Superstructural Reversible Logic

Z.A. Sparks Amr Sabry
School of Informatics and Computing
Indiana University, USA
zasparks@cs.indiana.edu sabry@cs.indiana.edu

Linear logic is a "resource-conscious" logic. We analyze the notion of resource maintained by linear logic proofs and argue that it only focuses on a "spatial" dimension of resources (e.g., memory locations) omitting another important "temporal" dimension of resources (e.g., control flow choices). By the same arguments used to explain the conventional spacetime trade-off in computation, these two resources are interchangeable and hence should be managed simultaneously. We therefore refine linear logic to maintain both spatial and temporal resources by adding more structural rules. The resulting logic proves only equivalences that computationally correspond to type isomorphisms. This property allows a study of reversible computation from a proof-theoretic perspective.

1 Introduction

In the original article introducing linear logic [8], Girard describes it as "a resource-conscious logic." However, linear logic does not treat "truth" or "information" as a resource. Indeed, it is known that linear logic allows us to "freely copy" and "foolishly discard" information in the sense of Shannon [17]. Consider the derivation to the right. The assumption $\mathbf{1} \oplus \mathbf{1}$ represents a boolean value and hence contains one bit of information. The conclusion represents the unit value, which has no information. Where did the information contained in the *bool* value go? One can answer this question more clearly by looking at the computational counterpart of the above proof, which is the function $\lambda x.\mathsf{if}\ x\ \mathsf{then}\ ()\ \mathsf{else}\ ()$ of type $\mathit{bool} \multimap \mathbf{1}$. The information in the input is represented by the *choice* of which branch is selected. This choice, which encodes one bit of information, is not recognized as a resource and hence is silently erased in the derivation.

$$\dfrac{\dfrac{}{\cdot \vdash \mathbf{1}}\ \mathbf{1}R}{\mathbf{1} \vdash \mathbf{1}}\ \mathbf{1}L \quad \dfrac{\dfrac{}{\cdot \vdash \mathbf{1}}\ \mathbf{1}R}{\mathbf{1} \vdash \mathbf{1}}\ \mathbf{1}L}{\mathbf{1} \oplus \mathbf{1} \vdash \mathbf{1}}\ \oplus L$$

Put differently, propositions in linear logic model *two* kinds of resources:

- multiplicative resources that combine using $\otimes$: these resources cannot be erased or duplicated and are carefully maintained in all proofs. As Wadler [20] argues, these resources naturally model *memory*. Following the terminology used in monoidal categories, we call such resources *spatial resources* since the object $A \otimes B$ is viewed as representing the parallel composition (in "space") of A and B, i.e., we have two objects that exist at the same time in different locations.
- additive resources that combine using $\oplus$: these resources can be erased and duplicated at will and are not maintained in proofs. These resources model *choices* and hence information. We call such resources *temporal resources* as we view the object $A \oplus B$ as representing a composition in which exactly one of A and B must exist at a given time, i.e., we have two objects that exist in the same location at different times.

This classification of resources is the first insight motivating our technical development. The remaining insight is that spatial resources can be traded for temporal resources and vice-versa. In linear logic, this trade-off is visible in examples such as [12]:

To appear in EPTCS.

```
copyNat zero      =  (zero, zero)
copyNat (succ n)  =  let (n1, n2) = copyNat n in
                     (succ n1, succ n2)
```

The function (or proof) has type $\mathbb{N} \multimap \mathbb{N} \otimes \mathbb{N}$ where $\mathbb{N}$ is the recursive type $1 \oplus 1 \oplus 1 \ldots$, which appears to violate the resource tracking infrastructure of linear logic. The catch is that the function needs an implicit temporal resource that allows it to explore the structure of the input natural number $1 \oplus 1 \oplus 1 \ldots$. Computationally, copying a location containing a natural number requires time proportional to the size of the number, i.e., we can "create" a new location containing n if we consume n time steps. In our setting, this trade-off between spatial and temporal resources is manifested by the fact that the following two propositions are equivalent $A \otimes (B \oplus C)$ and $(A \otimes B) \oplus (A \otimes C)$.

To summarize, our goal is to track what we call *spacetime* resources, i.e., the combination of spatial multiplicative resources and temporal additive resources. We achieve this by a variation of linear logic that maintains information about both $\otimes$ and $\oplus$ in the context in the spirit of the logic of bunched implications (BI) [11]. The resulting logic conserves information, maintains *all* resources, and is *reversible* in the sense that every proof of conclusion B from assumptions A can be reversed to produce A from B.

The rest of the paper introduces the rules for reversible logic, the relevant theorems about it, and discusses its connection to reversible computation. Before proceeding, we briefly justify and explain the words used in the title. The use of the phrase "reversible logic" in the literature is often associated with computation in reversible logic (hardware) circuits. There is only a superficial connection between the two usages that is the same sort of connection between the uses of "logic" in mathematical logic and logic gates.

Our use of "superstructural" alludes to "substructural" logics: describing a logic as "substructural" implicitly refers to the structural rules of intuitionistic or classical logic and conveys the idea that the logic *lacks* some of these usual rules. Although technically accurate, this description de-emphasizes the role of the structural rules. Conversely, our logic places heavy emphasis on its many structural rules, bringing them to the forefront of the presentation: hence the term "superstructural."

2 A Reversible Sequent Calculus

Inspired by our previous work on reversible computation [9] and the technical development of the logic of bunched implications (BI) [11], we refine linear logic to maintain information about both $\otimes$ and $\oplus$ in the contexts of assumptions. We call the resulting system Reversible Logic, or RL. Throughout this section, we will use $\vdash$ as the sequent symbol for RL when it is unambiguously a RL proof. Otherwise, we will distinguish between linear logic sequents, $\vdash_{LL}$, and RL sequents, $\vdash_{RL}$.

2.1 Spacetime contexts and resources

As Girard [8] notes, the way linear logic tracks multiplicative resources that combine with $\otimes$ is by using the "comma" connective in contexts as "hypocrisy" for $\otimes$. Informally, we argue that if our goal is to also maintain resources that combine with $\oplus$, then we should add another connective to contexts that stands for $\oplus$. Furthermore, as $\otimes$ and $\oplus$ are related in non-trivial ways, the resulting contexts should have a non-trivial algebra. Formally, propositions and contexts are defined as follows:

$$\begin{array}{rrcl} \text{Propositions} & A, B & ::= & \mathbf{0} \mid \mathbf{1} \mid A + B \mid A \times B \\ \text{Contexts} & \Gamma, \Delta & ::= & \mathbb{0} \mid \mathbb{1} \mid A \mid \Gamma \oplus \Gamma \mid \Gamma \otimes \Gamma \end{array}$$

$$\begin{array}{rcl} \Gamma \oplus \mathbb{0} & \sim & \Gamma \\ \Gamma_1 \oplus \Gamma_2 & \sim & \Gamma_2 \oplus \Gamma_1 \\ \Gamma_1 \oplus (\Gamma_2 \oplus \Gamma_3) & \sim & (\Gamma_1 \oplus \Gamma_2) \oplus \Gamma_3 \end{array} \qquad \begin{array}{rcl} \Gamma \otimes \mathbb{1} & \sim & \Gamma \\ \Gamma_1 \otimes \Gamma_2 & \sim & \Gamma_2 \otimes \Gamma_1 \\ \Gamma_1 \otimes (\Gamma_2 \otimes \Gamma_3) & \sim & (\Gamma_1 \otimes \Gamma_2) \otimes \Gamma_3 \end{array}$$

$$\begin{array}{rcl} \Gamma \otimes \mathbb{0} & \sim & \mathbb{0} \\ \Gamma_1 \otimes (\Gamma_2 \oplus \Gamma_3) & \sim & (\Gamma_1 \otimes \Gamma_2) \oplus (\Gamma_1 \otimes \Gamma_3) \end{array}$$

$$\frac{}{\Gamma \sim \Gamma} \qquad \frac{\Gamma_1 \sim \Gamma_2}{\Gamma_2 \sim \Gamma_1} \qquad \frac{\Gamma_1 \sim \Gamma_2 \qquad \Gamma_2 \sim \Gamma_3}{\Gamma_1 \sim \Gamma_3} \qquad \frac{\Gamma_1 \sim \Gamma_3 \qquad \Gamma_2 \sim \Gamma_4}{\Gamma_1 \oplus \Gamma_2 \sim \Gamma_3 \oplus \Gamma_4} \qquad \frac{\Gamma_1 \sim \Gamma_3 \qquad \Gamma_2 \sim \Gamma_4}{\Gamma_1 \otimes \Gamma_2 \sim \Gamma_3 \otimes \Gamma_4}$$

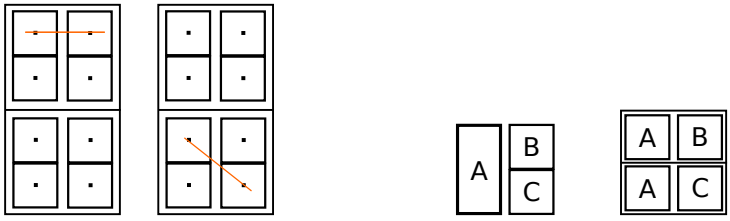
Figure 1: Contextual equivalences in RL

Contexts as trees with two forms of binary connectives and their units. One connective serves as the internalization of *spatial* conjunction, which represents having multiple resources in different places at the same time, and the other connective serves as the internalization of *temporal* conjunction, which represents having one resource that could be in any of several different states at any particular time. Treated as a whole, our contexts track *spacetime* resources. To make things easier to read, we use $\oplus$ and $\otimes$ instead of commas and semicolons for the contextual connectives. The propositional versions of these connectives are $+$ and $\times$, since our logic can also be interpreted as a deductive system for mathematical equality on these operators. (The formal result [5] is that the axioms of a *commutative semiring* constitute a sound and complete set of isomorphisms for finite types built from the empty type, the unit type, disjoint union, and product.)

To gain an intuition for resources, we think of the types as possible events, and of the values as consistent timelines of events. From this perspective, $A \times B$ denotes spatial composition, where both A and B must happen in any order, and $A + B$ denotes temporal composition, where exactly one of A and B must happen at a given time. The event 1 is then trivial, and 0 is impossible. We visualize the denotation of 1 as a point in space, and 0 as empty space. The denotation of $A + B$ is the denotation of A drawn on top of the denotation of B and separated by a border, and the denotation of $A \times B$ is just the denotation of A written next to the denotation of B. Now we can formally define a consistent timeline recursively on the structure of a diagram:

- An endpoint (the denotation of 1) has exactly one timeline through it. Empty space (the denotation of 0) has no timelines through it.
- A timeline through a stack of alternatives $A_1 + A_2 + ... + A_n$ is a timeline through exactly one of the possible alternatives.
- A timeline through a spatial composition of cells $A_1 \times A_2 \times ... \times A_n$ is a timeline through *all* of the cells, in any order.

For example, the following would be a consistent timeline through the diagram for $((2+2) \times (2+2)) + ((2+2) \times (2+2))$ times itself as shown in the left two boxes. The right two boxes show the trade-off between space and time encoded in distributivity. Since a timeline must pass through a column in exactly one point, the two boxes on the right have the same timelines:



$$\frac{\Gamma \vdash A \quad \Gamma' \vdash B}{\Gamma \otimes \Gamma' \vdash A \times B}\ \times R \qquad \frac{\Gamma[A \otimes B] \vdash C}{\Gamma[A \times B] \vdash C}\ \times L \qquad \frac{\Gamma \vdash A \quad \Gamma' \vdash B}{\Gamma \oplus \Gamma' \vdash A + B}\ +R \qquad \frac{\Gamma[A \oplus B] \vdash C}{\Gamma[A + B] \vdash C}\ +L$$

$$\frac{}{\mathbb{O} \vdash \mathbf{0}}\ \mathbf{0}R \qquad \frac{\Gamma[\mathbb{O}] \vdash C}{\Gamma[\mathbf{0}] \vdash C}\ \mathbf{0}L \qquad \frac{}{\mathbb{1} \vdash \mathbf{1}}\ \mathbf{1}R \qquad \frac{\Gamma[\mathbb{1}] \vdash C}{\Gamma[\mathbf{1}] \vdash C}\ \mathbf{1}L$$

Figure 2: Left and right rules in RL

2.2 Contextual equivalence

As mentioned above, contexts form a commutative semiring. The rules of contextual equivalence are shown in Fig. 1. We note that if contexts do not include $\oplus$, we recover linear logic contexts. The new rules encode the (monoidal) behavior of additive resources and, crucially, that multiplicative resources distribute over additive ones: $\Gamma_1 \otimes (\Gamma_2 \oplus \Gamma_3) \sim (\Gamma_1 \otimes \Gamma_2) \oplus (\Gamma_1 \otimes \Gamma_3)$. This rule allows multiplicative resources to be traded for additive ones and vice-versa, reifying the spacetime trade-off that is usually left implicit in programs. At first glance, distributivity appears to violate "meta-linearity," as it duplicates or erases a copy of Γ_1. Looking more closely, though, we find that although it violates *syntactic* linearity, it does not violate *semantic* linearity [6]—since $\oplus$ represents a choice between two alternatives, only one Γ_1 can exist at any time.

Distributivity and factoring provide a mechanism for choosing to delay or force choices, at will. This mechanism already exists implicitly in linear logic: consider an attempt to prove $A \oplus B, C \vdash (A \oplus B) \otimes C$. Without a predetermined proof search strategy or a cut rule, there are two choices: first, we could attempt to apply $\oplus L$:

$$\frac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\ id_A}{A \vdash A \oplus B}\ \oplus R_1 \quad \dfrac{}{C \vdash C}\ id_C}{A, C \vdash (A \oplus B) \otimes C}\ \otimes_R \quad \dfrac{\dfrac{\dfrac{}{B \vdash B}\ id_B}{B \vdash A \oplus B}\ \oplus R_2 \quad \dfrac{}{C \vdash C}\ id_C}{B, C \vdash (A \oplus B) \otimes C}\ \otimes_R}{A \oplus B, C \vdash (A \oplus B) \otimes C}\ \oplus L$$

However, note that this duplicates one of the branches of the proof. We might instead choose to apply $\otimes R$ first:

$$\frac{\dfrac{}{A \oplus B \vdash A \oplus B}\ id_{A \oplus B} \quad \dfrac{}{C \vdash C}\ id_C}{A \oplus B, C \vdash (A \oplus B) \otimes C}\ \otimes R$$

The former proof corresponds to deconstructing the choice first and using the common resource C later (that is, $(\Gamma_1 \otimes \Gamma_2) \oplus (\Gamma_1 \otimes \Gamma_3)$); the latter proof uses the common resource first, then proceeds to deconstruct the choice (that is, $\Gamma_1 \otimes (\Gamma_2 \oplus \Gamma_3)$). Both proofs are, in some sense, the same. In reversible logic, the structure of the context dictates which choice can be made, but they can be switched between at will.

2.3 Inference rules

We now define the left and right rules for our reversible sequent calculus, RL, shown in Fig. 2. As in BI, we want the left rules to be able to act on a proposition nested arbitrarily deeply within the context, so instead of writing Γ, A as in linear logic, we write $\Gamma[A]$ to mean a context Γ with a single hole somewhere in it that is filled by A. More generally, we can fill a hole with an arbitrary context, $\Gamma[\Delta]$. We can also

have a context with multiple holes in it, which we write with curly braces instead of square braces, $\Gamma\{\Delta\}$. The rules for $\times$ work the same as $\otimes$ in linear logic and BI, matching on the context on the right and decomposing the connective into its contextual version on the left. Now that we have a contextual version of $+$, its rules behave identically, transforming propositional $+$ into contextual $\oplus$. The units have similar rules, decomposing them into their contextual forms.

This is a small departure from linear logic, especially with the treatment of $\mathbf{0}$. The decomposition of $\mathbf{1}$ is also important since unlike in linear logic, this unit cannot always be eliminated. Recall the linear $\mathbf{1}L$ rule to the right. This only works because there is only one contextual connective, and $\cdot$, the contextual version of $\mathbf{1}$, is its unit. In RL, we have no such guarantee that in $\Gamma[\mathbb{1}]$, the hole is appearing as an immediate child of a $\otimes$ node, so we must rely on the contextual rules to eliminate it when possible.

$$\frac{\Gamma \vdash_{LL} C}{\Gamma, \mathbf{1} \vdash_{LL} C}\ \mathbf{1}L$$

So far, the rules appear tautological, just allowing us to shift focus from the proposition to the context and vice-versa. This is also the case for multiplicative linear logic, but in addition to this shift of focus, the linear logic context can be silently re-ordered. In our case, we add a rule that explicitly shuffles the resources in the context using the transformations in Fig. 1. The rule is similar to one in BI, with the exception that we treat the contextual equivalence as another derivation rather than a side condition:

$$\frac{\Gamma \sim \Gamma' \quad \Gamma' \vdash C}{\Gamma \vdash C}\ struct$$

For example, a proof that $A \times B \vdash B \times A$ would look as follows:

$$\dfrac{\dfrac{}{A \otimes B \sim B \otimes A} \quad \dfrac{\dfrac{}{B \vdash B}\ id \quad \dfrac{}{A \vdash A}\ id}{B \otimes A \vdash B \times A}\ \times R}{\dfrac{A \otimes B \vdash B \times A}{A \times B \vdash B \times A}\ \times L}\ struct$$

The left and right rules in RL act as wrappers around the combinators that act on contexts. We believe there may be ways to relax these restrictions and make writing proofs in RL more user-friendly, but for now it is intended just as a basic sequent calculus for reversible logic.

2.4 Internal Soundness and Completeness

To demonstrate that RL is internally sound and complete [18], we show that our sequent calculus as presented has admissible identity and cut rules.

Theorem 2.1 (Admissibility of identity)**.** *For all propositions A, $A \vdash A$.*

Proof. By induction on the proposition A. □

The proof of a usual statement of cut, unfortunately, does not proceed smoothly. We must first generalize the cut statement to multicut as in the BI metatheory [13]. This involves a slight change of notation: instead of writing $\Gamma[A]$ for a context Γ with one hole into which A gets substituted, we write $\Gamma\{A\}$ for a context Γ with zero or more holes, all of which contain A.

This generalization alone is not strong enough to prove cut for RL. The eventual result hinges on *contextual parametericity*, as follows:

Lemma 2.2 (Contextual parametericity)**.** *For all multi-hole contexts Γ and Γ' with no copies of A in them, if $\Gamma\{A\} \sim \Gamma'\{A\}$, then for all contexts Δ, $\Gamma\{\Delta\} \sim \Gamma'\{\Delta\}$.*

Proof. By induction on the context Γ. □

Due to the factoring rule (the inverse of distributivity), this lemma only holds with the condition that A does not occur in the contexts other than as a replacement in a hole. Otherwise, substituting in Δ would potentially prevent factoring from happening, even though it could happen before. For example, in the context $(A \otimes B) \oplus (A \otimes C)$, we can factor out A, but if we replace one copy with Δ, we get $(\Delta \otimes B) \oplus (A \otimes C)$ and the rule can no longer immediately be applied. With this lemma out of the way, we can finally state and prove cut:

Theorem 2.3 (Admissibility of cut)**.** *If* $\Delta \vdash A$, $\Gamma\{A\} \vdash C$, *and* A *does not occur outside a hole in* Γ, *then* $\Gamma\{\Delta\} \vdash C$.

Proof. As in BI [13], using the contextual parametericity lemma in the right commutative case of the *struct* rule. □

2.5 Reversibility

Since we have been writing this logic with the goal of reversibility in mind, it is important that we show that it is actually reversible. We can first show that each proof is a logical equivalence. This proof relies on the $\ulcorner - \urcorner$ operator from contexts to propositions, which turns each contextual connective and unit into its propositional counterpart.

Theorem 2.4 (Logical reversibility)**.** *If* $\Gamma \vdash A$, *then* $A \vdash \ulcorner \Gamma \urcorner$.

Proof. Another simple induction on the structure of the derivation of $\Gamma \vdash A$. □

The full version of the paper [19] develops a proof term assignment for RL that can used to show that every RL proof corresponds to a computation witnessing a type isomorphism.

2.6 Superstructural Approach to Logic

The above results demonstrate that the idea of tracking both $\otimes$ and $\oplus$ resources can be formalized in a richer refined variant of linear logic. The development however is confined to finite types, with no negation, and more crucially with no implication. We briefly discuss these current limitations in Sec. 3.

Of more immediate significance is that our technical development motivated by the desire to keep track of resources (re-)emphasizes the crucial nature of structural rules in logic. This resonates with the original motivation to linear logic and with various more recent approaches like the logic of bunched implications (BI) [11] and *deep inference* [2] but we believe we are the first to name the approach and embrace it fully to an extent that subsumes previous approaches.

In more detail, BI takes the structure of contexts seriously and adds an extra connective, which was the original inspiration for our own treatment of contexts. BI, however, explicitly eschews distributivity, which is central to RL; its contextual forms are distinguished by whether or not they allow weakening and contraction. Deep inference goes in a similar direction by focusing more on the structure and manipulation of contexts, but is explicitly a classical logic which is concerned with duality of rules, neither of which was a direction we wanted to go with superstructural logic. We also note that some work in mathematical logic [7] also replaces propositional connectives with separate connectives for structural rules, similar to our work.

Many of these approaches are subsumed by our much more comprehensive and foundational structure on contexts. Specifically, we argue that many of the logics that put some emphasis on the structure of contexts can be recovered from our superstructural logic by *adding* further structural rules to identify

additional contexts. As a concrete example (developed in full in the extended version of this paper [19]), it is possible to recover linear logic by adding the following additional rules and connectives:

$$\mathbb{0} \rightsquigarrow \Gamma \qquad \Gamma \rightsquigarrow \Gamma \circledR \Gamma \qquad \Gamma \rightsquigarrow \mathbb{T} \qquad \Gamma \oplus \Gamma \rightsquigarrow \Gamma$$

where ® is the contextual version of $\&$. These rules capture the implicit information effects inherent in linear logic in an explicit way, by allowing contraction for $\oplus$ and weakening for ®, as well as appropriate rules for their units.

This technique also generalizes to BI, which is directly expressible as a hybrid superstructural-substructural logic, since some of the computation happens outside the structural rules on contexts. We therefore put forward superstructural logic as a framework for studying low-level effects in logics that not even substructural logics can examine, while retaining the ability to work in a proof-theoretic setting.

3 Conclusion and Future Work

We have demonstrated a general approach for writing substructural or otherwise resource-conscious logic that places a heavy emphasis on the structural rules of the logic and subsumes prior approaches. We have used this superstructural approach to develop a sequent calculus for reversible logic.

The current system lacks implication as a connective. Following the original inspiration for linear logic, a *linear* implication is essentially a derived form with $A \multimap B$ defined as $A^{\perp} \parr B$. The definition of implication thus rests fundamentally on a definition of "negation" or in other words on notions of "symmetry" and "duality." The literature includes two distinct approaches to "symmetry" or "duality". One one hand, the historical view dating back to propositional logic and still present in linear logic is to have *one* notion of symmetry $^{\perp}$ that applies universally. In linear logic, this gives rise to conflated notions like "additive pairs" and "multiplicative sums." On the other hand, the literature also includes work which separates the additive from the multiplicative symmetry leading to "negative" and "fractional" types or propositions. This is most apparent in the categorical models based on monoidal categories where each of $\otimes$ and $\oplus$ carries the structure of a symmetric monoidal category that can be completed (i.e., extended with duals and hence higher-order functions) by adding an "opposite" to either $\otimes$ or $\oplus$ separately. Negative types have also appeared separately in various logics most notably Rauszer's *subtractive logic* [14, 15, 16]. Crolard [3, 4] explains the type $A - B$ as the type of *coroutines* with a local environment of type A and a continuation of type B. A notion of fractional types has also appeared in various logics most notably the Lambek-Grishin calculus [1] which includes products as well as a left division and right division which capture constraints on the valid contexts in which a formula can be used. The most consistent approach in our setting is to consider a dual for each resource separately (i.e. a negative symmetry and a fractional symmetry of propositions) and study the interactions between these two duals. We have some initial and promising results that we hope to report on in future work.

We conclude with two additional general directions for future research that are enabled by our results.

Reversible Proof Theory Using the machinery developed here, we plan to investigate reversible computation from a proof-theoretic perspective, which we believe will provide a significant advantage for understanding how to add more type-level features to a reversible programming language. In particular, the translation from a language with information effects to a reversible calculus given by James and Sabry [9] was syntax-directed rather than type-directed; we believe that we may be able to use existential types, as in closure conversion [10], to fix this problem and make sure source language terms of the same type translate to target language terms of the same type.

Computational interpretation. Coming up with a reversible logic is only the first step to studying reversible computation from a proof theoretic perspective; we also need a programming language with a reversible type system based on the logic. The extended version of the paper [19] includes a computational interpretation for RL, which we did not have space for in this paper. It acts as a level of wrappers on top of a simple reversible calculus based on type isomorphisms [9], but we believe the logical setting will allow more improvements in the syntax of the language, and easier extensions to the type system.

References

[1] Raffaella Bernardi & Michael Moortgat (2010): *Continuation semantics for the Lambek–Grishin calculus*. *Inf. Comput.* 208, pp. 397–416.

[2] Kai Brünnler (2003): *Deep inference and symmetry in classical proofs*. Logos Verlag.

[3] Tristan Crolard (2001): *Subtractive logic*. *Theoretical Computer Science* 254(1-2), pp. 151–185.

[4] Tristan Crolard (2004): *A Formulae-as-Types Interpretation of Subtractive Logic*. *Journal of Logic and Computation* 14(4), pp. 529–570.

[5] Marcelo P. Fiore, Roberto Di Cosmo & Vincent Balat (2002): *Remarks on Isomorphisms in Typed Lambda Calculi with Empty and Sum Types*. In: *LICS*.

[6] Marco Gaboardi, Luca Paolini & Mauro Piccolo (2011): *Linearity and PCF: A Semantic Insight!* In: *ICFP*.

[7] Nikolaos Galatos & Hiroakira Ono (2010): *Cut elimination and strong separation for substructural logics: An algebraic approach*. *Annals of Pure and Applied Logic* 161(9).

[8] Jean-Yves Girard (1987): *Linear Logic*. *Theoretical Computer Science*.

[9] Roshan P. James & Amr Sabry (2012): *Information effects*. In: *POPL*.

[10] Yasuhiko Minamide, Greg Morrisett & Robert Harper (1996): *Typed closure conversion*. In: *POPL*.

[11] Peter W. O'Hearn & David J. Pym (1999): *The Logic of Bunched Implications*. *Bulletin of Symbolic Logic*.

[12] Frank Pfenning (1998): *Recursive Types*. `http://www.cs.cmu.edu/~fp/courses/98-linear/handouts/rectypes.pdf`.

[13] David J Pym (2002): *The semantics and proof theory of the logic of bunched implications*. Kluwer Academic.

[14] Cecylia Rauszer (1974): *A formalization of the propositional calculus of H-B logic*. *Studia Logica* 33.

[15] Cecylia Rauszer (1974): *Semi-boolean algebras and their applications to intuitionistic logic with dual operators*. *Fundamenta Mathematicae* 83, pp. 219–249.

[16] Cecylia Rauszer (1980): *An algebraic and Kripke-style approach to a certain extension of intuitionistic logic*. In: *Dissertationes Mathmaticae*, 167, Institut Mathématique de l'Académie Polonaise des Sciences.

[17] Claude Shannon (1948): *A mathematical theory of communication*. *Bell Systems Technical Journal* 27.

[18] Robert J Simmons (2014): *Structural Focalization*. *ACM Transactions on Computational Logic (TOCL)* 15(3).

[19] Zachary Sparks & Amr Sabry (2014): *Superstructural Reversible Logic (Extended version)*. Available from `http://homes.soic.indiana.edu/zasparks/papers/revlog.pdf`.

[20] Philip Wadler (1993): *A Taste of Linear Logic*. In A. M. Borzyszkowski & S. Sokolowski, editors: *Mathematical Foundations of Computer Science*, Springer-Verlag LNCS 781, Gdańsk, Poland.