A taste of linear logic*

Philip Wadler

Department of Computing Science, University of Glasgow, G12 8QQ, Scotland
(wadler@dcs.glasgow.ac.uk)

Abstract. This tutorial paper provides an introduction to intuitionistic logic and linear logic, and shows how they correspond to type systems for functional languages via the notion of 'Propositions as Types'. The presentation of linear logic is simplified by basing it on the Logic of Unity. An application to the array update problem is briefly discussed.

1 Introduction

Some of the best things in life are free; and some are not.

Truth is free. Having proved a theorem, you may use this proof as many times as you wish, at no extra cost. Food, on the other hand, has a cost. Having baked a cake, you may eat it only once. If traditional logic is about truth, then linear logic is about food.

In traditional logic, if a fact is used to conclude another fact, the first fact is still available. For instance, given that A implies B and given A, one may deduce both A and B. In symbols, this is written as the judgement

$$A \to B,\ A \vdash A \times B \tag{i}$$

where $A \to B$ is read 'A implies B', and $A \times B$ is read 'A and B'. (It will shortly become apparent why $\times$ is written for 'and'.) The assumption A is used twice in the proof of this judgement, and that is reasonable because truth is free.

Traditional logic comes in many forms: the one we shall concentrate on is *intuitionistic* logic. This logic is of special interest because its terms are in one-to-one correspondence with a typed functional programming language. This correspondence goes by the name 'Curry-Howard isomorphism', after the logicians who discovered it, or by 'Propositions as types', because it views propositions of logic as types in a functional program. Under this correspondence, one proof of judgement (i) may be written as the program

$$f : A \to B,\ x : A \vdash (x, f\,(x)) : A \times B \tag{ii}$$

where f is a free variable denoting a function of type $A \to B$, and x is a free variable denoting a value of type A, and $(x, f\,(x))$ is a term denoting a pair of

* This is a slightly revised version of an invited lecture delivered at *Mathematical Foundations of Computer Science*, Gdansk, August-September 1993, Springer Verlag Lecture Notes in Computer Science 711.

type $A \times B$. The first component of the pair is the value x, the second component is the result of applying function f to value x. The free variable x appears twice in the term, just as assumption A is used twice in the proof of the corresponding judgement. This is reasonable if the value in x costs little to copy, so the name 'free' variable is doubly appropriate.

But not all things in life are free. To capture this notion, the logician Jean-Yves Girard devised linear logic, a 'resource conscious' logic. In linear logic, assumptions may not be freely copied, nor may they be foolishly discarded: each assumption is used exactly once. It is no longer the case that given A implies B and given A one can deduce both A and B. In symbols, this is written as the non-judgement

(iii) $$\langle A \multimap B \rangle, \langle A \rangle \not\vdash A \otimes B$$

where $A \multimap B$ is read 'consuming A yields B' and $A \otimes B$ is read 'both A and B', and angle brackets are written around each assumption to indicate that it must be used exactly once. Taking A to be the proposition 'I have a cake', and B to be the proposition 'I am full', we discover that one cannot have one's cake and eat it too.

Just as lambda calculus corresponds to intuitionistic logic, there is a programming calculus that correspond to linear logic. Corresponding to (iii) is the program

(iv) $$\langle f : A \multimap B \rangle, \langle x : A \rangle \not\vdash \langle x, f\,\langle x \rangle \rangle : A \otimes B.$$

If f was a function that takes a cake into a feeling of fullness, and x was a cake, then $\langle x, f\,\langle x \rangle \rangle$ is a pair consisting of the original cake and the fullness resulting from eating it. Just as judgement (iii) is invalid, program (iv) is ill-typed: the program that keeps a cake and eats it too has a type error.

Such a type system has less fanciful uses. It can allow for fine control over the efficient use of storage, and in particular for a new answer to the old question of how to handle arrays efficiently in a functional language.

This paper is intended as a general introduction to intutionistic logic and the Curry-Howard isomorphism, and to linear logic and some of its applications in functional programming. The presentation of linear logic is simplified by basing it on Girard's Logic of Unity, a refinement of the concept of linear logic.

The organisation follows the development above. Section 1 reviews intuitionistic logic. Section 2 describes how the Curry-Howard correspondence relates this logic to lambda calculus. Section 3 introduces linear logic. Section 4 uses the Curry-Howard correspondence to derive a linear lambda calculus, and outlines its applications. Section 5 discusses related work and concludes.

It is worth noting what is *not* covered. First, the paper concentrates on the application of linear logic to functional programming, and ignores other intriguing applications such as logic programming or concurrent programming. Second, the paper describes only *intuitionistic* linear logic, and ignores the related theory of *classical* linear logic. However, it is hoped that this paper will put the reader in a better position to appreciate the literature on the subject.

As this paper is mainly tutorial in nature, it is to a large extent a gloss on the work of others. It is a pleasure to acknowledge here some of the people who have

taught me these ideas: Jean-Yves Girard, Samson Abramsky, Martin Hyland, Yves Lafont, Gordon Plotkin, Vaughn Pratt, and Robert Seely.

2 Intuitionistic logic

Most computer scientists have seen one or another of the many ways of formulating logic. The version presented here is based on natural deduction, and specially emphasises the Contraction and Weakening rules, which are the key to understanding linear logic.

A *proposition* is built up from propositional constants using the combining forms *implies*, written $\rightarrow$, *and*, written $\times$, and *or*, written $+$. Let A, B, C range over propositions, and X range over propositional constants. Then the grammar of propositions is as follows.

$$A, B, C ::= X \mid A \rightarrow B \mid A \times B \mid A + B$$

An *assumption* is a sequence of zero or more propositions. Let Γ, Δ, Θ range over assumptions. A *judgement* has the form $\Gamma \vdash A$, meaning that from assumptions Γ one can conclude proposition A.

A *rule* consists of zero or more judgements written above a line, and one judgement written below. If all the judgements above the line are derivable, then the judgement below is derivable also. Note that there are three different levels of implication: $\rightarrow$ in a proposition, $\vdash$ in a judgement, and the line in a rule.

$$\frac{}{A \vdash A}\,\text{Id} \qquad \frac{\Gamma,\ \Delta \vdash A}{\Delta,\ \Gamma \vdash A}\,\text{Exchange}$$

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

$$\frac{\Gamma,\ A \vdash B}{\Gamma \vdash A \rightarrow B}\,{\rightarrow}\text{-I} \qquad \frac{\Gamma \vdash A \rightarrow B \qquad \Delta \vdash A}{\Gamma,\ \Delta \vdash B}\,{\rightarrow}\text{-E}$$

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma,\ \Delta \vdash A \times B}\,{\times}\text{-I} \qquad \frac{\Gamma \vdash A \times B \qquad \Delta,\ A,\ B \vdash C}{\Gamma,\ \Delta \vdash C}\,{\times}\text{-E}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A + B}\,{+}\text{-I}_1 \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A + B}\,{+}\text{-I}_2 \qquad \frac{\Gamma \vdash A + B \qquad \Delta,\ A \vdash C \qquad \Delta,\ B \vdash C}{\Gamma,\ \Delta \vdash C}\,{+}\text{-E}$$

Fig. 1. Intuitionistic logic

The rules for intuitionistic logic are shown in Figure 1. They come in three groups.

First is the lone *axiom*, Id, the only rule with no judgements above the line. This rule expresses tautology: from hypotheses A one can deduce conclusion A.

Second are the three *structural* rules, Exchange, Contraction, and Weakening. Exchange expresses that the order of hypotheses is irrelevant, Contraction expresses that any hypothesis may be duplicated, and Weakening expresses that any hypothesis may be discarded.

Third are the *logical* rules. These come in pairs. In the *introduction* rule $\to$-I a judgement ending in a proposition formed with $\to$ appears below the line, while in the *elimination* rule $\to$-E a judgment ending in a proposition formed with $\to$ appears above the line. Similarly for the other logical connectives, $\times$ and $+$.

Each rule has a straightforward logical reading. For instance, $\to$-I expresses the 'deduction theorem': if from assumptions Γ and A one can deduce B, then from assumption Γ alone one can deduce A implies B. Similarly, $\to$-E expresses 'modus ponens': if from assumptions Γ one can deduce that A implies B, and from assumptions Δ one can deduce A, then from assumptions Γ and Δ one can deduce B.

Derivations can be written as proof trees. The judgement derived is at the root, each branch represents a use of a rule, and the leaves are axioms. Here is a tree deriving the judgement $A \to B,\ A \vdash A \times B$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\text{Id} \qquad \dfrac{\dfrac{}{A \to B \vdash A \to B}\,\text{Id} \qquad \dfrac{}{A \vdash A}\,\text{Id}}{A \to B,\ A \vdash B}\,\to\text{-E}}{A,\ A \to B,\ A \vdash A \times B}\,\times\text{-I}}{A \to B,\ A,\ A \vdash A \times B}\,\text{Exchange}}{A \to B,\ A \vdash A \times B}\,\text{Contraction}$$

In this proof, the assumption A is used twice, once to conclude A itself, and once to conclude B. The double usage of A is justified by Contraction.

2.1 Alternative rules for conjunction

The rules for conjunction may also be expressed in an alternative form.

$$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \times B}\,\times\text{-I}' \qquad \dfrac{\Gamma \vdash A \times B}{\Gamma \vdash A}\,\times\text{-E}'_1 \qquad \dfrac{\Gamma \vdash A \times B}{\Gamma \vdash B}\,\times\text{-E}'_2$$

These are equivalent to the previous rules. We will see that the new rules may be derived from the old, plus Weakening and Contraction. Furthermore, the old rules may be derived from the new, plus Weakening, Contraction, $\to$-I, and $\to$-E.

The new $\times$-I$'$ is derived from the old $\times$-I together with Contraction.

$$\dfrac{\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma,\ \Gamma \vdash A \times B}\,\times\text{-I}}{\Gamma \vdash A \times B}\,\text{Contraction}$$

The double line stands for one use of Contraction for each assumption in Γ, and also hides some uses of Exchange to bring matching hypotheses together. The Exchange rule is so dull that its uses will not usually be mentioned.

The new $\times$-E$'_1$ is derived from the old $\times$-E plus Weakening.

$$\dfrac{\Gamma \vdash A \times B \qquad \dfrac{\dfrac{}{A \vdash A}\,\text{Id}}{A,\, B \vdash A}\,\text{Weakening}}{\Gamma \vdash A}\,\times\text{-E}$$

One might think that the Weakening rule is a little silly – why bother to make an extra assumption? – but this proof demonstrates its utility. The rule $\times$-E$'_2$ is derived similarly.

The other way around, $\times$-I can be derived from $\times$-I$'$ and Weakening.

$$\dfrac{\dfrac{\Gamma \vdash A}{\Gamma,\, \Delta \vdash A}\,\text{Weakening} \qquad \dfrac{\Delta \vdash B}{\Gamma,\, \Delta \vdash B}\,\text{Weakening}}{\Gamma,\, \Delta \vdash A \times B}\,\times\text{-I}'$$

The double lines stand for one use of Weakening for each assumption in Γ and Δ, and also hide some uses of Exchange.

Finally, $\times$-E can be derived from $\times$-E$'_1$ and $\times$-E$'_2$ togther with Contraction, $\rightarrow$-I, and $\rightarrow$-E.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Delta,\, A,\, B \vdash C}{\Delta,\, A \vdash B \to C}\,\to\text{-I}}{\Delta \vdash A \to B \to C}\,\to\text{-I} \qquad \dfrac{\Gamma \vdash A \times B}{\Gamma \vdash A}\,\times\text{-E}'_1}{\Delta,\, \Gamma \vdash B \to C}\,\to\text{-E} \qquad \dfrac{\Gamma \vdash A \times B}{\Gamma \vdash B}\,\times\text{-E}'_2}{\Delta,\, \Gamma,\, \Gamma \vdash C}\,\to\text{-E}}{\Gamma,\, \Delta \vdash C}\,\text{Contraction}$$

Note that the hypothesis $\Gamma \vdash A \times B$ is used *twice* in this proof, the multiple uses of Γ being reduced to a single use by Contraction.

The key difference of linear logic is that it restricts the use of Contraction and Weakening, so the two different rule sets are no longer equivalent. Instead of a single connective $\times$, there will be two connectives, $\otimes$ and &, one with rules analogous to $\times$-I and $\times$-E, and the other with rules analogous to $\times$-I$'$, $\times$-E$'_1$, and $\times$-E$'_2$.

2.2 An alternative to structural rules

The rules for Exchange, Weakening, and Contraction may appear unfamiliar to some readers, because logic is often presented in a different form that hides these

rules.

$$\frac{}{\Gamma,\, A,\, \Delta \vdash A}\,\mathrm{Id}'$$

$$\frac{\Gamma,\, A \vdash B}{\Gamma \vdash A \to B}\,{\to}\text{-I} \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\,{\to}\text{-E}'$$

The rule Id′ may be derived from Id, Weakening, and Exchange, and the rule $\to$-E′ may be derived from $\to$-E, Contraction, and Exchange. The system consisting of Id′, $\to$-I, and $\to$-E′ derives exactly the same judgements as the system consisting of Id, Exchange, Contraction, Weakening, $\to$-I, and $\to$-E. The other logical rules can be similarly modified, yielding an equivalent of the full system.

This system is convenient in that it eliminates all concern with the structural rules; but it has the disadvantage that it obscures the role of Contraction and Weakening.

2.3 Commuting conversions for structural rules

One lesson of the alternative system is that the order in which the structural rules are applied is irrelevant. This can be captured in our system by *commuting conversions*, which define an equivalence relation on proofs, written $\Longleftrightarrow$. Here are the commuting conversions for Contraction with the rule $\to$-E.

$$\frac{\dfrac{\Gamma,\, C,\, C \vdash A \to B}{\Gamma,\, C \vdash A \to B}\,\mathrm{Cont} \qquad \Delta \vdash A}{\Gamma,\, C,\, \Delta \vdash B}\,{\to}\text{-E} \;\Longleftrightarrow\; \frac{\dfrac{\Gamma,\, C,\, C \vdash A \to B \quad \Delta \vdash A}{\Gamma,\, C,\, C,\, \Delta \vdash B}\,{\to}\text{-E}}{\Gamma,\, C,\, \Delta \vdash B}\,\mathrm{Cont}$$

$$\frac{\Gamma \vdash A \to B \qquad \dfrac{\Delta,\, C,\, C \vdash A}{\Delta,\, C \vdash A}\,\mathrm{Cont}}{\Gamma,\, \Delta,\, C \vdash B}\,{\to}\text{-E} \;\Longleftrightarrow\; \frac{\dfrac{\Gamma \vdash A \to B \quad \Delta,\, C,\, C \vdash A}{\Gamma,\, \Delta,\, C,\, C \vdash B}\,{\to}\text{-E}}{\Gamma,\, \Delta,\, C \vdash B}\,\mathrm{Cont}$$

There are similar commuting conversions for Exchange, Contraction, and Weakening with each of the structural and logical rules.

As was already noted, a judgement is derivable in the system with explicit structural rules if and only if it is derivable in the system with structural rules 'built in'. With the commuting conversions, one can show not merely that the two systems derive the same judgements, but that proofs in the two systems are equivalent.

2.4 Proof reduction

A judgement may have several distinct derivations. Some of these may represent profoundly different proofs, while others may represent trivial variations. In particular, there is little point in having an introduction rule for a connective followed immediately by an elimination rule for the same connective.

Here is a sketch of a proof that introduces the proposition $A \to B$, only to immediately eliminate it.

$$\dfrac{\dfrac{\dfrac{\begin{array}{c} A \vdash A \ \cdots \\ \vdots u \\ \Gamma,\, A \cdots \vdash B \end{array}}{\Gamma,\, A \vdash B}}{\Gamma \vdash A \to B}\to\text{-I} \qquad \begin{array}{c} \vdots t \\ \Delta \vdash A \end{array}}{\Gamma,\, \Delta \vdash B}\to\text{-E}$$

Here t labels a subtree of the proof, ending in the judgement $\Delta \vdash A$, and u labels another subtree, ending in the judgement $\Gamma,\, A \vdash B$. Thanks to Weakening and Contraction, the assumption A may be used zero or more times in proof u, as indicated by ellipses. Each use corresponds to one appearance of the leaf $A \vdash A$, again as indicated by ellipses. The commuting conversions allow all uses of Weakening and Contraction on A in u to be moved to the end of the subtree, where they are indicated by the double line.

The unnecessary proposition $A \to B$ can be eliminated by replacing each occurrence of the leaf $A \vdash A$ in the proof u by a copy of the proof t of the judgement $\Gamma \vdash A$.

$$\dfrac{\begin{array}{c} \vdots t \\ \Delta \vdash A \cdots \\ \vdots u \\ \Gamma,\, \Delta \cdots \vdash B \end{array}}{\Gamma,\, \Delta \vdash B}$$

The Weakenings and Contractions previously performed on assumption A now need to be performed once for each assumption in Δ.

The replacement of the first proof by the second proof is called a *proof reduction*. If the assumption A is used more than once, then the proof t will be copied many times; so reduction does not always make a proof smaller. However, reduction is guaranteed to eliminate the unnecessary occurrence of $A \to B$ from the proof.

Similar proof reductions apply for the other connectives as well. Figure 2 shows the reduction rules for $\to$ and $\times$ and one of the two reduction rules for $+$ (the other is almost identical). Reductions form a partial order, denoted by $\Longrightarrow$. A proof to which no reductions apply is said to be in *normal form*. Remarkably, every proof has a normal form, and this normal form is unique modulo the commuting conversions.

A proof in normal form contains no extraneous propositions. More precisely, if a proof of a judgement $\Gamma \vdash A$ is in normal form, every proposition in the proof is a subformula of Γ or A. (The subformulas are defined in the obvious way; for instance, the subformulas of $A \to B$ are $A \to B$ itself plus the subformulas of A and B.)

For example, consider the judgement $A \to B,\, A \vdash B \times B$. One way to prove this is to use $\to$-I and Contraction to prove $\vdash B \to B \times B$, and then use $\to$-E

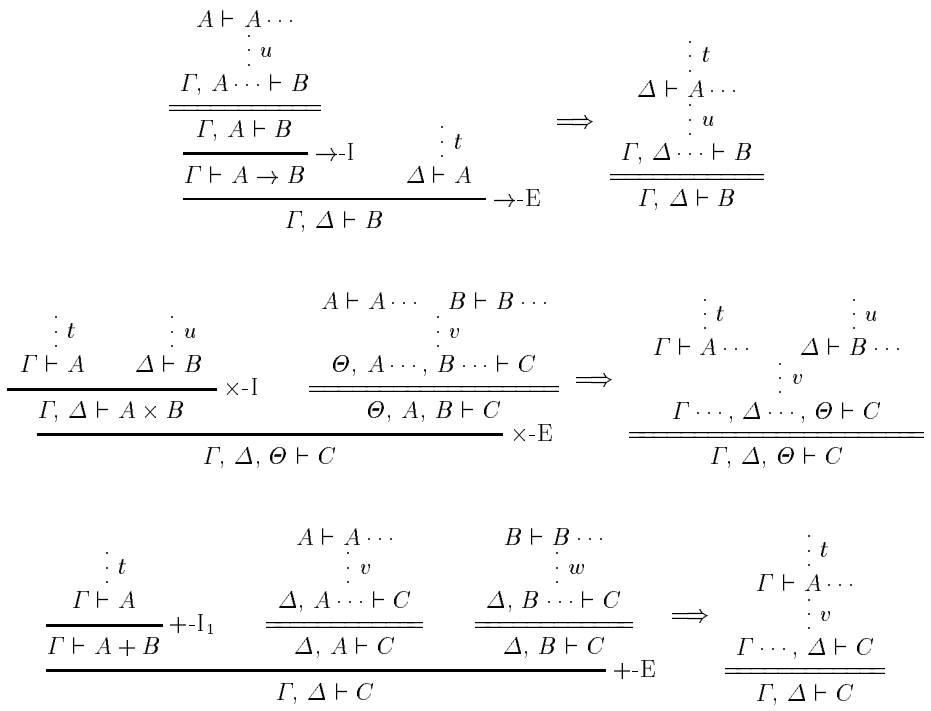

Fig. 2. Proof reduction for intuitionistic logic

to discharge the antecedent B with a proof of $A \to B,\ A \vdash B$. Here is the proof in full.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}\,\mathrm{Id} \qquad \dfrac{}{B \vdash B}\,\mathrm{Id}}{B,\, B \vdash B \times B}\,\times\text{-I}}{B \vdash B \times B}\,\mathrm{Contraction}}{\vdash B \to (B \times B)}\,\to\text{-I}\ (\dagger) \qquad \dfrac{\dfrac{}{A \to B \vdash A \to B}\,\mathrm{Id} \qquad \dfrac{}{A \vdash A}\,\mathrm{Id}}{A \to B,\, A \vdash B}\,\to\text{-E}}{A \to B,\, A \vdash B \times B}\,\to\text{-E}\ (\dagger)$$

This proof contains a proposition, $B \to B \times B$, that is not a subformula of the judgement proved. It also contains an introduction rule immediately followed by an elimination rule, each marked with (†). Applying the reduction rule for $\to$

results in the following proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \to B \vdash A \to B}\,\text{Id} \qquad \dfrac{}{A \vdash A}\,\text{Id}}{A \to B,\, A \vdash B}\,{\to}\text{-E} \qquad \dfrac{\dfrac{}{A \to B \vdash A \to B}\,\text{Id} \qquad \dfrac{}{A \vdash A}\,\text{Id}}{A \to B,\, A \vdash B}\,{\to}\text{-E}}{A \to B,\, A \to B,\, A \vdash B \times B}\,{\times}\text{-I}}{A \to B,\, A \vdash B \times B}\,\text{Contraction}$$

Contraction on B in the first proof has been replaced in the second by copying the proof of $A \to B,\ A \vdash B$ twice, and applying Contraction to the assumptions $A \to B$ and A. The second proof is in normal form, and does indeed satisfy the subformula property.

3 Intuitionistic terms

We now augment our judgements to contain terms. From the logicians' point of view, terms encode proofs. From the computer scientists' point of view, terms are a programming language for which the corresponding logic provides a type system.

In a constructive logic, propositions can be read as types, and proofs read as terms of the corresponding type. Implication is read as function space: a proof of $A \to B$ is a function that takes any proof of A into a proof of B. Conjunction is read as cartesian product: a proof of $A \times B$ is a pair consisting of a proof of A and a proof of B. Disjunction is read as disjoint sum: a proof of $A + B$ is either a proof of A or a proof of B. Henceforth *type* will be a synonym for proposition.

Terms encode the rules of the logic. There is one term form for the axiom Id, namely variables, and one term form for each of the logical rules. An introduction rule corresponds to a term that *constructs* a value of the corresponding type, while an elimination rule corresponds to a term that *deconstructs* a value of the corresponding type. There are no term forms for the structural rules, Exchange, Contraction, and Weakening. Let x, y, z range over variables, and s, t, u, v, w range over terms. The grammar of terms is as follows.

$$\begin{aligned} s, t, u, v, w ::= {} & x \\ \mid {} & \lambda x.\, u \mid s\,(t) \\ \mid {} & (t, u) \mid \text{case } s \text{ of } (x, y) \to v \\ \mid {} & \text{inl}\,(t) \mid \text{inr}\,(u) \mid \text{case } s \text{ of inl}\,(x) \to v;\ \text{inr}\,(y) \to w \end{aligned}$$

Assumptions are now written in the form

$$x_1 : A_1,\ \ldots,\ x_n : A_n$$

where $x_1, \ldots, x_n$ are distinct variables, $A_1, \ldots, A_n$ are types, and $n \geq 0$. As before, let Γ and Δ range over assumptions. Write Γ, Δ to denote the concatenation of assumptions; an implicit side condition of such a concatenation is that Γ and Δ contain distinct variables.

Judgements are now written in the form $\Gamma \vdash t : A$. One can think of Γ as a declaration specifying the types of the free variables of term t, which itself has type A.

Each rule is decorated with variables and terms, as shown in Figure 3.

$$\frac{}{x : A \vdash x : A}\ \text{Id} \qquad \frac{\Gamma,\ \Delta \vdash t : A}{\Delta,\ \Gamma \vdash t : A}\ \text{Exchange}$$

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

$$\frac{\Gamma,\ x : A \vdash u : B}{\Gamma \vdash \lambda x.\, u : A \to B}\ \to\text{-I} \qquad \frac{\Gamma \vdash s : A \to B \qquad \Delta \vdash t : A}{\Gamma,\ \Delta \vdash s\,(t) : B}\ \to\text{-E}$$

$$\frac{\Gamma \vdash t : A \qquad \Delta \vdash u : B}{\Gamma,\ \Delta \vdash (t, u) : A \times B}\ \times\text{-I} \qquad \frac{\Gamma \vdash s : A \times B \qquad \Delta,\ x : A,\ y : B \vdash v : C}{\Gamma,\ \Delta \vdash \text{case } s \text{ of } (x, y) \to v : C}\ \times\text{-E}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \text{inl}\,(t) : A + B}\ +\text{-I}_1 \qquad \frac{\Gamma \vdash u : B}{\Gamma \vdash \text{inr}\,(u) : A + B}\ +\text{-I}_2$$

$$\frac{\Gamma \vdash s : A + B \qquad \Delta,\ x : A \vdash v : C \qquad \Delta,\ y : B \vdash w : C}{\Gamma,\ \Delta \vdash \text{case } s \text{ of inl}\,(x) \to v;\ \text{inr}\,(y) \to w : C}\ +\text{-E}$$

Fig. 3. Intuitionistic types

If a concatenation of the form Γ, Δ appears in a rule, then Γ and Δ must contain distinct variables. Thus in the $\to$-I rule, the variable x may not appear in Γ, so there is no shadowing of variables.

A variable in the assumption can appear more than once in a term only if Contraction is used. In the Contraction rule, the notation $t[x/y, x/z]$ stands for term t with variable x replacing all occurrences of variables y and z. Similarly, a variable in the assumption must appear in the corresponding term unless Weakening is used.

Here is the proof of the judgement $A \to B,\ A \vdash A \times B$, decorated with terms.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{y : A \vdash y : A}\ \text{Id} \qquad \dfrac{\dfrac{}{f : A \to B \vdash f : A \to B}\ \text{Id} \qquad \dfrac{}{z : A \vdash z : A}\ \text{Id}}{f : A \to B,\ z : A \vdash f\,(z) : B}\ \to\text{-E}}{y : A,\ f : A \to B,\ z : A \vdash (y, f\,(z)) : A \times B}\ \times\text{-I}}{f : A \to B,\ y : A,\ z : A \vdash (y, f\,(z)) : A \times B}\ \text{Exchange}}{f : A \to B,\ x : A \vdash (x, f\,(x)) : A \times B}\ \text{Contraction}$$

The judgement at the root of this tree encodes the entire proof tree. In general, the judgement at the bottom of a labeled proof tree uniquely encodes the entire tree, modulo the commuting conversions for the structural rules.

3.1 Alternate terms for conjunction

The alternate rules for conjunction elimination correspond to alternate term forms for deconstructing pairs, $\mathrm{fst}\,(s)$ and $\mathrm{snd}\,(s)$.

$$\frac{\Gamma \vdash s : A \times B}{\Gamma \vdash \mathrm{fst}\,(s) : A}\times\text{-E}'_1 \qquad \frac{\Gamma \vdash s : A \times B}{\Gamma \vdash \mathrm{snd}\,(s) : B}\times\text{-E}'_2$$

Just as the various rules can be expressed in terms of each other, the various terms can be defined in terms of each other. The terms 'fst' and 'snd' can be defined using 'case'.

$$\begin{aligned}\mathrm{fst}\,(s) &= \mathrm{case}\ s\ \mathrm{of}\ (x, y) \to x\\ \mathrm{snd}\,(s) &= \mathrm{case}\ s\ \mathrm{of}\ (x, y) \to y\end{aligned}$$

Conversely, the term 'case' can be defined using 'fst' and 'snd'.

$$\mathrm{case}\ s\ \mathrm{of}\ (x, y) \to v = (\lambda x.\, \lambda y.\, v)\ (\mathrm{fst}\,(s))\ (\mathrm{snd}\,(s))$$

These definitions can be read off from the corresponding proof trees seen previously.

3.2 Term reduction

Since terms encode proofs, the proof reductions seen previously can be encoded more compactly as term reductions. The operation of substituting a proof subtree for a leaf becomes the operation of substituting a term for a variable.

$$\begin{aligned}(\lambda x.\, u)\ (t) &\Longrightarrow u[t/x]\\ \mathrm{case}\ (t, u)\ \mathrm{of}\ (x, y) \to v &\Longrightarrow v[t/x, u/y]\\ \mathrm{case}\ \mathrm{inl}\,(t)\ \mathrm{of}\ \mathrm{inl}\,(x) \to v;\ \mathrm{inr}\,(y) \to w &\Longrightarrow v[t/x]\\ \mathrm{case}\ \mathrm{inr}\,(u)\ \mathrm{of}\ \mathrm{inl}\,(x) \to v;\ \mathrm{inr}\,(y) \to w &\Longrightarrow w[u/y]\end{aligned}$$

These rules are familiar to any programmer; in particular, the first is the beta reduction rule of lambda calculus.

The properties of proof reduction carry over into properties of term reduction. The correspondence of proof reductions with term reductions guarantees that reducing a well-typed term yields a well-typed term; this is called the *Subject Reduction* property. The uniqueness of normal forms for proofs corresponds to the uniqueness of normal forms for terms, which is analogous to the Church-Rosser theorem for untyped lambda calculus. The existence of normal forms for proofs means that every reduction sequence starting with a well-typed term eventually terminates; a property which certainly does *not* hold for untyped lambda calculus.

It is fascinating to observe that the notion of proof reduction, which was formulated to demonstrate the properties of proof systems, corresponds precisely to term reduction, which was formulated as a model of computation. This is the essence of the Curry-Howard isomorphism.

3.3 Fixpoints

Since every typed term has a normal form, the typed term calculus is strictly less expressive than untyped lambda calculus. To regain the power to express every computable function, one adds for each type A a constant

$$\mathit{fix} : (A \to A) \to A$$

with the reduction rule

$$\mathit{fix}\,(f) \Longrightarrow f\,(\mathit{fix}\,(f)).$$

A term containing this constant may not have a normal form, even if it is well typed. In particular, for every type A, the judgement $\vdash \mathit{fix}\,(\lambda x.\, x) : A$ is derivable, but this term has no normal form. Perhaps that is just as well, since this judgement corresponds to a proof that *any* proposition A is true!

4 Linear logic

Traditional logic encourages reckless use of resources. Contraction profligately duplicates assumptions, Weakening foolishly discards assumptions. This makes sense for logic, where truth is free; and it makes sense for some programming languages, where copying a value is as cheap as copying a pointer. But it is not always sensible. Linear logic encourages more careful use of resources. It is a logic for the 90s. If you lean to the right, view it as a logic of realistic accounting: no more free assumptions. If you lean to the left, view it as an eco-logic: resources must be conserved.

The obvious thing to do is to simply get rid of Contraction and Weakening entirely, but that is perhaps too severe. Our desire is to find a logic that allows control over Contraction and Weakening, but is still powerful enough that traditional intuitionistic logic may be embedded within it. In programming language terms, this corresponds to a language that allows control over duplication and discarding for some variables, but is still powerful enough that all traditional programs may be expressed within it.

So rather than getting rid of Contraction and Weakening entirely, we shall 'bell the cat'. If Contraction or Weakening are used in a proof, then this will be explicitly visible in the proposition proved.

The absence of Contraction and Weakening profoundly changes the nature of the logical connectives. Implication is now written $A \multimap B$ and pronounced 'consume A yielding B'. (The symbol $\multimap$ on its own is pronounced 'lollipop'.) As noted previously, in the absence of Contraction and Weakening, there are two distinct ways to formulate conjunction, corresponding to two distinct connectives in linear logic. These are written $A \otimes B$, pronounced 'both A and B', and $A \,\&\, B$, pronounced 'choose from A and B'. (The symbols $\otimes$ and $\&$ are pronounced 'tensor' and 'with'.) Disjunction is written $A \oplus B$ and still pronounced 'either A or B'. Finally, a new form of proposition is introduced to indicate where Contraction or Weakening may be used. It is written $!A$ and pronounced 'of course A'. (The symbol ! is pronounced 'pling' or 'bang!'.)

As before, let A, B, C range over propositions, and X range over propositional constants. The grammar of linear propositions is as follows.

$$A, B, C ::= X \mid A \multimap B \mid A \otimes B \mid A \,\&\, B \mid A \oplus B \mid !A$$

The particular logical system that we will study is based on Girard's Logic of Unity, which is a refinement of linear logic. This achieves some significant technical simplifications by using *two* forms of assumption. One form of assumption, called *linear*, does not allow Contraction or Weakening, and is written in angle brackets, $\langle A \rangle$. The other form of assumption, called *intuitionistic*, does allow Contraction and Weakening, and is written in square brackets, $[A]$.

As we shall see, an assumption of the form $\langle !A \rangle$ is in a sense equivalent to an assumption of the form $[A]$. However, they differ in that a linear assumption – even one of the form $\langle !A \rangle$ – must be used *exactly once* in a proof, while an intuitionistic assumption may be used any number of times.

As before, let Γ, Δ range over sequences of zero or more assumptions, which may be of either sort. Write $[\Gamma]$ for a sequence of zero or more intuitionistic assumptions.

Judgements, as before, are written in the form $\Gamma \vdash A$. It is only assumptions that are labeled with angle or square brackets – these brackets only appear to the left of $\vdash$, never to the right.

The rules of linear logic are shown in Figure 4. There are now two axioms, $\langle \text{Id} \rangle$ for a linear assumption and [Id] for an intuitionistic assumption. As one might expect, the rules Contraction, Weakening, and the logical rules for ! use intuitionistic assumptions. The logical rules for the other conectives, $\multimap$, $\otimes$, &, and $\oplus$, all use linear assumptions.

Apart from the switch to linear assumptions, the logical rules for $\multimap$, $\otimes$, and $\oplus$ are identical to the logical rules for $\to$, $\times$, and $+$; and the logical rules for & are identical to the alternate logical rules for $\times$. Note that $\otimes$-I uses two distinct assumption lists, while &-I uses the same list twice.

Before looking in detail at the rules for !, let's take a moment to explore the meaning other logical connectives.

4.1 A logic of resources

We can read the new rules in terms of combinations of resources. Take A to be the proposition 'I have ten zlotys', B to be the proposition 'I have a pizza', and C to be the proposition 'I have a cake'. Adding to our logic the axioms

$$\langle A \rangle \vdash B \qquad \langle A \rangle \vdash C$$

expresses that for ten zlotys I may buy a pizza, and for ten zlotys I may buy a cake. Instantiating $\otimes$-I yields

$$\frac{\langle A \rangle \vdash B \qquad \langle A \rangle \vdash C}{\langle A \rangle, \langle A \rangle \vdash B \otimes C}\ \otimes\text{-I}$$

$$\frac{}{\langle A\rangle \vdash A}\,\langle\mathrm{Id}\rangle \qquad \frac{}{[A] \vdash A}\,[\mathrm{Id}] \qquad \frac{\Gamma,\ \Delta \vdash A}{\Delta,\ \Gamma \vdash A}\,\mathrm{Exchange}$$

$$\frac{\Gamma,\ [A],\ [A] \vdash B}{\Gamma,\ [A] \vdash B}\,\mathrm{Contraction} \qquad \frac{\Gamma \vdash B}{\Gamma,\ [A] \vdash B}\,\mathrm{Weakening}$$

$$\frac{[\Gamma] \vdash A}{[\Gamma] \vdash\ !A}\,!\text{-I} \qquad \frac{\Gamma \vdash\ !A \qquad \Delta,\ [A] \vdash B}{\Gamma,\ \Delta \vdash B}\,!\text{-E}$$

$$\frac{\Gamma,\ \langle A\rangle \vdash B}{\Gamma \vdash A \multimap B}\,\multimap\text{-I} \qquad \frac{\Gamma \vdash A \multimap B \qquad \Delta \vdash A}{\Gamma,\ \Delta \vdash B}\,\multimap\text{-E}$$

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma,\ \Delta \vdash A \otimes B}\,\otimes\text{-I} \qquad \frac{\Gamma \vdash A \otimes B \qquad \Delta,\ \langle A\rangle,\ \langle B\rangle \vdash C}{\Gamma,\ \Delta \vdash C}\,\otimes\text{-E}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}\,\&\text{-I} \qquad \frac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash A}\,\&\text{-E}_1 \qquad \frac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash B}\,\&\text{-E}_2$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\,\oplus\text{-I}_1 \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\,\oplus\text{-I}_2 \qquad \frac{\Gamma \vdash A \oplus B \qquad \Delta,\ \langle A\rangle \vdash C \qquad \Delta,\ \langle B\rangle \vdash C}{\Gamma,\ \Delta \vdash C}\,\oplus\text{-E}$$

Fig. 4. Linear logic

meaning that for twenty zlotys I can buy both a pizza and a cake. Instantiating &-I yields

$$\frac{\langle A\rangle \vdash B \qquad \langle A\rangle \vdash C}{\langle A\rangle \vdash B \,\&\, C}\,\&\text{-I}$$

meaning that for ten zlotys I can buy whichever I choose from a cake and a pizza. Instantiating $\oplus$-I$_1$ yields

$$\frac{\langle A\rangle \vdash B}{\langle A\rangle \vdash B \oplus C}\,\oplus\text{-I}_1$$

meaning that for ten zlotys I can buy either a pizza or a cake. But I no longer have a choice: the proof implies that I must buy a pizza. If the bakery closes, so that $\langle A\rangle \vdash C$ is no longer an axiom, then $\langle A\rangle \vdash B \,\&\, C$ is no longer provable, but $\langle A\rangle \vdash B \oplus C$ remains provable as long as the pizzeria remains open.

Take D to be the proposition 'I am happy'. Then $\langle B \otimes C\rangle \vdash D$ expresses that I will be happy given *both* a pizza and a cake; and $\langle B \,\&\, C\rangle \vdash D$ expresses that I will be happy given *my choice* from a pizza and a cake; and $\langle B \oplus C\rangle \vdash D$ expresses that I will be happy given *either* a pizza or a cake, I don't care which.

For any A and B, one can prove $\langle A \& B\rangle \vdash A \oplus B$. Indeed, there are two proofs, one choosing A and one choosing B. But there is no way to prove the converse,

$\langle A \oplus B \rangle \vdash A \,\&\, B$. Furthermore, neither $\langle A \otimes B \rangle \vdash A \,\&\, B$ nor $\langle A \,\&\, B \rangle \vdash A \otimes B$ is provable.

Thanks to the absence of Contraction for linear assumptions, it is impossible to prove $\langle A \rangle \vdash A \otimes A$; disappointingly, ten zlotys cannot magically become twenty. Thanks to the absence of Weakening, it is impossible to prove $\langle A \otimes A \rangle \vdash A$; reassuringly, twenty zlotys will not mysteriously become ten.

4.2 Unlimited resources

A linear assumption, $\langle A \rangle$, can be thought of as supplying exactly one occurrence of A, while an intuitionistic assumption, $[A]$, can be thought of as supplying an unlimited number of occurrences of A: multiple occurrences of A may be supplied if Contraction is used, and no occurrences of A may be supplied if Weakening is used. Taking A to be 'I have ten zlotys', then the linear assumption $\langle A \rangle$ holds if I have a ten zloty note in my pocket, while the intuitionistic assumption $[A]$ holds if I have an unlimited supply of ten zloty notes.

It is instructive to compare the two axioms.

$$\frac{}{\langle A \rangle \vdash A}\,\langle\text{Id}\rangle \qquad \frac{}{[A] \vdash A}\,[\text{Id}]$$

The axiom on the left says that if I have ten zlotys in my pocket, I can reach in and extract ten zlotys. The axiom on the right says that if I have an indefinitely large supply of zloty notes, then from it I may withdraw a single ten zloty note.

If $\Gamma, \langle A \rangle \vdash B$ is provable, then $\Gamma, [A] \vdash B$ is also provable.

$$\dfrac{\dfrac{\Gamma, \langle A \rangle \vdash B}{\Gamma \vdash A \multimap B}\,\multimap\text{-I} \qquad \dfrac{}{[A] \vdash A}\,[\text{Id}]}{\Gamma, [A] \vdash B}\,\multimap\text{-E}$$

Taking $\langle A \rangle \vdash B$ to mean that for ten zlotys I can buy a pizza, it follows that $[A] \vdash B$, so if I have an indefinitely large supply of ten zloty notes then I can still buy a pizza.

Since the logical connectives are defined in terms of linear assumptions, some method is required to turn an intuitionistic assumption into a linear one. This is supplied by the connective !. An intuitionistic assumption $[A]$ is equivalent to a linear assumption $\langle !A \rangle$. Thus, the proposition $!A$ holds when there is an unlimited supply of A.

In the !-I rule, $[\Gamma]$ is written to indicate that all the assumptions must be intuitionistic. The rule states that if from such an assumption list one can prove A, then from the same assumption list one can prove $!A$. In other words, if given unlimited assumptions one can derive a single A, then from the same assumptions one can also derive an unlimited number of A. Instantiating !-I yields

$$\frac{[A] \vdash B}{[A] \vdash\, !B}\,!\text{-I}.$$

If I have an unlimited supply of ten zloty notes, then not only can I buy one pizza, there is no limit to the number I can buy.

The rule !-E states that an intuitionistic assumption $[A]$ may be satisfied by a proof of the proposition $!A$. Adding to our logic the axiom

$$[B] \vdash D$$

expresses that an unlimited supply of pizza leads to happiness. Instantiating !-E yields

$$\frac{[A] \vdash\ !B \qquad [B] \vdash D}{[A] \vdash D}\ !\text{-E}.$$

Given an unlimited supply of zlotys, I can produce an unlimited supply of pizza, and this in turn produces happiness.

A linear assumption $\langle !A \rangle$ is equivalent to an intuitionistic assumption $[A]$. That is, $\Gamma, \langle !A \rangle \vdash B$ is provable if and only if $\Gamma, [A] \vdash B$ is provable. In the forward direction, this is shown with the aid of !-I.

$$\dfrac{\dfrac{\Gamma, \langle !A \rangle \vdash B}{\Gamma \vdash\ !A \multimap B}\ \multimap\text{-I} \qquad \dfrac{\dfrac{}{[A] \vdash A}\ [\text{Id}]}{[A] \vdash\ !A}\ !\text{-I}}{\Gamma, [A] \vdash B}\ \multimap\text{-E}$$

In the reverse direction, it is shown with the aid of !-E.

$$\dfrac{\dfrac{}{\langle !A \rangle \vdash\ !A}\ \langle \text{Id} \rangle \qquad \Gamma, [A] \vdash B}{\Gamma, \langle !A \rangle \vdash B}\ !\text{-E}$$

One might therefore ask: Why bother with assumptions of the form $[A]$? Why not just use $\langle !A \rangle$ instead? There is a good reason for distinguishing the two, which will be explained after proof reductions are discussed.

The propositions $!(A \,\&\, B)$ and $!A \otimes\, !B$ are equivalent in linear logic. That is, both of the following are provable:

$$\langle !(A \,\&\, B) \rangle \vdash\ !A \otimes\, !B,$$
$$\langle !A \otimes\, !B \rangle \vdash\ !(A \,\&\, B).$$

Consider the following two airlines. On one, there is a single steward, who can be called an unlimited number of times. Whenever called, he will give you your choice of either a cup of coffee or a cup of tea. On the other, there are separate coffee and tea stewards, each of whom can be called an unlimited number of times, and each of whom provides a cup of the appropriate beverage when called. Thanks to the equivalence displayed above, you realize that you should not let this distinction determine your choice of airline.

4.3 Embedding intuitionistic logic in linear logic

One reason for incorporating ! into our logic is so that it has sufficient power that intuitionistic logic can be embedded within it. The idea is to find a translation of the connectives $\to$, $\times$ and $+$ into the connectives !, $\multimap$, &, and $\oplus$, such that a judgement $\Gamma \vdash A$ is provable in intuitionistic logic if and only if the corresponding judgement $[\Gamma] \vdash A$ is provable in linear logic. As one might expect, the embedding takes (unlabeled) assumptions of the intuitionistic judgement into intuitionistic assumptions in the linear judgement. This is necessary because Contraction and Weakening may be applied to any assumption in an intuitionistic proof.

To find the translation for $\to$, compare the rules for $\to$ in Figure 1 with the rules for $\multimap$ in Figure 4. The key difference is that A appears as an intuitionistic assumption in $\to$-I, but as a linear assumption in $\multimap$-I. This makes it reasonable to define

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

With this definition, the intuitionistic rules can be defined from the corresponding rules of linear logic, together with the rules for !-I and !-E.

$$\dfrac{\dfrac{}{\langle !A \rangle \vdash !A}\,\langle \mathrm{Id} \rangle \qquad \Gamma,\, [A] \vdash B}{\dfrac{\Gamma,\, \langle !A \rangle \vdash B}{\Gamma \vdash !A \multimap B}\,\multimap\text{-I}}\,!\text{-E} \qquad\qquad \dfrac{\Gamma \vdash !A \multimap B \qquad \dfrac{[\Delta] \vdash B}{[\Delta] \vdash !B}\,!\text{-I}}{\Gamma,\, [\Delta] \vdash B}\,\multimap\text{-E}$$

The derived rules are slightly more general than required, in that Γ may contain assumptions of either type, though $[\Delta]$ is indeed restricted to contain intuitionistic assumptions.

A similar comparison of the rules for $\times$ with the rules for &, and of the rules for $+$ with the rules for $\otimes$, yields the complete embedding.

$$\begin{aligned} A \to B &= !A \multimap B \\ A \times B &= A \,\&\, B \\ A + B &= !A \oplus !B \end{aligned}$$

It is an easy exercise to work out the corresponding derived rules.

Comparing the rules for $\times$ with the rules for $\otimes$ yields an alternative embedding.

$$A \times B = !A \otimes !B$$

Observe that $\langle !A \otimes !B \rangle \vdash A \,\&\, B$ but that $\langle A \,\&\, B \rangle \not\vdash !A \otimes !B$. So in some sense the embedding with & is tighter than the embedding with $\otimes$.

4.4 Proof reductions

The new rules for proof reduction are show in Figure 5. Whereas before the double bars standing for potential occurrences of Contraction and Weakening were ubiquitous, now they appear only in the reduction rule for !. The rules for

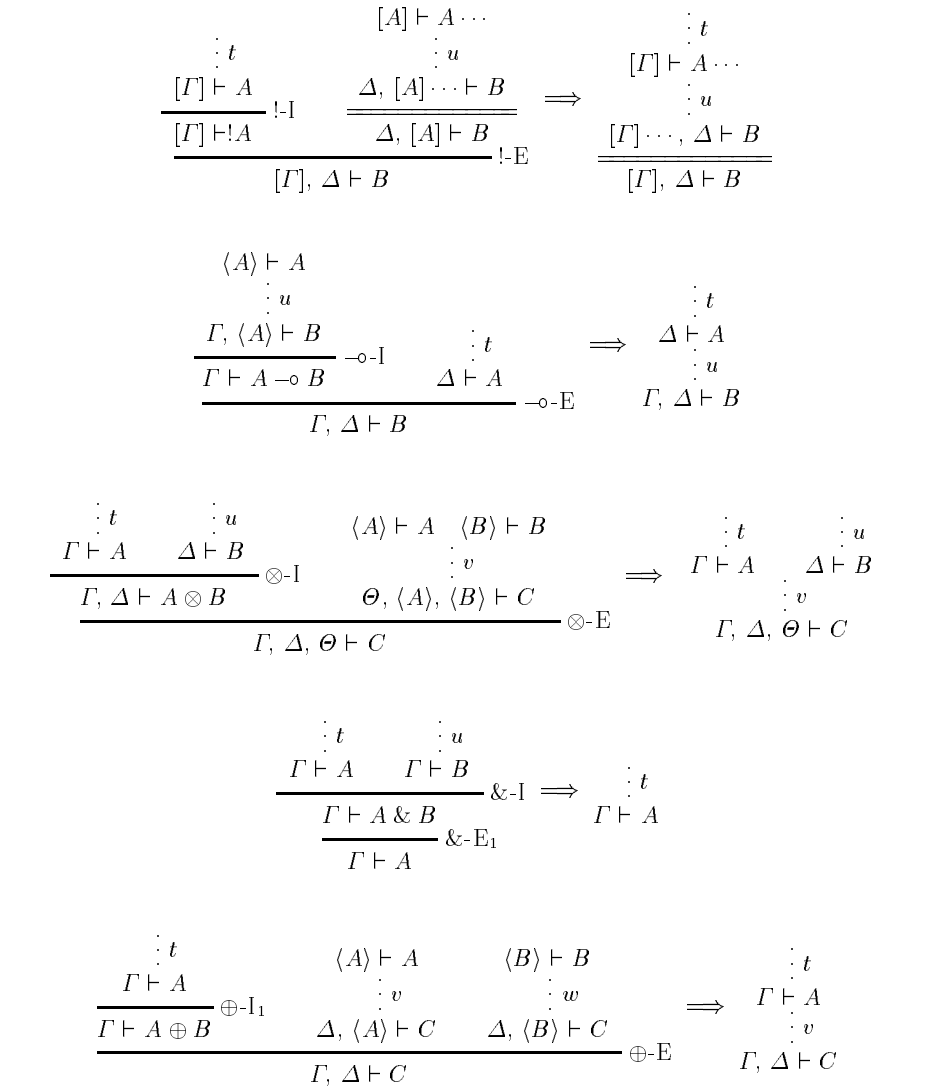

Fig. 5. Proof reduction for linear logic

$\multimap$, $\otimes$, $\oplus$ are simpler than the corresponding rules for $\to$, $\times$, and $+$, because the linear connectives involve only linear assumptions.

In intuitionistic logic, substituting a proof of $\Delta \vdash A$ for each occurrence of the leaf $A \vdash A$ in a proof tree of $\Gamma, A \vdash B$ may cause the assumption list Δ to appear zero, one, or many times in the result: $\Gamma, \Delta \cdots \vdash B$. A number of contractions and weakenings then yields $\Gamma, \Delta \vdash B$. (See, for example, the rule

for $\to$ in Figure 2.)

But in linear logic, substituting a proof of $\Delta \vdash A$ for each occurrence of the leaf $\langle A \rangle \vdash A$ in a proof tree for $\Gamma, \langle A \rangle \vdash B$ will cause the assumptions Δ to appear exactly once in the result: $\Gamma, \Delta \vdash B$. (See, for example, the rule for $\multimap$ in Figure 5.) Hence, simpler reduction rules!

4.5 The need for intuitionistic assumptions

As noted previously, an intuitionistic assumption $[A]$ is equivalent to a linear assumption $\langle !A \rangle$. So why bother with intuitionistic assumptions? One is tempted to define a simpler version of linear logic by replacing $[A]$ by $\langle !A \rangle$ in the rules Contraction and Weakening, and adopting some variant of the [Id], !-I, and !-E rules. Surely such a system would be better?

Alas, this system is *too* simple. The problem shows up when one considers proof reduction. Observe that a judgement can prove a proposition begining with ! even if none of the assumptions begin with !.

$$\dfrac{\dfrac{}{\langle A \multimap\, !B \rangle \vdash A \multimap\, !B}\,\langle \mathrm{Id} \rangle \qquad \dfrac{}{\langle A \rangle \vdash A}\,\langle \mathrm{Id} \rangle}{\langle A \multimap\, !B \rangle, \langle A \rangle \vdash\, !B}\,\multimap\text{-E}$$

Now consider the following proof, which uses a variant Cont$'$ of the contraction rule, in which assumptions of the form $[A]$ have been replaced by assumptions of the form $\langle !A \rangle$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\langle !B \rangle \vdash\, !B}\,\langle \mathrm{Id} \rangle \qquad \dfrac{}{\langle !B \rangle \vdash\, !B}\,\langle \mathrm{Id} \rangle}{\langle !B \rangle, \langle !B \rangle \vdash\, !B \otimes\, !B}\,\otimes\text{-I}}{\langle !B \rangle \vdash\, !B \otimes\, !B}\,\mathrm{Cont}'}{\vdash\, !B \multimap (!B \otimes\, !B)}\,\multimap\text{-I } (\dagger) \qquad \begin{array}{c}\vdots\\ \langle A \multimap\, !B \rangle, \langle A \rangle \vdash\, !B\end{array}}{\langle A \multimap\, !B \rangle, \langle A \rangle \vdash\, !B \otimes\, !B}\,\multimap\text{-E } (\dagger)$$

This contains a $\multimap$-I rule followed immediately by a $\multimap$-E rule, both marked with $(\dagger)$. Applying proof reduction yields the following proof.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \langle A \multimap\, !B \rangle, \langle A \rangle \vdash\, !B\end{array} \qquad \begin{array}{c}\vdots\\ \langle A \multimap\, !B \rangle, \langle A \rangle \vdash\, !B\end{array}}{\langle A \multimap\, !B \rangle, \langle A \rangle, \langle A \multimap\, !B \rangle, \langle A \rangle \vdash\, !B \otimes\, !B}\,\otimes\text{-I}}{\langle A \multimap\, !B \rangle, \langle A \rangle \vdash\, !B \otimes\, !B}\,\mathrm{Cont}'$$

But this proof is *illegal*! Contraction has been applied to assumptions $\langle A \multimap\, !B \rangle$ and $\langle A \rangle$, which is not allowed.

The problem is that the soundness of the proof reduction rule for $\multimap$ depends on the fact that in a proof of $\Gamma, \langle A \rangle \vdash B$, the assumption $\langle A \rangle$ is used exactly

once. But this is no longer true with the modified Contraction and Weakening rules.

This problem does not arise for the system given here. The closest one can come to the bad proof is the following.

$$\dfrac{\dfrac{}{\langle !B\rangle \vdash !B}\,\langle\text{Id}\rangle \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{[B]\vdash B}\,[\text{Id}]}{[B]\vdash !B}\,!\text{-I} \qquad \dfrac{\dfrac{}{[B]\vdash B}\,[\text{Id}]}{[B]\vdash !B}\,!\text{-I}}{[B],\,[B]\vdash !B\otimes !B}\,\otimes\text{-I}}{[B]\vdash !B\otimes !B}\,\text{Cont}}{\langle !B\rangle \vdash !B\otimes !B}\,!\text{-E}}{\vdash !B\multimap (!B\otimes !B)}\,\multimap\text{-I}\ (\dagger) \qquad \begin{array}{c}\vdots\\ \langle A\multimap !B\rangle,\,\langle A\rangle\vdash !B\end{array}}{\langle A\multimap !B\rangle,\,\langle A\rangle\vdash !B\otimes !B}\,\multimap\text{-E}\ (\dagger)$$

Applying proof reduction yields the following proof.

$$\dfrac{\begin{array}{c}\vdots\\ \langle A\multimap !B\rangle,\,\langle A\rangle\vdash !B\end{array} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{[B]\vdash B}\,[\text{Id}]}{[B]\vdash !B}\,!\text{-I} \qquad \dfrac{\dfrac{}{[B]\vdash B}\,[\text{Id}]}{[B]\vdash !B}\,!\text{-I}}{[B],\,[B]\vdash !B\otimes !B}\,\otimes\text{-I}}{[B]\vdash !B\otimes !B}\,\text{Cont}}{\langle A\multimap !B\rangle,\,\langle A\rangle\vdash !B\otimes !B}\,!\text{-E}$$

Thanks to !-E, the linear assumption $\langle !B\rangle$ is used exactly once in the proof of $\langle !B\rangle \vdash !B\otimes !B$, and so proof reduction poses no problem.

It is indeed possible to formulate a system of linear logic without introducing intuitionistic assumptions. One way to do so is to replace the rules [Id], Contraction, Weakening, !-I, and !-E of Figure 4 by the rules in Figure 6. The rule !-E$'_1$ combines [Id] and !-E, and the rules !-E$'_2$ and !-E$'_3$ correspond to Contraction and Weakening. This system has been explored by Benton, Bierman, de Paiva, and Hyland [2]. This system is considerably more complex, especially when one compares !-I with !-I$'$.

$$\dfrac{\Gamma_1\vdash !A_1 \quad\cdots\quad \Gamma_n\vdash !A_n \quad \langle !A_1\rangle,\,\ldots,\,\langle !A_n\rangle\vdash B}{\Gamma_1,\,\ldots,\,\Gamma_n\vdash !B}\,!\text{-I}'$$

$$\dfrac{\Gamma\vdash !A \quad \Delta,\,\langle A\rangle\vdash B}{\Gamma,\,\Delta\vdash B}\,!\text{-E}'_1 \qquad \dfrac{\Gamma\vdash !A \quad \Delta,\,\langle !A\rangle,\,\langle !A\rangle\vdash B}{\Gamma,\,\Delta\vdash B}\,!\text{-E}'_2 \qquad \dfrac{\Gamma\vdash !A \quad \Delta\vdash B}{\Gamma,\,\Delta\vdash B}\,!\text{-E}'_3$$

Fig. 6. Alternate rules for linear logic

5 Linear types

At last, all the pieces are in place to carry through our plan. We have seen intuitionistic logic and how it induces a typed functional language, and we have seen linear logic. Having set everything up carefully, it is straightforward to induce a typed language based on linear logic.

As before, there is one term form for variables, and one for each of the logical introduction and elimination rules. The term forms follow closely what has already been seen for intuitionistic logic. To distinguish the new term forms, they are generally written with angle brackets rather than round brackets. There are two pair constructors, $\langle t, u \rangle$ for types of the form $A \otimes B$, and $\langle\langle t, u \rangle\rangle$ for types of the form $A \,\&\, B$. The grammar of terms is as follows.

$$
\begin{array}{rcl}
s, t, u, v, w & ::= & x \\
& \mid & \lambda\langle x \rangle . \, u \mid s \, \langle t \rangle \\
& \mid & !t \mid \text{case } s \text{ of } !x \to u \\
& \mid & \langle t, u \rangle \mid \text{case } s \text{ of } \langle x, y \rangle \to v \\
& \mid & \langle\langle t, u \rangle\rangle \mid \text{fst } \langle s \rangle \mid \text{snd } \langle s \rangle \\
& \mid & \text{inl} \, \langle t \rangle \mid \text{inr} \, \langle u \rangle \mid \text{case } s \text{ of inl} \, \langle x \rangle \to v; \text{ inr} \, \langle y \rangle \to w
\end{array}
$$

Assumptions now take two forms: linear, written $\langle x : A \rangle$, and intuitionistic, written $[x : A]$. As before, let Γ, Δ range over lists of zero or more assumptions, where all of the variables are distinct, and let $[\Gamma]$ denote a list containing only intuitionistic assumptions. Judgements have the form $\Gamma \vdash t : A$.

The rules for linear terms are shown in Figure 7.

Here is a sample proof augmented with terms.

$$
\dfrac{\dfrac{}{\langle x : !B \rangle x : !B}\,\langle\text{Id}\rangle \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{[y : B] \vdash y : B}\,[\text{Id}]}{[y : B] \vdash !y : !B}\,!\text{-I} \qquad \dfrac{\dfrac{}{[y : B] \vdash y : B}\,[\text{Id}]}{[y : B] \vdash !y : !B}\,!\text{-I}}{[y : B], [y : B] \vdash \langle !y, !y \rangle : !B \otimes !B}\,\otimes\text{-I}}{[y : B] \vdash \langle !y, !y \rangle : !B \otimes !B}\,\text{Cont}}{\dfrac{\langle x : !B \rangle \vdash \text{case } x \text{ of } !y \to \langle !y, !y \rangle : !B \otimes !B}{\vdash \lambda\langle x \rangle . \, \text{case } x \text{ of } !y \to \langle !y, !y \rangle : !B \multimap !B \otimes !B}\,\multimap\text{-I}}\,!\text{-E}
$$

As before, the judgement at the root uniquely encodes the entire proof tree, modulo the commuting conversions.

5.1 Term reductions

Proof reductions may be read off as term reductions. As before, the properties of proof reductions guarantee that every term has a unique normal form, and

$$\frac{}{\langle x : A\rangle \vdash x : A}\ \langle\mathrm{Id}\rangle \qquad \frac{}{[x : A] \vdash x : A}\ [\mathrm{Id}] \qquad \frac{\Gamma,\ \Delta \vdash t : A}{\Delta,\ \Gamma \vdash t : A}\ \mathrm{Exchange}$$

$$\frac{\Gamma,\ [y : A],\ [z : A] \vdash u : B}{\Gamma,\ [x : A] \vdash u[x/y,\ x/z] : B}\ \mathrm{Contraction} \qquad \frac{\Gamma \vdash u : B}{\Gamma,\ [x : A] \vdash u : B}\ \mathrm{Weakening}$$

$$\frac{[\Gamma] \vdash t : A}{[\Gamma] \vdash\ !t : !A}\ !\text{-I} \qquad \frac{\Gamma \vdash s : !A \qquad \Delta,\ [x : A] \vdash u : B}{\Gamma,\ \Delta \vdash \mathrm{case}\ s\ \mathrm{of}\ !x \to u : B}\ !\text{-E}$$

$$\frac{\Gamma \vdash t : A \qquad \Delta \vdash u : B}{\Gamma,\ \Delta \vdash \langle t, u\rangle : A \otimes B}\ \otimes\text{-I} \qquad \frac{\Gamma \vdash s : A \otimes B \qquad \Delta,\ \langle x : A\rangle,\ \langle y : B\rangle \vdash v : C}{\Gamma,\ \Delta \vdash \mathrm{case}\ s\ \mathrm{of}\ \langle x, y\rangle \to v : C}\ \otimes\text{-E}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash \langle\langle t, u\rangle\rangle : A\ \&\ B}\ \&\text{-I} \qquad \frac{\Gamma \vdash s : A\ \&\ B}{\Gamma \vdash \mathrm{fst}\,\langle s\rangle : A}\ \&\text{-E}_1 \qquad \frac{\Gamma \vdash s : A\ \&\ B}{\Gamma \vdash \mathrm{snd}\,\langle s\rangle : B}\ \&\text{-E}_2$$

$$\frac{\Gamma,\ \langle x : A\rangle \vdash u : B}{\Gamma \vdash \lambda\langle x\rangle.\, u : A \multimap B}\ \multimap\text{-I} \qquad \frac{\Gamma \vdash s : A \multimap B \qquad \Delta \vdash t : A}{\Gamma,\ \Delta \vdash s\,\langle t\rangle : B}\ \multimap\text{-E}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathrm{inl}\,\langle t\rangle : A \oplus B}\ \oplus\text{-I}_1 \qquad \frac{\Gamma \vdash u : B}{\Gamma \vdash \mathrm{inr}\,\langle u\rangle : A \oplus B}\ \oplus\text{-I}_2$$

$$\frac{\Gamma \vdash s : A \oplus B \qquad \Delta,\ \langle x : A\rangle \vdash v : C \qquad \Delta,\ \langle y : B\rangle \vdash w : C}{\Gamma,\ \Delta \vdash \mathrm{case}\ s\ \mathrm{of}\ \mathrm{inl}\,\langle x\rangle \to v;\ \mathrm{inr}\,\langle y\rangle \to w : C}\ \oplus\text{-E}$$

Fig. 7. Linear types

that term reduction preserves well typing.

$$\begin{aligned}
\mathrm{case}\ !t\ \mathrm{of}\ !x \to u &\Longrightarrow u[t/x]\\
(\lambda\langle x\rangle.\, u)\,\langle t\rangle &\Longrightarrow u[t/x]\\
\mathrm{case}\ \langle t, u\rangle\ \mathrm{of}\ \langle x, y\rangle \to v &\Longrightarrow v[t/x, u/y]\\
\mathrm{fst}\,\langle\ \langle\langle t, u\rangle\rangle\ \rangle &\Longrightarrow t\\
\mathrm{snd}\,\langle\ \langle\langle t, u\rangle\rangle\ \rangle &\Longrightarrow u\\
\mathrm{case}\ \mathrm{inl}\,\langle t\rangle\ \mathrm{of}\ \mathrm{inl}\,\langle x\rangle \to v;\ \mathrm{inr}\,\langle y\rangle \to w &\Longrightarrow v[t/x]\\
\mathrm{case}\ \mathrm{inr}\,\langle u\rangle\ \mathrm{of}\ \mathrm{inl}\,\langle x\rangle \to v;\ \mathrm{inr}\,\langle y\rangle \to w &\Longrightarrow w[u/y]
\end{aligned}$$

Thanks to linearity, the only substitution in the above that may duplicate a variable is that for !. In a lazy language, this means that only evaluation of ! requires overwriting.

It is possible to arrange an implementation so that a variable corresponding to a linear assumption contains the sole pointer to a value. However, this requires that a term of the form $!t$ be re-evaluated *each* time it is examined, which is prohibitively expensive for most purposes.

More commonly, a term of the form $!t$ will be overwitten with its value the first time it is accessed, and future accesses will copy a pointer to that value. In this case, a variable corresponding to a linear assumption may not contain the sole pointer to a value. However, the absence of Contraction and Weakening on linear assumptions makes it possible to guarantee the following useful property: if a variable corresponding to a linear assumption ever contains the sole pointer to a variable, then it will continue to do so. Some applications of this property will be discussed later.

5.2 Embedding intuitionistic logic into linear logic

As we have seen, the connectives of intuitionistic logic can be regarded as abbreviations for combinations of connectives in linear logic.

$$\begin{aligned} A \to B &= !A \multimap B \\ A \times B &= A \,\&\, B \\ A + B &= !A \oplus !B \end{aligned}$$

Similarly, the term calculus of intuitionistic logic can be regarded as abbreviations for combinations of linear terms.

$$\begin{aligned} \lambda x\,.\,u &= \lambda\langle x'\rangle.\,\mathrm{case}\ x'\ \mathrm{of}\ !x \to u \\ t\,(u) &= t\,\langle !u\rangle \\ (t,u) &= \langle\!\langle t,u\rangle\!\rangle \\ \mathrm{fst}\,(s) &= \mathrm{fst}\,\langle s\rangle \\ \mathrm{snd}\,(s) &= \mathrm{snd}\,\langle s\rangle \\ \mathrm{inl}\,(t) &= \mathrm{inl}\,\langle !t\rangle \\ \mathrm{inr}\,(u) &= \mathrm{inr}\,\langle !u\rangle \\ \mathrm{case}\ s\ \mathrm{of}\ \mathrm{inl}\,(x) \to v;\ \mathrm{inr}\,(y) \to w &= \mathrm{case}\ s\ \mathrm{of} \\ &\qquad \mathrm{inl}\,\langle x'\rangle \to \mathrm{case}\ x'\ \mathrm{of}\ !x \to v; \\ &\qquad \mathrm{inr}\,\langle y'\rangle \to \mathrm{case}\ y'\ \mathrm{of}\ !y \to w \end{aligned}$$

An intuitionistic judgement $\Gamma \vdash t : A$ is provable if and only if the corresponding linear judgement $[\Gamma] \vdash t : A$ is provable. In this way, the ordinary lambda calculus can be regarded as a subset of the linear lambda calculus.

There is an alternate translation for the product types.

$$A \times B = !A \otimes !B$$

This gives rise to an alternate translation for terms.

$$\begin{aligned} (t,u) &= \langle !t, !u\rangle \\ \mathrm{case}\ s\ \mathrm{of}\ (x,y) \to v &= \mathrm{case}\ s\ \mathrm{of}\ \langle x', y'\rangle \to \\ &\qquad \mathrm{case}\ x'\ \mathrm{of}\ !x \to \\ &\qquad\quad \mathrm{case}\ y'\ \mathrm{of}\ !y \to v \end{aligned}$$

In some circumstances, this alternate translation may be more efficient.

The embedding of intutionistic logic into linear logic also works for the fixpoint constant.

$$\mathit{fix} : !(!A \multimap A) \multimap A$$

This has the following reduction rule.

$$\mathit{fix}\,\langle !f\rangle \Longrightarrow f\,\langle !\mathit{fix}\langle !f\rangle\rangle$$

5.3 Array update

As mentioned, if a variable corresponding to a linear assumption ever contains the sole pointer to a value, then it will continue to do so. This can be exploited to provide a solution to the old problem of in-place update for arrays.

An array (of type Arr) is a mapping of indices (of type Ix) to values (of type Val). The usual operations provided on arrays are as follows.

$$\begin{aligned} \mathit{new} &: \mathit{Val} \to \mathit{Arr} \\ \mathit{lookup} &: \mathit{Ix} \to \mathit{Arr} \to \mathit{Val} \\ \mathit{update} &: \mathit{Ix} \to \mathit{Val} \to \mathit{Arr} \to \mathit{Arr} \end{aligned}$$

Here $\mathit{new}\,(v)$ returns an array with each location initialised to v, and $\mathit{lookup}\,(i)\,(a)$ returns the value at location i in array a, and $\mathit{update}\,(i)\,(v)\,(a)$ returns an array identical to a except that location i contains value v. The trouble with this is that the update operation can be prohibitively expensive, as it may need to copy the entire array.

A version of these operators may be devised for the linear type calculus that places a newly allocated array in a variable corresponding to a linear assumption. Since the variable contains the sole pointer to the array, one can guarantee that it will continue to do so. Hence the update operation may safely be performed in place. Here are the new versions of the operations.

$$\begin{aligned} \mathit{new} &: !\,\mathit{Val} \multimap (\mathit{Arr} \multimap \mathit{Arr} \otimes X) \multimap X \\ \mathit{lookup} &: !\mathit{Ix} \multimap \mathit{Arr} \multimap \mathit{Arr} \otimes !\,\mathit{Val} \\ \mathit{update} &: !\mathit{Ix} \multimap !\,\mathit{Val} \multimap \mathit{Arr} \multimap \mathit{Arr} \end{aligned}$$

Here $\mathit{new}\,\langle v\rangle\,\langle f\rangle$ allocates a new array a with each location initialised to v, and then computes $f\,\langle a\rangle$ which will return a pair $\langle a', x\rangle$, and then deallocates the final array a' and returns the value x; thus *new* acts very much like a block, in that it allocates an array, processes it, and deallocates it. The call $\mathit{lookup}\,\langle i\rangle\,\langle a\rangle$ returns the pair $\langle a, v\rangle$ where v is the value in location i of a. Note that since *lookup* is passed the sole pointer to the array, it must return the array it is passed. As before, the call $\mathit{update}\,\langle i\rangle\,\langle v\rangle\,\langle a\rangle$ returns an array identical to a except that location i contains value v; but since this call is passed the sole pointer to array a, it may be safely implemented by updating the array in place.

For example, evaluating

$$\begin{aligned} \mathit{new}\,\langle 6\rangle\,\langle \lambda\langle a_0\rangle.\ & \text{case } \mathit{lookup}\,\langle 1\rangle\,\langle a_0\rangle \text{ of } \langle a_1, x\rangle \to \\ & \text{case } \mathit{update}\,\langle 2\rangle\,\langle x+1\rangle\,\langle a_1\rangle \text{ of } a_2 \to \\ & \text{case } \mathit{lookup}\,\langle 2\rangle\,\langle a_2\rangle \text{ of } \langle a_3, y\rangle \to \\ & \langle a_3, x \times y\rangle\rangle \end{aligned}$$

returns 42. (In the second line, case t of $x \to u$ is a convenient abbreviation for $(\lambda\langle x\rangle.\, u)\ (t)$.)

This approach requires further refinement. The form given here is too unwieldy for convenient use. But it should give a hint as to the practical applications of linear logic.

6 Conclusions and related work

Traditional logic has close ties to computing in general and functional languages in particular. The Curry-Howard isomorphism specifies a precise correspondence between intuitionistic logic, on the one hand, and typed lambda calculus, on the other [4, 11].

As a result, logicians and computer scientists sometimes discover the same system independently, usually with the logician getting there first. The type inference algorithm published by Milner in 1978, is at the heart of the type system used in ML, Miranda, and Haskell [14]. Unbeknown to Milner, the same idea was published by Hindley in 1969 [9]. Reynolds described a polymorphic lambda calculus in 1974 that generalised Milner's type system, and also generalised the power of generic type variables in languages such as Ada [17, 18]. Unbeknown to Reynolds, the same generalisation was described by Girard in 1972 [5].

In 1987, Girard published the first description of linear logic [6]. By now, the computer scientists and the logicians had realised that they had something to say to one another: the seminal paper appeared in *Theoretical Computer Science*. Computer scientists have been active ever since, exploring the applications of linear logic to computing.

Early computational models were discussed by Lafont [12] and Holström [10]. Abramsky wrote a highly influential paper that explored computing applications of both intuitionistic and classical linear logic [1]. Other models have been discussed by Chirimar, Gunter, and Riecke [3], Lincoln and Mitchell [13], Reddy [16], and Wadler [21, 22].

The particular formulation of linear logic presented here is based on Girard's Logic of Unity, a refinement of linear logic [7]. This overcomes some technical problems with other presentations of linear logic, some of which are discussed by Benton, Bierman, de Paiva, and Hyland [2], and Wadler [23, 24]. Much of the insight for this work comes from categorical models of linear logic [19, 15]. The particular system presented here was suggested to the author by Girard, and a similar system has been suggested by Plotkin.

For further background on traditional logic see the wonderful introduction by Girard, Lafont, and Taylor [8], and for further details on linear logic see the helpful textbook by Troelstra [20].

Acknowledgements. I thank Cordy Hall for detailed and timely comments, and Stefan Kahrs and John Hatcliff for suggesting improvements. The paper was produced using Knuth's Tex, Lamport's Latex, Taylor's tree macros, and style macros from Springer-Verlag.

References

1. S. Abramsky, Computational interpretations of linear logic. Presented at *Workshop on Mathematical Foundations of Programming Language Semantics*, 1990. To appear in *Theoretical Computer Science*.
2. N. Benton, G. Bierman, V. de Paiva, and M. Hyland, Type assignment for intuitionistic linear logic. Technical report 262, Computing Laboratory, University of Cambridge, August 1992.
3. J. Chirimar, C. A. Gunter, and J. G. Riecke. Linear ML. In *Symposium on Lisp and Functional Programming*, ACM Press, San Francisco, June 1992.
4. H. B. Curry and R. Feys, *Combinatory Logic*, North Holland, 1958.
5. J.-Y. Girard, *Interprétation functionelle et élimination des coupures dans l'arithmétique d'ordre supérieure*. Ph.D. thesis, Université Paris VII, 1972.
6. J.-Y. Girard, Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
7. J.-Y. Girard, On the unity of logic. Manuscript, 1991.
8. J.-Y. Girard, Y. Lafont, and P. Taylor, *Proofs and types*, Cambridge University Press, 1989.
9. R. Hindley, The principal type scheme of an object in combinatory logic. *Trans. Am. Math. Soc.*, 146:29–60, December 1969.
10. S. Holmström, A linear functional language. Draft paper, Chalmers University of Technology, 1988.
11. W. A. Howard, The formulae-as-types notion of contstruction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, Academic Press, 1980. (The original version was circulated privately in 1969.)
12. Y. Lafont, The linear abstract machine. *Theoretical Computer Science*, 59:157–180, 1988.
13. P. Lincoln and J. Mitchell, Operational aspects of linear lambda calculus. In *7'th Symposium on Logic in Computer Science*, IEEE Press, Santa Cruz, California, June 1992.
14. R. Milner, A theory of type polymorphism in programming. *J. Comput. Syst. Sci.*, 17:348–375, 1978.
15. V. Pratt, Event spaces and their linear logic. In *AMAST '91: Algebraic Methodology And Software Technology*, Iowa City, Springer Verlag LNCS, 1992.
16. U. S. Reddy, A typed foundation for directional logic programming. In E. Lamma and P. Mello, editors, *Extensions of logic programming*, Lecture Notes in Artificial Intelligence 660, Springer-Verlag, 1993.
17. J. C. Reynolds, Towards a theory of type structure. In B. Robinet, editor, *Proc. Colloque sur la Programmation*, LNCS 19, Springer-Verlag.
18. J. C. Reynolds, Three approaches to type structure. In *Mathematical Foundations of Software Development*, LNCS 185, Springer-Verlag, 1985.
19. R. A. G. Seely, Linear logic, $*$-autonomous categories, and cofree coalgebras. In *Categories in Computer Science and Logic*, June 1989. AMS Contemporary Mathematics 92.
20. A. S. Troelstra, *Lectures on Linear Logic*. CSLI Lecture Notes, 1992.
21. P. Wadler, Linear types can change the world! In *IFIP TC 2 Working Conference on Programming Concepts and Methods*, Sea of Galilee, Israel, April 1990. Published as M. Broy and C. Jones, editors, *Programming Concepts and Methods*, North Holland, 1990.

22. P. Wadler, Is there a use for linear logic? In *Conference on Partial Evaluation and Semantics-Based Program Manipulation (PEPM)*, ACM Press, New Haven, Connecticut, June 1991.
23. P. Wadler, There's no substitute for linear logic. Presented at *Workshop on Mathematical Foundations of Programming Language Semantics*, Oxford, April 1992.
24. P. Wadler, A syntax for linear logic. Presented at *Conference on Mathematical Foundations of Programming Language Semantics*, New Orleans, April 1993.

This article was processed using the LaTeX macro package with LLNCS style