CS 428/528 Lecture 17: Linear Logic

March 26, 2024

Based on Abramsky's TCS93 paper, Curien's 2018 tutorial, and Orchard et al. ICFP19 paper

Computational Interpretation of Linear Logic [Abramsky TCS 1993]

Computational interpretations of linear logic

Samson Abramsky

Department of Computing, Imperial College of Science, Technology and Medicine, 180 Queen's Gate, London SW7 2BZ, UK

Abstract

Abramsky, S., Computational interpretations of linear logic, Theoretical Computer Science 111 (1993) 3–57.

We study Girard's linear logic from the point of view of giving a concrete computational interpretation of the logic, based on the Curry–Howard isomorphism. In the case of Intuitionistic linear logic, this leads to a refinement of the lambda calculus, giving finer control over order of evaluation and storage allocation, while maintaining the logical content of programs as proofs, and computation as cut-elimination. In the classical case, it leads to a concurrent process paradigm with an operational semantics in the style of Berry and Boudol's chemical abstract machine. This opens up a promising new approach to the parallel implementation of functional programming languages; and offers the prospect of typed concurrent programming in which correctness is guaranteed by the typing.

Review: Natural Deduction

sequent calculus, we present "natural deduction in sequent form", in which the objects being derived are sequents

$$A_1, \ldots, A_n \vdash A.$$

(We use Γ, Δ to range over sequences of formulas, including the empty sequence; and write Γ, Δ for concatenation of sequences.) What distinguishes the system as natural deduction is the form of the rules for each connective: these are structured into *introduction rules* and *elimination rules.*

Review: Natural Deduction for Intuitionistic Logic

Axiom:

$$(\text{Id})\ \frac{}{A \vdash A}$$

Structural rules:

$$(\text{Exchange})\ \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}$$

$$(\text{Contraction})\ \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B} \qquad (\text{Weakening})\ \frac{\Gamma \vdash B}{\Gamma, A \vdash B}$$

Logical rules:

$$(\wedge \text{I})\ \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \qquad (\wedge \text{E})\ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \quad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$$

$$(\supset \text{I})\ \frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B} \qquad (\supset \text{E})\ \frac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

$$(\vee \text{I})\ \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \quad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \qquad (\vee \text{E})\ \frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$$

Review: Curry-Howard Correspondance

their proofs: a proof of a conjunction is a *pair* of proofs of the conjuncts; a proof of an implication $A \supset B$ is a (constructive) *function* mapping proofs of A to proofs of B; a proof of a disjunction $A \vee B$ is *either* a proof of A or a proof of B, together with the information as to which disjunct was actually proved. Thus, propositions are viewed as *data types*:

$$A \wedge B = A \times B \quad \text{(Cartesian product)},$$

$$A \supset B = A \Rightarrow B, \quad \text{(function space)},$$

$$A \vee B = A + B \quad \text{(disjoint union)}.$$

We present the term assignment as a version of natural deduction in which the objects being derived now have the form

$$x_1 : A_1, \ldots, x_n : A_n \vdash t : A$$

Review: Curry-Howard Correspondance

Axiom:

$$\text{(Id)}\ \frac{}{x{:}A \vdash x{:}A}$$

Structural rules:

$$\text{(Exchange)}\ \frac{\Gamma, x{:}A, y{:}B, \Delta \vdash t{:}C}{\Gamma, y{:}B, x{:}A, \Delta \vdash t{:}C}$$

$$\text{(Contraction)}\ \frac{\Gamma, x{:}A, y{:}A \vdash t{:}B}{\Gamma, z{:}A \vdash t[z/x, z/y]{:}B} \qquad \text{(Weakening)}\ \frac{\Gamma \vdash t{:}B}{\Gamma, z{:}A \vdash t{:}B}$$

Review: Curry-Howard Correspondance

Logical rules:

$$(\wedge \mathrm{I})\ \frac{\Gamma \vdash t:A \quad \Gamma \vdash u:B}{\Gamma \vdash \langle t, u\rangle : A \wedge B} \qquad (\wedge \mathrm{E})\ \frac{\Gamma \vdash t:A \wedge B}{\Gamma \vdash \mathsf{fst}(t):A} \quad \frac{\Gamma \vdash t:A \wedge B}{\Gamma \vdash \mathsf{snd}(t):B}$$

$$(\supset \mathrm{I})\ \frac{\Gamma, x:A \vdash t:B}{\Gamma \vdash \lambda x.t : A \supset B} \qquad (\supset \mathrm{E})\ \frac{\Gamma \vdash t:A \supset B \quad \Gamma \vdash u:A}{\Gamma \vdash tu:B}$$

$$(\vee \mathrm{I})\ \frac{\Gamma \vdash t:A}{\Gamma \vdash \mathsf{inl}(t):A \vee B} \quad \frac{\Gamma \vdash u:B}{\Gamma \vdash \mathsf{inr}(u):A \vee B}$$

$$(\vee \mathrm{E})\ \frac{\Gamma \vdash t:A \vee B \quad \Gamma, x:A \vdash u:C \quad \Gamma, y:B \vdash v:C}{\Gamma \vdash \mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}(x) \Rightarrow u \,|\, \mathsf{inr}(y) \Rightarrow v : C}$$

Review: Operational Semantics (Lazy)

Canonical forms:

$$\lambda x.t \qquad \langle t, u\rangle \qquad \mathsf{inl}(t) \qquad \mathsf{inr}(u)$$

Evaluation relation:
This is defined inductively, as the least satisfying the following clauses:

$$\frac{}{\langle t, u\rangle \Downarrow \langle t, u\rangle} \qquad \frac{t \Downarrow \langle u, v\rangle \quad u \Downarrow c}{\mathsf{fst}(t) \Downarrow c} \qquad \frac{t \Downarrow \langle u, v\rangle \quad v \Downarrow c}{\mathsf{snd}(t) \Downarrow c}$$

$$\frac{}{\lambda x.t \Downarrow \lambda x.t} \qquad \frac{t \Downarrow \lambda x.v \quad v[u/x] \Downarrow c}{tu \Downarrow c}$$

$$\frac{}{\mathsf{inl}(t) \Downarrow \mathsf{inl}(t)} \qquad \frac{}{\mathsf{inr}(u) \Downarrow \mathsf{inr}(u)}$$

$$\frac{t \Downarrow \mathsf{inl}(w) \quad u[w/x] \Downarrow c}{\mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}(x) \Rightarrow u \mid \mathsf{inr}(y) \Rightarrow v \Downarrow c}$$

$$\frac{t \Downarrow \mathsf{inr}(w) \quad v[w/y] \Downarrow c}{\mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}(x) \Rightarrow u \mid \mathsf{inr}(y) \Rightarrow v \Downarrow c}$$

Review: Operational Semantics (Eager)

Canonical forms:

$$\lambda x.t \qquad \langle c, d \rangle \qquad \mathsf{inl}(c) \qquad \mathsf{inr}(d)$$

where c, d are canonical forms.

Evaluation relation:

$$\frac{t \Downarrow c \quad u \Downarrow d}{\langle t, u \rangle \Downarrow \langle c, d \rangle} \qquad \frac{t \Downarrow \langle c, d \rangle}{\mathsf{fst}(t) \Downarrow c} \qquad \frac{t \Downarrow \langle c, d \rangle}{\mathsf{snd}(t) \Downarrow d}$$

$$\frac{}{\lambda x.t \Downarrow \lambda x.t} \qquad \frac{t \Downarrow \lambda x.v \quad u \Downarrow c \quad v[c/x] \Downarrow d}{tu \Downarrow d}$$

$$\frac{t \Downarrow c}{\mathsf{inl}(t) \Downarrow \mathsf{inl}(c)} \qquad \frac{u \Downarrow d}{\mathsf{inr}(u) \Downarrow \mathsf{inr}(d)}$$

$$\frac{t \Downarrow \mathsf{inl}(c) \quad u[c/x] \Downarrow d}{\mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}(x) \Rightarrow u | \mathsf{inr}(y) \Rightarrow v \Downarrow d}$$

$$\frac{t \Downarrow \mathsf{inr}(c) \quad v[c/y] \Downarrow d}{\mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}(x) \Rightarrow u | \mathsf{inr}(y) \Rightarrow v \Downarrow d}$$

Review: Sequent Calculus

2.4. Sequent calculus

We now review the sequent calculus presentation of intuitionistic logic. The objects derived in this calculus are exactly the same sequents $\Gamma \vdash A$ as in (our version of) natural deduction. The difference appears in the form of the rules. Firstly, the Cut rule

$$\frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}$$

Logical rules:

$$(\wedge \mathrm{R}) \ \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \wedge B} \qquad (\wedge \mathrm{L}) \ \frac{\Gamma, A \vdash C}{\Gamma, A \wedge B \vdash C} \quad \frac{\Gamma, B \vdash C}{\Gamma, A \wedge B \vdash C}$$

$$(\supset \mathrm{R}) \ \frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B} \qquad (\supset \mathrm{L}) \ \frac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Gamma, A \supset B, \Delta \vdash C}$$

$$(\vee \mathrm{R}) \ \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \quad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \qquad (\vee \mathrm{L}) \ \frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C}$$

Review: Sequent Calculus (Curry-Howard)

Cut rule:

$$\frac{\Gamma \vdash t:A \quad x:A, \Delta \vdash u:B}{\Gamma, \Delta \vdash u[t/x]:B}$$

Logical rules:

$$(\wedge \mathrm{R}) \ \frac{\Gamma \vdash t:A \quad \Delta \vdash u:B}{\Gamma, \Delta \vdash \langle t, u \rangle : A \wedge B}$$

$$(\wedge \mathrm{L}) \ \frac{\Gamma, x:A \vdash t:C}{\Gamma, z:A \wedge B \vdash t[\mathsf{fst}(z)/x]:C} \quad \frac{\Gamma, y:B \vdash u:C}{\Gamma, z:A \wedge B \vdash u[\mathsf{snd}(z)/y]:C}$$

$$(\supset \mathrm{R}) \ \frac{\Gamma, x:A \vdash t:B}{\Gamma \vdash \lambda x.t : A \supset B} \qquad (\supset \mathrm{L}) \ \frac{\Gamma \vdash t:A \quad x:B, \Delta \vdash u:C}{\Gamma, f:A \supset B, \Delta \vdash u[(ft)/x]:C}$$

$$(\vee \mathrm{R}) \ \frac{\Gamma \vdash t:A}{\Gamma \vdash \mathsf{inl}(t):A \vee B} \quad \frac{\Gamma \vdash u:B}{\Gamma \vdash \mathsf{inr}(u):A \vee B}$$

$$(\vee \mathrm{L}) \ \frac{\Gamma, x:A \vdash u:C \quad \Gamma, y:B \vdash v:C}{\Gamma, z:A \vee B \vdash \mathsf{case}\ z\ \mathsf{of}\ \mathsf{inl}(x) \Rightarrow u \,|\, \mathsf{inr}(y) \Rightarrow v : C}$$

Note that terms generated by cut-free proofs are in *normal form*; in partic

Intuitionistic Linear Logic (ILL)

$$\text{(Contraction)}\ \frac{\Gamma, x:A, y:A \vdash t:B}{\Gamma, z:A \vdash t[z/x, z/y]:B} \qquad \text{(Weakening)}\ \frac{\Gamma \vdash t:B}{\Gamma, z:A \vdash t:B}$$

Axiom:

$$\text{(Id)}\ \frac{}{A \vdash A}$$

Structural rule:

$$\text{(Exchange)}\ \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}$$

Cut rule:

$$\frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}$$

Intuitionistic Linear Logic (ILL)

- If we wish to use *both* components of the pair, on the unique occasion when we use the input, then we lose the ability to project. This leads to the multiplicative version of conjunction, the tensor product $A \otimes B$.
- If we wish to project, then on the unique occasion when we use the input, we must choose to take *either* the first *or* the second projection, and this is the only part of the input we will ever see. So the additive conjunction $A \& B$ appears as a kind of *choice* – an "external" choice in the terminology of CSP [19], since it is made by the consumer of the datum.
- The disjoint sum (additive disjunction) $A \oplus B$ appears as an *internal* choice, since it is at the discretion of the producer of the datum whether the choice is made from A – a value of the form $\mathsf{inl}(t)$, or from B – a value of the form $\mathsf{inr}(u)$.
- The linear implication $A \multimap B$ is the type of functions which use their argument exactly once, internalizing linear inference.

Intuitionistic Linear Logic (ILL)

Logical rules:

$$(\mathbf{1}\mathrm{R})\ \frac{}{\vdash \mathbf{1}} \qquad (\mathbf{1}\mathrm{L})\ \frac{\Gamma \vdash A}{\Gamma, \mathbf{1} \vdash A}$$

$$(\otimes \mathrm{R})\ \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \qquad (\otimes \mathrm{L})\ \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}$$

$$(\multimap \mathrm{R})\ \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \qquad (\multimap \mathrm{L})\ \frac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Gamma, A \multimap B, \Delta \vdash C}$$

$$(\& \mathrm{R})\ \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B} \qquad (\& \mathrm{L})\ \frac{\Gamma, A \vdash C}{\Gamma, A \& B \vdash C} \quad \frac{\Gamma, B \vdash C}{\Gamma, A \& B \vdash C}$$

$$(\circledast \mathrm{R})\ \frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B} \quad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B} \qquad (\oplus \mathrm{L})\ \frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}$$

Intuitionistic Linear Logic (ILL)

Menu (price 17 Euros)

Quiche or Salad
Chicken or Fish
Banana or "Surprise du Chef"*

(*) either "*Profiteroles*" or "*Tarte Tatin*"

$$17E \vdash \left\{ \begin{array}{c} (Q\&S) \\ \otimes \\ (C\&F) \\ \otimes \\ (B\&(P \oplus T)) \end{array} \right.$$

We can recognize here some of the meanings that we already discussed. The right of the sequent is what you can get for 17 Euros. The tensor tells that for this price you get one "entrée", one dish and one dessert. The difference between & and $\oplus$ is a bit more subtle, and the game interpretation is helpful here. So let us start again from the beginning, considering a play between the restaurant manager (the Player) and the customer (the Opponent). It is the Player's responsibility to split the $17E$ into three parts, corresponding to the cost of the three parts of the meal. May be, this is done as follows:

$$\frac{5E \vdash Q\&S \quad 8E \vdash C\&F \quad 4E \vdash B\&(P \oplus T)}{17E \vdash (Q\&S) \otimes (C\&F) \otimes (B\&(P \oplus T))}$$

Intuitionistic Linear Logic (ILL)

Now let the Opponent challenge $5E \vdash Q \& S$:

$$\frac{5E \vdash Q \quad 5E \vdash S}{5E \vdash Q \& S}$$

which reads as: both Quiche and Salad are available to the customer, but he can get only one, and it is *his* choice of picking one of the antecedents and to order, say, a Quiche. Thus the additive conjunction can be understood as a ... disjunction embodying a notion of *external choice* (remember that in our example the customer is the Opponent, or the context, or the environment). Let us now analyse a proof of $4E \vdash B \& (P \oplus T)$:

$$\frac{4E \vdash B \quad \dfrac{4E \vdash T}{4E \vdash P \oplus T}}{4E \vdash B \& (P \oplus T)}$$

Suppose that the Opponent chooses the Surprise. Then it is the Player's turn, who justifies $4E \vdash P \oplus T$ using the right $\oplus$ rule. So, the Opponent will get a Tarte Tatin, but the choice was in the Player's hands. Thus $\oplus$ has an associated meaning of *internal choice*. In summary, two forms of choice, external and internal, are modelled by & and $\oplus$, respectively. In the case of $\oplus$, whether A or B is chosen is controlled by the rule, that is, by the Player. In the case of &, the choice between A or B is a choice of one of the antecedents of the rule, and is in the hands of the Opponent.

Intuitionistic Linear Logic (ILL)

Logical rules:

$$(\mathbf{1}\mathrm{R})\ \frac{}{\vdash \mathbf{1}} \qquad (\mathbf{1}\mathrm{L})\ \frac{\Gamma \vdash A}{\Gamma, \mathbf{1} \vdash A}$$

$$(\otimes \mathrm{R})\ \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \qquad (\otimes \mathrm{L})\ \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}$$

$$(\multimap \mathrm{R})\ \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \qquad (\multimap \mathrm{L})\ \frac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Gamma, A \multimap B, \Delta \vdash C}$$

$$(\& \mathrm{R})\ \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B} \qquad (\& \mathrm{L})\ \frac{\Gamma, A \vdash C}{\Gamma, A \& B \vdash C} \quad \frac{\Gamma, B \vdash C}{\Gamma, A \& B \vdash C}$$

$$(\circledast \mathrm{R})\ \frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B} \quad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B} \qquad (\oplus \mathrm{L})\ \frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}$$

Intuitionistic Linear Logic (ILL)

These connectives by themselves are far too weak to provide useful expressive power. This is regained by reintroducing weakening and contraction in a controlled form, not as omnipresent structural rules, but reflected into a datatype, the exponential $!A$ ("Of course A"). The effect is to make as many copies of a value of type A available as may be needed.

That we have recovered adequate expressive power is witnessed by the fact that intuitionistic logic can be interpreted in linear logic with the above connectives. In particular, intuitionistic implication is recovered by

$$A \supset B = !A \multimap B.$$

$$(!\mathrm{R})\ \frac{!\Gamma \vdash A}{!\Gamma \vdash !A} \qquad (\text{Dereliction})\ \frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B}$$

($!\Gamma$ means a sequence of the form $!A_1, \ldots, !A_k$.)

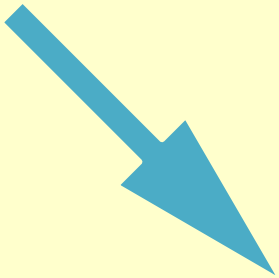
$$(\text{Contraction})\ \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B} \qquad (\text{Weakening})\ \frac{\Gamma \vdash B}{\Gamma, !A \vdash B}$$

Bounded Linear Logic (BLL)

$$(!\mathrm{R})\ \frac{!\Gamma \vdash A}{!\Gamma \vdash !A} \qquad (\text{Dereliction})\ \frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B}$$

($!\Gamma$ means a sequence of the form $!A_1, \ldots, !A_k$.)

$$(\text{Contraction})\ \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B} \qquad (\text{Weakening})\ \frac{\Gamma \vdash B}{\Gamma, !A \vdash B}$$



in Sections 3 and 4. If Γ is $A_1, \ldots, A_n$ we write $!_{\vec{y}}\Gamma$ for $!_{y_1}A_1, \ldots, !_{y_n}A_n$.

The rules for storage naturally induce polynomials:

$$\textit{Storage}\ \frac{!_{\vec{y}}\Gamma \vdash A}{!_{x\vec{y}}\Gamma \vdash !_x A} \qquad \textit{Weakening}\ \frac{\Gamma \vdash B}{\Gamma, !_0 A \vdash B}$$

$$\textit{Contraction}\ \frac{\Gamma, !_x A, !_y A \vdash B}{\Gamma, !_{x+y} A \vdash B} \qquad \textit{Dereliction}\ \frac{\Gamma, A \vdash B}{\Gamma, !_1 A \vdash B}.$$

We may interpret these rules in second-order RLL, by translating $!_x A$ as

$$\underbrace{1 \otimes A \otimes \cdots \otimes A}_{x\ \otimes\text{'s}}$$

Intuitionistic LL (Curry-Howard)

patterns. We use X, Y, Z to range over finite sets of variables. Now $\mathscr{P}_X$, the set of patterns with variables in X, is defined as follows:

$$*, \_ \in \mathscr{P}_\emptyset \qquad \langle x, \_ \rangle, \langle \_, x \rangle, !x \in \mathscr{P}_{\{x\}} \qquad x \otimes y, x @ y \in \mathscr{P}_{\{x,y\}}$$

We can now define $\mathscr{T}_X$, the linear terms with free variables in X, inductively as follows:

- $x \in \mathscr{T}_{\{x\}}$
- $* \in \mathscr{T}_\emptyset$
- $t \in \mathscr{T}_X, u \in \mathscr{T}_Y, X \cap Y = \emptyset \Rightarrow t \otimes u, tu \in \mathscr{T}_{X \cup Y}$
- $t, u \in \mathscr{T}_X \Rightarrow \langle t, u \rangle \in \mathscr{T}_X$
- $t \in \mathscr{T}_X \Rightarrow \mathsf{inl}(t), \mathsf{inr}(t), !t \in \mathscr{T}_X$
- $t \in \mathscr{T}_{X \cup \{x\}}, x \notin X \Rightarrow \lambda x.t \in \mathscr{T}_X$
- $t \in \mathscr{T}_X, p \in \mathscr{P}_Y, u \in \mathscr{T}_{Y \cup Z}, X \cap Z = Y \cap Z = \emptyset$
  $\Rightarrow$ let t be p in $u \in \mathscr{T}_{X \cup Z}$
- $t \in \mathscr{T}_X, u \in \mathscr{T}_{Z \cup \{x\}}, v \in \mathscr{T}_{Z \cup \{y\}}, X \cap Z = \{x, y\} \cap Z = \emptyset$
  $\Rightarrow$ case t of $\mathsf{inl}(x) \Rightarrow u | \mathsf{inr}(y) \Rightarrow v \in \mathscr{T}_{X \cup Z}$

Intuitionistic LL (Curry-Howard)

We now present the assignment of linear terms to proofs in intuitionistic linear logic, in the same style as the term assignment for sequent calculus given in the previous section. Our sequents now have the form

$$x_1:A_1, \ldots, x_k:A_k \vdash t:A$$

where the A_i are linear formulae (built from the connectives $\mathbf{1}$, $\otimes$, $\multimap$, $\&$, $\oplus$, $!$), the x_i are distinct variables, and $t \in \mathscr{T}_X$, $X = \{x_1, \ldots, x_k\}$. Note that the rules presented below are subject to the implicit constraint that the linearity conditions for well-formedness of terms are satisfied. This constraint can always be met, e.g. by using distinct variables for all instances of the Axiom.

Axiom:

$$\text{(Id)}\ \frac{}{x:A \vdash x:A}$$

Structural rule:

$$\text{(Exchange)}\ \frac{\Gamma, x:A, y:B, \Delta \vdash t:C}{\Gamma, y:B, x:A, \Delta \vdash t:C}$$

Cut rule:

$$\frac{\Gamma \vdash t:A \quad x:A, \Delta \vdash u:B}{\Gamma, \Delta \vdash u[t/x]:B}$$

Logical rules:

$$\text{(1R)}\ \frac{}{\vdash *:\mathbf{1}} \qquad \text{(1L)}\ \frac{\Gamma \vdash t:A}{\Gamma, z:\mathbf{1} \vdash \mathsf{let}\ z\ \mathsf{be}\ *\ \mathsf{in}\ t:A}$$

$$\text{($\otimes$R)}\ \frac{\Gamma \vdash t:A \quad \Delta \vdash u:B}{\Gamma, \Delta \vdash t \otimes u:A \otimes B} \qquad \text{($\otimes$L)}\ \frac{\Gamma, x:A, y:B \vdash t:C}{\Gamma, z:A \otimes B \vdash \mathsf{let}\ z\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ t:C}$$

$$\text{($\multimap$R)}\ \frac{\Gamma, x:A \vdash t:B}{\Gamma \vdash \lambda x.t:A \multimap B} \qquad \text{($\multimap$L)}\ \frac{\Gamma \vdash t:A \quad x:B, \Delta \vdash u:C}{\Gamma, f:A \multimap B, \Delta \vdash u[(ft)/x]:C}$$

Intuitionistic LL (Curry-Howard)

$$(\&\mathrm{R})\ \frac{\Gamma \vdash t:A \quad \Gamma \vdash u:B}{\Gamma \vdash \langle t, u\rangle : A \& B}$$

$$(\&\mathrm{L})\ \frac{\Gamma, x:A \vdash t:C}{\Gamma, z:A\&B \vdash \mathsf{let}\ z\ \mathsf{be}\ \langle x, \_\rangle\ \mathsf{in}\ t:C}$$

$$\frac{\Gamma, y:B \vdash t:C}{\Gamma, z:A\&B \vdash \mathsf{let}\ z\ \mathsf{be}\ \langle \_, y\rangle\ \mathsf{in}\ t:C}$$

$$(\oplus\mathrm{R})\ \frac{\Gamma \vdash t:A}{\Gamma \vdash \mathsf{inl}(t):A \oplus B} \quad \frac{\Gamma \vdash u:B}{\Gamma \vdash \mathsf{inr}(u):A \oplus B}$$

$$(\oplus\mathrm{L})\ \frac{\Gamma, x:A \vdash u:C \quad \Gamma, y:B \vdash v:C}{\Gamma, z:A \oplus B \vdash \mathsf{case}\ z\ \mathsf{of}\ \mathsf{inl}(x) \Rightarrow u \,|\, \mathsf{inr}(y) \Rightarrow v:C}$$

$$(!\mathrm{R})\ \frac{!\Gamma \vdash t:A}{!\Gamma \vdash !t:!A}$$

$$(\text{Dereliction})\ \frac{\Gamma, x:A \vdash t:B}{\Gamma, z:!A \vdash \mathsf{let}\ z\ \mathsf{be}\ !x\ \mathsf{in}\ t:B}$$

$$(\text{Contraction})\ \frac{\Gamma, x:!A, y:!A \vdash t:B}{\Gamma, z:!A \vdash \mathsf{let}\ z\ \mathsf{be}\ x@y\ \mathsf{in}\ t:B}$$

$$(\text{Weakening})\ \frac{\Gamma \vdash t:B}{\Gamma, z:!A \vdash \mathsf{let}\ z\ \mathsf{be}\ \_\ \mathsf{in}\ t:B}$$

Intuitionistic LL (Curry-Howard)

Operational semantics

We now give an operational semantics for the linear term calculus in exactly the same style, and with the same supporting intuitions as the semantics of the λ-calculus given in the previous section. However, one notable difference emerges, as immediate evidence of the more refined computational content of the linear types. Whereas intuitionistic logic was perfectly neutral as to which evaluation strategy to adopt for the λ-calculus, in linear logic the logical structure of the types gives a clear indication as to which form of evaluation to employ:

- For a term of tensor type $A \otimes B$ we know that any consumer (e.g. a destructor context let $[\cdot]$ be $x \otimes y$ in u) will evaluate this term to a pair, and *use both components*. This clearly indicates eager evaluation. Similarly, a term of type $A \multimap B$ must evaluate to an abstraction, which when applied to any argument will evaluate it exactly once. Evaluation of the argument exactly once is the slogan of call-by-value [37]. Finally, any consumer of a term of type $A \oplus B$ will evaluate it to a term of the form $\mathsf{inl}(t)$ or $\mathsf{inr}(t)$, and then use t in evaluating the appropriate arm of a case statement; so once again, eager evaluation is indicated.
- On the other hand, a value of type $A \& B$ will evaluate to a pair, *exactly one component of which* will be used in any given context. Since we cannot predict which component will be used, it is clear that evaluating either component in advance of their actual use will lead in general to redundant computation. Thus lazy evaluation is indicated here. Again, a value of type $!A$ may be discarded altogether, so evaluation in advance of actual use may lead to redundant computation, and lazy evaluation is indicated.

Thus, we get a classification:

- $\otimes$, $\multimap$, $\oplus$ (eager evaluation)
- $\&$, $!$ (lazy evaluation).

Intuitionistic LL (Curry-Howard)

Canonical forms:

$$\langle t, u\rangle \qquad !t$$

$$* \qquad c \otimes d \qquad \lambda x.t \qquad \mathsf{inl}(c) \qquad \mathsf{inr}(d)$$

where c, d are canonical.

Evaluation relation:

$$\frac{}{* \Downarrow *} \qquad \frac{t \Downarrow * \quad u \Downarrow c}{\mathsf{let}\ t\ \mathsf{be} * \mathsf{in}\ u \Downarrow c}$$

$$\frac{t \Downarrow c \quad u \Downarrow d}{t \otimes u \Downarrow c \otimes d} \qquad \frac{t \Downarrow c \otimes d \quad u[c/x, d/y] \Downarrow e}{\mathsf{let}\ t\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ u \Downarrow e}$$

$$\frac{}{\lambda x.t \Downarrow \lambda x.t} \qquad \frac{t \Downarrow \lambda x.v \quad u \Downarrow c \quad v[c/x] \Downarrow d}{tu \Downarrow d}$$

$$\frac{}{\langle t, u\rangle \Downarrow \langle t, u\rangle} \qquad \frac{t \Downarrow \langle v, w\rangle \quad v \Downarrow c \quad u[c/x] \Downarrow d}{\mathsf{let}\ t\ \mathsf{be}\ \langle x, \_\rangle\ \mathsf{in}\ u \Downarrow d}$$

$$\frac{t \Downarrow \langle v, w\rangle \quad w \Downarrow c \quad u[c/y] \Downarrow d}{\mathsf{let}\ t\ \mathsf{be}\ \langle \_, y\rangle\ \mathsf{in}\ u \Downarrow d}$$

$$\frac{t \Downarrow c}{\mathsf{inl}(t) \Downarrow \mathsf{inl}(c)} \qquad \frac{u \Downarrow d}{\mathsf{inr}(u) \Downarrow \mathsf{inr}(d)}$$

$$\frac{t \Downarrow \mathsf{inl}(c) \quad u[c/x] \Downarrow d}{\mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}(x) \Rightarrow u | \mathsf{inr}(y) \Rightarrow v \Downarrow d}$$

$$\frac{t \Downarrow \mathsf{inr}(c) \quad v[c/y] \Downarrow d}{\mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}(x) \Rightarrow u | \mathsf{inr}(y) \Rightarrow v \Downarrow d}$$

$$\frac{}{!t \Downarrow !t} \qquad \frac{t \Downarrow !v \quad v \Downarrow c \quad u[c/x] \Downarrow d}{\mathsf{let}\ t\ \mathsf{be}\ !x\ \mathsf{in}\ u \Downarrow d}$$

$$\frac{t \Downarrow !v \quad u \Downarrow c}{\mathsf{let}\ t\ \mathsf{be}\ \_\ \mathsf{in}\ u \Downarrow c} \qquad \frac{t \Downarrow !v \quad u[!v/x, !v/y] \Downarrow c}{\mathsf{let}\ t\ \mathsf{be}\ x@y\ \mathsf{in}\ u \Downarrow c}$$

Classical Linear Logic (CLL)

The basic step in the extension from intuitionistic to classical linear logic is the introduction of the *linear negation* $A^{\perp}$. The idea is that this will obey the same kind of laws as classical negation, while constructive content is retained through linearity. This requires the introduction of a number of new connectives, as duals to the existing ones: $\perp$ as dual to $\mathbf{1}$, $\parr$ ("par") as dual to $\otimes$, ? ("why not") as dual to !, and $\exists$ as dual to $\forall$. (The two additive constructs & and $\oplus$ will be dual to each other in CLL.) Linear negation is then characterized by the following laws:

$$
\begin{aligned}
&A^{\perp\perp} = A \\
&\mathbf{1}^{\perp} = \perp \\
&(A \otimes B)^{\perp} = A^{\perp} \parr B^{\perp} \\
&(A \,\&\, B)^{\perp} = A^{\perp} \oplus B^{\perp} \\
&(!A)^{\perp} = ?A^{\perp} \\
&(\forall \alpha . A)^{\perp} = \exists \alpha . A^{\perp} \\
&A \multimap B = A^{\perp} \parr B.
\end{aligned}
\tag{18}
$$

The syntax of linear formulas in CLL is then defined as follows. Formulas are built from propositional variables α, β, γ and their linear negations $\alpha^{\perp}, \beta^{\perp}, \gamma^{\perp}$ by the following connectives and quantifiers:

Units	$\mathbf{1}$	$\perp$
Multiplicatives	$\otimes$	$\parr$
Additives	&	$\oplus$
Exponentials	!	?
Quantifiers	$\forall$	$\exists$.

Classical Linear Logic (CLL)

The proof system for CLL is a fully symmetric sequent calculus, in which sequents have the form

$$\Gamma \vdash \Delta,$$

with the intended meaning that the formula $\otimes \Gamma \multimap \parr \Delta$ is valid. However, a considerable economy is gained by observing that a sequent $\Gamma \vdash \Delta$ is equivalent, by (18), to the sequent $\vdash \Gamma^{\perp}, \Delta$; so it is sufficient to consider right-sided sequents only. The sequent calculus presentation of CLL can then be given as follows:

Axiom $\dfrac{}{\vdash A^{\perp}, A}$ Exchange $\dfrac{\vdash \Gamma, A, B, \Delta}{\vdash \Gamma, B, A, \Delta}$ Cut $\dfrac{\vdash \Gamma, A \quad \vdash \Delta, A^{\perp}}{\vdash \Gamma, \Delta}$

Unit $\dfrac{}{\vdash \mathbf{1}}$ Perp $\dfrac{\vdash \Gamma}{\vdash \Gamma, \perp}$

Times $\dfrac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}$ Par $\dfrac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}$

With $\dfrac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}$ Plus (i) $\dfrac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}$ Plus (ii) $\dfrac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}$

Dereliction $\dfrac{\vdash \Gamma, A}{\vdash \Gamma, ?A}$ Of Course $\dfrac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}$

Weakening $\dfrac{\vdash \Gamma}{\vdash \Gamma, ?A}$ Contraction $\dfrac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}$

All $\dfrac{\vdash \Gamma, A}{\vdash \Gamma, \forall \alpha . A}$ $(*)$ Exists $\dfrac{\vdash \Gamma, A[B/\alpha]}{\vdash \Gamma, \exists \alpha . A}$

CLL Computational Interpretation

6.1. Syntax of proof expressions

Firstly, a point of terminology: we shall use *list* to mean finite sequence. We define a number of syntactic categories:

- A set $\mathcal{N}$ of *names*, ranged over by x, y, z. We use $\bar{x}, \bar{y}, \bar{z}$ to range over lists of names.
- *Terms* have one of the forms

x			
$*$	$\circledast$		
$t \otimes u$	$t \parr u$		
$\mathsf{inl}(t)$	$\mathsf{inr}(u)$	$\bar{x}(P \,[]\, Q)$	
$?t$	$-$	$t @ u$	$\bar{x}(P)$

where t, u are terms, and P, Q are proof expressions.

- *Coequations* have the form $t \perp u$, where t, u are terms. We use Θ, Ξ to range over lists of coequations.
- *Proof expressions* have the form $\Theta; \bar{t}$, where Θ is a list of coequations, and $\bar{t}$ is a list of terms. We use P, Q to range over proof expressions.

CLL Computational Interpretation

Notation. The occurrences of $x_1, \ldots, x_k$ in a term of the form $x_1, \ldots, x_k(P \square Q)$ or $x_1, \ldots, x_k(P)$ are said to be *passive*; all other occurrences are *active*. If e is some syntactic expression (term, coequation, proof expression, etc.), we write $\mathcal{N}(e)$ for the set of names occurring in e, and $A\mathcal{N}(e)$ ($\mathrm{P}\mathcal{N}(e)$) for the set of names occurring actively (passively) in e.

We shall now define an assignment of proof expressions to sequent proofs in CLL. The idea is that, to each proof Π of a sequent $\vdash A_1, \ldots, A_k$, we will assign a proof expression $\Theta; t_1, \ldots, t_k$, where Θ corresponds to the uses of the Cut rule in Π.

To ensure that suitable linearity constraints are satisfied, we shall adopt the following name convention (cf. the variable convention in [3]): *different* names are introduced for each instance of the Axiom, With and Of Course rules.

CLL Computational Interpretation

Proof expression assignment for CLL

$$\text{Axiom}\ \frac{}{\vdash ; x{:}A^{\perp}, x{:}A} \qquad \text{Exchange}\ \frac{\vdash \Theta; \Gamma, t{:}A, u{:}B, \Delta}{\vdash \Theta; \Gamma, u{:}B, t{:}A, \Delta}$$

$$\text{Cut}\ \frac{\vdash \Theta; \Gamma, t{:}A \quad \vdash \Xi; \Delta, u{:}A^{\perp}}{\vdash \Theta, \Xi, t \perp u; \Gamma, \Lambda}$$

$$\text{Unit}\ \frac{}{\vdash ; * {:} \mathbf{1}} \qquad \text{Perp}\ \frac{\vdash \Theta; \Gamma}{\vdash \Theta; \Gamma, \circledast : \perp}$$

$$\text{Times}\ \frac{\vdash \Theta; \Gamma, t{:}A \quad \vdash \Xi; \Delta, u{:}B}{\vdash \Theta, \Xi; \Gamma, \Delta, t \otimes u{:}A \otimes B} \qquad \text{Par}\ \frac{\vdash \Theta; \Gamma, t{:}A, u{:}B}{\vdash \Theta; \Gamma, t \parr u{:}A \parr B}$$

$$\text{With}\ \frac{\vdash \Theta; \bar{t}{:}\Gamma, t{:}A \quad \vdash \Xi; \bar{u}{:}\Gamma, u{:}B}{\vdash ; \bar{x}{:}\Gamma, \bar{x}(\Theta; \bar{t}, t \,[]\, \Xi; \bar{u}, u){:}A \,\&\, B}$$

$$\text{Plus (i)}\ \frac{\vdash \Theta; \Gamma, t{:}A}{\vdash \Theta; \Gamma, \mathsf{inl}(t){:}A \oplus B} \qquad \text{Plus (ii)}\ \frac{\vdash \Theta; \Gamma, u{:}B}{\vdash \Theta; \Gamma, \mathsf{inr}(u){:}A \oplus B}$$

CLL Computational Interpretation

$$\text{Dereliction}\ \frac{\vdash \Theta;\Gamma, t:A}{\vdash \Theta;\Gamma, ?t:?A} \qquad \text{Weakening}\ \frac{\vdash \Theta;\Gamma}{\vdash \Theta;\Gamma, \_:?A}$$

$$\text{Contraction}\ \frac{\vdash \Theta;\Gamma, t:?A, u:?A}{\vdash \Theta;\Gamma, t\,@\,u:?A}$$

$$\text{Of Course}\ \frac{\vdash \Theta;\bar{t}:?\Gamma, t:A}{\vdash ;\bar{x}:?\Gamma, \bar{x}(\Theta;\bar{t}, t):!A}$$

$$\text{All}\ \frac{\vdash \Theta;\Gamma, t:A}{\vdash \Theta;\Gamma, t:\forall\alpha.A}\ (*) \qquad \text{Exists}\ \frac{\vdash \Theta;\Gamma, t:A[B/\alpha]}{\vdash \Theta;\Gamma, t:\exists\alpha.A}$$

CLL Computational Interpretation

6.2. Operational semantics: the linear CHAM

We now complete our computational interpretation of classical linear logic by giving an operational semantics for proof expressions. Rather than directly defining the relation of evaluation to canonical form, we shall define a one-step transition relation on proof expressions, and define canonical forms as certain *normal forms* with respect to this relation. This is because the notion of computation for proof expressions is inherently *parallel*; the model is that the coequations form a pool of concurrent processes. In fact, our presentation of the operational semantics fits very nicely into the framework of the chemical abstract machine proposed recently by Berry and Boudol [5] as a paradigm for concurrent abstract machines. They describe the basic ideas thus:

> Most available concurrency models are based on architectural concepts, e.g. networks of processes communicating by means of ports or channels. Such concepts convey a rigid geometrical vision of concurrency. Our chemical abstract machine model is based on a radically different paradigm ... where the concurrent components are freely "moving" in the system and communicate when they come into contact. ...

CLL Computational Interpretation

Structural rules

There are two basic structural rules:

- $t \perp u \leftrightharpoons u \perp t$
- $t \perp u, t' \perp u' \leftrightharpoons t' \perp u', t \perp u$

The first says that each coequation can be regarded as a multiset of exactly two terms, the second that lists of coequations can be regarded as multisets.

These rules can be applied in any context:

$$\frac{\Theta \leftrightharpoons \Xi}{C[\Theta] \leftrightharpoons C[\Xi]}$$

The basic metarule for the CHAM refers to the transition relation $\longrightarrow$ to be defined below.

Magical mixing rule:

$$\frac{P \leftrightharpoons^* P' \quad P' \longrightarrow Q' \quad Q' \leftrightharpoons^* Q}{P \longrightarrow Q}$$

Reaction context rule:

$$\frac{\Theta \longrightarrow \Xi}{\Theta_1, \Theta, \Theta_2; \bar{t} \longrightarrow \Theta_1, \Xi, \Theta_2; \bar{t}}$$

CLL Computational Interpretation

Reaction rules

These rules describe how lists of adjacent coequations react, giving rise to new lists.

Notation. Given $\bar{x} = x_1, \ldots, x_k, \bar{t} = t_1, \ldots, t_k$, we write $\bar{x} \perp \bar{t}$ to denote the list $x_1 \perp t_1, \ldots, x_k \perp t_k$.

Communication:

$$t \perp x, x \perp u \longrightarrow t \perp u$$

Unit:

$$* \perp \circledast \longrightarrow$$

Pair:

$$t \otimes u \perp t' \parr u' \longrightarrow t \perp t', u \perp u'$$

Case Left:

$$\bar{x}(\Theta; \bar{t}, t \,[]\, \Xi; \bar{u}, u) \perp \mathsf{inl}(v) \longrightarrow \Theta, \bar{x} \perp \bar{t}, t \perp v$$

Case Right:

$$\bar{x}(\Theta; \bar{t}, t \,[]\, \Xi; \bar{u}, u) \perp \mathsf{inr}(v) \longrightarrow \Xi, \bar{x} \perp \bar{u}, u \perp v$$

Reaction context rule:

$$\frac{\Theta \longrightarrow \Xi}{\Theta_1, \Theta, \Theta_2; \bar{t} \longrightarrow \Theta_1, \Xi, \Theta_2; \bar{t}}$$

CLL Computational Interpretation

From the computational aspect, the most interesting rules are those for the additives and exponentials. In both cases we have lazy types – & and ! – which in the concurrent framework must be implemented by some form of explicit *synchronization*. This is the role of the forms $\bar{x}(P \,[]\, Q)$ and $\bar{x}(P)$. In both cases, proof expressions are suspended from execution, and only resumed when sufficient information is available (or, in more computational terms, when sufficient demand has been generated). In the case of the additives, the With rule (which under the classical dualities is equivalent to the intuitionistic rule ($\oplus$ L)) corresponds to a case statement, i.e. a choice between two alternatives. Clearly, we only want to evaluate that expression corresponding to the alternative actually chosen; so we must *wait* until the choice is made. This is done when the term $\bar{x}(P \,[]\, Q)$ is cut against a term denoting a proof of the dual $(A \,\&\, B)^{\perp} = A^{\perp} \oplus B^{\perp}$, of the form inl($t$), where t is a proof of $A^{\perp}$, or inr(u), where u is a proof of $B^{\perp}$; hence the Case Left and Case Right rules. So we must defer any evaluation of the proofs of the side formulas Γ of the With rule until this choice is made. (Indeed, we don't even know till then whether these proofs should be taken as

CLL Computational Interpretation

Read:

$$\bar{x}(\Theta;\bar{t}, t) \perp ?u \longrightarrow \Theta, \bar{x} \perp \bar{t}, t \perp u$$

Discard:

$$\bar{x}(P) \perp \_ \longrightarrow x_1 \perp \_, \ldots, x_k \perp \_$$

Copy:

$$\bar{x}(P) \perp u \mathbin{@} v \longrightarrow \bar{x} \perp (\bar{x}^l \mathbin{@} \bar{x}^r), \bar{x}(P)^l \perp u, \bar{x}(P)^r \perp v$$

Similar considerations apply to the rules for the exponentials. The idea here is that the term of type $?A^{\perp}$ specifies how many copies of the term of type $!A$ are required; each of the terms for the side formulas $?\Gamma$ of the Of Course rule which generated the $!A$ term $\bar{x}(P)$ must then be directed to ask for a corresponding multiple of copies from its "input".

CLL Computational Interpretation

Notational interlude: *variants*

We shall need to consider *variants* of terms t occurring in a proof expression P, i.e. copies of t in which all names have been replaced by "fresh" names not already occurring in P. In order to implement this global condition in a local way, we need a little extra structure. We fix a bijection $\mathcal{N} \cong N_0 \times \{l, r\}^*$, and extend the name convention so that when a name $x \leftrightarrow \langle x_0, s \rangle$ is introduced in a proof expression, the x_0 component is distinct from that of any name already occurring in the expression. Now given a term t, we define t^l, t^r to be the result of replacing each occurrence of a name $x \leftrightarrow \langle x_0, s \rangle$ in t by $y \leftrightarrow \langle x_0, sl \rangle$, $z \leftrightarrow \langle x_0, sr \rangle$, respectively. The idea is that the following invariant is established by the proof expression assignment and maintained by the transition relation to be defined below:

For all distinct names $x \leftrightarrow \langle x_0, s \rangle$, $y \leftrightarrow \langle y_0, t \rangle$ occurring in P, $x_0 = y_0$ implies that s is incompatible with t (i.e. they have no upper bound in the prefix ordering).

CLL Computational Interpretation

Cleanup rule

Finally, we have a rule which tidies up a computation by consolidating information back into the main body $\bar{t}$ of a proof expression $\Theta;\bar{t}$. This is somewhat analogous to collecting the answer substitution from a PROLOG computation.

Cleanup:

$$x \perp t, \Theta;\bar{t} \longrightarrow \Theta;\bar{t}[t/x] \quad (x \in A\mathcal{N}(\bar{t})).$$

We can now define the *result* of a computation. A proof expression $P = \Theta;\bar{t}$ is *canonical* if it is a $\longrightarrow$-normal form, and each coequation in Θ has the form $x \perp t$ or $t \perp x$ for some name x. P is *cut-free* if Θ is empty.

Definition 6.1. We define $P \Downarrow Q$ – *P evaluates to canonical form Q* – by:

$$P \Downarrow Q \overset{\text{def}}{\Leftrightarrow} P \longrightarrow^* Q,\ Q \text{ canonical.}$$

Graded Modal Types & Coeffects

Quantitative Program Reasoning with Graded Modal Types

DOMINIC ORCHARD, University of Kent, UK
VILEM-BENJAMIN LIEPELT, University of Kent, UK
HARLEY EADES III, Augusta University, USA

In programming, some data acts as a resource (e.g., file handles, channels) subject to usage constraints. This poses a challenge to software correctness as most languages are agnostic to constraints on data. The approach of linear types provides a partial remedy, delineating data into resources to be used but never copied or discarded, and unconstrained values. Bounded Linear Logic provides a more fine-grained approach, quantifying non-linear use via an indexed-family of modalities. Recent work on *coeffect types* generalises this idea to *graded comonads*, providing type systems which can capture various program properties. Here, we propose the umbrella notion of *graded modal types*, encompassing coeffect types and dual notions of type-based effect reasoning via *graded monads*. In combination with linear and indexed types, we show that graded modal types provide an expressive type theory for quantitative program reasoning, advancing the reach of type systems to capture and verify a broader set of program properties. We demonstrate this approach via a type system embodied in a fully-fledged functional language called Granule, exploring various examples.

CCS Concepts: • **Theory of computation → Modal and temporal logics**; **Program specifications**; **Program verification**; *Linear logic*; *Type theory*.

Additional Key Words and Phrases: graded modal types, linear types, coeffects, implementation

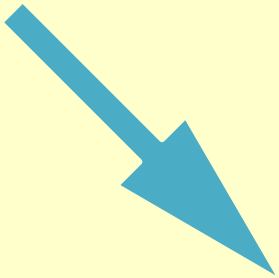
ACM Reference Format:
Dominic Orchard, Vilem-Benjamin Liepelt, and Harley Eades III. 2019. Quantitative Program Reasoning with Graded Modal Types. *Proc. ACM Program. Lang.* 3, ICFP, Article 110 (August 2019), 30 pages. https://doi.org/10.1145/3341714

Bounded Linear Logic (BLL)

$$(!\mathrm{R})\ \frac{!\Gamma \vdash A}{!\Gamma \vdash !A} \qquad (\text{Dereliction})\ \frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B}$$

($!\Gamma$ means a sequence of the form $!A_1, \ldots, !A_k$.)

$$(\text{Contraction})\ \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B} \qquad (\text{Weakening})\ \frac{\Gamma \vdash B}{\Gamma, !A \vdash B}$$



in Sections 3 and 4. If Γ is $A_1, \ldots, A_n$ we write $!_{\vec{y}}\Gamma$ for $!_{y_1}A_1, \ldots, !_{y_n}A_n$.

The rules for storage naturally induce polynomials:

$$\textit{Storage}\ \frac{!_{\vec{y}}\Gamma \vdash A}{!_{x\vec{y}}\Gamma \vdash !_x A} \qquad \textit{Weakening}\ \frac{\Gamma \vdash B}{\Gamma, !_0 A \vdash B}$$

$$\textit{Contraction}\ \frac{\Gamma, !_x A, !_y A \vdash B}{\Gamma, !_{x+y} A \vdash B} \qquad \textit{Dereliction}\ \frac{\Gamma, A \vdash B}{\Gamma, !_1 A \vdash B}.$$

We may interpret these rules in second-order RLL, by translating $!_x A$ as

$$\underbrace{1 \otimes A \otimes \cdots \otimes A}_{x\ \otimes\text{'s}}$$

Graded Modal Types: GrMini

Types and terms of GrMini are those of the linear λ-calculus with two additional pieces of syntax for introducing and eliminating values of the graded necessity type $\Box_r A$:

$$t ::= x \mid t_1\ t_2 \mid \lambda x.t \mid [t] \mid \mathbf{let}\,[x] = t_1\ \mathbf{in}\ t_2 \qquad A, B ::= A \multimap B \mid \Box_r A \qquad (\textit{terms and types})$$

The usual syntax of the λ-calculus, with variables x, is extended with the term-former $[t]$ which promotes a term to a graded modality, typed by $\Box_r A$, as shall be seen in the typing rules. The term $\mathbf{let}\,[x] = t_1\ \mathbf{in}\ t_2$ dually provides elimination for graded modal types. The graded modality $\Box_r A$ is an indexed family of type constructors whose indices r range over the elements of a *resource algebra*—in this case, a semiring $(\mathcal{R}, +, 0, \cdot, 1)$—whose operations echo the structure of the proof/typing rules. This semiring parameterises GrMini as a meta-level entity. In contrast, Gr can internally

Graded Modal Types: GrMini

Typing judgments are of the form $\Gamma \vdash t : A$ with typing contexts Γ of the form:

$$\Gamma ::= \emptyset \mid \Gamma, x : A \mid \Gamma, x : [A]_r \qquad (\textit{contexts})$$

Contexts are either empty $\emptyset$, or can be extended with a linear variable assumption $x : A$ or a *graded assumption* $x : [A]_r$. For a graded assumption, x can behave non-linearly, with substructural behaviour captured by the semiring element r, which describes x's use in a term. We will denote the domain of a context Γ, the set of variables assigned a type in the context, by $|\Gamma|$.

Typing for the linear λ-calculus fragment is then given by the rules:

$$\frac{}{x : A \vdash x : A}\ \textsc{var} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B}\ \textsc{abs} \qquad \frac{\Gamma_1 \vdash t_1 : A \multimap B \quad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + \Gamma_2 \vdash t_1\ t_2 : B}\ \textsc{app} \qquad \frac{\Gamma \vdash t : A}{\Gamma, [\Delta]_0 \vdash t : A}\ \textsc{weak}$$

Definition 3.1. [Context concatenation] Two contexts can be concatenated if they contain disjoint sets of linear assumptions. Furthermore, graded assumptions appearing in both contexts are combined using the additive operation of the semiring $+$. Concatenation $+$ is specified as follows:

$$(\Gamma, x : A) + \Gamma' = (\Gamma + \Gamma'), x : A \quad \text{iff } x \notin |\Gamma'| \qquad \emptyset + \Gamma = \Gamma$$
$$\Gamma + (\Gamma', x : A) = (\Gamma + \Gamma'), x : A \quad \text{iff } x \notin |\Gamma| \qquad \Gamma + \emptyset = \Gamma$$
$$(\Gamma, x : [A]_r) + (\Gamma', x : [A]_s) = (\Gamma + \Gamma'), x : [A]_{(r+s)}$$

Graded Modal Types: GrMini

The next three rules employ the remaining semiring structure, typing the additional syntax as well as connecting linear assumptions to graded assumptions:

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A]_1 \vdash t : B}\ \textsc{der} \qquad \frac{[\Gamma] \vdash t : A}{r \cdot [\Gamma] \vdash [t] : \Box_r A}\ \textsc{pr} \qquad \frac{\Gamma_1 \vdash t_1 : \Box_r A \quad \Gamma_2, x : [A]_r \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \mathbf{let}\,[x] = t_1\,\mathbf{in}\,t_2 : B}\ \textsc{let}$$

Dereliction (DER) converts a linear assumption to be graded, marked with 1. Subsequently, the semiring element 1 relates to linearity, though in GR (§4) it does not exactly denote linear use as $x : [A]_1 \vdash t : B$ does not imply $x : A \vdash t : B$ for all semirings once ordering is added to allow approximation. *Promotion* (PR) introduces graded necessity with grade r, propagating this grade to the assumptions via scalar multiplication of the context by r. For tracking number of uses, the rule states that to produce the *capability* to reuse t of type A exactly r times requires that all the input requirements for t are provided r times over, hence we multiply the context by r.

Definition 3.2. [Scalar context multiplication] Assuming that a context contains only graded assumptions, denoted $[\Gamma]$ in typing rules, then Γ can be multiplied by a semiring element $r \in R$:

$$r \cdot \emptyset = \emptyset \qquad\qquad r \cdot (\Gamma, x : [A]_s) = (r \cdot \Gamma), x : [A]_{(r \cdot s)}$$

Graded Modal Types: Granule

4.1 Syntax

A core subset of the surface-level syntax for Granule is given by the following grammar:

$$t ::= \underbrace{x \mid t_1\ t_2 \mid \lambda p.t}_{\lambda\text{-calculus}} \mid \underbrace{[t]}_{\text{box}} \mid \underbrace{n \mid C\ t_0 \ldots t_n}_{\text{constructors}} \mid \underbrace{\mathbf{let}\ \langle p \rangle \leftarrow t_1\ \mathbf{in}\ t_2 \mid \langle t \rangle}_{\text{monadic metalanguage}} \qquad (\textit{terms})$$

$$p ::= \underbrace{x}_{\text{variables}} \mid \underbrace{\_}_{\text{wildcard}} \mid \underbrace{[p]}_{\text{unbox}} \mid \underbrace{n \mid C\ p_0 \ldots p_n}_{\text{constructors}} \qquad (\textit{patterns})$$

The "boxing" (promotion) construct $[t]$ is dualised by the unboxing pattern $[p]$, replacing the specialised let-binding syntax of GrMini, which is now syntactic sugar:

$$\mathbf{let}\,[p] = t_1\ \mathbf{in}\ t_2 \triangleq (\lambda[p].t_2)\ t_1 \qquad (\textit{syntactic sugar})$$

The syntax of GrMini types is extended and a syntactic category of kinds is also now included:

$$A, B, R, E ::= A \to B \mid K \mid \alpha \mid A\ B \mid A\ \mathrm{op}\ B \mid \Box_c A \mid \Diamond_\varepsilon A \qquad (\textit{types})$$

$$\kappa ::= \mathsf{Type} \mid \mathsf{Coeffect} \mid \mathsf{Effect} \mid \mathsf{Predicate} \mid \kappa_1 \to \kappa_2 \mid \uparrow A \qquad (\textit{kinds})$$

$$\mathrm{op} ::= + \mid * \mid - \mid \leq \mid < \mid \geq \mid > \mid = \mid \neq \mid \sqcup \mid \sqcap \qquad (\textit{type operators})$$

$$K ::= \mathsf{Int} \mid \mathsf{Char} \mid () \mid \times \mid \mathsf{IO} \mid \mathsf{Nat} \mid \mathsf{Level} \mid \mathsf{Ext} \mid \mathsf{Interval} \qquad (\textit{type constructors})$$

Graded Modal Types: Granule

GrMini was parameterised at the meta-level by a semiring, providing a system with one graded necessity modality. Granule instead allows various graded modalities, with different index domains, to be used simultaneously within the same program. The type $\Box_c A$ captures different graded necessity modalities identified by the type of the grade c which is an element of a *resource algebra*: a pre-ordered semiring $(\mathcal{R}, +, 0, \cdot, 1, \sqsubseteq)$ with monotonic multiplication and addition which may be partial. We colour in blue general resource algebra operations, and necessity grades in typing rules.

Within types, necessity grade terms c (which we call *coeffects*) have the following syntax:

$$c ::= \alpha \mid c_1 + c_2 \mid c_1 \cdot c_2 \mid 0 \mid 1 \mid c_1 \sqcup c_2 \mid c_1 \sqcap c_2 \mid \mathsf{flatten}(c_1, R, c_2, S) \qquad (\textit{coeffects})$$
$$\mid n \mid \mathsf{Private} \mid \mathsf{Public} \mid c_1..c_2 \mid \infty \mid (c_1, c_2)$$

Definition 4.1. [Exact usage] The coeffect type Nat has the resource algebra given by the usual natural numbers semiring $(\mathbb{N}, +, 0, \cdot, 1, \equiv)$, but notably with *discrete ordering* $\equiv$ giving exact usage analysis in Granule (see §2). Thus, meet and join are only defined on matching inputs.

Graded Modal Types: Granule

Definition 4.2. [Security levels] The coeffect type Level provides a way of capturing confidentiality requirements and enforcing noninterference, with a three-point lattice of security levels {Irrelevant $\sqsubseteq$ Private $\sqsubseteq$ Public} with 0 = Irrelevant, 1 = Private, $+$ = $\sqcup$ (join of the induced lattice), and if r = Irrelevant or s = Irrelevant then $r \cdot s$ = Irrelevant otherwise $r \cdot s = r \sqcup s$.

Multiplication $\cdot$ is such that if a value is used publicly, all of its dependencies must also be public; a private value can depend on public and private values. Recall that $+$ represents contraction (i.e., a split in the dataflow of a value). Therefore, a dependency used publicly must be permitted for public use even if it used elsewhere privately, thus Public+Private = Public. Since 0 = Irrelevant (bottom element), we can weaken at any level by approximation. Similarly, since 1 = Private, we can essentially derelict at either Private or Public by approximation. The appendix (Lemma A.1) gives the proof that this resource algebra is a preordered semiring.

Graded Modal Types: Granule

Definition 4.3. [Intervals] The Interval constructor is unary, of kind Coeffect $\rightarrow$ Coeffect, where Interval R is inhabited by pairs of R elements, giving lower and upper bounds. Thus, Interval R is the semiring over $\{c..d \mid c \in R \wedge d \in R \wedge c \sqsubseteq_R d\}$, i.e., pairs written with the Granule syntax $c..d$, where the first component is less than the second (according to the preorder on R). Units are $0 = 0_R..0_R$ and $1 = 1_R..1_R$ and the operations and pre-order are defined as in interval arithmetic:

$$c_l..c_u + d_l..d_u = (c_l +_R d_l)..(c_u +_R d_u)$$

$$c_l..c_u \cdot d_l..d_u = (c_l \cdot d_l \sqcap_R c_l \cdot d_u \sqcap_R c_u \cdot d_l \sqcap_R c_u \cdot d_u)..(c_l \cdot d_l \sqcup_R c_l \cdot d_u \sqcup_R c_u \cdot d_l \sqcup_R c_u \cdot d_u)$$

$$c_l..c_u \sqsubseteq d_l..d_u = (d_l \sqsubseteq_R c_l) \wedge (c_u \sqsubseteq_R d_u)$$

For Interval Nat (used in Section 2 to capture lower and upper bounds on usage), multiplication simplifies to $c_l..c_u \cdot d_l..d_u = (c_l \cdot d_l)..(c_u \cdot d_u)$. Whilst Nat is discrete, the implementation uses the $\leq$ natural number ordering to form the Interval Nat resource algebra so that it can properly capture lower and upper bounds on use.

Graded Modal Types: Granule

Definition 4.4. [Extended coeffects] For a resource algebra R, applying the unary constructor Ext R extends the resource algebra with an element ∞ (i.e., Ext $R = R \cup \{\infty\}$) with operations:

$$r + s = \begin{cases} \infty & (r = \infty) \vee (s = \infty) \\ r +_R s & \textit{otherwise} \end{cases} \qquad r \cdot s = \begin{cases} 0_R & (r = 0_R) \vee (s = 0_R) \\ \infty & ((r = \infty) \wedge (s \neq 0_R)) \vee ((s = \infty) \wedge (r \neq 0_R)) \\ r \cdot_R s & \textit{otherwise} \end{cases}$$

The pre-order for Ext R is that of R, but with $r \sqsubseteq \infty$ for all r. Some Section 2 examples used coeffects of kind Interval (Ext Nat), where $0..\infty$ captures arbitrary ("Cartesian" usage), providing a type analysis akin to the ! modality of linear logic. In Granule, the type "`A []`" is an alias for "`A [0..∞]`".

Definition 4.5. [Products] Given two resource algebras R and S, we can form a product resource algebra $R \times S$ whose operations are the pairwise application of the operations for R and S, e.g., $(r, s)+(r', s') = (r +_R r', s +_S s')$. This is useful for composing grades together to capture multiple properties at once. We treat products as commutative and associative.

Graded Modal Types: Granule

Finally, an inter-resource algebra operation "flatten" describes how to sequentially compose two levels of grading, which occurs when we have nested pattern matching on nested graded modalities—a novel feature. Consider the following example, which takes a value inside two layers of graded modalities, pattern matches on both simultaneously, and then uses the value:

```
unpack : (Int [2]) [3] → Int
unpack [[x]] = x + x + x + x + x + x
```

Here, double unboxing computes the multiplication of the two grades, capturing that x is used six times. What if we have two different graded modalities (i.e., graded by different coeffect types)? The flatten operation is used here, taking two coeffect terms and their types (i.e., $r : R$ and $r' : R'$), computing a coeffect term describing composition of r and r', resolved to a particular (possibly different) coeffect type. If $\mathsf{flatten}(r, R, r', R') = s : S$ then we can type the following:

$$\lambda[[x]].[x] \ : \ \forall\{\alpha : \mathsf{Type}, r : \uparrow R, r' : \uparrow R', s : \uparrow S\} \ . \ \Box_{r'}(\Box_{r}\alpha) \to \Box_{s}\alpha$$

Currently, flatten is defined in Granule as follows (but can be easily extended at a later date):

Definition 4.6. For the built-in resource algebras, flatten is the symmetric congruence closure of:

$$\begin{array}{ll}
\mathsf{flatten}(r, \mathsf{Ext\ Nat}, s, \mathsf{Ext\ Nat}) = r \cdot s : \mathsf{Ext\ Nat} & \mathsf{flatten}(r, \mathsf{Nat}, s, \mathsf{Ext\ Nat}) = r \cdot s : \mathsf{Ext\ Nat} \\
\mathsf{flatten}(r, R, r_1..r_2, \mathsf{Interval}\ R) = (r \cdot r_1)..(r \cdot r_2) : \mathsf{Interval}\ R & \mathsf{flatten}(r, \mathsf{Nat}, s, \mathsf{Nat}) \quad = r \cdot s : \mathsf{Nat} \\
\mathsf{flatten}(r, R, (r_1, s_1), R \times S) \quad = (r \cdot r_1, s_1) : R \times S & \mathsf{flatten}(r, \mathsf{Level}, s, \mathsf{Level}) \ = r \sqcap s : \mathsf{Level} \\
\mathsf{flatten}(s, S, (r_1, s_1), R \times S) \quad = (r_1, s \cdot s_1) : R \times S & \mathsf{flatten}(r, R, s, S) \ \mid R \neq S = (r, s) : R \times S
\end{array}$$

Thus for Nat we flatten using multiplication, and similarly when combining Nat with an Ext Nat (resolving to the larger type Ext Nat). For levels, we take the meet, i.e., $\Box_{\mathsf{Public}}(\Box_{\mathsf{Private}}\alpha)$ is flattened to $\Box_{\mathsf{Private}}\alpha$, avoiding leakage. For two different resource algebras, flatten forms a product, giving a composite analysis. Note, flatten is a homomorphism with respect to the resource algebra operations. The next section shows how flatten is used in typing.