \sharp^* F, \varDelta \to G}{\varGamma, \sharp^+ F, \varDelta \to G} \quad \dfrac{\varGamma, F, \varDelta \to G}{\varGamma, \sharp^* F, \varDelta \to G} \quad \dfrac{\varGamma, \varDelta \to G}{\varGamma, \sharp^* F, \varDelta \to G}$$

Notice that there is no need for explicit contraction and only $\sharp^*$ allows for weakening. We accommodate contraction into the introduction rules, namely, by modifying the rules where there is context splitting, such as in the rules $\setminus_L$. In particular, one should decide in which branch a formula C is necessarily be used. This is accomplished by using adequately $\sharp^+ F$ and $\sharp^* F$. For example, some rules for $\setminus_L$ are shown below:

$$\dfrac{\sharp^* C, \varGamma_2 \to F \quad \varGamma_1, \sharp^+ C, G, \varGamma_3 \to H}{\varGamma_1, \sharp^+ C, \varGamma_2, F \setminus G, \varGamma_3 \to H}$$

$$\dfrac{\sharp^+ C, \varGamma_2 \to F \quad \varGamma_1, \sharp^* C, G, \varGamma_3 \to H}{\varGamma_1, \sharp^+ C, \varGamma_2, F \setminus G, \varGamma_3 \to H}$$

$$\frac{\sharp^* C, \Gamma_2 \to F \quad \Gamma_1, \sharp^* C, G, \Gamma_3 \to H}{\Gamma_1, \sharp^* C, \Gamma_2, F \setminus G, \Gamma_3 \to H}$$

In the first rule, the C is necessarily used in the right premise, while in the left premise one can chose to use C or not. SLLM# contains similar symmetric rules where C is necessarily moved to the left premise. Also it contains the corresponding rules for $/_L$.

SLLM# also include the following more refined right-introduction rules for ! and ∇. where Γ_1^*, Γ_2^* are lists containing only formulas of the form $\sharp^* H$.

$$\frac{F \to G}{\Gamma_1^*, !F, \Gamma_2^* \to !G} \quad \frac{F \to G}{\Gamma_1^*, \nabla F, \Gamma_2^* \to \nabla G}$$

Notice how the decision that all formulas in Γ_1, Γ_2 represent zero copies is made in the rules above.

Theorem 3. *Let Γ, G be a list of formulas not containing $\sharp^*$ nor $\sharp^+$. A sequent $\Gamma \to G$ is provable in the* SLLM *if and only if it is provable in* SLLM#.

Completeness follows from straightforward proof by induction on the size of proofs. One needs to slightly generalize the inductive hypothesis. Soundness follows from the fact that contractions are local and can be permuted below every other rule.

4.2 Focused Proof System

First proposed by Andreoli [1] for Linear Logic, focused proof systems reduce proof search space by distinguishing rules which have don't know non-determinism, classified as *positive*, from rules which have don't care non-determinism, classified as *negative*. We classify the rules $\cdot_R, \setminus_L, /_L$ as positive and the rules $\cdot_L, \setminus_R, /_R$ as negative. Non-atomic formulas of the form $F \cdot G$ are classified as positive, $\nabla F, \sharp^* F, \sharp^+ F$ and $!F$ are classified as modal formulas, and formulas of the form $F \setminus G$ and F / G as negative.

The focused proof system manipulates the following types of sequents, where Γ_1 and Γ_2 are possibly empty lists of non-positive formulas, Θ is *multiset* of formulas, and N_N is a non-negative formula. Intitively, Θ will contain all formulas of the form ∇F. As they allow exchange rule, their collection can be treated as a multiset. Γ_1, Γ_2 contain formulas that cannot be introduced by negative rules.

- **Negative Phase:** $\Theta : [\Gamma_1], \Uparrow\Delta, [\Gamma_2] \to [N_N]$ and $\Theta : [\Gamma_1], \Uparrow\Delta, [\Gamma_2] \to \Uparrow F$. Intuitively, the formulas in Δ and F are eagerly introduced whenever they negative rules are applicable, as one does not lose completeness in doing so.
- **Border:** $\Theta : [\Gamma_1] \to [N_N]$. These are sequents for which no negative rule can be applied. At this moment, one has to decide on a formula starting a positive phase.
- **Positive Phase:** $\Theta : [\Gamma_1], \Downarrow F, [\Gamma_2] \to [N_N]$ and $\Theta : [\Gamma_1] \to \Downarrow F$, where only the formula F is focused on.

The focused proof system SLLMF is composed by the rules in Figures 1 and 2. Intuitively, reaction rules R_{L1}, R_{L2}, R_R and negative phase rules are applied until no more rules are applicable. Then a decision rule is applied which focuses on one formula. One needs, however, to be careful on whether the focused formula's main connective is $\sharp^+$ or $\sharp^*$. If it is the former, then we have committed to one copy of the formula and therefore, it can be modified to be $\sharp^*$, while the latter does not change.

The number of rules in Figure 2 simply reflects the different cases emerging due to the presence or not of formulas whose main connective is $\sharp^+$ or $\sharp^*$. For example, $\backslash_{L2}$ considers the case when the splitting of the context occurs exactly on a formula $\sharp^+ C$. In this case, the decision is to commit to use a copy of C in the right-premise, thus containing $\sharp^+ C$, while $\sharp^* C$ on the left-premise. The other rules follow the same reasoning.

Finally, notice that in the rules $I, !_R$ and ∇_R the context may contain formulas with main connective $\sharp^*$ in their conclusion, but not in their premise. This illustrates the lazy decision of how many copies of a formula are needed.

Theorem 4. *A sequent $\Gamma \to G$ is provable in* SLLM# *if and only if the sequent $\cdot : \Uparrow\Gamma \to \Uparrow G$ is provable in* SLLMF.

The proof of this theorem follows the same ideas as detailed in [26] and [12].

Corollary 1. *Let Γ, G be a list of formulas not containing $\sharp^*$ nor $\sharp^+$. A sequent $\Gamma \to G$ is provable in* SLLM *if and only if the sequent $\cdot : \Uparrow\Gamma \to \Uparrow G$ is provable in* SLLMF.

Remark 4. The focused proof system introduced above enables the use of more sophisticated search mechanisms. For example, lazy methods can reduce the non-determinism caused by the great number of introduction rules caused by managing $\#^*, \#^+$, *e.g.*, Figure 2.

5 Complexity

In this section, we investigate the complexity of SLLM. In particular, we show that it is undecidable in general, by encoding Turing computations. This encoding also illustrate how the focused proof system SLLMF reduces non-determinism. We then identify decidable fragments for SLLM.

5.1 Encoding Turing Computations in SLLM

Any Turing instruction I is encoded by $!\nabla A_I$ with an appropriate A_I.
E.g., an instruction $I : q\xi \to q'\eta R$

if in state q looking at symbol ξ, replace it by η, move the tape head one cell to the right, and go into state q',

Positive Phase Rules

$$\frac{}{\cdot : [\Gamma_1^*, A, \Gamma_2^*] \to \Downarrow A}\ I \qquad \frac{}{\cdot : [\Gamma_1^*, \sharp^{+/*} A, \Gamma_2^*] \to \Downarrow A}\ I \qquad \frac{\Theta_1 : [\Gamma_1] \to \Downarrow F \quad \Theta_2 : [\Gamma_1] \to \Downarrow G}{\Theta_1, \Theta_2 : [\Gamma_1, \Gamma_2] \to \Downarrow F \cdot G}\ \cdot_R$$

$$\frac{\cdot : \Uparrow F \to \Uparrow G}{\cdot : [\Gamma_1^*, !F, \Gamma_2^*] \to \Downarrow !G}\ !_R \qquad \frac{\cdot : \Uparrow F \to \Uparrow G}{F : [\Gamma^*] \to \Downarrow \nabla G}\ \nabla_R$$

Negative Phase Rules

$$\frac{\Theta : [\Gamma_1], \Uparrow F, G, \Delta, [\Gamma_2] \to \mathcal{R}}{\Theta : [\Gamma_1], \Uparrow F \cdot G, \Delta, [\Gamma_2] \to \mathcal{R}}\ \cdot_L \qquad \frac{\Theta : \Uparrow F, [\Gamma] \to \Uparrow G}{\Theta : [\Gamma] \to \Uparrow F \setminus G}\ \setminus_R \qquad \frac{\Theta : [\Gamma], \Uparrow G \to \Uparrow F}{\Theta : [\Gamma] \to \Uparrow F \,/\, G}\ /_R$$

Decision and Reaction Rules

$$\frac{\Theta : [\Gamma_1], \Uparrow N_P^{\nabla}, [\Gamma_2] \to [G]}{\Theta : [\Gamma_1], \Downarrow N_P^{\nabla}, [\Gamma_2] \to [G]}\ R_L \qquad \frac{\Theta : [\Gamma] \to \Uparrow N_{NA}}{\Theta : [\Gamma] \to \Downarrow N_{NA}}\ R_R$$

$$\frac{\Theta : [\Gamma_1, N_P], \Uparrow \Delta, [\Gamma_2] \to \mathcal{R}}{\Theta : [\Gamma_1], \Uparrow N_P, \Delta, [\Gamma_2] \to \mathcal{R}}\ \Uparrow_{L1} \qquad \frac{\Theta, F : [\Gamma_1], \Uparrow \Delta, [\Gamma_2] \to \mathcal{R}}{\Theta : [\Gamma_1], \Uparrow \nabla F, \Delta, [\Gamma_2] \to \mathcal{R}}\ \Uparrow_{L2}$$

$$\frac{\Theta : [\Gamma] \to [N_N]}{\Theta : [\Gamma] \to \Uparrow N_N}\ R_R \qquad \frac{[\Gamma] \to \Downarrow G}{[\Gamma] \to [G]}\ D_R \qquad \frac{\Theta : [\Gamma_1], \Downarrow N_P, [\Gamma_2] \to [G]}{\Theta : [\Gamma_1, N_P, \Gamma_2] \to [G]}\ D_1$$

$$\frac{\Theta : [\Gamma_1], \Downarrow F, [\Gamma_2] \to [G]}{\Theta, F : [\Gamma_1, \Gamma_2] \to [G]}\ D_2 \qquad \frac{\Theta : [\Gamma_1, \sharp^{+} F, \Gamma_2] \to [G]}{\Theta : [\Gamma_1, !F, \Gamma_2] \to [G]}\ D_3$$

$$\frac{\Theta : [\Gamma_1, \sharp^{*} F], \Downarrow F, [\Gamma_2] \to [G]}{\Theta : [\Gamma_1, \sharp^{+/*} F, \Gamma_2] \to [G]}\ D_4$$

Fig. 1. Focused proof system for SLLM#. N_N represent a non-negative formula. N_{NA} represent a non-atomic, non-negative formula. N_P represents a non-positive formula whose main connective is not ∇. N_P^{∇} represents a non-positive formula. We use $\mathcal{R}$ for both $\Uparrow G$ or $[N_N]$. We use $\sharp^{+/*}F$ for both $\sharp^{+}F$ and $\sharp^{*}F$. We use Γ^* for a possibly empty list of formulas of the form $\sharp^{*}F$.

is encoded by $!\nabla A_i$ where

$$A_i = [(q_i \cdot \xi_i) \setminus (\eta_i \cdot q_i')]$$

Let M lead from an initial configuration, represented as $B_1 \cdot q_1 \cdot \xi \cdot B_2$, to the final configuration q_0 (the tape is empty).

We demonstrate how focusing improves search with the encoding of Turing computations. For example, an instruction that write to the tape and moves to the right has the form: $A_i = [(q_i \cdot \xi_i) \setminus (\eta_i \cdot q_i')]$ while the Turing Machine (TM) configuration is encoded as in the sequent context: $[B_1, q_1, \xi, B_2]$ specifies A TM at state q_1 looking at the symbol ξ in the tape B_1, ξ, B_2.

Assuming that $A_1, \ldots, A_n$ is used at least once, $!\nabla A_1, \ldots, !\nabla A_n$ specifies the behavior of the TM. This becomes transparent with focusing. The following focused derivation illustrate how a copy of an instruction encoding, below A_1, can be made available to be used. Recall that the one has to look at the derivation

$$\frac{\Theta_1 : [\Gamma_2], \to \Downarrow F \quad \Theta_2 : [\Gamma_1], \Downarrow G, [\Gamma_3] \to [H]}{\Theta_1, \Theta_2 : [\Gamma_1, \Gamma_2], \Downarrow F \setminus G, [\Gamma_3] \to [H]} \setminus_{L1}^{\star}$$

$$\frac{\Theta_1 : [\sharp^* C, \Gamma_2], \to \Downarrow F \quad \Theta_2 : [\Gamma_1, \sharp^+ C], \Downarrow G, [\Gamma_3] \to [H]}{\Theta_1, \Theta_2 : [\Gamma_1, \sharp^+ C, \Gamma_2], \Downarrow F \setminus G, [\Gamma_3] \to [H]} \setminus_{L2}$$

$$\frac{\Theta_1 : [\sharp^+ C, \Gamma_2], \to \Downarrow F \quad \Theta_2 : [\Gamma_1, \sharp^* C], \Downarrow G, [\Gamma_3] \to [H]}{\Theta_1, \Theta_2 : [\Gamma_1, \sharp^+ C, \Gamma_2], \Downarrow F \setminus G, [\Gamma_3] \to [H]} \setminus_{L3}$$

$$\frac{\Theta_1 : [\sharp^* C, \Gamma_2], \to \Downarrow F \quad \Theta_2 : [\Gamma_1, \sharp^* C], \Downarrow G, [\Gamma_3] \to [H]}{\Theta_1, \Theta_2 : [\Gamma_1, \sharp^* C, \Gamma_2], \Downarrow F \setminus G, [\Gamma_3] \to [H]} \setminus_{L4}$$

$$\frac{\Theta_1 : [\Gamma_2], \to \Downarrow G \quad \Theta_2 : [\Gamma_1], \Downarrow F, [\Gamma_3] \to [H]}{\Theta_1, \Theta_2 : [\Gamma_1], \Downarrow F \,/\, G, [\Gamma_2, \Gamma_3] \to [H]} /_{L1}^{\dagger}$$

$$\frac{\Theta_1 : [\Gamma_2, \sharp^* C], \to \Downarrow G \quad \Theta_2 : [\Gamma_1], \Downarrow F, [\sharp^+ C, \Gamma_3] \to [H]}{\Theta_1, \Theta_2 : [\Gamma_1], \Downarrow F \,/\, G, [\Gamma_2, \sharp^+ C, \Gamma_3] \to [H]} /_{L2}$$

$$\frac{\Theta_1 : [\Gamma_2, \sharp^+ C], \to \Downarrow G \quad \Theta_2 : [\Gamma_1], \Downarrow F, [\sharp^* C, \Gamma_3] \to [H]}{\Theta_1, \Theta_2 : [\Gamma_1], \Downarrow F \,/\, G, [\Gamma_2, \sharp^+ C, \Gamma_3] \to [H]} /_{L3}$$

$$\frac{\Theta_1 : [\Gamma_2, \sharp^* C], \to \Downarrow G \quad \Theta_2 : [\Gamma_1], \Downarrow F, [\sharp^* C, \Gamma_3] \to [H]}{\Theta_1, \Theta_2 : [\Gamma_1], \Downarrow F \,/\, G, [\Gamma_2, \sharp^* C, \Gamma_3] \to [H]} /_{L4}$$

Fig. 2. Focused left introduction rules for $/$ and $\setminus$. The proviso $\star$ in $\setminus_{L1}$ states that the left-most formula of Γ_2 or the right-most formula of Γ_1 are not of the form $\sharp^* F$ nor $\sharp^+ F$. The proviso $\dagger$ in $/_{L1}$ is states that the right-most formula of Γ_2 or the left-most formula of Γ_3 are not of the form $\sharp^* F$ nor $\sharp^+ F$.

from bottom-up.

$$\cfrac{\cfrac{\cfrac{A_1 : [\sharp^* \nabla A_1, \ldots, !\nabla A_n, B_1, q_1, \xi, B_2] \to [q_0]}{\cdot : \Downarrow \nabla A_1, [\sharp^* \nabla A_1, \ldots, !\nabla A_n, B_1, q_1, \xi, B_2] \to [q_0]} R_L, \Uparrow_2}{\cdot : [\sharp^+ \nabla A_1, \ldots, !\nabla A_n, B_1, q_1, \xi, B_2] \to [q_0]} D_4}{\cdot : [!\nabla A_1, \ldots, !\nabla A_n, B_1, q_1, \xi_1, B_2] \to [q_0]} D_3$$

Notice that A_1 is placed in the Θ context. This means that it can be moved at any place. Also notice that since one copy of A_1 is used, $\sharp^+ \nabla A_1$ is replaced by $\sharp^* \nabla A_1$.

The following derivation continues from the premise of the derivation above by focusing on A_1, $\mathcal{A} = \sharp^* \nabla A_1, \ldots, !\nabla A_n$:

$$\cfrac{\cfrac{\cfrac{\cfrac{}{\cdot : [q_1] \to \Downarrow q_1} I \quad \cfrac{}{\cdot : [\eta_1] \to \Downarrow \eta_1} I}{\cdot : [q_1, \xi] \to \Downarrow q_1 \cdot \xi_1} \cdot_R \quad \cfrac{\cfrac{\cdot : [\mathcal{A}, B_1, \eta_1 \cdot q_1', B_2] \to [q_0]}{\cdot : [\mathcal{A}, B_1], \Uparrow \eta_1 \cdot q_1', [B_2] \to [q_0]} \cdot_L, 2\times \Uparrow_{L_1}}{\cdot : [\mathcal{A}, B_1], \Downarrow \eta_1 \cdot q_1', [B_2] \to [q_0]} R_L}{\cdot : [\mathcal{A}, B_1, q_1, \xi], \Downarrow (q_1 \cdot \xi_1) \setminus (\eta_1 \cdot q_1'), [B_2] \to [q_0]} \setminus_L}{A_1 : [\mathcal{A}, B_1, q_1, \xi, B_2] \to [q_0]} D_4$$

Notice that the resulting premise (at the right of the tree) specifies exactly the TM tape resulting from executing the instruction specified by A_1.

Moreover, notice that for the rule D_4, there are many options on where exactly to use A_1. However, it will only work if done as above. This is because otherwise it would not be possible to apply the initial rules as to the left of the derivation above. This reduces considerably the non-determinism involved for proof search.

Finally, once the final configuration q_0 (our Turing machine is responsible for garbage collection, so the configuration is just q_0, with no symbols on the tape) is reached, one finishes the proof:

$$\dfrac{\dfrac{}{\cdot : [\sharp^* A_1, \ldots, \sharp^* A_n, q_0] \to \Downarrow q_0}\ I}{\cdot : [\sharp^* A_1, \ldots, \sharp^* A_n, q_0] \to [q_0]}\ D_R$$

The focusing discipline guarantees that the structure of the proof described above is the only one available. Therefore our encoding soundly and faithfully encodes Turing computations, resulting in the following theorem. However, notice additionally, that due to the non-determinism due to the ! left introduction, the encoding of Turing computations is not on the level of derivations, but on the level of proofs, following the terminology in [24].

The absence of Weakening seems to reduce the expressive power of our system SLLM in the case where not all instructions might have been applied within a particular computation. However, we are still able to get a strong positive statement.

Theorem 5. *We establish a strong one-to-one correspondence between Turing computations and focused derivations in* SLLM*.*

Namely, given a subset I_1, I_2, …, I_m of Turing instructions, the following two statements are equivalent:

(a) The deterministic Turing machine M leads from an initial configuration, represented as $B_1 \cdot q_1 \cdot \xi \cdot B_2$, to the final configuration q_0 so that I_1, I_2, …, I_m are only those instructions that have been actually *applied in the given Turing computation.*

(b) A sequent of the following specified form is derivable in SLLM*.*

$$!\nabla A_{I_1},\ !\nabla A_{I_2},\ \ldots,\ !\nabla A_{I_m},\ B_1 \cdot q_1 \cdot \xi \cdot B_2\ \vdash\ q_0 \tag{9}$$

Corollary 2. *The derivability problem for* SLLM *is undecidable.*

Proof. Assume the contrary: a decision algorithm α decides whether any sequent in SLLM is derivable or not. In particular, for any Turing machine M and any initial configuration of M, α decides whether any sequent of the form (9) is derivable or not, where $B_1 \cdot q_1 \cdot \xi \cdot B_2$ represents the initial configuration.

Then for each of the subsets $\{I_1, I_2, \ldots, I_m\}$ of the instructions of M, we apply α to the corresponding sequent of the form (9).

If all results are negative then we can conclude that M does not terminate on the initial configuration, represented as $B_1 \cdot q_1 \cdot \xi \cdot B_2$.

Otherwise, M does terminate.

Since the halting problem for Turing machines is undecidable, we conclude that α is impossible.

5.2 Decidable Fragments: Syntactically defined

An advantage of our approach is that, unlike the contraction rule, we can syntactically control the multiplexing to provide decidable fragments still sufficient for applications.

Theorem 6. *If we bound k in the multiplexing rule in the calculus* SLLM *with a fixed constant k_0, such a fragment becomes decidable.*

Proof Sketch. Each application of Multiplexing of the form:

$$\frac{\Gamma_1, \overbrace{A, A, \ldots, A}^{k \text{ times}}, \Gamma_2 \to C}{\Gamma_1, !A, \Gamma_2 \to C}\ !_L\ (1 \leq k \leq k_0)$$

multiplies the number of formulas with the factor k_0, which provides an upper bound on the size S of the sequents involved:

$$S = O(S_0 \cdot k_0^n)$$

here S_0 is the size of the input, and n is a bound on the nesting depth of the !-formulas. It suffices to apply a non-deterministic decision procedure but generally on the sequents of exponential size. ■

Theorem 7. *In the case where we bound k in the multiplexing rule in the calculus* SLLM *with a fixed constant k_0, and, in addition, we bound the depth of nesting of $!A$, we get NP-completeness.*

Remark 5. In fact Theorem 7 gives NP-procedures for parsing complex and compound sentences in many practically important cases.

Remark 6. The strong lower bound is given by the following.

The !-free fragment that invokes only one implication, $(A\backslash B)$, and ∇A is still NP-complete.

6 Concluding Remarks

In this paper we have introduced SLLM, a non-commutative intuitionistic linear logic system with two subexponentials. One subexponential implements permutation and the other one obeys the multiplexing rule, which is a weaker, miniature version of contraction. Our system was inspired by *subexponentials* [23], *linear logic* [6], *light linear logic* [7], *soft linear logic* [16], and *easy linear logic* [10].

We have also provided a complete focused proof system for our calculus SLLM. We have illustrated the expressive power of the focused system by modelling computational processes.

The general aim is to develop more refined and efficient procedures for the miniature versions of non-commutative systems, e.g., Lambek calculus and its extensions, based on the multiplexing rule. We aim to ensure a reasonable balance between the expressive power of the formal systems in question and the complexity of their algorithmic implementation. The calculus SLLM, with multiplexing instead of contraction, provides simultaneously three properties: cut elimination, substitution, and Lambek's restriction.

One particular advantage of our system SLLM to systems in our previous work [13, 12] is the fact that it naturally incorporates Lambek's non-emptiness restriction, which is incompatible with stronger systems involving contraction [13]. Lambek's non-emptiness restriction plays a crucial role in applications of substructural (Lambek-style) calculi in formal linguistics (type-logical grammars). Indeed, overcoming this impossibility result is one of our main motivations in looking for a system that would satisfy cut-elimination, substitution, and the Lambek's restriction. The new system proposed in this paper is our proposed solution to this subtle and challenging problem.

Moreover, there is no direct way of reducing the undecidability result in this paper, say, to the undecidability results from our previous papers [11, 13, 15] by a logical translation or representation of the logical systems. Since a number of those systems are also undecidable, there are of course Turing reductions both ways to the system in this paper. However, the Turing reductions factor through the new representation of Turing machines introduced in this paper. That is, the undecidability result in this paper is a new result.

Also the focused proof system proposed here has innovations to the paper [12]. For example, our proof system contains relevant subexponentials that do not allow contraction, something that was not addressed in [12]. Indeed such subexponentials have also been left out of other focused proof systems, e.g., in the papers [23, 25].

Another advantage of this approach is that, unlike the contraction rule, we can syntactically control the multiplexing to provide feasible fragments still sufficient for linguistic applications.

As future work, we are investigating how to extend the systems proposed in this paper with additives. In particular, the proposed focused proof system using the introduced connectives $\#^*, \#^+$ may have to be extended in order to support additives. This is because it does not seem possible with these connectives to make sure that the same number of copies of a contractable formula are used in both premises when introducing an additive connective.

References

1. Andreoli, J.-M.: Logic programming with focusing proofs in linear logic. J. Log. Comput. 2(3): 297-347 (1992)
2. Barry, G., Hepple, M., Leslie, N., Morrill, G.: Proof Figures and Structural Operators for Categorial Grammar. In Proceedings of the Fifth Conference of the European Chapter of the Association for Computational Linguistics, Berlin, 1991.
3. Carpenter, B.: Type-logical semantics. MIT Press, 1998.
4. Cervesato, I., Pfenning, F.: A linear logic framework. In Proceedings, Eleventh Annual IEEE Symposium on Logic in Computer Science, pages 264–275, New Brunswick, New Jersey, July 1996. IEEE Computer Society Press.

5. Cervesato, I., Pfenning, F.: A linear logical framework. Inform. Comput. 179(1): 19–75 (2002).
6. Girard, J.-Y.: Linear logic. Theor. Comput. Sci. 50(1): 1–101 (1987)
7. Girard, J.-Y.: Light linear logic. Inform. and Comput. 143(2): 175–204 (1998)
8. Kanovich, M.I.: Horn programming in linear logic is NP-complete. In LICS 1992, pp. 200–210.
9. Kanovich, M.I., Okada, M., Scedrov, A.: Phase semantics for light linear logic. Theor. Comput. Sci. 294(3): 525–549 (2003).
10. Kanovich, M.I.: Light linear logics with controlled weakening: Expressibility, confluent strong normalization. Annals Pure Appl. Logic 163(7): 854–874 (2012)
11. Kanovich, M., Kuznetsov, S., Nigam, V., Scedrov, A.: Subexponentials in non-commutative linear logic. Math. Struct. Comput. Sci. 29(8): 1217–1249 (2018). (a special issue in celebration of D. Miller's 60th birthday)
12. Kanovich, M., Kuznetsov, S., Nigam, V., Scedrov, A.: A logical framework with commutative and non-commutative Subexponentials. In IJCAR 2018, LNCS/LNAI vol. 10900, Springer, 2018, pp. 228–245.
13. Kanovich, M., Kuznetsov, S., Scedrov, A.: Reconciling Lambek's restriction, cut-elimination, and substitution in the presence of exponential modalities. J. Log. Comput., to appear. (arXiv: 1608.02254)
14. Kanovich, M., Kuznetsov, S., Scedrov, A.: L-models and R-models for Lambek calculus enriched with additives and the multiplicative unit. WoLLIC 2019, LNCS vol. 11541, Springer, 2019, pp. 373–391.
15. Kanovich, M., Kuznetsov, S., Scedrov, A.: Undecidability of the lambek calculus with a relevant modality. In FG 2015/2016, LNCS vol. 9804, Springer, 2016, pp. 240–256.
16. Lafont, Y.: Soft linear logic and polynomial time. Theor. Comput. Sci. 318(1–2): 163–180 (2004)
17. Lambek, J.: The mathematics of sentence structure. Amer. Math. Monthly, 65(3): 154–170 (1958)
18. Lincoln, P., Mitchell, J.C., Scedrov, A., Shankar, N.: Decision problems for propositional linear logic. Annals Pure Appl. Logic 56(1–3): 239–311 (1992)
19. Moot, R., Retoré, C.: The logic of categorial grammars. A deductive account of natural language syntax and semantics. LNCS vol. 6850, Springer, 2012.
20. Moortgat, M.: Constants of grammatical reasoning. In Bouma, G., Hinrichs, E., Kruijff, G.-J., and Oehrle, R. (eds): Constraints and Resources in Natural Language Syntax and Semantics. CSLI Publications, Stanford, 1999, pp. 195–219.
21. Morrill, G.: Categorial grammar: logical syntax, semantics, and processing. Oxford Univ. Press, 2011.
22. Morrill, G., Valentín, O.: Computational Coverage of TLG: Nonlinearity. In Kanazawa, M., Moss, L., de Paiva, V. (eds.): Proceedings of NLCS'15. Third Workshop on Natural Language and Computer Science, EPiC vol. 32, EasyChair, 2015, pp. 51–63.
23. Nigam, N., Miller, D.: Algorithmic specifications in linear logic with subexponentials. In PPDP, pp. 129–140, 2009
24. Nigam, V., Miller, D.: A framework for proof systems. J. of Automated Reasoning. 45(2): 157–188 (2010).
25. Nigam, V., Pimentel, E., Reis, G.: An extended framework for specifying and reasoning about proof systems. J. Log. Comput., 26(2):539-576, 2016.
26. Saurin, A.: Une étude logique du contrôle. PhD Thesis, 2008.
27. Yetter, D. N.: Quantales and (noncommutative) linear logic J. Symb. Log. 55(1): 41–64 (1990).