Linear Logic

Frank Pfenning
Carnegie Mellon University

$\boxed{\text{Draft of May 4, 1998}}$

Material for the course *Lineare Logik mit Anwendungen*, Technische Hochschule Darmstadt, Sommersemester 1996. Revised for the course *Linear Logic* at Carnegie Mellon University, Spring 1998. Material for this course is available at
`http://www.cs.cmu.edu/~fp/courses/linear.html`.

Please send comments to `fp@cs.cmu.edu`

This material is in rough draft form and is likely to contain errors. Furthermore, citations are in no way adequate or complete. Please do not cite or distribute this document.

This work was supported by NSF Grants CCR-9303383 and CCR-9619684.

Contents

Chapter 1

Natural Deduction

> Ich wollte zunächst einmal einen Formalismus aufstellen, der dem wirklichen Schließen möglichst nahe kommt. So ergab sich ein „Kalkül des natürlichen Schließens“.
>
> — Gerhard Gentzen
> *Untersuchungen über das logische Schließen* [Gen35]

In this chapter we explore ways to define logics, or, which comes to the same thing, ways to give meaning to logical connectives. Our fundamental notion is that of a *judgment* based on *evidence*. For example, we might make the judgment "*It is raining*" based on visual evidence. Or we might make the judgment "*'A implies A' is true for any proposition A*" based on a derivation. The use of the notion of a judgment as conceptual prior to the notion of proposition has been advocated by Martin-Löf [ML85a, ML85b]. Certain forms of judgments frequently recur and have therefore been investigated in their own right, prior to logical considerations. Two that we will use are *hypothetical judgments* and *parametric jugments* (the latter is sometimes called *general judgment* or *schematic judgment*).

A hypothetical judgment has the form "J_2 *under hypothesis* J_1". We consider this judgment evident if we are prepared to make the judgment J_2 once provided with evidence for J_1. Formal evidence for a hypothetical judgment is a *hypothetical derivation* where we can freely use the hypothesis J_1 in the derivation of J_2. Note that hypotheses need not be used, and could be used more than once. In contrast, derivations of *linear hypothetical judgments* introduced later are restricted so the the linear hypotheses must be used exactly once.

A parametric judgment has the form "J *for any* a" where a is a *parameter* which may occur in J. We make this judgment if we are prepared to make the judgment $[O/a]J$ for arbitrary objects O of the right category. Here $[O/a]J$ is our notation for substituting the object O for parameter a in the judgment J. Formal evidence for a parametric judgment J is a *parametric derivation* with free occurrences of the parameter a.

Formal evidence for a judgment in form of a derivation is usually written in two-dimensional notation:

$$\begin{array}{c}\mathcal{D}\\ J\end{array}$$

if $\mathcal{D}$ is a derivation of J. For the sake of brevity we sometimes use the alternative notation $\mathcal{D} of J$. A hypothetical judgment is written as

$$\begin{array}{c}\overline{J_1}\,u\\ \vdots\\ J_2\end{array}$$

where u is a label which identifies the hypothesis J_1. We use the labels to guarantee that hypotheses which are introduced during the reasoning process are not used outside their scope.

The separation of the notion of judgment and proposition and the corresponding separation of the notion of evidence and proof sheds new light on various styles that have been used to define logical systems.

An axiomatization in the style of Hilbert [Hil22], for example, arises when one defines a judgment "*A is true*" without the use of hypothetical judgments. Such a definition is highly economical in its use of judgments, which has to be compensated by a liberal use of implication in the axioms. When we make proof structure explicit in such an axiomatization, we arrive at combinatory logic [Cur30].

A categorical logic [LS86] arises when the basic judgment is not truth, but entailment "*A entails B*". Once again, presentations are highly economical and do not need to seek recourse in complex judgment forms (at least for the propositional fragment). But derivations often require many hypotheses, which means that we need to lean rather heavily on conjunction here. Proofs are realized by morphisms which are an integral part of the machinery of category theory.

While these are interesting and in many ways useful approaches to logic specification, neither of them comes particularly close to capturing the practice of mathematical reasoning. This was Gentzen's point of departure for the design of a system of *natural deduction* [Gen35]. From our point of view, this system is based on the simple judgment "*A is true*", but relies critically on hypothetical and parametric judgments. In addition to being extremely elegant, it has the great advantage that one can define all logical connectives without reference to any other connective. This principle of modularity extends to the meta-theoretic study of natural deduction and simplifies considering fragments and extension of logics. Since we will consider many fragments and extension, this *orthogonality* of the logical connectives is a critical consideration. There is another advantage to natural deduction, namely that its proofs are isomorphic to the terms in a λ-calculus via the so-called Curry-Howard isomorphism [How69], which establishes many connections to functional programming.

Finally, we arrive at the *sequent calculus* (also introduced by Gentzen in his seminal paper [Gen35]) when we split the single judgment of truth into two:

"*A is an assumption*" and "*A is a true conclusion*". While we still employ the machinery of parametric and hypothetical judgments, we now need an explicit rule to state that "*A is an assumption*" is sufficient evidence for "*A is a true conclusion*". The reverse, namely that if "*A is a true conclusion*" then "*A may be used as an assumption*" is the Cut rule which he proved to be redundant in his *Hauptsatz*. For Gentzen the sequent calculus was primarily a technical device to prove consistency of his system of natural deduction, but it exposes many details of the fine structure of proofs in such a clear manner, that most presentations of linear and related logics employ sequent calculus. The laws governing the structure of proofs, however, are more complicated than the Curry-Howard isomorphism for natural deduction might suggest and are still the subject of study [Her95, Pfe95].

We choose natural deduction as our definitional formalism as the purest and most widely applicable. Later we justify the sequent calculus as a calculus of proof search for natural deduction and explicitly relate the two forms of presentation.

We begin by introducing natural deduction for intuitionistic logic, exhibiting its basic principles. We then enrich it to capture linear connectives.

1.1 Intuitionistic Natural Deduction

The system of natural deduction we describe below is basically Gentzen's system NJ [Gen35] or the system which may be found in Prawitz [Pra65]. The calculus of natural deduction was devised by Gentzen in the 1930's out of a dissatisfaction with axiomatic systems in the Hilbert tradition, which did not seem to capture mathematical reasoning practices very directly. Instead of a number of axioms and a small set of inference rules, valid deductions are described through inference rules only, which at the same time explain the meaning of the logical quantifiers and connectives in terms of their proof rules.

A language of (first-order) *terms* is built up from *variables* x, y, *etc.*, *function symbols* f, g, *etc.*, each with a unique arity, and *parameters* a, b, *etc.* in the usual way.

$$\textit{Terms} \quad t \quad ::= \quad x \mid a \mid f(t_1, \ldots, t_n)$$

A constant c is simply a function symbol with arity 0 and we write c instead of $c()$. Exactly which function symbols are available is left unspecified in the general development of predicate logic and only made concrete for specific theories, such as the theory of natural numbers. However, variables and parameters are always available. We will use t and s to range over terms.

The language of *propositions* is built up from *predicate symbols* P, Q, *etc.* and terms in the usual way.

$$\begin{array}{lll} \textit{Propositions} \quad A & ::= & P(t_1, \ldots, t_n) \mid A_1 \wedge A_2 \mid A_1 \supset A_2 \mid A_1 \vee A_2 \mid \neg A \\ & & \mid \bot \mid \top \mid \forall x.\, A \mid \exists x.\, A \end{array}$$

A propositional constant P is simply a predicate symbol with no arguments and we write P instead of $P()$. We will use A, B, and C to range over propositions.

Exactly which predicate symbols are available is left unspecified in the general development of predicate logic and only made concrete for specific theories.

The notions of *free* and *bound* variables in terms and propositions are defined in the usual way: the variable x is bound in propositions of the form $\forall x.\ A$ and $\exists x.\ A$. We use parentheses to disambiguate and assume that $\wedge$ and $\vee$ bind more tightly than $\supset$. It is convenient to assume that propositions have no free individual variables; we use parameters instead where necessary. Our notation for substitution is $[t/x]A$ for the result of substituting the term t for the variable x in A. Because of the restriction on occurrences of free variables, we can assume that t is free of individual variables, and thus capturing cannot occur.

The main judgment of natural deduction is "C *is true*" written as $\vdash C$, from hypotheses $\vdash A_1, \ldots, \vdash A_n$. We will model this as a hypothetical judgment. This means that certain structural properties of derivations are tacitly assumed, independently of any logical inferences. In essence, these assumptions explain what hypothetical judgments are.

Hypothesis. If we have a hypothesis $\vdash A$ than we can conclude $\vdash A$.

Weakening. Hypotheses need not be used.

Duplication. Hypotheses can be used more than once.

Exchange. The order in which hypotheses are introduced is irrelevant.

In natural deduction each logical connective and quantifier is characterized by its *introduction rule(s)* which specifies how to infer that a conjunction, disjunction, *etc.* is true. The *elimination rule* for the logical constant tells what other truths we can deduce from the truth of a conjunction, disjunction, *etc.* Introduction and elimination rules must match in a certain way in order to guarantee that the rules are meaningful and the overall system can be seen as capturing mathematical reasoning.

The first is a *local soundness* property: if we introduce a connective and then immediately eliminate it, we should be able to erase this detour and find a more direct derivation of the conclusion without using the connective. If this property fails, the elimination rules are too strong: they allow us to conclude more than we should be able to know.

The second is a *local completeness* property: we can eliminate a connective in a way which retains sufficient information to reconstitute it by an introduction rule. If this property fails, the elimination rules are too weak: the do not allow us to conclude everything we should be able to know.

We provide evidence for local soundness and completeness of the rules by means of *local reduction* and *expansion* judgments, which relate proofs of the same proposition.

One of the important principles of natural deduction is that each connective should be defined only in terms of inference rules without reference to other logical connectives or quantifiers. We refer to this as *orthogonality* of the connectives. It means that we can understand a logical system as a whole by understanding each connective separately. It also allows us to consider fragments and extensions directly and it means that the investigation of properties of a logical system can be conducted in a modular way.

We now show the introduction and elimination rules, local reductions and expansion for each of the logical connectives in turn. The rules are summarized on page 1.1.

Conjunction. $A \wedge B$ should be true if both A and B are true. Thus we have the following introduction rule.

$$\frac{\vdash A \qquad \vdash B}{\vdash A \wedge B} \wedge\mathrm{I}$$

If we consider this as a complete definition, we should be able to recover both A and B if we know $A \wedge B$. We are thus led to two elimination rules.

$$\frac{\vdash A \wedge B}{\vdash A} \wedge\mathrm{E_L} \qquad \frac{\vdash A \wedge B}{\vdash B} \wedge\mathrm{E_R}$$

To check our intuition we consider a deduction which ends in an introduction followed by an elimination:

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \vdash A\end{array} \qquad \begin{array}{c}\mathcal{E}\\ \vdash B\end{array}}{\vdash A \wedge B} \wedge\mathrm{I}}{\vdash A} \wedge\mathrm{E_L}$$

Clearly, it is unnecessary to first introduce the conjunction and then eliminate it: a more direct proof of the same conclusion from the same (or fewer) assumptions would be simply

$$\begin{array}{c}\mathcal{D}\\ \vdash A\end{array}$$

Formulated as a transformation or *reduction* between derivations we have

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \vdash A\end{array} \qquad \begin{array}{c}\mathcal{E}\\ \vdash B\end{array}}{\vdash A \wedge B} \wedge\mathrm{I}}{\vdash A} \wedge\mathrm{E_L} \quad \Longrightarrow_R \quad \begin{array}{c}\mathcal{D}\\ \vdash A\end{array}$$

and symmetrically

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \vdash A\end{array} \qquad \begin{array}{c}\mathcal{E}\\ \vdash B\end{array}}{\vdash A \wedge B} \wedge\mathrm{I}}{\vdash B} \wedge\mathrm{E_R} \quad \Longrightarrow_R \quad \begin{array}{c}\mathcal{E}\\ \vdash B\end{array}$$

The new judgment

$$\begin{array}{c}\mathcal{D}\\ \vdash A\end{array} \quad \Longrightarrow_R \quad \begin{array}{c}\mathcal{E}\\ \vdash A\end{array}$$

relates derivations with the same conclusion. We say $\mathcal{D}$ *locally reduces to* $\mathcal{E}$. Since local reductions are possible for both elimination rules for conjunction, our rules are locally sound. To show that the rules are locally complete we show how to reintroduce a conjunction from its components in the form of a local expansion.

$$\begin{array}{c}\mathcal{D}\\ \vdash A \wedge B\end{array} \quad \Longrightarrow_E \quad \dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \vdash A \wedge B\end{array}}{\vdash A}\wedge\mathrm{E_L} \qquad \dfrac{\begin{array}{c}\mathcal{D}\\ \vdash A \wedge B\end{array}}{\vdash B}\wedge\mathrm{E_R}}{\vdash A \wedge B}\wedge\mathrm{I}$$

Implication. To derive $\vdash A \supset B$ we assume $\vdash A$ and then derive $\vdash B$. Written as a hypothetical judgment:

$$\dfrac{\begin{array}{c}\overline{\vdash A}\,u\\ \vdots\\ \vdash B\end{array}}{\vdash A \supset B}\supset\mathrm{I}^u$$

We must be careful that the hypothesis $\vdash A$ is available only in the derivation above the premiss. We therefore label the inference with the name of the hypothesis u, which must not be used already as the name for a hypothesis in the derivation of the premiss. We say that the hypothesis $\vdash A$ labelled u is *discharged* at the inference labelled $\supset\mathrm{I}^u$. A derivation of $\vdash A \supset B$ describes a construction by which we can transform a derivation of $\vdash A$ into a derivation of $\vdash B$: we substitute the derivation of $\vdash A$ wherever we used the assumption $\vdash A$ in the hypothetical derivation of $\vdash B$. The elimination rule expresses this: if we have a derivation of $\vdash A \supset B$ and also a derivation of $\vdash A$, then we can obtain a derivation of $\vdash B$.

$$\dfrac{\vdash A \supset B \qquad \vdash A}{\vdash B}\supset\mathrm{E}$$

The local reduction rule carries out the substitution of derivations explained above.

$$\dfrac{\dfrac{\begin{array}{c}\overline{\vdash A}\,u\\ \mathcal{D}\\ \vdash B\end{array}}{\vdash A \supset B}\supset\mathrm{I}^u \qquad \begin{array}{c}\mathcal{E}\\ \vdash A\end{array}}{\vdash B}\supset\mathrm{E} \quad \Longrightarrow_R \quad \begin{array}{c}\dfrac{\mathcal{E}}{\vdash A}\,u\\ \mathcal{D}\\ \vdash B\end{array}$$

The final derivation depends on all the hypotheses of $\mathcal{E}$ and $\mathcal{D}$ except u, for which we have substituted $\mathcal{E}$. An alternative notation for this substitution of derivations for hypotheses is $[\mathcal{E}/u]\mathcal{D} :: \ \vdash B$. The local reduction described above may significantly increase the overall size of the derivation, since the deduction $\mathcal{E}$ is substituted for each occurrence of the assumption labeled u in $\mathcal{D}$ and may thus be replicated many times. The local expansion simply rebuilds the implication.

$$\begin{array}{c}\mathcal{D}\\ \vdash A \supset B\end{array} \quad \Longrightarrow_E \quad \dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \vdash A \supset B\end{array} \qquad \dfrac{}{\vdash A}u}{\vdash B}\supset\mathrm{E}}{\vdash A \supset B}\supset\mathrm{I}^u$$

Disjunction. $A \vee B$ should be true if either A is true or B is true. Therefore we have two introduction rules.

$$\dfrac{\vdash A}{\vdash A \vee B}\vee\mathrm{I_L} \qquad \dfrac{\vdash B}{\vdash A \vee B}\vee\mathrm{I_R}$$

If we have a hypothesis $\vdash A \vee B$, we do not know how it might be inferred. That is, a proposed elimination rule

$$\dfrac{\vdash A \vee B}{\vdash A}?$$

would be incorrect, since a deduction of the form

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{E}\\ \vdash B\end{array}}{\vdash A \vee B}\vee\mathrm{I_R}}{\vdash A}?$$

cannot be reduced. As a consequence, the system would be *inconsistent*: if we have at least one theorem (B, in the example) we can prove every formula (A, in the example). How do we use the assumption $A \vee B$ in informal reasoning? We often proceed with a proof by cases: we prove a conclusion C under the assumption A and also show C under the assumption B. We then conclude C, since either A or B by assumption. Thus the elimination rule employs two hypothetical judgments.

$$\dfrac{\vdash A \vee B \qquad \begin{array}{c}\dfrac{}{\vdash A}u_1\\ \vdots\\ \vdash C\end{array} \qquad \begin{array}{c}\dfrac{}{\vdash B}u_2\\ \vdots\\ \vdash C\end{array}}{\vdash C}\vee\mathrm{E}^{u_1,u_2}$$

Now one can see that the introduction and elimination rules match up in two reductions. First, the case that the disjunction was inferred by $\lor\mathrm{I_L}$.

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \vdash A\end{array}}{\vdash A \lor B}\lor\mathrm{I_L} \qquad \begin{array}{c}\overline{\vdash A}\,u_1\\ \mathcal{E}_1\\ \vdash C\end{array} \qquad \begin{array}{c}\overline{\vdash B}\,u_2\\ \mathcal{E}_2\\ \vdash C\end{array}}{\vdash C}\lor\mathrm{E}^{u_1,u_2} \qquad \Longrightarrow_R \qquad \begin{array}{c}\dfrac{\mathcal{D}}{\vdash A}\,u_1\\ \mathcal{E}_1\\ \vdash C\end{array}$$

The other reduction is symmetric.

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \vdash B\end{array}}{\vdash A \lor B}\lor\mathrm{I_R} \qquad \begin{array}{c}\overline{\vdash A}\,u_1\\ \mathcal{E}_1\\ \vdash C\end{array} \qquad \begin{array}{c}\overline{\vdash B}\,u_2\\ \mathcal{E}_2\\ \vdash C\end{array}}{\vdash C}\lor\mathrm{E}^{u_1,u_2} \qquad \Longrightarrow_R \qquad \begin{array}{c}\dfrac{\mathcal{D}}{\vdash B}\,u_2\\ \mathcal{E}_2\\ \vdash C\end{array}$$

As in the reduction for implication, the resulting derivation may be longer than the original one. The local expansion is more complicated than for the previous connectives, since we first have to distinguish cases and then reintroduce the disjunction in each branch.

$$\begin{array}{c}\mathcal{D}\\ \vdash A \lor B\end{array} \qquad \Longrightarrow_E \qquad \dfrac{\begin{array}{c}\mathcal{D}\\ \vdash A \lor B\end{array} \qquad \dfrac{\overline{\vdash A}\,u_1}{\vdash A \lor B}\lor\mathrm{I_L} \qquad \dfrac{\overline{\vdash B}\,u_2}{\vdash A \lor B}\lor\mathrm{I_R}}{\vdash A \lor B}\lor\mathrm{E}^{u_1,u_2}$$

Negation. In order to derive $\neg A$ we assume A and try to derive a contradiction. Thus it seems that negation requires falsehood, and, indeed, in most literature on constructive logic, $\neg A$ is seen as an abbreviation of $A \supset \bot$. In order to give a self-contained explanation of negation by an introduction rule, we employ a judgment that is parametric in a propositional parameter p: If we can derive *any* p from the hypothesis A we conclude $\neg A$.

$$\dfrac{\begin{array}{c}\overline{\vdash A}\,u\\ \vdots\\ \vdash p\end{array}}{\vdash \neg A}\neg\mathrm{I}^{p,u} \qquad\qquad \dfrac{\vdash \neg A \qquad \vdash A}{\vdash C}\neg\mathrm{E}$$

The elimination rule follows from this view: if we know $\vdash \neg A$ and $\vdash A$ then we can conclude any formula C is true. In the form of a local reduction:

$$\dfrac{\dfrac{\begin{array}{c}\overline{\vdash A}\,u\\ \mathcal{D}\\ \vdash p\end{array}}{\vdash \neg A}\,\neg\mathrm{I}^{p,u} \qquad \begin{array}{c}\mathcal{E}\\ \vdash A\end{array}}{\vdash C}\,\neg\mathrm{E} \quad \Longrightarrow_R \quad \begin{array}{c}\dfrac{\mathcal{E}}{\vdash A}\,u\\ [C/p]\mathcal{D}\\ \vdash C\end{array}$$

The substitution $[C/p]\mathcal{D}$ is valid, since $\mathcal{D}$ is parametric in p. The local expansion is similar to the case for implication.

$$\begin{array}{c}\mathcal{D}\\ \vdash \neg A\end{array} \quad \Longrightarrow_E \quad \dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \vdash \neg A\end{array} \qquad \overline{\vdash A}\,u}{\vdash p}\,\neg\mathrm{E}}{\vdash \neg A}\,\neg\mathrm{I}^{p,u}$$

Truth. There is only an introduction rule for $\top$:

$$\dfrac{}{\vdash \top}\,\top\mathrm{I}$$

Since we put no information into the proof of $\top$, we know nothing new if we have an assumption $\top$ and therefore we have no elimination rule and no local reduction. It may also be helpful to think of $\top$ as a 0-ary conjunction: the introduction rule has 0 premisses instead of 2 and we correspondingly have 0 elimination rules instead of 2. The local expansion allows the replacement of any derivation of $\top$ by $\top$I.

$$\begin{array}{c}\mathcal{D}\\ \vdash \top\end{array} \quad \Longrightarrow_E \quad \dfrac{}{\vdash \top}\,\top\mathrm{I}$$

Falsehood. Since we should not be able to derive falsehood, there is no introduction rule for $\bot$. Therefore, if we can derive falsehood, we can derive everything.

$$\dfrac{\vdash \bot}{\vdash C}\,\bot\mathrm{E}$$

Note that there is no local reduction rule for $\bot$E. It may be helpful to think of $\bot$ as a 0-ary disjunction: we have 0 instead of 2 introduction rules and we correspondingly have to consider 0 cases instead of 2 in the elimination rule. Even though we postulated that falsehood should not be derivable, falsehood could clearly be a consequence of contradictory assumption. For example, $\vdash$

$A \wedge \neg A \supset \bot$ is derivable. While there is no local reduction rule, there still is a local expansion in analogy to the case for disjunction.

$$\begin{array}{c}\mathcal{D}\\ \vdash \bot\end{array} \quad \Longrightarrow_E \quad \dfrac{\begin{array}{c}\mathcal{D}\\ \vdash \bot\end{array}}{\vdash \bot}\bot\mathrm{E}$$

Universal Quantification. Under which circumstances should $\vdash \forall x.\ A$ be true? This clearly depends on the domain of quantification. For example, if we know that x ranges over the natural numbers, then we can conclude $\forall x.\ A$ if we can prove $[0/x]A$, $[1/x]A$, *etc.* Such a rule is not effective, since it has infinitely many premisses. Thus one usually retreats to rules such as induction. However, in a general treatment of predicate logic we would like to prove statements which are true for *all* domains of quantification. Thus we can only say that $\forall x.\ A$ should be provable if $[a/x]A$ is provable for a new parameter a about which we can make no assumption. Conversely, if we know $\forall x.\ A$, we know that $[t/x]A$ for any term t.

$$\dfrac{\vdash [a/x]A}{\vdash \forall x.\ A}\forall\mathrm{I}^a \qquad\qquad \dfrac{\vdash \forall x.\ A}{\vdash [t/x]A}\forall\mathrm{E}$$

The label a on the introduction rule is a reminder the parameter a must be "new", that is, it may not occur in any uncancelled assumption in the proof of $[a/x]A$ or in $\forall x.\ A$ itself. In other words, the derivation of the premiss must parametric in a. The local reduction carries out the substitution for the parameter.

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \vdash [a/x]A\end{array}}{\vdash \forall x.\ A}\forall\mathrm{I}}{\vdash [t/x]A}\forall\mathrm{E} \quad \Longrightarrow_R \quad \begin{array}{c}[t/a]\mathcal{D}\\ \vdash [t/x]A\end{array}$$

Here, $[t/a]\mathcal{D}$ is our notation for the result of substituting t for the parameter a throughout the deduction $\mathcal{D}$. For this substitution to preserve the conclusion, we must know that a does not already occur in A. Similarly, we would change the hypotheses if a occurred free in any of the undischarged hypotheses of $\mathcal{D}$. This might render a larger proof incorrect. As an example, consider the formula $\forall x. \forall y.\ P(x) \supset P(y)$ which should clearly not be true for all predicates P. The following is *not* a deduction of this formula.

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{\vdash P(a)}\,u}{\vdash \forall x.\, P(x)}\,\forall\mathrm{I}^a?}{\vdash P(b)}\,\forall\mathrm{E}}{\vdash P(a) \supset P(b)}\,{\supset}\mathrm{I}^u}{\vdash \forall y.\, P(a) \supset P(y)}\,\forall\mathrm{I}^b}{\vdash \forall x.\, \forall y.\, P(x) \supset P(y)}\,\forall\mathrm{I}^a$$

The flaw is at the inference marked with "?," where a is free in the hypothesis labelled u. Applying a local proof reduction to the (incorrect) $\forall$I inference followed by $\forall$E leads to the the assumption $[b/a]P(a)$ which is equal to $P(b)$. The resulting derivation

$$\cfrac{\cfrac{\cfrac{\cfrac{}{\vdash P(b)}\,u}{\vdash P(a) \supset P(b)}\,{\supset}\mathrm{I}^u}{\vdash \forall y.\, P(a) \supset P(y)}\,\forall\mathrm{I}^b}{\vdash \forall x.\, \forall y.\, P(x) \supset P(y)}\,\forall\mathrm{I}^a$$

is once again incorrect since the hypothesis labelled u should read $P(a)$, not $P(b)$.

The local expansion for universal quantification is much simpler.

$$\begin{array}{c} \mathcal{D} \\ \vdash \forall x.\, A \end{array} \quad \Longrightarrow_E \quad \cfrac{\cfrac{\begin{array}{c}\mathcal{D} \\ \vdash \forall x.\, A\end{array}}{\vdash [a/x]A}\,\forall\mathrm{E}}{\vdash \forall x.\, A}\,\forall\mathrm{I}^a$$

Existential Quantification. We conclude that $\exists x.\, A$ is true when there is a term t such that $[t/x]A$ is true.

$$\cfrac{\vdash [t/x]A}{\vdash \exists x.\, A}\,\exists\mathrm{I}$$

When we have an assumption $\exists x.\, A$ we do not know for which t it is the case that $[t/x]A$ holds. We can only assume that $[a/x]A$ holds for some parameter a about which we know nothing else. Thus the elimination rule resembles the one for disjunction.

$$\dfrac{\vdash \exists x.\, A \qquad \begin{array}{c}\overline{\vdash [a/x]A}\,u\\ \vdots \\ \vdash C\end{array}}{\vdash C}\,\exists\mathrm{E}^{a,u}$$

The restriction is similar to the one for $\forall$I: the parameter a must be new, that is, it must not occur in $\exists x.\, A$, C, or any assumption employed in the derivation of the second premiss. In the reduction rule we have to perform two substitutions: we have to substitute t for the parameter a and we also have to substitute for the hypothesis labelled u.

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \vdash [t/x]A\end{array}}{\exists x.\, A}\,\exists\mathrm{I} \qquad \begin{array}{c}\overline{\vdash [a/x]A}\,u\\ \mathcal{E} \\ \vdash C\end{array}}{\vdash C}\,\exists\mathrm{E}^{a,u} \quad \Longrightarrow_R \quad \begin{array}{c}\dfrac{\mathcal{D}}{\vdash [t/x]A}\,u\\ {[t/a]}\mathcal{E} \\ \vdash C\end{array}$$

The proviso on occurrences of a guarantees that the conclusion and hypotheses of $[t/a]\mathcal{E}$ have the correct form. The local expansion for existential quantification is also similar to the case for disjunction.

$$\begin{array}{c}\mathcal{D}\\ \vdash \exists x.\, A\end{array} \quad \Longrightarrow_E \quad \dfrac{\begin{array}{c}\mathcal{D}\\ \vdash \exists x.\, A\end{array} \qquad \dfrac{\overline{\vdash [a/x]A}\,u}{\vdash \exists x.\, A}\,\exists\mathrm{I}}{\vdash \exists x.\, A}\,\exists\mathrm{E}^{a,u}$$

Here is a simple example of a natural deduction. We attempt to show the process by which such a deduction may have been generated, as well as the final deduction. The three vertical dots indicate a gap in the derivation we are trying to construct, with hypotheses and their consequences shown above and the desired conclusion below the gap.

$$\begin{array}{c}\vdots\\ \vdash A \wedge (A \supset B) \supset B\end{array} \quad \leadsto \quad \dfrac{\begin{array}{c}\overline{\vdash A \wedge (A \supset B)}\,u\\ \vdots \\ \vdash B\end{array}}{\vdash A \wedge (A \supset B) \supset B}\,{\supset}\mathrm{I}^u$$

$$\leadsto \quad \dfrac{\begin{array}{c}\dfrac{\dfrac{}{\vdash A \wedge (A \supset B)}\,u}{\vdash A}\,\wedge\mathrm{E_L} \\ \vdots \\ \vdash B\end{array}}{\vdash A \wedge (A \supset B) \supset B}\,\supset\!\mathrm{I}^u \qquad \leadsto \quad \dfrac{\begin{array}{c}\dfrac{\dfrac{}{\vdash A \wedge (A \supset B)}\,u}{\vdash A}\,\wedge\mathrm{E_L} \qquad \dfrac{\dfrac{}{\vdash A \wedge (A \supset B)}\,u}{\vdash A \supset B}\,\wedge\mathrm{E_R} \\ \vdots \\ \vdash B\end{array}}{\vdash A \wedge (A \supset B) \supset B}\,\supset\!\mathrm{I}^u$$

$$\leadsto \quad \dfrac{\begin{array}{c}\dfrac{\dfrac{\dfrac{}{\vdash A \wedge (A \supset B)}\,u}{\vdash A \supset B}\,\wedge\mathrm{E_R} \qquad \dfrac{\dfrac{}{\vdash A \wedge (A \supset B)}\,u}{\vdash A}\,\wedge\mathrm{E_L}}{\vdash B}\,\supset\!\mathrm{E} \\ \vdots \\ \vdash B\end{array}}{\vdash A \wedge (A \supset B) \supset B}\,\supset\!\mathrm{I}^u$$

$$\leadsto \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A \wedge (A \supset B)}\,u}{\vdash A \supset B}\,\wedge\mathrm{E_R} \qquad \dfrac{\dfrac{}{\vdash A \wedge (A \supset B)}\,u}{\vdash A}\,\wedge\mathrm{E_L}}{\vdash B}\,\supset\!\mathrm{E}}{\vdash A \wedge (A \supset B) \supset B}\,\supset\!\mathrm{I}^u$$

The symbols A and B in this derivation stand for arbitrary propositions; we can thus established a judgment parametric in A and B. In other words, every instance of this derivation (substituting arbitrary propositions for A and B) is a valid derivation.

Below is a summary of the rules of intuitionistic natural deduction.

Introduction Rules

$$\dfrac{\vdash A \qquad \vdash B}{\vdash A \wedge B}\,\wedge\mathrm{I}$$

$$\dfrac{\vdash A}{\vdash A \vee B}\,\vee\mathrm{I}_{\mathrm{L}} \qquad \dfrac{\vdash B}{\vdash A \vee B}\,\vee\mathrm{I}_{\mathrm{R}}$$

$$\dfrac{\begin{array}{c}\overline{\vdash A}\,u\\ \vdots\\ \vdash B\end{array}}{\vdash A \supset B}\,{\supset}\mathrm{I}^{u}$$

$$\dfrac{\begin{array}{c}\overline{\vdash A}\,u\\ \vdots\\ \vdash p\end{array}}{\vdash \neg A}\,\neg\mathrm{I}^{p,u}$$

$$\dfrac{}{\vdash \top}\,\top\mathrm{I}$$

no $\bot$ introduction

$$\dfrac{\vdash [a/x]A}{\vdash \forall x.\ A}\,\forall\mathrm{I}^{a}$$

$$\dfrac{\vdash [t/x]A}{\vdash \exists x.\ A}\,\exists\mathrm{I}$$

Elimination Rules

$$\dfrac{\vdash A \wedge B}{\vdash A}\,\wedge\mathrm{E}_{\mathrm{L}} \qquad \dfrac{\vdash A \wedge B}{\vdash B}\,\wedge\mathrm{E}_{\mathrm{R}}$$

$$\dfrac{\vdash A \vee B \qquad \begin{array}{c}\overline{\vdash A}\,u_1\\ \vdots\\ \vdash C\end{array} \qquad \begin{array}{c}\overline{\vdash B}\,u_2\\ \vdots\\ \vdash C\end{array}}{\vdash C}\,\vee\mathrm{E}^{u_1,u_2}$$

$$\dfrac{\vdash A \supset B \qquad \vdash A}{\vdash B}\,{\supset}\mathrm{E}$$

$$\dfrac{\vdash A \qquad \vdash \neg A}{\vdash C}\,\neg\mathrm{E}$$

no $\top$ elimination

$$\dfrac{\vdash \bot}{\vdash C}\,\bot\mathrm{E}$$

$$\dfrac{\vdash \forall x.\ A}{\vdash [t/x]A}\,\forall\mathrm{E}$$

$$\dfrac{\vdash \exists x.\ A \qquad \begin{array}{c}\overline{\vdash [a/x]A}\,u\\ \vdots\\ \vdash C\end{array}}{\vdash C}\,\exists\mathrm{E}^{a,u}$$

1.2 Classical Logic

The inference rules so far only model *intuitionistic logic*, and some classically true propositions such as $A \vee \neg A$ (for an arbitrary A) are not derivable, as we will see in Section **??**. There are three commonly used ways one can construct a system of *classical natural deduction* by adding one additional rule of inference. $\bot_C$ is called *Proof by Contradiction* or *Rule of Indirect Proof*, $\neg\neg_C$ is the *Double Negation Rule*, and XM is referred to as *Excluded Middle*.

$$\dfrac{\dfrac{}{\neg A}u \\ \vdots \\ \bot}{A}\bot_C^u \qquad \dfrac{\neg\neg A}{A}\neg\neg_C \qquad \dfrac{}{A \vee \neg A}\text{XM}$$

The rule for classical logic (whichever one chooses to adopt) breaks the pattern of introduction and elimination rules. One can still formulate some reductions for classical inferences, but natural deduction is at heart an intuitionistic calculus. The symmetries of classical logic are much better exhibited in sequent formulations of the logic. In Exercise 1.3 we explore the three ways of extending the intuitionistic proof system and show that they are equivalent.

Another way to obtain a natural deduction system for classical logic is to allow multiple conclusions (see, for example, Parigot [Par92]).

1.3 Localizing Hypotheses

In the formulation of natural from Section 1.1 correct use of hypotheses and parameters is a global property of a derivation. We can localize it by annotating each judgment in a derivation by the available parameters and hypotheses. Since hypotheses and their restrictions are critical for linear logic, we give here a formulation of natural deduction for intuitionistic logic with localized hypotheses, but not parameters. For this we need a notation for hypotheses which we call a *context*.

$$\textit{Contexts} \quad \Gamma \quad ::= \quad \cdot \mid \Gamma, u{:}A$$

Here, "$\cdot$" represents the empty context, and $\Gamma, u{:}A$ adds hypothesis $\vdash A$ labelled u to Γ. We assume that each label u occurs at most once in a context in order to avoid ambiguities. The main judgment can then be written as $\Gamma \vdash A$, where

$$\cdot, u_1{:}A_1, \ldots, u_n{:}A_n \vdash A$$

stands for

$$\dfrac{}{\vdash A_1}u_1 \quad \ldots \dfrac{}{\vdash A_n}u_n \\ \vdots \\ \vdash A$$

in the notation of Section 1.1.

We use a few important abbreviations in order to make this notation less cumbersome. First of all, we may omit the leading "$\cdot$" and write, for example, $u_1{:}A_1, u_2{:}A_2$ instead of $\cdot, u_1{:}A_1, u_2{:}A_2$. Secondly, we denote concatenation of contexts by overloading the comma operator as follows.

$$\begin{array}{rcl} \Gamma, \cdot & = & \Gamma \\ \Gamma, (\Gamma', u{:}A) & = & (\Gamma, \Gamma'), u{:}A \end{array}$$

With these additional definitions, the localized version of our rules are as follows.

Introduction Rules

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\wedge\mathrm{I}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\vee\mathrm{I_L} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\vee\mathrm{I_R}$$

$$\frac{\Gamma, u{:}A \vdash B}{\Gamma \vdash A \supset B}\supset\mathrm{I}^u$$

$$\frac{\Gamma, u{:}A \vdash p}{\Gamma \vdash \neg A}\neg\mathrm{I}^{p,u}$$

$$\frac{}{\Gamma \vdash \top}\top\mathrm{I}$$

no $\perp$ *introduction*

$$\frac{\Gamma \vdash [a/x]A}{\Gamma \vdash \forall x.\ A}\forall\mathrm{I}^a$$

$$\frac{\Gamma \vdash [t/x]A}{\Gamma \vdash \exists x.\ A}\exists\mathrm{I}$$

Elimination Rules

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\wedge\mathrm{E_L} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\wedge\mathrm{E_R}$$

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, u_1{:}A \vdash C \qquad \Gamma, u_2{:}B \vdash C}{\Gamma \vdash C}\vee\mathrm{E}^{u_1,u_2}$$

$$\frac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B}\supset\mathrm{E}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash \neg A}{\Gamma \vdash C}\neg\mathrm{E}$$

no $\top$ *elimination*

$$\frac{\Gamma \vdash \perp}{\Gamma \vdash C}\perp\mathrm{E}$$

$$\frac{\Gamma \vdash \forall x.\ A}{\Gamma \vdash [t/x]A}\forall\mathrm{E}$$

$$\frac{\Gamma \vdash \exists x.\ A \qquad \Gamma, u{:}[a/x]A \vdash C}{\Gamma \vdash C}\exists\mathrm{E}^{a,u}$$

We also have a new rule for hypotheses which was an implicit property of the hypothetical judgments before.

$$\frac{}{\Gamma_1, u{:}A, \Gamma_2 \vdash A}u$$

Other general assumptions about hypotheses, namely that they may be used arbitrarily often in a derivation and that their order does not matter, are indirectly reflected in these rules. Note that if we erase the context Γ from the judgments throughout a derivation, we obtain a derivation in the original notation.

When we discussed local reductions in order to establish local soundness, we used the notation

$$\begin{array}{c} \dfrac{\mathcal{D}}{\vdash A}u \\ \mathcal{E} \\ \vdash C \end{array}$$

for the result of substituting the derivation $\mathcal{D}$ of $\vdash A$ for all uses of the hypothesis $\vdash A$ labelled u in $\mathcal{E}$. We would now like to reformulate the property with localized hypotheses. In order to prove that the (now explicit) hypotheses behave as expected, we use the principle of *structural induction* over derivations. Simply put, we prove a property for all derivations by showing that, whenever it holds for the premisses of an inference, it holds for the conclusion. Note that we have to show the property outright when the rule under consideration has no premisses, which amounts to the base cases forof the induction.

Theorem 1.1 (Structural Properties of Hypotheses) *The following properties hold for intuitionistic natural deduction.*

1. *(Exchange) If* $\Gamma_1, u_1{:}A, \Gamma_2, u_2{:}B, \Gamma_2 \vdash C$ *then* $\Gamma_1, u_2{:}B, \Gamma_2, u_1{:}A, \Gamma_2 \vdash C$.
2. *(Weakening) If* $\Gamma_1, \Gamma_2 \vdash C$ *then* $\Gamma_1, u{:}A, \Gamma_2 \vdash C$.
3. *(Contraction) If* $\Gamma_1, u_1{:}A, \Gamma_2, u_2{:}A, \Gamma_2 \vdash C$ *then* $\Gamma_1, u{:}A, \Gamma_2, \Gamma_3 \vdash C$.
4. *(Substitution) If* $\Gamma_1, u{:}A, \Gamma_2 \vdash C$ *and* $\Gamma_1 \vdash A$ *then* $\Gamma_1, \Gamma_2 \vdash C$.

Proof: The proof is in each case by straightforward induction over the structure of the first given derivation.

In the case of exchange, we appeal to the inductive assumption on the derivations of the premisses and construct a new derivation with the same inference rule. Algorithmically, this means that we exchange the hypotheses labelled u_1 and u_2 in every judgment in the derivation.

In the case of weakening and contraction, we proceed similarly, either adding the new hypothesis $u{:}A$ to every judgment in the derivation (for weakening), or replacing uses of u_1 and u_2 by u (for contraction).

For substitution, we apply the inductive assumption to the premisses of the given derivation $\mathcal{D}$ until we reach hypotheses. If the hypothesis is different from u we can simply erase $u{:}A$ (which is unused) to obtain the desired derivation. If the hypothesis is $u{:}A$ the derivation looks like

$$\mathcal{D} = \dfrac{}{\Gamma_1, u{:}A, \Gamma_2 \vdash A}u$$

so $C = A$ in this case. We are also given a derivation $\mathcal{E}$ of $\Gamma_1 \vdash A$ and have to construct a derivation $\mathcal{F}$ of $\Gamma_1, \Gamma_2 \vdash A$. But we can just repeatedly apply weakening to $\mathcal{E}$ to obtain $\mathcal{F}$. Algorithmically, this means that, as expected, we substitute the derivation $\mathcal{E}$ (possibly weakened) for uses of the hypotheses $u{:}A$ in $\mathcal{D}$. Note that in our original notation, this weakening has no impact, since unused hypotheses are not apparent in a derivation. □

It is also possible to localize the derivations themselves, using *proof terms*. As we will see in Chapter 4, these proof terms form a λ-calculus closely related to functional programming. When parameters, hypotheses, and proof terms are all localized our main judgment becomes decidable. In the terminology of Martin-Löf [ML94], the main judgment is then *analytic* rather than *synthetic*. We no longer need to go outside the judgment itself in order to collect evidence for it: An analytic judgment encapsulates its own evidence.

1.4 Exercises

Exercise 1.1 Prove the following by natural deduction using only intuitionistic rules when possible. We use the convention that $\supset$, $\wedge$, and $\vee$ associate to the right, that is, $A{\supset}B{\supset}C$ stands for $A{\supset}(B{\supset}C)$. $A \equiv B$ is a syntactic abbreviation for $(A \supset B) \wedge (B \supset A)$. Also, we assume that $\wedge$ and $\vee$ bind more tightly than $\supset$, that is, $A \wedge B \supset C$ stands for $(A \wedge B) \supset C$. The scope of a quantifier extends as far to the right as consistent with the present parentheses. For example, $(\forall x.\ P(x) \supset C) \wedge \neg C$ would be disambiguated to $(\forall x.\ (P(x) \supset C)) \wedge (\neg C)$.

1. $\vdash A \supset B \supset A$.
2. $\vdash A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$.
3. (Peirce's Law). $\vdash ((A \supset B) \supset A) \supset A$.
4. $\vdash A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$.
5. $\vdash A \supset (A \wedge B) \vee (A \wedge \neg B)$.
6. $\vdash (A \supset \exists x.\ P(x)) \equiv \exists x.\ (A \supset P(x))$.
7. $\vdash ((\forall x.\ P(x)) \supset C) \equiv \exists x.\ (P(x) \supset C)$.
8. $\vdash \exists x.\ \forall y.\ (P(x) \supset P(y))$.

Exercise 1.2 We write $A \vdash B$ if B follows from hypothesis A and $A \dashv\vdash B$ for $A \vdash B$ and $B \vdash A$. Which of the following eight parametric judgments are derivable intuitionistically?

1. $(\exists x.\ A) \supset B \dashv\vdash \forall x.\ (A \supset B)$
2. $A \supset (\exists x.\ B) \dashv\vdash \exists x.\ (A \supset B)$
3. $(\forall x.\ A) \supset B \dashv\vdash \exists x.\ (A \supset B)$
4. $A \supset (\forall x.\ B) \dashv\vdash \forall x.\ (A \supset B)$

Provide natural deductions for the valid judgments. You may assume that the bound variable x does not occur in B (items 1 and 3) or A (items 2 and 4).

Exercise 1.3 Show that the three ways of extending the intuitionistic proof system are equivalent, that is, the same formulas are deducible in all three systems.

Exercise 1.4 Assume we had omitted disjunction and existential quantification and their introduction and elimination rules from the list of logical primitives. In the classical system, give a definition of disjunction and existential quantification (in terms of other logical constants) and show that the introduction and elimination rules now become *admissible rules of inference*. A rule of inference is *admissible* if any deduction using the rule can be transformed into one without using the rule.

Exercise 1.5 Assume we would like to design a system of natural deduction for a simple temporal logic. The main judgment is now "A *is true at time* t" written as

$$\vdash^t A.$$

1. Explain how to modify the given rules for natural deduction to this more general judgment and show the rules for implication and universal quantification.
2. Write out introduction and elimination rules for the temporal operator $\bigcirc A$ which should be true if A is true at the next point in time. Denote the "next time after t" by $t + 1$.
3. Show the local reductions and expansions which show the local soundness and completness of your rules.
4. Write out introduction and elimination rules for the temporal operator $\Box A$ which should be true if A is true at all times.
5. Show the local reductions and expansions.

Exercise 1.6 Design introduction and elimination rules for the connectives

1. $A \equiv B$, usually defined as $(A \supset B) \wedge (B \supset A)$,
2. $A \mid B$ (exclusive or), usually defined as $(A \wedge \neg B) \vee (\neg A \wedge B)$,

without recourse to other logical constants or operators. Also show the corresponding local reductions and expansions.

Chapter 2

Intuitionistic Linear Logic

Linear logic, in its original formulation by Girard [Gir87] and many subsequent investigations was presented as a refinement of classical logic. This calculus of *classical linear logic* can be cleanly related to classical logic and exhibits many pleasant symmetries. On the other hand, a number of applications in logic and functional programming can be treated most directly using the intuitionistic version. In this chapter we present the basic system of natural deduction defining intuitionistic linear logic. Further surveys and introductions to linear logic can be found in [Lin92, Sce93, Tro92]. A historical introduction [Došen93] and context for linear and other substructural logics outside computer science can be found in [SHD93].

We introduce linear logic by enriching our judgment forms by a *linear hypothetical judgment*. A linear hypothetical judgment has the form "J_2 provided resource J_1". We consider this judgment evident if we are prepared to make judgment J_2 when provided with the resource J_1. A *resource* or *linear hypothesis* behaves like an ordinary hypothesis except that it must be used exactly once.

As an example, consider the basic judgment "*I own* X" for objects X. We would be prepared to make the linear hypothetical judgment

"*I own book* b provided *I own \$5*"

if we know that book b costs five dollars and that it is available. If we ever actually had \$5, we could then achieve a situation in which we owned the book. Obviously, we would no longer own the five dollars, since they would have been consumed in the process of obtaining the book. It is clear that we would not be prepared to make the judgment above if the book costs ten dollars due to insufficient resources. But we would reject the judgment even if the book cost only one dollar, since we would not use all the given resources.

We can already see that evidence for a judgment of this form is a derivation in which we have to keep track of resources. Implicit here is a notion of state and change of state which is not present in traditional mathematical logic. For this reason, linear logic is often referred to as a "logic of state".

In the following section we develop the logical connectives of linear logic, based on the notion of linear hypothetical judgment.

2.1 Purely Linear Natural Deduction

The main judgment of *purely linear natural deduction* is "A is true assuming linear hypotheses $A_1, \ldots, A_n$". Later we also admit unrestricted hypotheses, but we postpone this complication. We refer to $A_1, \ldots A_n$ as *resources* and A as the *goal* to be achieved. As for hypotheses in Section 1.3 we localize resources into a context Δ and write

$$\Delta \vdash A.$$

Just as for ordinary hypotheses, we can substitute concrete evidence for a linear hypothesis for its use in a derivation. The corresponding *substitution principle* is the following:

If $\Delta_1, w{:}A, \Delta_2 \vdash C$ and $\Delta \vdash A$ then $\Delta_1, \Delta, \Delta_2 \vdash C$.

Intuitively, it states that if we have a derivation of $\vdash C$ from resources Δ_1, Δ_2 and the additional resource $\vdash A$ labelled w, and if we have a derivation of $\vdash A$ requiring resources Δ, then we can obtain a derivation of $\vdash C$ from resources Δ_1, Δ, and Δ_2.

Resources also satisfy the principle of exchange, since their order is irrelevant.

If $\Delta_1, w_1{:}A, \Delta_2, w_2{:}B, \Delta_3 \vdash C$ then $\Delta_1, w_2{:}B, \Delta_2, w_1{:}A, \Delta_3 \vdash C$.

Note that unlike unrestricted hypotheses, resources cannot be weakened or contracted since they must be used exactly once. Weakening would allow resources to remain unused, while contraction would allow resources to be used more than once.

Finally, the use of a resource is restricted (when compared to hypotheses in the intuitionistic case) so that there are no other resources remaining.

$$\frac{}{\cdot, w{:}A \vdash A}\, w$$

We now examine, connective by connective, the introduction and elimination rules of linear logic and check their local soundness and completeness. In the local reductions and expansion we will have to carefully check the preservation of resources.

Simultaneous Conjunction. Assume we have some resources Δ and we want to achieve goals A and B. We then need to split our resources Δ into Δ_1 and Δ_2 and show that with resources Δ_1 we can achieve A and with Δ_2 we can achieve B. The introduction rule for the corresponding connective of *simultaneous conjunction*, written $A \otimes B$ and read "A *tensor* B", then requires a notation for splitting resources (when viewed bottom-up) or merging resources (when viewed top-down). We merge $\Delta_1 \times \Delta_2$ according to the following rules.

$$\begin{array}{rcl} \cdot \times \cdot & = & \cdot \\ \Delta_1 \times (\Delta_2, u{:}A) & = & (\Delta_1 \times \Delta_2), u{:}A \\ (\Delta_1, u{:}A) \times \Delta_2 & = & (\Delta_1 \times \Delta_2), u{:}A \end{array}$$

Note that this is a non-deterministic operation, since the last two rules might both be applicable. We use the convention that any way to merge two contexts in an inference rules yields a valid inference. With this preparation we can now state the introduction rule for simultaneous conjunction.

$$\frac{\Delta_1 \vdash A \qquad \Delta_2 \vdash B}{\Delta_1 \times \Delta_2 \vdash A \otimes B}\otimes\mathrm{I}$$

The elimination rule should capture what we can achieve if we know that we can achieve both A and B simultaneously from some hypothetical resources Δ. We reason as follows: If with A, B, and additional resources Δ' we could achieve goal C, then we could achieve C from resources Δ and Δ'.

$$\frac{\Delta \vdash A \otimes B \qquad \Delta', w_1{:}A, w_2{:}B \vdash C}{\Delta' \times \Delta \vdash C}\otimes\mathrm{E}^{w_1,w_2}$$

The way we achieve C is to commit resources Δ to achieving A and B by the derivation of the left premiss and then using the remaining resources Δ' together with A and B to achieve C.

As before, we should check that the rules above are locally sound and complete. First, the local reduction

$$\frac{\dfrac{\begin{array}{c}\mathcal{D}_1\\ \Delta_1 \vdash A\end{array} \qquad \begin{array}{c}\mathcal{D}_2\\ \Delta_2 \vdash B\end{array}}{\Delta_1 \times \Delta_2 \vdash A \otimes B}\otimes\mathrm{I} \qquad \begin{array}{c}\mathcal{E}\\ \Delta', w_1{:}A, w_2{:}B \vdash C\end{array}}{\Delta' \times \Delta_1 \times \Delta_2 \vdash C}\otimes\mathrm{E}^{w_1,w_2} \quad \Longrightarrow_R \quad \begin{array}{c}[\mathcal{D}_1/w_1][\mathcal{D}_2/w_2]\mathcal{E}\\ \Delta' \times \Delta_1 \times \Delta_2 \vdash C\end{array}$$

which requires two substitutions for linear hypotheses and the application of the substitution principle. We have also used exchange implicitly on the right hand side: if $\Delta', \Delta_1, \Delta_2 \vdash C$ then $\Delta' \times \Delta_1 \times \Delta_2 \vdash C$ due to the principle of exchange. We will use exchange tacitly from now on together with the substitution principle. The derivation on the right shows that the elimination rules are not too strong: we cannot obtain more judgments than we used to introduce the simultaneous conjunction.

For local completeness we have the following expansion.

$$\begin{array}{c}\mathcal{D}\\ \Delta \vdash A \otimes B\end{array} \quad \Longrightarrow_E \quad \frac{\begin{array}{c}\mathcal{D}\\ \Delta \vdash A \otimes B\end{array} \qquad \dfrac{\dfrac{}{\cdot, w_1{:}A \vdash A}w_1 \qquad \dfrac{}{\cdot, w_2{:}B \vdash B}w_2}{\cdot, w_1{:}A, w_2{:}B \vdash A \otimes B}\otimes\mathrm{I}}{\Delta \vdash A \otimes B}\otimes\mathrm{E}^{w_1,w_2}$$

The derivation on the right verifies that the elimination rules are strong enough so that the simultaneous conjunction can be reconstituted from the parts we obtain from the elimination rule.

Alternative Conjunction. Assume there are two books b_1 and b_2, each of which costs five dollars. If we had five dollars, we could by each one, but not both at the same time. It is our choice which one we buy and it therefore is a form of conjunction. We call it *alternative conjunction* $A \& B$ and pronounce it "A with B". It is sometimes also called *internal choice*. In its introduction rule, the resources are made available in both premisses, since we have to make a choice which among A and B we want to achieve.

$$\frac{\Delta \vdash A \qquad \Delta \vdash B}{\Delta \vdash A \& B} \& \mathrm{I}$$

Consquently, if we have a resource $A \& B$, we can recover either A or B, but not both simultaneously. Therefore we have two elimination rules.

$$\frac{\Delta \vdash A \& B}{\Delta \vdash A} \& \mathrm{E_L} \qquad \frac{\Delta \vdash A \& B}{\Delta \vdash B} \& \mathrm{E_R}$$

The local reductions formalize the reasoning above.

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_1\\ \Delta \vdash A\end{array} \qquad \begin{array}{c}\mathcal{D}_2\\ \Delta \vdash B\end{array}}{\Delta \vdash A \& B} \& \mathrm{I}}{\Delta \vdash A} \& \mathrm{E_L} \quad \Longrightarrow_R \quad \begin{array}{c}\mathcal{D}_1\\ \Delta \vdash A\end{array}$$

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_1\\ \Delta \vdash A\end{array} \qquad \begin{array}{c}\mathcal{D}_2\\ \Delta \vdash B\end{array}}{\Delta \vdash A \& B} \& \mathrm{I}}{\Delta \vdash B} \& \mathrm{E_R} \quad \Longrightarrow_R \quad \begin{array}{c}\mathcal{D}_2\\ \Delta \vdash B\end{array}$$

We recognize these rules from intuitionistic natural deduction, where the context Γ is also made available in both premisses. The embedding of intuitionistic in linear logic will therefore map intuitionistic conjunction $A \wedge B$ to alternative conjunction $A \& B$. The expansion is also already familiar.

$$\begin{array}{c}\mathcal{D}\\ \Delta \vdash A \& B\end{array} \quad \Longrightarrow_E \quad \dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \Delta \vdash A \& B\end{array}}{\Delta \vdash A} \& \mathrm{E_L} \qquad \dfrac{\begin{array}{c}\mathcal{D}\\ \Delta \vdash A \& B\end{array}}{\Delta \vdash B} \& \mathrm{E_R}}{\Delta \vdash A \& B} \& \mathrm{I}$$

Linear Implication. The *linear implication* or *resource implication* internalizes the linear hypothetical judgment at the level of propositions. We say $A \multimap B$ for the goal of achieving B with resource A.

$$\frac{\Delta, w{:}A \vdash B}{\Delta \vdash A \multimap B} \multimap\mathrm{I}^w$$

If we know $A \multimap B$ we can obtain B from a derivation of A.

$$\frac{\Delta \vdash A \multimap B \qquad \Delta' \vdash A}{\Delta \times \Delta' \vdash B} \multimap\mathrm{E}$$

As in the case for simultaneous conjunction, we have to split the resources, devoting Δ to achieving $A \multimap B$ and Δ' to achieving A.

The local reduction carries out the expected substitution for the linear hypothesis.

$$\frac{\dfrac{\begin{matrix}\mathcal{D}\\ \Delta, w{:}A \vdash B\end{matrix}}{\Delta \vdash A \multimap B} \multimap\mathrm{I}^w \qquad \begin{matrix}\mathcal{E}\\ \Delta' \vdash A\end{matrix}}{\Delta \times \Delta' \vdash B} \multimap\mathrm{E} \quad \Longrightarrow_R \quad \begin{matrix}[\mathcal{E}/w]\mathcal{D}\\ \Delta \times \Delta' \vdash B\end{matrix}$$

The rules are also locally complete, as witnessed by the local expansion.

$$\begin{matrix}\mathcal{D}\\ \Delta \vdash A \multimap B\end{matrix} \quad \Longrightarrow_E \quad \dfrac{\dfrac{\begin{matrix}\mathcal{D}\\ \Delta \vdash A \multimap B\end{matrix} \qquad \dfrac{}{\cdot, w{:}A \vdash A}\,w}{\Delta, w{:}A \vdash B} \multimap\mathrm{E}}{\Delta \vdash A \multimap B} \multimap\mathrm{I}^w$$

Unit. The trivial goal which requires no resources is written as $\mathbf{1}$.

$$\frac{}{\cdot \vdash \mathbf{1}} \mathbf{1}\mathrm{I}$$

If we can achieve $\mathbf{1}$ from some resources Δ we know that we can consume all those resources.

$$\frac{\Delta \vdash \mathbf{1} \qquad \Delta' \vdash C}{\Delta' \times \Delta \vdash C} \mathbf{1}\mathrm{E}$$

The rules above and the local reduction and expansion can be seen as a case of 0-ary simultaneous conjunction. In particular, we will see that $\mathbf{1} \otimes A$ is equivalent to A.

$$\frac{\dfrac{}{\cdot \vdash \mathbf{1}} \mathbf{1}\mathrm{I} \qquad \begin{matrix}\mathcal{E}\\ \Delta' \vdash C\end{matrix}}{\Delta' \vdash C} \mathbf{1}\mathrm{E} \quad \Longrightarrow_R \quad \begin{matrix}\mathcal{E}\\ \Delta' \vdash C\end{matrix}$$

$$\begin{matrix}\mathcal{D}\\ \Delta \vdash \mathbf{1}\end{matrix} \quad \Longrightarrow_E \quad \frac{\begin{matrix}\mathcal{D}\\ \Delta \vdash \mathbf{1}\end{matrix} \qquad \dfrac{}{\cdot \vdash \mathbf{1}} \mathbf{1}\mathrm{I}}{\Delta \vdash \mathbf{1}} \mathbf{1}\mathrm{E}$$

Top. There is also a goal which consumes all resources. It is the unit of alternative conjunction and follows the laws of intuitionistic truth.

$$\frac{}{\Delta \vdash \top}\ \top\mathrm{I}$$

There is no elimination rule for $\top$ and consequently no local reduction (it is trivially locally sound). The local expansion replaces an arbitrary derivation by the rule above.

$$\begin{array}{c}\mathcal{D}\\ \Delta \vdash \top\end{array} \quad \Longrightarrow_E \quad \frac{}{\Delta \vdash \top}\ \top\mathrm{I}$$

Disjunction. The *disjunction* $A \oplus B$ (also called *external choice*) is characterized by two introduction rules.

$$\frac{\Delta \vdash A}{\Delta \vdash A \oplus B}\ \oplus\mathrm{I_L} \qquad \frac{\Delta \vdash B}{\Delta \vdash A \oplus B}\ \oplus\mathrm{I_R}$$

As in the case for intuitionistic disjunction, we therefore have to distinguish two cases when we know that we can achieve $A \oplus B$.

$$\frac{\Delta \vdash A \oplus B \qquad \Delta', w_1{:}A \vdash C \qquad \Delta', w_2{:}B \vdash C}{\Delta' \times \Delta \vdash C}\ \oplus\mathrm{E}^{w_1,w_2}$$

Note that resources Δ' appear in both branches, since only one of those two derivations will actually be used to achieve C, depending on the derivation of $A \oplus B$. This can be seen from the local reductions.

$$\frac{\dfrac{\begin{array}{c}\mathcal{D}\\ \Delta \vdash A\end{array}}{\Delta \vdash A \oplus B}\ \oplus\mathrm{I_L} \qquad \begin{array}{c}\mathcal{E}_1\\ \Delta', w_1{:}A \vdash C\end{array} \qquad \begin{array}{c}\mathcal{E}_2\\ \Delta', w_2{:}B \vdash C\end{array}}{\Delta' \times \Delta \vdash C}\ \oplus\mathrm{E}^{w_1,w_2} \quad \Longrightarrow_R \quad \begin{array}{c}[\mathcal{D}/w_1]\mathcal{E}_1\\ \Delta' \times \Delta \vdash C\end{array}$$

$$\frac{\dfrac{\begin{array}{c}\mathcal{D}\\ \Delta \vdash B\end{array}}{\Delta \vdash A \oplus B}\ \oplus\mathrm{I_L} \qquad \begin{array}{c}\mathcal{E}_1\\ \Delta', w_1{:}A \vdash C\end{array} \qquad \begin{array}{c}\mathcal{E}_2\\ \Delta', w_2{:}B \vdash C\end{array}}{\Delta' \times \Delta \vdash C}\ \oplus\mathrm{E}^{w_1,w_2} \quad \Longrightarrow_R \quad \begin{array}{c}[\mathcal{D}/w_2]\mathcal{E}_2\\ \Delta' \times \Delta \vdash C\end{array}$$

The local expansion is also familiar from intuitionistic disjunction.

$$\begin{array}{c}\mathcal{D}\\ \Delta \vdash A \oplus B\end{array} \quad \Longrightarrow_E \quad \frac{\begin{array}{c}\mathcal{D}\\ \Delta \vdash A \oplus B\end{array} \qquad \dfrac{\dfrac{}{\cdot, w_1{:}A \vdash A}\ w_1}{\cdot, w_1{:}A \vdash A \oplus B}\ \vee\mathrm{I_L} \qquad \dfrac{\dfrac{}{\cdot, w_2{:}B \vdash B}\ w_2}{\cdot, w_2{:}B \vdash A \oplus B}\ \vee\mathrm{I_R}}{\Delta \vdash A \oplus B}\ \vee\mathrm{E}^{w_1,w_2}$$

Impossibility. The *impossibility* $\mathbf{0}$ is the case of a disjunction between zero alternatives and the unit of $\oplus$. There is no introduction rule. In the elimination rule we have to consider no branches.

$$\frac{\Delta \vdash \mathbf{0}}{\Delta' \times \Delta \vdash C}\,\mathbf{0}\mathrm{E}$$

There is no local reduction, since there is no introduction rule. However, as in the case of falsehood in intuitionistic logic, we have a local expansion.

$$\begin{array}{c}\mathcal{D}\\ \Delta \vdash \mathbf{0}\end{array} \quad \Longrightarrow_E \quad \frac{\begin{array}{c}\mathcal{D}\\ \Delta \vdash \mathbf{0}\end{array}}{\Delta \vdash \mathbf{0}}\,\mathbf{0}\mathrm{E}$$

Universal Quantification. Quantifiers do not interact much with linearity, since we make no restrictions on occurrences of the parameter. They are included here for reference, but we omit the local reductions and expansion which are given in Section 1.1.

$$\frac{\Delta \vdash [a/x]A}{\Delta \vdash \forall x.\, A}\,\forall\mathrm{I}^a \qquad\qquad \frac{\Delta \vdash \forall x.\, A}{\Delta \vdash [t/x]A}\,\forall\mathrm{E}$$

Existential Quantification. The idea remains the same as in the intuitionistic case, except that we have to split resources among the premisses of the elimination rule.

$$\frac{\Delta \vdash [t/x]A}{\Delta \vdash \exists x.\, A}\,\exists\mathrm{I} \qquad\qquad \frac{\Delta \vdash \exists x.\, A \qquad \Delta', w{:}[a/x]A \vdash C}{\Delta' \times \Delta \vdash C}\,\exists\mathrm{E}^{a,w}$$

We omit the local reduction and expansion, which are trivial modification of the rules in Section 1.1.

This concludes the purely linear operators. Negation and another version of falsehood are postponed to Section **??**, since they may be formally definable, but their interpretation is somewhat questionable in the context we have established so far.

The connectives we have introduced may be classified as to whether the resources are split among the premisses or distributed to the premisses. Connectives of the former kind are called *multiplicative*, the latter *additive*. For example, we might refer to simultaneous conjunction also as *multiplicative conjunction* and to *alternative conjunction* as *additive conjunction*. When we line up the operators against each other, we notice some gaps. For example, there seems to be only a multiplicative implication, but no additive implication. Dually, there seems to be only an additive disjunction, but no multiplicative disjunction. This is not an accident and is pursued further in Exercise 2.3.

2.2 Intuitionistic Hypotheses in Linear Logic

So far, the main judgment permits only linear hypotheses. This means that the logic is too weak to embed intuitionistic logic, and we have failed so far to design a true extension. We now generalize the main judgment to

$$\Gamma; \Delta \vdash A$$

which we read as "*under unrestricted hypotheses* Γ *with resources* Δ *we can achieve goal* A". The hypotheses Γ are intended to satisfy all the structural properties of Section 1.3, that is, exchange, weakening, contraction, and substitution. Substitution is not completely straightforward, since we have to consider the interaction with linear hypotheses. It now reads as follows:

If $(\Gamma_1, u{:}A, \Gamma_2); \Delta \vdash C$ and $\Gamma_1; \cdot \vdash A$ then $(\Gamma_1, \Gamma_2); \Delta \vdash C$.

It is critical to understand why the derivation of $\vdash A$ may not use any linear hypotheses. This is because in the construction of the resulting derivation of $\vdash C$ we substitute the derivation of $\vdash A$ for any use of the hypothesis $u{:}A$ in the given derivation of C. But this hypothesis $u{:}A$ is unrestricted and my be used many times. The substitution could therefore replicate any resources used in the derivation of $\vdash A$, violating the basic principle that resources are used exactly once.

For a similar reason, we can only use an unrestricted hypothesis if there are no resources (which otherwise would not be used as required).

$$\frac{}{(\Gamma_1, u{:}A, \Gamma_2); \cdot \vdash A}\, u$$

All the other rules we presented for pure linear logic are extended by adding the unrestricted context to premisses and conclusion (see the rule summary on page 30). We now reflect the unrestricted hypotheses in the language of proposition by reintroducing the corresponding operator of *intuitionistic implication*.

Intuitionistic Implication. The intuistionistic implication is the familiar one, where we have to be careful in the elimination rule to capture the restriction on the substitution property.

$$\frac{(\Gamma, u{:}A); \Delta \vdash B}{\Gamma; \Delta \vdash A \supset B}\,{\supset}\mathrm{I}^u \qquad \frac{\Gamma; \Delta \vdash A \supset B \qquad \Gamma; \cdot \vdash A}{\Gamma; \Delta \vdash B}\,{\supset}\mathrm{E}$$

The local reduction uses the substitution principle for unrestricted hypotheses.

$$\frac{\dfrac{\begin{array}{c}\mathcal{D}\\ (\Gamma, u{:}A); \Delta \vdash B\end{array}}{\vdash A \supset B}\,{\supset}\mathrm{I}^u \qquad \begin{array}{c}\mathcal{E}\\ \Gamma; \cdot \vdash A\end{array}}{\vdash B}\,{\supset}\mathrm{E} \quad \Longrightarrow_R \quad \begin{array}{c}[\mathcal{D}/u]\mathcal{E}\\ \Gamma; \Delta \vdash B\end{array}$$

In Exercise 2.1 you are asked to show that the rules would be locally unsound (that is, local reduction is not possible), if the second premiss in the elimination rule would be allowed to depend on linear hypotheses. The local expansion is simpler.

$$
\begin{array}{c}\mathcal{D}\\ \Gamma;\Delta\vdash A\supset B\end{array}
\quad\Longrightarrow_E\quad
\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ (\Gamma,u{:}A);\Delta\vdash A\supset B\end{array}\qquad \dfrac{}{(\Gamma,u{:}A);\cdot\vdash A}\,u}{(\Gamma,u{:}A);\Delta\vdash B}\,{\supset}\mathrm{E}}{\Gamma;\Delta\vdash A\supset B}\,{\supset}\mathrm{I}^u
$$

Notice that we weaken $\mathcal{D}$ with the added (and unused) hypothesis u, which does not affect the structure of the derivation. This is not visible in the formulation of the local reduction in Section 1.1, since we did not make the hypotheses explicit.

"Of Course" Modality. Girard [Gir87] observed that there is an even simpler way to connect intuitionistic and linear logic by internalizing the notion of *intuitionistic truth* via a modal operator $!A$ he called "*of course* A" (often pronounced "*bang* A)". A formula is intuitionistically true if it can be derived without the use of any restricted resources.

$$
\dfrac{\Gamma;\cdot\vdash A}{\Gamma;\cdot\vdash {!A}}\,!\mathrm{I}
$$

The elimination rule states that if we can derive $\vdash {!A}$ than we are allowed to use A as an unrestricted hypothesis.

$$
\dfrac{\Gamma;\Delta\vdash {!A}\qquad (\Gamma,u{:}A);\Delta'\vdash C}{\Gamma;(\Delta'\times\Delta)\vdash C}\,!\mathrm{E}^u
$$

This pair of rules is locally sound and complete.

$$
\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \Gamma;\cdot\vdash A\end{array}}{\Gamma;\cdot\vdash {!A}}\,!\mathrm{I}\qquad \begin{array}{c}\mathcal{E}\\ (\Gamma,u{:}A);\Delta'\vdash C\end{array}}{\Gamma;\Delta'\vdash C}\,!\mathrm{E}^u
\quad\Longrightarrow_R\quad
\begin{array}{c}[\mathcal{D}/u]\mathcal{E}\\ \Gamma;\Delta'\vdash C\end{array}
$$

$$
\begin{array}{c}\mathcal{D}\\ \Gamma;\Delta\vdash {!A}\end{array}
\quad\Longrightarrow_E\quad
\dfrac{\begin{array}{c}\mathcal{D}\\ \Gamma;\Delta\vdash {!A}\end{array}\qquad \dfrac{\dfrac{}{(\Gamma,u{:}A);\cdot\vdash A}\,u}{(\Gamma,u{:}A);\cdot\vdash {!A}}\,!\mathrm{I}}{\Gamma;\Delta\vdash {!A}}\,!\mathrm{E}^u
$$

Using the *of course* modality, one can define the intuitionstic implication $A\supset B$ as $(!A)\multimap B$. It was this observation which gave rise to Girard's development of linear logic. Under this interpretation, the introduction and elimination rules for intuitionistic implication are *derived rules of inference* (see Exercise 2.2).

We now summarize the rules of intuitionistic linear logic. A very similar calculus was developed and analyzed in the categorical context by Barber [Bar96]. It differs from more traditional treatments by Abramsky [Abr93], Troelstra [Tro93], Bierman [Bie94] and Albrecht et al. [ABCJ94] in that structural rules remain completely implicit. The logic we consider here comprises the following logical operators.

Formulas	A	$::=$	P	Atoms
			$\mid A_1 \multimap A_2 \mid A_1 \otimes A_2 \mid \mathbf{1}$	Multiplicatives
			$\mid A_1 \& A_2 \mid \top \mid A_1 \oplus A_2 \mid \mathbf{0}$	Additives
			$\mid \forall x.\ A \mid \exists x.\ A$	Quantifiers
			$\mid A \supset B \mid\ !A$	Exponentials

It is instructive to compare the rules below with those of intuitionistic natural deduction on page 13, keeping in mind that hypotheses were left implicit in that formulation.

Hypotheses.

$$\frac{}{\Gamma; (\cdot, w{:}A) \vdash A}\, w \qquad \frac{}{(\Gamma_1, u{:}A, \Gamma_2); \cdot \vdash A}\, u$$

Multiplicative Connectives.

$$\frac{\Gamma; \Delta_1 \vdash A \qquad \Gamma; \Delta_2 \vdash B}{\Gamma; (\Delta_1 \times \Delta_2) \vdash A \otimes B}\, \otimes\mathrm{I} \qquad \frac{\Gamma; \Delta \vdash A \otimes B \quad \Gamma; (\Delta', w_1{:}A, w_2{:}B) \vdash C}{\Gamma; (\Delta' \times \Delta) \vdash C}\, \otimes\mathrm{E}^{w_1,w_2}$$

$$\frac{\Gamma; (\Delta, w{:}A) \vdash B}{\Gamma; \Delta \vdash A \multimap B}\, \multimap\mathrm{I}^{w} \qquad \frac{\Gamma; \Delta \vdash A \multimap B \qquad \Gamma; \Delta' \vdash A}{\Gamma; (\Delta \times \Delta') \vdash B}\, \multimap\mathrm{E}$$

$$\frac{}{\Gamma; \cdot \vdash \mathbf{1}}\, \mathbf{1}\mathrm{I} \qquad \frac{\Gamma; \Delta \vdash \mathbf{1} \qquad \Gamma; \Delta' \vdash C}{\Gamma; (\Delta' \times \Delta) \vdash C}\, \mathbf{1}\mathrm{E}$$

Additive Connectives.

$$\frac{\Gamma; \Delta \vdash A \qquad \Gamma; \Delta \vdash B}{\Gamma; \Delta \vdash A \& B}\, \&\mathrm{I} \qquad \frac{\Gamma; \Delta \vdash A \& B}{\Gamma; \Delta \vdash A}\, \&\mathrm{E_L} \qquad \frac{\Gamma; \Delta \vdash A \& B}{\Gamma; \Delta \vdash B}\, \&\mathrm{E_R}$$

$$\frac{}{\Gamma; \Delta \vdash \top}\, \top\mathrm{I} \qquad \textit{no } \top \textit{ elimination}$$

$$\frac{\Gamma; \Delta \vdash A}{\Gamma; \Delta \vdash A \oplus B}\oplus\mathrm{I_L} \qquad \frac{\Gamma; \Delta \vdash A \oplus B \quad \Gamma; (\Delta', w_1{:}A) \vdash C \quad \Gamma; (\Delta', w_2{:}B) \vdash C}{\Gamma; (\Delta' \times \Delta) \vdash C}\oplus\mathrm{E}^{w_1,w_2}$$

$$\frac{\Gamma; \Delta \vdash B}{\Gamma; \Delta \vdash A \oplus B}\oplus\mathrm{I_R}$$

$$\textit{no } \mathbf{0} \textit{ introduction} \qquad \frac{\Gamma; \Delta \vdash \mathbf{0}}{\Gamma; (\Delta' \times \Delta) \vdash C}\mathbf{0}\mathrm{E}$$

Quantifiers.

$$\frac{\Gamma; \Delta \vdash [a/x]A}{\Gamma; \Delta \vdash \forall x.\ A}\forall\mathrm{I}^a \qquad \frac{\Gamma; \Delta \vdash \forall x.\ A}{\Gamma; \Delta \vdash [t/x]A}\forall\mathrm{E}$$

$$\frac{\Gamma; \Delta \vdash [t/x]A}{\Gamma; \Delta \vdash \exists x.\ A}\exists\mathrm{I} \qquad \frac{\Gamma; \Delta \vdash \exists x.\ A \qquad \Gamma; (\Delta', w{:}[a/x]A) \vdash C}{\Gamma; (\Delta' \times \Delta) \vdash C}\exists\mathrm{E}^{a,w}$$

Exponentials.

$$\frac{(\Gamma, u{:}A); \Delta \vdash B}{\Gamma; \Delta \vdash A \supset B}\supset\mathrm{I}^u \qquad \frac{\Gamma; \Delta \vdash A \supset B \qquad \Gamma; \cdot \vdash A}{\Gamma; \Delta \vdash B}\supset\mathrm{E}$$

$$\frac{\Gamma; \cdot \vdash A}{\Gamma; \cdot \vdash {!A}}!\mathrm{I} \qquad \frac{\Gamma; \Delta \vdash {!A} \qquad (\Gamma, u{:}A); \Delta' \vdash C}{\Gamma; (\Delta' \times \Delta) \vdash C}!\mathrm{E}^u$$

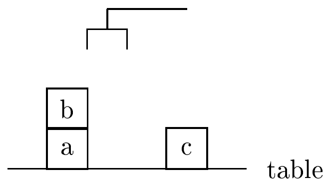
2.3 Two Examples

In this section we practice exploiting the connectives of linear logic to express situations involving resources and state. The first example is a *menu* consisting of various courses which can be obtained for 200 french francs.

Menu A: FF 200	$\mathrm{FF}(200) \multimap$
Onion Soup or *Clear Broth*	$((\mathrm{OS} \& \mathrm{CB})$
Honey-Glazed Duck	$\otimes\, \mathrm{HGD}$
Peas or *Red Cabbage* (according to season)	$\otimes\, (\mathrm{P} \oplus \mathrm{RC})$
New Potatoes	$\otimes\, \mathrm{NP}$
Chocolate Mousse (FF 30 extra)	$\otimes\, ((\mathrm{FF}(30) \multimap \mathrm{CM}) \& \mathbf{1})$
Coffee (unlimited refills)	$\otimes\, \mathrm{C}$ $\otimes\, (!\mathrm{C}))$

Note the two different informal uses of "or", one modelled by an alternative conjunction and one by a disjunction. The option of ordering chocolate mousse is also represented by an alternative conjunction: we can choose $(\mathrm{FF}(30) \multimap \mathrm{CM}) \& \mathbf{1}$ to obtain nothing ($\mathbf{1}$) or pay another 30 francs to obtain the mousse.

The second is perhaps more typical of uses of linear logic in computer science applications. We use it to model a planning problem in the so-called *blocks world* in which a robot arm can manipulate blocks, trying to achieve some goal.



We use the following primitive propositions.

$\mathrm{on}(x, y)$	block x is on block y
$\mathrm{tb}(x)$	block x is on the table
$\mathrm{holds}(x)$	robot arm holds block x
empty	robot arm is empty
$\mathrm{clear}(x)$	the top of block x is clear

A planning problem is represented as judgment

$$\Gamma_0; \Delta_0 \vdash A_0$$

where Γ_0 represent the rules which describe the legal operations, Δ_0 is the initial state represented as a context of the propositions which are true, and A is the goal to be achieved. For example, the situation in the picture above would be represented by

$$\Delta_0 = \cdot, \mathrm{empty}, \mathrm{tb}(a), \mathrm{on}(b, a), \mathrm{clear}(b), \mathrm{tb}(c), \mathrm{clear}(c)$$

where we have omitted labels for the sake of brevity. The rules are represented by unrestricted hypotheses, since they may be used arbitrarily often in the course of solving a problem. We use the following for rules for picking up or putting down and object. We use the convention that simultaneous conjunction $\otimes$ binds more tightly than linear implication $\multimap$.

$$\begin{array}{lcll}
\Gamma_0 & = & \cdot, & \\
 & & \mathrm{geton} & : \ \forall x.\ \forall y.\ \mathrm{empty} \otimes \mathrm{clear}(x) \otimes \mathrm{on}(x,y) \multimap \mathrm{holds}(x) \otimes \mathrm{clear}(y), \\
 & & \mathrm{gettb} & : \ \forall x.\ \mathrm{empty} \otimes \mathrm{clear}(x) \otimes \mathrm{tb}(x) \multimap \mathrm{holds}(x), \\
 & & \mathrm{puton} & : \ \forall x.\ \forall y.\ \mathrm{holds}(x) \otimes \mathrm{clear}(y) \multimap \mathrm{empty} \otimes \mathrm{on}(x,y) \otimes \mathrm{clear}(x), \\
 & & \mathrm{puttb} & : \ \forall x.\ \mathrm{holds}(x) \multimap \mathrm{empty} \otimes \mathrm{tb}(x) \otimes \mathrm{clear}(x).
\end{array}$$

Each of these represents a particular possible action, assuming that it can be carried out successfully. Matching the left-hand side of one these rules will consume the corresponding resources so that, for example, the proposition *empty* with no longer be available after the *geton* action has been applied.

The goal that we would like to achieve $\mathrm{on}(a, b)$, for example, is represented with the aid of using $\top$.

$$A_0 = \mathrm{on}(a, b) \otimes \top$$

Any derivation of the judgment

$$\Gamma_0; \Delta_0 \vdash A_0$$

represents a plan for achieving the goal A_0 from the initial situation state Δ_0.

We now go through a derivation of the particular example above, omitting the unrestricted resources Γ_0 which do not change throughout the derivation. Our first goal is to derive

$$\cdot, \mathrm{empty}, \mathrm{tb}(a), \mathrm{on}(b, a), \mathrm{clear}(b), \mathrm{tb}(c), \mathrm{clear}(c), \mathrm{empty} \vdash \mathrm{on}(a, b) \otimes \top$$

By using $\otimes$I twice we can prove

$$\cdot, \mathrm{empty}, \mathrm{on}(b, a), \mathrm{clear}(b) \vdash \mathrm{empty} \otimes \mathrm{clear}(b) \otimes \mathrm{on}(b, a)$$

Using the intuitionistic hypothesis rule for *geton* followed by $\forall$E twice and $\multimap$ E we obtain

$$\cdot, \mathrm{empty}, \mathrm{clear}(b), \mathrm{on}(b, a) \vdash \mathrm{holds}(b) \otimes \mathrm{clear}(a)$$

Now we use $\otimes$E with the derivation above as our left premiss, to prove our overall goal, leaving us with the goal to derive

$$\cdot, \mathrm{tb}(a), \mathrm{tb}(c), \mathrm{clear}(c), \mathrm{holds}(b), \mathrm{clear}(a) \vdash \mathrm{on}(a, b) \otimes \top$$

as our right premiss. Observe how the original resources Δ_0 have been split between the two premisses, and the results from the left premiss derivation, holds(b) and clear(a) have been added to the description of the situation. The new subgoal has exactly the same form as the original goal (in fact, the conclusion has not changed), but applying the unrestricted assumption *geton* has changed our state.

Proceeding in the same manner, using the rule *puttb* next leaves us with the subgoal

$$\cdot, \mathrm{tb}(a), \mathrm{tb}(c), \mathrm{clear}(c), \mathrm{clear}(a), \mathrm{empty}, \mathrm{clear}(b), \mathrm{tb}(b) \vdash \mathrm{on}(a,b) \otimes \top$$

We now apply *gettb* using a for x and proceeding as above which gives us a derivation of $\vdash$ holds(a). Instead of $\otimes$E, we use the substitution principle yielding the subgoal

$$\cdot, \mathrm{tb}(c), \mathrm{clear}(c), \mathrm{clear}(b), \mathrm{tb}(b), \mathrm{holds}(a) \vdash \mathrm{on}(a,b) \otimes \top$$

With same technique, this time using *puton*, we obtain the subgoal

$$\cdot, \mathrm{tb}(c), \mathrm{clear}(c), \mathrm{tb}(b), \mathrm{empty}, \mathrm{on}(a,b), \mathrm{clear}(a) \vdash \mathrm{on}(a,b) \otimes \top$$

Now we can conclude the derivation with the $\otimes$I rule, distributing resource on(a, b) to the left premiss, which follows immediately as hypothesis, and distributing the remaining resources to the right premiss, where $\top$ follows by $\top$I, ignoring all resources.

Note that different derivations of the original judgment represent different sequences of actions (see Exercise 2.4).

2.4 Embedding Intuitionistic Logic

Our goal in this section is to show that intuitionistic linear logic (ILL) is a refinement of intuitionistic logic (IL) in the sense that we can translate each formula of IL into ILL in a way that preserves derivability. Actually, we will try to achieve more: not only should it be possible to preserve derivability, but the translation should also preserve the structure of derivations as much as possible. This will allow us to make stronger statements regarding the connection between proof search and reduction in the two calculi when we investigate specific applications.

The guiding principle in the definition of the translation $()^+$ of IL formulas into ILL is the idea that the judgment $\Gamma \vdash A$ of IL is interpreted as the judgment $\Gamma^+; \cdot \vdash A^+$ of ILL. In other words, all intuitionistic assumptions become unrestricted hypotheses. We design the translation so that a derivation $\mathcal{D} :: (\Gamma \vdash A)$ can be translated directly to a derivation $\mathcal{D}^+ :: (\Gamma^+; \cdot \vdash A^+)$. We omit negation here, which is left to Exercise 2.10.

The only real question arises in the cases for conjunction and truth, since they split into two possible connectives each. We model them here with additive linear connectives. Another translation is explored in Exercise 2.9. For most connectives, however, we have little choice. Contexts are translated by simply translating the formulas occurring in them.

$$
\begin{array}{rcl}
P^+ & = & P \\
(A \wedge B)^+ & = & A^+ \& B^+ \\
(A \supset B)^+ & = & (!A^+) \multimap B^+ \\
(A \vee B)^+ & = & (!A^+) \oplus (!B^+) \\
(\bot)^+ & = & \mathbf{0} \\
(\top)^+ & = & \top \\
(\forall x.\, A)^+ & = & \forall x.\, A^+ \\
(\exists x.\, A)^+ & = & \exists x.\, !A^+ \\
\\
(\cdot)^+ & = & \cdot \\
(\Gamma, u{:}A)^+ & = & \Gamma^+, u{:}A^+
\end{array}
$$

To illustrate Girard's original decomposition of $A \supset B$ into $(!A) \multimap B$ we do not use intuitionistic implication in linear logic, even though it would certainly be reasonable to translate $(A \supset B)^+ = A^+ \supset B^+$.

Lemma 2.1 (Embedding) *If $\Gamma \vdash A$ in IL then $\Gamma^+; \cdot \vdash A^+$ in ILL.*

Proof: By induction over the structore of $\mathcal{D} :: (\Gamma \vdash A)$. The computational contents of this proof is a compositional translation of derivations $\mathcal{D}$ to derivations $\mathcal{D}^+ :: (\Gamma^+; \cdot \vdash A^+)$. □

An attempt to prove the other direction in a similar manner will fail, since a natural deduction of $\Gamma^+; \cdot \vdash A^+$ may have many subdeductions with a conclusion which is not of this form. For example, if the deduction ends in $\multimap E$ the premises contain the new formula A which may not necessarily be the translation of an intuitionistic formula. In the next section we show a way to prove the opposite direction based on normal derivations. A simpler way is to translate each linear connective into its intuitionistic counterpart and show that the resulting judgment is derivable. This reverse translation $()^-$ should have the property that $(A^+)^- = A$

$$
\begin{array}{rcl}
P^- & = & P \\
(A \& B)^- = (A \otimes B)^- & = & A^- \wedge B^- \\
(A \multimap B)^- = (A \supset B)^- & = & A^- \supset B^- \\
(A \oplus B)^- & = & A^- \vee B^- \\
(\mathbf{0})^- & = & \bot \\
(\top)^- = (\mathbf{1})^- & = & \top \\
(\forall x.\, A)^- & = & \forall x.\, A^- \\
(\exists x.\, A)^- & = & \exists x.\, A^- \\
(!A)^- & = & A^- \\
\\
(\cdot)^- & = & \cdot \\
(\Gamma, u{:}A)^- & = & \Gamma^-, u{:}A^-
\end{array}
$$

The last two rules are also used to map a linear context Δ to the corresponding intuitionistic context Δ^-.

Property 2.2 $(A^+)^- = A$

Proof: By induction on the structure of A. $\square$

Lemma 2.3 (Conservativity) *If* $\Gamma; \Delta \vdash A$ *in ILL then* $\Gamma^-, \Delta^- \vdash A^-$ *in IL.*

Proof: By induction on the structure of $\mathcal{D} :: (\Gamma; \Delta \vdash A)$. $\square$

Theorem 2.4 (Conservative Embedding) *The translation* $()^+$ *is a conservative embedding from IL into ILL.*

Proof: From Lemmas 2.1 and 2.3 and Property 2.2. $\square$

2.5 Normal Deductions

An intuitive strategy in constructing natural deductions is to apply introduction rules backwards to break the conclusion into subgoals and to apply elimination rules to hypotheses until the two meet. This strategy is in fact complete which has numerous consequences. One of the most important is *consistency* of the logic, that is, not every proposition is true. This is closely related to the local soundness property we have investigated for each of the connectives.

We call natural deductions which have been constructed with the strategy sketched above *normal.* Normalcy is a judgment about derivations, just as truth is a judgment about propositions. It is awkward to write out and reason about judgments on derivations, but there are standard techniques to avoid them. The most commonly used is to reformulate the judgment on derivations as a judgment on objects, in this case propositions. Instead of judging a derivation to be normal, the judgment expresses that "*A has a normal derivation*".

In our situation one judgment will not be sufficient, since we need to describe bottom-up reasoning (introduce the main connective of the conclusion) and top-down reasoning (eliminate the main connective of the hypothesis). Thus we have two mutually dependent judgments

$\Gamma; \Delta \vdash A \Uparrow$	A has a normal derivation, and
$\Gamma; \Delta \vdash A \Downarrow$	A has an atomic derivation,

where the latter formalizes the top-down reasoning from hypotheses (intuitionistic or linear). These judgments are defined by the following inference rules.

Hypotheses.

$$\frac{}{\Gamma; (\cdot, w{:}A) \vdash A \Downarrow} w \qquad \frac{}{(\Gamma_1, u{:}A, \Gamma_2); \cdot \vdash A \Downarrow} u$$

Multiplicative Connectives.

$$\frac{\Gamma;\Delta_1 \vdash A\ \uparrow \qquad \Gamma;\Delta_2 \vdash B\ \uparrow}{\Gamma;(\Delta_1\times\Delta_2)\vdash A\otimes B\ \uparrow}\otimes\mathrm{I} \qquad \frac{\Gamma;\Delta\vdash A\otimes B\ \downarrow \qquad \Gamma;(\Delta', w_1{:}A, w_2{:}B)\vdash C\ \uparrow}{\Gamma;(\Delta'\times\Delta)\vdash C\ \uparrow}\otimes\mathrm{E}^{w_1,w_2}$$

$$\frac{\Gamma;(\Delta, w{:}A)\vdash B\ \uparrow}{\Gamma;\Delta\vdash A\multimap B\ \uparrow}\multimap\mathrm{I}^w \qquad \frac{\Gamma;\Delta\vdash A\multimap B\ \downarrow \qquad \Gamma;\Delta'\vdash A\ \uparrow}{\Gamma;\Delta\times\Delta'\vdash B\ \downarrow}\multimap\mathrm{E}$$

$$\frac{}{\Gamma;\cdot\vdash \mathbf{1}\ \uparrow}\mathbf{1}\mathrm{I} \qquad \frac{\Gamma;\Delta\vdash\mathbf{1}\ \downarrow \qquad \Gamma;\Delta'\vdash C\ \uparrow}{\Gamma;(\Delta'\times\Delta)\vdash C\ \uparrow}\mathbf{1}\mathrm{E}$$

Additive Connectives.

$$\frac{\Gamma;\Delta\vdash A\ \uparrow \qquad \Gamma;\Delta\vdash B\ \uparrow}{\Gamma;\Delta\vdash A\&B\ \uparrow}\&\mathrm{I} \qquad \frac{\Gamma;\Delta\vdash A\&B\ \downarrow}{\Gamma;\Delta\vdash A\ \downarrow}\&\mathrm{E_L} \qquad \frac{\Gamma;\Delta\vdash A\&B\ \downarrow}{\Gamma;\Delta\vdash B\ \downarrow}\&\mathrm{E_R}$$

$$\frac{}{\Gamma;\Delta\vdash\top\ \uparrow}\top\mathrm{I} \qquad \textit{No } \top \textit{ elimination rule}$$

$$\frac{\Gamma;\Delta\vdash A\ \uparrow}{\Gamma;\Delta\vdash A\oplus B\ \uparrow}\oplus\mathrm{I_L} \qquad \frac{\Gamma;\Delta\vdash B\ \uparrow}{\Gamma;\Delta\vdash A\oplus B\ \uparrow}\oplus\mathrm{I_R}$$

$$\frac{\Gamma;\Delta\vdash A\oplus B\ \downarrow \qquad \Gamma;(\Delta', w_1{:}A)\vdash C\ \uparrow \qquad \Gamma;(\Delta', w_2{:}B)\vdash C\ \uparrow}{\Gamma;(\Delta'\times\Delta)\vdash C\ \uparrow}\oplus E^{w_1,w_2}$$

$$\textit{No } \mathbf{0} \textit{ introduction rule} \qquad \frac{\Gamma;\Delta\vdash 0\ \downarrow}{\Gamma;(\Delta'\times\Delta)\vdash C\ \uparrow}\mathbf{0}\mathrm{E}$$

Quantifiers.

$$\frac{\Gamma;\Delta\vdash [a/x]A\ \uparrow}{\Gamma;\Delta\vdash \forall x.\ A\ \uparrow}\forall\mathrm{I}^a \qquad \frac{\Gamma;\Delta\vdash\forall x.\ A\ \downarrow}{\Gamma;\Delta\vdash [t/x]A\ \downarrow}\forall\mathrm{E}$$

$$\frac{\Gamma;\Delta\vdash [t/x]A\ \uparrow}{\Gamma;\Delta\vdash\exists x.\ A\ \uparrow}\exists\mathrm{I} \qquad \frac{\Gamma;\Delta\vdash\exists x.\ A\ \downarrow \qquad \Gamma;(\Delta', w{:}[a/x]A)\vdash C\ \uparrow}{\Gamma;(\Delta'\times\Delta)\vdash C\ \uparrow}\exists\mathrm{E}^{a,w}$$

Exponentials.

$$\frac{(\Gamma, u{:}A); \Delta \vdash B \uparrow}{\Gamma; \Delta \vdash A \supset B \uparrow} \supset\mathrm{I}^u \qquad \frac{\Gamma; \Delta \vdash A \supset B \downarrow \qquad \Gamma; \cdot \vdash A \uparrow}{\Gamma; \Delta \vdash B \downarrow} \supset\mathrm{E}$$

$$\frac{\Gamma; \cdot \vdash A \uparrow}{\Gamma; \cdot \vdash !A \uparrow} !I \qquad \frac{\Gamma; \Delta \vdash !A \downarrow \qquad (\Gamma, u{:}A); \Delta' \vdash C \uparrow}{\Gamma; (\Delta' \times \Delta) \vdash C \uparrow} !\mathrm{E}^u$$

Coercion.

$$\frac{\Gamma; \Delta \vdash A \downarrow}{\Gamma; \Delta \vdash A \uparrow} \downarrow\uparrow$$

The coercion $\downarrow\uparrow$ states that all atomic derivations should be considered normal. From the point of view of proof search this means that we can complete the derivation when forward and backward reasoning arrive at the same proposition. It easy to see that these judgments just restrict the set of derivations.

Property 2.5 (Soundness of Normal Derivations)

1. *If* $\Gamma; \Delta \vdash A \uparrow$ *then* $\Gamma; \Delta \vdash A$.
2. *If* $\Gamma; \Delta \vdash A \downarrow$ *then* $\Gamma; \Delta \vdash A$.

Proof: *By simultaneous induction on the given derivations. The computational contents of this proof are the obvious structural translation from* $\mathcal{N} :: (\Gamma; \Delta \vdash A \uparrow)$ *to* $\mathcal{N}^- :: (\Gamma; \Delta \vdash A)$ *and from* $\mathcal{A} :: (\Gamma; \Delta \vdash A \downarrow)$ *to* $\mathcal{A}^- :: (\Gamma; \Delta \vdash A)$. *Note that the coercion* $\downarrow\uparrow$ *disappears, since the translation of the premiss and conclusion are identical.* □

The corresponding completeness theorem, namely that $\Gamma; \Delta \vdash A$ implies $\Gamma; \Delta \vdash A \uparrow$, also holds, but is quite difficult to prove. This is the subject of the Normalization Theorem 2.19. Together with the two judgments about atomic and normal derivations, we have refined substitution principles. Since hypotheses are atomic, they permit only the substitution of atomic derivations for hypotheses.

Lemma 2.6 (Substitution Principles for Normal Derivations)

1. *If* $\Gamma; (\Delta_1, w{:}A, \Delta_2) \vdash C \uparrow$ *and* $\Gamma; \Delta \vdash A \downarrow$ *then* $\Gamma; (\Delta_1, \Delta, \Delta_2) \vdash C \uparrow$
2. *If* $\Gamma; (\Delta_1, w{:}A, \Delta_2) \vdash C \downarrow$ *and* $\Gamma; \Delta \vdash A \downarrow$ *then* $\Gamma; (\Delta_1, \Delta, \Delta_2) \vdash C \downarrow$
3. *If* $(\Gamma_1, u{:}A, \Gamma_2); \Delta \vdash C \uparrow$ *and* $\Gamma_1; \cdot \vdash A \downarrow$ *then* $(\Gamma_1, \Gamma_2); \Delta \vdash C \uparrow$
4. *If* $(\Gamma_1, u{:}A, \Gamma_2); \Delta \vdash C \downarrow$ *and* $\Gamma_1; \cdot \vdash A \downarrow$ *then* $(\Gamma_1, \Gamma_2); \Delta \vdash C \downarrow$

Proof: By straightforward inductions over the structure of the first of the given derivations. □

A first immediate connection to local reductions is the following.

Property 2.7

1. *If* $\mathcal{N} :: (\Gamma; \Delta \vdash A \uparrow)$ *then* $\mathcal{N}^- :: (\Gamma; \Delta \vdash A)$ *contains no local redex.*
2. *If* $\mathcal{A} :: (\Gamma; \Delta \vdash A \downarrow)$ *then* $\mathcal{A}^- :: (\Gamma; \Delta \vdash A)$ *contains no local redex.*

Proof: By induction on the structure of $\mathcal{N}$ and $\mathcal{A}$, inspecting the possible forms of local redices in each case. □

We can now also give an alternative way to describe the connection between IL and ILL by showing the normal deductions can be translated in the opposite directions quite easily. We write $!\Delta$ for a context of the form $\cdot, u_1{:}!A_1, \ldots, u_n{:}!A_n$.

Lemma 2.8

1. *If* $\Gamma^+; !\Delta^+ \vdash A^+ \uparrow$ *in ILL then* $\Gamma, \Delta \vdash A$ *in IL.*
2. *If* $\Gamma^+; !\Delta^+ \vdash !A^+ \uparrow$ *in ILL then* $\Gamma, \Delta \vdash A$ *in IL.*
3. *If* $\Gamma^+; !\Delta^+ \vdash C \downarrow$ *in ILL then either* $C = B^+$ *or* $C = !B^+$ *for some* B *and* $\Gamma, \Delta \vdash B$ *in IL.*

Proof: By simultaneous induction on the structures of $\mathcal{N} :: (\Gamma^+; !\Delta^+ \vdash A^+ \uparrow)$ and $\mathcal{A} :: (\Gamma^+; !\Delta^+ \vdash C \downarrow)$. □

2.6 Cut-Free Sequent Calculus

The sequent calculus can be seen as a calculus of proof search for natural deductions. In this section we try to transcribe the process of searching for a *normal* natural deduction into an inference system. In the context of sequent calculus, proof search is seen entirely as the bottom-up construction of a derivation. This means that elimination rules must be turned "upside-down" so they can also be applied bottom-up rather than top-down. A sequent has the form $\Gamma; \Delta \Longrightarrow C$, where Γ corresponds to unrestricted hypotheses Δ to linear hypotheses, and C to the conclusion.

In terms of *judgments* we interpret a sequent via a splitting of the judgment "*A is true*" into two judgments: "*A is a resource*" and "*A is a true conclusion*". Ignoring unrestricted hypothesis for the moment, the main judgment

$$(\cdot, w_1{:}A_1, \ldots, w_n{:}A_n) \Longrightarrow C$$

expresses

> Under the linear hypothesis that we have resources $A_1, \ldots, A_n$ we judge C to be a true conclusion.

Adding unrestricted hypotheses, the judgment

$$(\cdot, u_1{:}B_1, \ldots, u_m{:}B_m); (\cdot, w_1{:}A_1, \ldots, w_n{:}A_n) \Longrightarrow C$$

expresses

> Under the unrestricted hypotheses that we have resources $B_1, \ldots, B_m$ and linear hyptheses that we have resources $A_1, \ldots, A_n$, we judge C to be a true conclusion.

This interpretation means that we now have an explicit inference rule which relates the judgment "*A is a resource*" to the judgment "*A is a true conclusion*". We call the resulting sequent an *initial sequent* and write I.

$$\frac{}{\Gamma; (\cdot, w{:}A) \Longrightarrow A}\,\mathrm{I}(w)$$

The remaining rules are divided into *right* and *left* rules, which correspond to the *introduction* and *elimination* rules of natural deduction, respectively. The right rules apply to the conclusion, while the left rules apply to resources. Since resources may be either linear or unrestricted, our notation would require two versions of each left rule. Instead we add one more hypothesis rule which allows us to copy an unrestricted to a linear hypothesis. This rule is labelled DL for *dereliction*.

$$\frac{(\Gamma_1, u{:}A, \Gamma_2); (\Delta, w{:}A) \Longrightarrow C}{(\Gamma_1, u{:}A, \Gamma_2); \Delta \Longrightarrow C}\,\mathrm{DL}^w(u)$$

In the following, we adhere to common practice and omit labels on hypotheses and consequently also on the justifications of the inference rules. The reader should keep in mind, however, that this is just a short-hand, and that there are, for example, two *different* derivations of $(\cdot, A, A); \cdot \Longrightarrow A$, one using the first copy of A and one using the second.

Finally, we permit implicit uses of exchange in the conclusion in order to move the principal proposition of a rule to the right-most position. In other words, we write Δ, A instead of $\Delta_1, w{:}A, \Delta_2$. We repeat the rules from above in their abbreviated form and the give the remaining left and right rules.

Hypotheses.

$$\frac{}{\Gamma; A \Longrightarrow A}\,\mathrm{I} \qquad \frac{(\Gamma, A); (\Delta, A) \Longrightarrow C}{(\Gamma, A); \Delta \Longrightarrow C}\,\mathrm{DL}$$

Multiplicative Connectives.

$$\frac{\Gamma; \Delta, A \Longrightarrow B}{\Gamma; \Delta \Longrightarrow A \multimap B} \multimap R \qquad \frac{\Gamma; \Delta_1 \Longrightarrow A \qquad \Gamma; \Delta_2, B \Longrightarrow C}{\Gamma; \Delta_1 \times \Delta_2, A \multimap B \Longrightarrow C} \multimap L$$

$$\frac{\Gamma; \Delta_1 \Longrightarrow A \qquad \Gamma; \Delta_2 \Longrightarrow B}{\Gamma; \Delta_1 \times \Delta_2 \Longrightarrow A \otimes B} \otimes R \qquad \frac{\Gamma; \Delta, A, B \Longrightarrow C}{\Gamma; \Delta, A \otimes B \Longrightarrow C} \otimes L$$

$$\frac{}{\Gamma; \cdot \Longrightarrow \mathbf{1}} \mathbf{1}R \qquad \frac{\Gamma; \Delta \Longrightarrow C}{\Gamma; \Delta, \mathbf{1} \Longrightarrow C} \mathbf{1}L$$

Additive Connectives.

$$\frac{\Gamma; \Delta \Longrightarrow A \qquad \Gamma; \Delta \Longrightarrow B}{\Gamma; \Delta \Longrightarrow A \& B} \& R \qquad \frac{\Gamma; \Delta, A \Longrightarrow C}{\Gamma; \Delta, A \& B \Longrightarrow C} \& L_1 \qquad \frac{\Gamma; \Delta, B \Longrightarrow C}{\Gamma; \Delta, A \& B \Longrightarrow C} \& L_2$$

$$\frac{}{\Gamma; \Delta \Longrightarrow \top} \top R \qquad \textit{No } \top \textit{ left rule}$$

$$\frac{\Gamma; \Delta \Longrightarrow A}{\Gamma; \Delta \Longrightarrow A \oplus B} \oplus R_1 \qquad \frac{\Gamma; \Delta \Longrightarrow B}{\Gamma; \Delta \Longrightarrow A \oplus B} \oplus R_2 \qquad \frac{\Gamma; \Delta, A \Longrightarrow C \qquad \Gamma; \Delta, B \Longrightarrow C}{\Gamma; \Delta, A \oplus B \Longrightarrow C} \oplus L$$

$$\textit{No } \mathbf{0} \textit{ right rule} \qquad \frac{}{\Gamma; \Delta, \mathbf{0} \Longrightarrow C} \mathbf{0}L$$

Quantifiers.

$$\frac{\Gamma; \Delta \Longrightarrow [a/x]A}{\Gamma; \Delta \Longrightarrow \forall x.\ A} \forall R^a \qquad \frac{\Gamma; \Delta, [t/x]A \Longrightarrow C}{\Gamma; \Delta, \forall x.\ A \Longrightarrow C} \forall L$$

$$\frac{\Gamma; \Delta \Longrightarrow [t/x]A}{\Gamma; \Delta \Longrightarrow \exists x.\ A} \exists R \qquad \frac{\Gamma; \Delta, [a/x]A \Longrightarrow C}{\Gamma; \Delta, \exists x.\ A \Longrightarrow C} \exists L^a$$

Exponentials.

$$\frac{(\Gamma, A); \Delta \Longrightarrow B}{\Gamma; \Delta \Longrightarrow A \supset B} \supset\mathrm{R} \qquad \frac{\Gamma; \cdot \Longrightarrow A \qquad \Gamma; \Delta, B \Longrightarrow C}{\Gamma; \Delta, A \supset B \Longrightarrow C} \supset\mathrm{L}$$

$$\frac{\Gamma; \cdot \Longrightarrow A}{\Gamma; \cdot \Longrightarrow\, !A} !\mathrm{R} \qquad \frac{(\Gamma, A); \Delta \Longrightarrow C}{\Gamma; \Delta, !A \Longrightarrow C} !\mathrm{L}$$

We have the following theorems relating normal natural deductions and sequent derivations.

Theorem 2.9 (Soundness of Sequent Derivations)
If $\Gamma; \Delta \Longrightarrow A$ *then* $\Gamma; \Delta \vdash A \uparrow$.

Proof: By induction on the structure of the derivation of $\Gamma; \Delta \Longrightarrow A$. Initial sequents are translated to the $\downarrow\uparrow$ coercion, and use of an unrestricted hypothesis follows by a substitution principle (Lemma 2.6). For right rules we apply the corresponding introduction rules. For left rules we either directly construct a derivation of the conclusion after an appeal to the induction hypothesis ($\otimes$L, **1**L, $\otimes$L, **0**L, $\exists$L, !L) or we appeal to a substitution principle of atomic natural deductions for hypotheses ($\multimap$ L, &L_1, &L_2, $\forall$L, $\supset$L). □

Theorem 2.10 (Completeness of Sequent Derivations)

1. *If* $\Gamma; \Delta \vdash A \uparrow$ *then there is a sequent derivation of* $\Gamma; \Delta \Longrightarrow A$, *and*
2. *if* $\Gamma; \Delta \vdash A \downarrow$ *then for any formula* C *and derivation of* $\Gamma; \Delta', A \Longrightarrow C$ *there is a derivation of* $\Gamma; (\Delta' \times \Delta) \Longrightarrow C$.

Proof: By simultaneous induction on the structure of the derivations of $\Gamma; \Delta \vdash A \uparrow$ and $\Gamma; \Delta \vdash A \downarrow$. □

2.7 Another Example: Distributed Systems

Another class of examples for linear logic is the description of concurrent systems. Linear logic can be used to represent whole classes of concurrent systems, such as Petri Nets [MOM91] or Milner's π-calculus [MPP92]. At present we are concerned only with the basic principles. We also now employ sequent derivations instead of natural deductions to model computations.

Unlike the planning example, in distributed computation there is no overall goal, just an evolution of state. Thus the right-hand side of the judgment should be "empty", which we model by **0**, the impossible goal. Thus the basic representation of a distributed system is

$$\Gamma_0; \Delta \Longrightarrow \mathbf{0}$$

where Γ_0 are the rules of computation and Δ is the state (including the processes, messages, *etc.*). A partial derivation

$$\begin{array}{c} \Gamma_0; \Delta_1 \Longrightarrow \mathbf{0} \\ \vdots \\ \Gamma_0; \Delta_0 \Longrightarrow \mathbf{0} \end{array}$$

represents a computation from Δ_0 to Δ_1.

We consider a simple example with the following atomic propositions.

$\mathrm{send}(x, y, m)$	x is sending the message m to y
$\mathrm{message}(x, y, m)$	message m from x to y is in transit
$\mathrm{listen}(y)$	y is listening for messages addressed to y
$\mathrm{received}(y, x, m)$	y has received message m from x

The computation rules are linear implications, available as unrestricted hypotheses.

$$\begin{array}{rcl} \mathrm{sendMsg} & : & \forall x.\, \forall y.\, \forall m.\, \mathrm{send}(x, y, m) \multimap \mathrm{message}(x, y, m) \\ \mathrm{receiveMsg} & : & \forall x.\, \forall y.\, \forall m.\, \mathrm{message}(x, y, m) \otimes \mathrm{listen}(y) \\ & & \qquad \multimap \mathrm{listen}(y) \otimes \mathrm{received}(y, x, m) \end{array}$$

Here, y continues to listen after it has received and stored a message. However, the receiver cannot distinguish the order in which messages were sent or received. This can be modelled by explicitly adding time stamps to the predicates above, or by using *non-commutative linear logic* (see Chapter **??**). Protocols for communication which require acknowledgments and other complex exchanges can be modelled based on the simple ideas above. For example, to express that a message may be lost, we can add the following rule.

$$\mathrm{loseMsg} \quad : \quad \mathrm{message}(x, y, m) \multimap \mathbf{1}$$

Here and below we omit universal quantifiers for the sake of brevity: all free variables in a rule are implicitly universally quantified on the outside.

Other connectives also have interesting computational interpretations. For example, a global abort message from x can be implemented using $\mathbf{0}$.

$$\mathrm{abortSys} \quad : \quad \mathrm{abort}(x) \otimes \mathrm{authorized}(x) \multimap \mathbf{0}$$

If x is authorized and aborts, we obtain $\mathbf{0}$ as part of the state from which we can prove anything and terminate the computation. However, there is nothing in the reading of derivations as deductions which would force this to be “immediate”: other computations could still proceed.

Alternative conjunction represents non-deterministic choice. Since we intend that any derivation represents a legal computation, this means a resource $A \& B$ could evolve to either A or B. For example, if storing a message might fail in the sense that it simply disappears, we can specify:

$$\mathrm{receiveMsg}' \quad : \quad \mathrm{message}(x, y, m) \otimes \mathrm{listen}(y) \multimap \mathrm{listen}(y) \otimes (\mathrm{received}(y, x, m) \& \mathbf{1})$$

Quantifiers can also be used to advantage. For example, we can send a message to anyone.

$$\text{sendMsgAny} \quad : \quad \text{sendany}(x, m) \multimap (\forall y.\, \text{message}(x, y, m))$$

However, this message can only ever be seen by one recipient. If we want to publish a message so everyone can see it, and see it as often as they like without storing it locally, we can specify:

$$\text{publishMsg} \quad : \quad \text{publish}(x, m) \multimap\, !(\forall y.\, \text{message}(x, y, m))$$

Some protocols establish "new connections", and some security protocols require the sender to generate a "fresh" message which has never been seen before. We can model both of these with an existential quantifier, since its left rule will introduce a new parameter, which may not occur in the present state.

$$\text{sendFresh} \quad : \quad \text{fresh}(x, y) \multimap \exists m.\, \text{message}(x, y, m)$$

The reader is invited to verify how sequent derivations, constructed in a bottom-up fashion, model computations. In each case we match the left-hand side of a linear implication in Γ_0 against components of the state, and then add the components of the right-hand side.

2.8 Deductions with Lemmas

One common way to find or formulate a proof is to introduce a lemma. In the sequent calculus, the introduction and use of a lemma during proof search is modelled by the rules of cut, *Cut* for lemmas used as linear hypotheses, and *Cut*! for lemmas used unrestrictedly. The corresponding rule for intuitionistic logic is due to Gentzen [Gen35]. We write $\Gamma; \Delta \overset{+}{\Longrightarrow} A$ for the judgment that A can be derived with the rules from before, plus one of the two cut rules below.

$$\frac{\Gamma; \Delta \overset{+}{\Longrightarrow} A \qquad \Gamma; (\Delta', A) \overset{+}{\Longrightarrow} C}{\Gamma; \Delta' \times \Delta \overset{+}{\Longrightarrow} C}\, \mathit{Cut} \qquad \frac{\Gamma; \cdot \overset{+}{\Longrightarrow} A \qquad (\Gamma, A); \Delta' \overset{+}{\Longrightarrow} C}{\Gamma; \Delta' \overset{+}{\Longrightarrow} C}\, \mathit{Cut}!$$

Note that the linear context in the left premiss of the *Cut*! rule must be empty, because the new hypothesis A in the right premiss is unrestricted in its use.

On the side of natural deduction, these rules correspond to substitution principles. They can be related to normal and atomic derivations only if we allow an additional coercion from normal to atomic derivations. This is because the left premiss corresponds to a derivation of $\Gamma; \Delta \vdash A \uparrow$ which can be substituted into a derivation of $\Gamma; \Delta', A \vdash C \uparrow$ only have the additional coercion has been applied. Of course, the resulting deductions are no longer normal in the sense we defined before, so we write $\Gamma; \Delta \vdash^{+} A \downarrow$ and $\Gamma; \Delta \vdash^{+} A \uparrow$. These judgments are defined with the same rules as $\Gamma; \Delta \vdash A \uparrow$ and $\Gamma; \Delta \vdash A \downarrow$, plus the following coercion.

$$\frac{\Gamma; \Delta \vdash^+ A \uparrow}{\Gamma; \Delta \vdash^+ A \downarrow} \uparrow\downarrow$$

It is now easy to prove that arbitrary natural deductions can be annotated with $\uparrow$ and $\downarrow$, since we can arbitrarily coerce back and forth between the two judgments.

Theorem 2.11 *If* $\Gamma; \Delta \vdash A$ *then* $\Gamma; \Delta \vdash^+ A \uparrow$ *and* $\Gamma; \Delta \vdash^+ A \downarrow$

Proof: By induction on the structure of $\mathcal{D} :: (\Gamma; \Delta \vdash A)$. □

Theorem 2.12

1. *If* $\Gamma; \Delta \vdash^+ A \uparrow$ *then* $\Gamma; \Delta \vdash A$.
2. *If* $\Gamma; \Delta \vdash^+ A \downarrow$ *then* $\Gamma; \Delta \vdash A$.

Proof: My mutual induction on $\mathcal{N} :: (\Gamma; \Delta \vdash^+ A \uparrow)$ and $\mathcal{A} :: (\Gamma; \Delta \vdash^+ A \downarrow)$. □

It is also easy to relate the Cut rules to the new coercions (and thereby to natural deductions), plus four substitution principles.

Property 2.13

1. *If* $\Gamma; (\Delta', w{:}A) \vdash^+ C \uparrow$ *and* $\Gamma; \Delta \vdash^+ A \downarrow$ *then* $\Gamma; (\Delta' \times \Delta) \vdash^+ C \uparrow$.
2. *If* $\Gamma; (\Delta', w{:}A) \vdash^+ C \downarrow$ *and* $\Gamma; \Delta \vdash^+ A \downarrow$ *then* $\Gamma; (\Delta' \times \Delta) \vdash^+ C \downarrow$.
3. *If* $(\Gamma, u{:}A); \Delta' \vdash^+ C \uparrow$ *and* $\Gamma; \cdot \vdash^+ A \downarrow$ *then* $\Gamma; \Delta' \vdash^+ C \uparrow$.
4. *If* $(\Gamma, u{:}A); \Delta' \vdash^+ C \downarrow$ *and* $\Gamma; \cdot \vdash^+ A \downarrow$ *then* $\Gamma; \Delta' \vdash^+ C \downarrow$.

Proof: By mutual induction on the structure of the given derivations. □

We can now extend Theorems 2.9 and 2.10 to relate sequent derivations with Cut to natural deductions with explicit lemmas.

Theorem 2.14 (Soundness of Sequent Derivations with Cut)
If $\Gamma; \Delta \stackrel{+}{\Longrightarrow} A$ *then* $\Gamma; \Delta \vdash^+ A \uparrow$.

Proof: As in Theorem 2.9 by induction on the structure of the derivation of $\Gamma; \Delta \stackrel{+}{\Longrightarrow} A$. An inference with one of the new rules *Cut* or *Cut*! is translated into an application of the $\uparrow\downarrow$ coercion followed by an appeal to one of the substitution principles in Property 2.13. □

Theorem 2.15 (Completeness of Sequent Derivations with Cut)

1. *If* $\Gamma; \Delta \vdash^+ A \uparrow$ *then there is a sequent derivation of* $\Gamma; \Delta \stackrel{+}{\Longrightarrow} A$, *and*

2. *if* $\Gamma; \Delta \vdash^{+} A \downarrow$ *then for any formula* C *and derivation of* $\Gamma; (\Delta', A) \stackrel{+}{\Longrightarrow} C$ *there is a derivation of* $\Gamma; (\Delta' \times \Delta) \stackrel{+}{\Longrightarrow} C$.

Proof: As in the proof of Theorem 2.10 by induction on the structure of the given derivations. In the new case of the $\uparrow\downarrow$ coercion, we use the rule of *Cut*. The other new rule, *Cut*!, is not needed for this proof, but is necessary for the proof of admissibility of cut in the next section. □

2.9 Cut Elimination

We viewed the sequent calculus as a calculus of proof search for natural deduction. The proofs of the soundness theorems 2.10 and 2.15 provide ways to translate cut-free sequent derivations into normal natural deductions, and sequent derivations with cut into arbitrary natural deductions.

This section is devoted to showing that the two rules of cut are redundant in the sense that any derivation in the sequent calculus which makes use of the rules of cut can be translated to one that does not. Taken together with the soundness and completeness theorems for the sequent calculi with and without cut, this has many important consequences.

First of all, a proof search procedure which looks only for cut-free sequent derivations will be complete: any derivable proposition can be proven this way. When the cut rule

$$\frac{\Gamma; \Delta \stackrel{+}{\Longrightarrow} A \qquad \Gamma; \Delta', A \stackrel{+}{\Longrightarrow} C}{\Gamma; \Delta' \times \Delta \stackrel{+}{\Longrightarrow} C}\ \mathit{Cut}$$

is viewed in the bottom-up direction the way it would be used during proof search, it introduces a new and arbitrary proposition A. Clearly, this introduces a great amount of non-determinism into the search. The cut elimination theorem now tells us that we never need to use this rule. All the remaining rules have the property that the premisses contain only instances of propositions in the conclusion, or parts thereof. This latter property is often called the *subformula property*.

Secondly, it is easy to see that the logic is *consistent*, that is, not every proposition is provable. In particular, the sequent $\cdot; \cdot \Longrightarrow \mathbf{0}$ does not have a cut-free derivation, because there is simply no rule which could be applied to infer it! This property clearly fails in the presence of cut: it is *prima facie* quite possible that the sequent $\cdot; \cdot \stackrel{+}{\Longrightarrow} \mathbf{0}$ is the conclusion of the cut rule.

Along the same lines, we can show that a number of propositions are *not derivable* in the sequent calculus and therefore not true as defined by the natural deduction rules. Examples of this kind are given at the end of this section.

We prove cut elimination by showing that the two cut rules are *admissible rules of inference* in the sequent calculus without cut. An inference rule is admissible if whenever we can find derivations for its premisses we can find a derivation of its conclusion. This should be distinguished from a *derived rule of* *inference* which requires a direct derivation of the conclusion from the premisses. We can also think of a derived rule as an evident hypothetical judgment where the premisses are (unrestricted) hypotheses.

Derived rules of inference have the important property that they remain evident under any extension of the logic. An admissible rule, on the other hand, represents a global property of the deductive system under consideration and may well fail when the system is extended. Of course, every derived rule is also admissible.

Theorem 2.16 (Admissibility of Cut)

1. *If* $\Gamma; \Delta \Longrightarrow A$ *and* $\Gamma; (\Delta', A) \Longrightarrow C$ *then* $\Gamma; \Delta' \times \Delta \Longrightarrow C$.
2. *If* $\Gamma; \cdot \Longrightarrow A$ *and* $(\Gamma, A); \Delta' \Longrightarrow C$ *then* $\Gamma; \Delta' \Longrightarrow C$.

Proof: By nested inductions on the structure of the cut formula A and the given derivations, where induction hypothesis (1) has priority over (2). To state this more precisely, we refer to the given derivations as $\mathcal{D} :: (\Gamma; \Delta \Longrightarrow A)$, $\mathcal{D}' :: (\Gamma; \cdot \Longrightarrow A)$, $\mathcal{E} :: (\Gamma; (\Delta, A) \Longrightarrow C)$, and $\mathcal{E}' :: ((\Gamma, A); \Delta' \vdash C)$. Then we may appeal to the induction hypothesis whenever

a. the cut formula A is strictly smaller, or

b. the cut formula A remains the same, but we appeal to induction hypothesis (1) in the proof of (2) (but when we appeal to (2) in the proof of (1) the cut formula must be strictly smaller), or

c. the cut formula A and the derivation $\mathcal{E}$ remain the same, but the derivation $\mathcal{D}$ becomes smaller, or

d. the cut formula A and the derivation $\mathcal{D}$ remain the same, but the derivation $\mathcal{E}$ or $\mathcal{E}'$ becomes smaller.

Here, we consider a formula smaller it is an immediate subformula, where $[t/x]A$ is considered a subformula of $\forall x.\, A$, since it contains fewer quantifiers and logical connectives. A derivation is smaller if it is an immediate subderivation, where we allow weakening by additional unrestricted hypothesis in one case (which does not affect the structure of the derivation).

The cases we have to consider fall into 5 classes:

Initial Cuts: One of the two premisses is an initial sequent. In these cases the cut can be eliminated directly.

Principal Cuts: The cut formula A was just inferred by a right rule in $\mathcal{D}$ and by a left rule in $\mathcal{E}$. In these cases we appeal to the induction hypothesis (possibly several times) on smaller cut formulas (item (a) above).

Dereliction Cut: The cases for the *Cut*! rule are treated as right commutative cuts (see below), except for the rule of dereliction which requires an appeal to induction hypothesis (1) with the same cut formula (item (b) above).

Left Commutative Cuts: The cut formula A is a side formula of the last inference in $\mathcal{D}$. In these cases we may appeal to the induction hypotheses with the same cut formula, but smaller derivation $\mathcal{D}$ (item (c) above).

Right Commutative Cuts: The cut formula A is a side formula of the last inference in $\mathcal{E}$. In these cases we may appeal to the induction hypotheses with the same cut formula, but smaller derivation $\mathcal{E}$ or $\mathcal{E}'$ (item (d) above).

[*Some cases to be filled in later.*]

□

Using the admissibility of cut, the cut elimination theorem follows by a simple structural induction.

Theorem 2.17 (Cut Elimination)
If $\Gamma; \Delta \stackrel{+}{\Longrightarrow} C$ *then* $\Gamma; \Delta \Longrightarrow C$.

Proof: By induction on the structure of $\mathcal{D} :: (\Gamma; \Delta \stackrel{+}{\Longrightarrow} C)$. In each case except *Cut* or *Cut*! we simply appeal to the induction hypothesis on the derivations of the premisses and use the corresponding rule in the cut-free sequent calculus. For the *Cut* and *Cut*! rules we appeal to the induction hypothesis and then admissibility of cut (Theorem 2.16) on the resulting derivations. □

2.10 Consequences of Cut Elimination

As a first consequence, we see that linear logic is *consistent*: not every proposition can be proved. A proof of consistency for both intuitionistic and classical logic was Gentzen's original motivation for the development of the sequent calculus and his proof of cut elimination.

Theorem 2.18 (Consistency of Intuitionistic Linear Logic)
$\cdot; \cdot \vdash \mathbf{0}$ *is not derivable.*

Proof: If the judgment were derivable, by Theorems 2.11, 2.15, and 2.17, there must be a cut-free sequent derivation of $\cdot; \cdot \Longrightarrow \mathbf{0}$. But there is no rule with which we could infer this sequent (there is no right rule for $\mathbf{0}$), and so it cannot be derivable. □

A second consequence is that every natural deduction can be translated to a normal natural deduction. The necessary construction is implicit in the proofs of the soundness and completeness theorems for sequent calculi and the proofs of admissibility of cut and cut elimination. In Chapter 4 we will see a much more direct, but in other respects more complicated proof.

Theorem 2.19 (Normalization for Natural Deductions)
If $\Gamma; \Delta \vdash A$ *then* $\Gamma; \Delta \vdash A \uparrow$.

Proof: Directly, using theorems from this chapter. Assume $\Gamma; \Delta \vdash A$. Then

$\Gamma; \Delta \vdash^{+} A$ by Theorem 2.11,
$\Gamma; \Delta \overset{+}{\Longrightarrow} A$ by completeness of sequent derivations with cut (Theorem 2.15),
$\Gamma; \Delta \Longrightarrow A$ by cut elimination (Theorem 2.17), and
$\Gamma; \Delta \vdash A \uparrow$ by soundness of cut-free sequent derivations (Theorem 2.9).

□

2.11 Exercises

Exercise 2.1 Give a counterexample which shows that the elimination $\supset$E would be locally unsound if its second premiss were allowed to depend on linear hypotheses.

Exercise 2.2 If we *define* intuitionistic implication $A \supset B$ in linear logic as an abbreviation for $(!A) \multimap B$, then the given introduction and elimination rules become *derived rules of inference*. Prove this by giving a derivation for the conclusion of the $\supset$E rule from its premisses under the interpretation, and similarly for the $\supset$I rule.

For the other direction, show how $!A$ could be defined from intuitionistic implication or speculate why this might not be possible.

Exercise 2.3 [*To be filled in: an exercise exploring the "missing connectives" of multiplicative disjunction and additive implication.*]

Exercise 2.4 In the blocks world example from Section 2.3, sketch the derivation for the same goal A_0 and initial situation Δ_0 in which block b is put on block c, rather than the table.

Exercise 2.5 Model the *Towers of Hanoi* in linear logic in analogy with our modelling of the blocks world.

1. Define the necessary atomic propositions and their meaning.
2. Describe the legal moves in *Towers of Hanoi* as unrestricted hypotheses Γ_0 independently from the number of towers or disks.
3. Represent the initial situation of three towers, where two are empty and one contains two disks in a legal configuration.
4. Represent the goal of legally stacking the two disks on some arbitrary other tower.
5. Sketch the proof for the obvious 3-move solution as in Section 2.3.

Exercise 2.6 Consider if $\otimes$ and $\&$ can be distributed over $\oplus$ or *vice versa*. There are four different possible equivalences based on eight possible entailments. Give natural deductions for the entailments which hold.

Exercise 2.7 In this exercise we explore distributive and related *interaction laws* for linear implication. In intuitionistic logic, for example, we have the following $(A \wedge B) \supset C \dashv\vdash A \supset (B \supset C)$ and $A \supset (B \wedge C) \dashv\vdash (A \supset B) \wedge (A \supset C)$, where $\dashv\vdash$ is mutual entailment as in Exercise 1.2.

In linear logic, we now write $A \dashv\vdash A'$ for linear mutual entailment, that is, A' follows from linear hypothesis A and *vice versa*. Write out appropriate interaction laws or indicate none exists, for each of the following propositions.

1. $A \multimap (B \otimes C)$
2. $(A \otimes B) \multimap C)$
3. $A \multimap \mathbf{1}$
4. $\mathbf{1} \multimap A$
5. $A \multimap (B \& C)$
6. $(A \& B) \multimap C$
7. $A \multimap \top$
8. $\top \multimap A$
9. $A \multimap (B \oplus C)$
10. $(A \oplus B) \multimap C$
11. $A \multimap \mathbf{0}$
12. $\mathbf{0} \multimap A$
13. $A \multimap (B \multimap C)$
14. $(A \multimap B) \multimap C$

Note that an interaction law exists only if there is a mutual linear entailment—we are not interested if one direction holds, but not the other.

Give the derivations in both directions for one of the interaction laws of a binary connective $\otimes$, $\&$, $\oplus$, or $\multimap$, and for one of the interaction laws of a logical constant $\mathbf{1}$, $\top$, or $\mathbf{0}$.

Exercise 2.8 Extend the interaction laws from Exercise 2.7 by laws showing how linear implication interacts with existential and universal quantification.

Exercise 2.9 Design an alternative translation $()^*$ from formulas and natural deductions in intuitionistic logic to intuitionistic linear logic in which conjunction ($\wedge$) and truth ($\top$) are mapped to simultaneous conjunction ($\otimes$) and its unit ($\mathbf{1}$) instead of the additive connectives as in $()^+$. Prove the correctness of the embedding and discuss the relative merits of the two translations.

Exercise 2.10 Extend the embedding from from Section 2.4 to encompass intuitionistic propositions $\neg A$ without adding any connectives to the linear logic. Modify the statements and proofs of embedding and conservativity (if necessary) and show the proof cases concerned with negation.

Exercise 2.11 Find a derivation $\mathcal{D} :: (\Gamma; \Delta \vdash A)$ which contains no local redex, but which is not normal in the sense that there is no derivation $\mathcal{N} :: (\Gamma; \Delta \vdash A \uparrow)$ such that $\mathcal{N}^{-} = \mathcal{D}$.

Exercise 2.12 Internalize the notion of mutual linear entailment from Exercise 2.7 as a new linear connective $A \multimapboth A'$.

1. Give introduction and elimination rules. Your rules should be orthogonal to all other connectives and not mention, for example, linear implication.
2. Are your rules locally sound and complete? Give the local reduction and expansions, if they exist.
3. Annotate your rules, extending the definitions of normal and atomic derivations.
4. Give right and left sequent rules corresponding to the introduction and elimination rules, respectively.
5. Show the new cases in the proofs of soundness and completeness of the sequent calculus with respect to natural deduction (Theorems 2.9 and 2.10).
6. Show a new principal case in the proof of admissibility of cut (Theorem 2.16).
7. Would you classify the new connective as multiplicative, additive, or exponential? Can it be defined from the linear connectives introduced in Sections 2.1 and 2.2 in such a way that your introduction and elimination rules become derived rules of inference? If so, give the definition, if not explain informally why it is not possible.

Chapter 3

Proof Search

Linear logic as introduced by Girard and presented in the previous chapter is a rich system for the formalization of reasoning involving state. It conservatively extends intuitionistic logic and can therefore also serve as the logical basis for general constructive mathematics. Searching for proofs in such an expressive logic is difficult, and one should not expect silver bullets.

Depending on the problem, proof search in linear logic can have a variety of applications. In the domain of planning problems (see Section 2.3) searching for a proof means searching for a plan. In the domain of concurrent computation (see Section 2.7) searching for a proof means searching for possible computations. In the domain of logic programming (which we investigate in detail in Chapter **??**), searching for a proof according to a fixed strategy is the basic paradigm of computation. In the domain of functional programming and type theory (which we investigate in Chapter 4), searching for a proof means searching for a program satisfying a given specification.

Each application imposes different requirements on proof search, but there are underlying basic techniques which recur frequently. In this chapter we take a look at some basic techniques, to be exploited in subsequent chapters.

3.1 Bottom-Up Proof Search and Inversion

The literature is not in agreement on the terminology, but we refer to the process of creating a derivation from the desired judgment on upward as *bottom-up* proof search. A snap-shot of a bottom-up search is a partial derivation, with undecided judgments at the top. Our goal is to derive all remaining judgments, thereby completing a proof.

We proceed by selecting a judgment which remains to be derived and an inference rule with which it might be inferred. We also may need to determine exactly how the conclusion of the rule matches the judgment. For example, in the $\otimes$R rule we need to decide how to split the linear hypotheses between the two premisses. After these choices have been made, we reduce the goal of deriving the judgment to a number of subgoals, one for each premiss of the selected rule. If there are no premisses, the subgoal is solved. If there are no subgoals left, we have derived the original judgment.

Using this simple intuition, the cut elimination theorem (Theorem 2.17) directly implies decidability of pure propositional linear logic as defined in Section 2.1, that is, linear logic without unrestricted resources and without the exponential connectives *of course* "$!A$" and intuitionistic implication "$A \supset B$".

Theorem 3.1 (Decidability of Propositional Pure Linear Logic) *Pure linear logic with connectives $\multimap$, $\otimes$, $\mathbf{1}$, $\&$, $\top$, $\oplus$, and $\mathbf{0}$ is decidable.*

Proof: We know by cut elimination and other results from Chapter 2.2 that $\cdot; \Delta \vdash A$ iff $\cdot; \Delta \Longrightarrow A$. Every premiss of every sequent rule in the pure fragment of linear logic without cut contains fewer connectives and quantifiers than the conclusion. Every branch must therefore be finite. Furthermore, there are only finitely many different inference rules which can be used to infer any given conclusion, and every rule has at most two premisses. Therefore, the space of possible cut-free sequent derivations of a purely linear judgment is finite and derivability is decidable. □

After Section 3.2 we see that this theorem still holds even if we admit quantifiers, but that it fails if we allow unrestricted hypotheses (even without quantifiers).

The second observation about bottom-up proof search is that some rules are *invertible*, that is, the premisses are derivable whenever the conclusion is derivable. The usual direction states that the conclusion is evident whenver the premisses are. Invertible rules can safely be applied whenever possible without losing completeness, although some care must be taken to retain a terminating procedure in the presence of unrestricted hypotheses. We also separate *weakly invertible* rules, which only apply when there are no linear hypotheses (besides possibly the principal proposition of the inference rule). For example, we cannot apply the $\mathbf{1}$R whenever the judgment is $\Gamma; \Delta \vdash \mathbf{1}$, although it is safe to do so when there are no linear hypotheses. Similarly, we cannot use the initial sequent rule to infer $\Gamma; \Delta, A \Longrightarrow A$ unless $\Delta = \cdot$. Strongly invertible rules apply regardless of any other hypotheses.

Theorem 3.2 (Inversion Lemmas) *The following table lists invertible, weakly invertible, and non-invertible rule in intuitionistic linear logic.*

Strongly Invertible	*Weakly Invertible*	*Not Invertible*
$\multimap$R		$\multimap$L
$\otimes$L, $\mathbf{1}$L	$\mathbf{1}$R	$\otimes$R
$\&$R, $\top$R		$\&\mathrm{L}_1, \&\mathrm{L}_2$
$\oplus$L, $\mathbf{0}$L		$\oplus\mathrm{R}_1, \oplus\mathrm{R}_2$
$\forall$R, $\exists$L		$\forall$L, $\exists$R
$\supset$R, !L	!R	$\supset$L
	I	DL

Proof: For invertible rule we prove that each premiss follows from the conclusion. For non-invertible rules we give a counterexample. The two sample case below are representative: for invertible rules we apply admissibility of cut, for non-invertible rules we consider a sequent with the same proposition on the left and right.

Case: $\multimap$R is invertible. We have to show that $\Gamma; (\Delta, A) \Longrightarrow B$ is derivable whenver $\Gamma; \Delta \Longrightarrow A \multimap B$ is derivable, so we assume $\Gamma; \Delta \Longrightarrow A \multimap B$. We also have $\Gamma; (\cdot, A, A \multimap B) \Longrightarrow B$, which follows by one $\multimap$L rule from two initial sequents. From the admissibility of cut (Theorem 2.16) we then obtain directly $\Gamma; (\Delta, A) \Longrightarrow B$.

Case: $\multimap$L is not invertible. Consider $\cdot; (\cdot, A \multimap B) \Longrightarrow A \multimap B$ for parameters A and B. There is only one way to use $\multimap$L to infer this, which leads to $\cdot; \cdot \Longrightarrow A$ and $\cdot; (\cdot, B) \Longrightarrow A \multimap B$, neither of which is derivable. Therefore $\multimap$L is not invertible in general.

□

As a final, general property for bottom-up proof search we show that we can restrict ourselved to initial sequents of the form $\Gamma; (\cdot, P) \Longrightarrow P$, where P is an atomic proposition. We write $\Gamma; \Delta \overset{-}{\Longrightarrow} A$ for the restricted judgment whose rules are as for $\Gamma; \Delta \Longrightarrow A$, except that initial sequents are restricted to atomic propositions. Obviously, if $\Gamma; \Delta \overset{-}{\Longrightarrow} A$ then $\Gamma; \Delta \Longrightarrow A$.

Theorem 3.3 (Completeness of Atomic Initial Sequents) *If* $\Gamma; \Delta \Longrightarrow A$ *then* $\Gamma; \Delta \overset{-}{\Longrightarrow} A$.

Proof: By induction on the the structure of $\mathcal{D} :: (\Gamma; \Delta \Longrightarrow A)$. In each case except initial sequents, we appeal directly to the induction hypothesis and infer $\Gamma; \Delta \overset{-}{\Longrightarrow} A$ from the results. For initial sequents, we use an auxiliary induction on the structure of the formula A. We show only one case—the others are similar in that they follow the local expansions, translated from natural deduction to the setting of the sequent calculus. If local completeness did not hold for a connective, then atomic initial sequents would be incomplete as well.

Case: $\mathcal{D} = \dfrac{}{\Gamma; (\cdot, A_1 \otimes A_2) \Longrightarrow A_1 \otimes A_2}\,\mathrm{I}$, where $A = A_1 \otimes A_2$. Then we construct

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_1' \\ \Gamma; (\cdot, A_1) \overset{-}{\Longrightarrow} A_1\end{array} \qquad \begin{array}{c}\mathcal{D}_2' \\ \Gamma; (\cdot, A_2) \overset{-}{\Longrightarrow} A_2\end{array}}{\Gamma; (\cdot, A_1, A_2) \overset{-}{\Longrightarrow} A_1 \otimes A_2}\,\otimes\mathrm{R}}{\Gamma; (\cdot, A_1 \otimes A_2) \overset{-}{\Longrightarrow} A_1 \otimes A_2}\,\otimes\mathrm{L}$$

where $\mathcal{D}_1'$ and $\mathcal{D}_2'$ exist by induction hypothesis on A_1 and A_2.

□

The theorems in this section lead to a search procedure with the following general outline:

1. Pick a subgoal to solve.
2. Decide to apply a right rule to the consequent or a left rule to a hypothesis.
3. Determine the remaining parameters (either how to split the hypotheses, or on the terms which may be required).
4. Apply the rule in the backward direction, reducing the goal to possibly several subgoals.

A lot of choices remain in this procedure. They can be classified according to the type of choice which must be made. This classification will guide us in the remainder of this chapter, as we discuss how to reduce the inherent non-determinism in the procedure above.

- Conjunctive choices. We know all subgoals have to be solved, but the order in which we attempt to solve them is not determined. In the simplest case, this is a form of *don't-care non-determinism*, since all subgoals have to be solved. In practice, it is not that simple since subgoals may interact once other choices have been made more deterministic. Success is a special case of conjunctive choice with no conjuncts.
- Disjunctive choices. We don't know which left or right rule to apply. Invertible rules are always safe, but once they all have been applied, many possibilities may remain. This is a form of *don't-know non-determinism*, since a sequence of correct guesses will lead to a derivation if there is one. In practice, this may be solved via backtracking, for example. Failure is a special case of a disjunctive choice with zero alternatives.
- Universal choices. In the $\forall$R and $\exists$L rules we have to choose a new parameter. Fortunately, this is a trivial choice, since *any* new parameter will work, and its name is not important. Hence this is a form of don't-care non-determinism.
- Existential choices. In the $\exists$R and $\forall$L rules we have to choose a term t to substitute for the bound variable. Since there are potentially infinitely many terms (depending on the domain of quantification), this is a form of don't-know non-determinism. In practice, this is solved by *unification*, discussed in the next section.

3.2 Unification

When proving a proposition of the form $\exists x.\ A$ by its right rule in the sequent calculus, we must supply a term t and then prove $[t/x]A$. The domain of quantification may include infinitely many terms (such as the natural numbers), so this choice cannot be resolved simply by trying all possible terms t. Similarly, when we use a hypothesis of the form $\forall x.\ A$ we must supply a term t to substitute for x.

Fortunately, there is a better technique called *unification* which is sound and complete for syntactic equality between terms. The basic idea is quite simple: we postpone the choice of t and instead substitute a new *existential variable* (often called *meta-variable* or *logic variable*) X for x and continue with the bottom-up construction of a derivation. When we reach initial sequents we check if there is a substitution for the existential variables such that the hypothesis matches the conclusion. If so, we apply this instantiation globally to the partial derivation and continue to search for proofs of other subgoals. Finding an instantiation for existential variables under which two propositions or terms match is called *unification*. It is decidable if a unifying substitution or *unifier* exists, and if so, we can effectively compute it in linear time. Moreover, we can do so with a minimal commitment and we do not need to choose between various possible unifiers.

Because of its central importance, unification has been thoroughly investigated. Herbrand [Her30] is given credit for the first description of a unification algorithm in a footnote of his thesis, but it was not until 1965 that it was introduced into automated deduction through the seminal work by Alan Robinson [Rob65, Rob71]. The first algorithms were exponential, and later almost linear [Hue76, MM82] and linear algorithms [MM76, PW78] were discovered. In the practice of theorem proving, generally variants of Robinson's algorithm are still used, due to its low constant overhead on the kind of problems encountered in practice. For further discussion and a survey of unification, see [Kni89]. We describe a variant of Robinson's algorithm.

Before we describe the unification algorithm itself, we relate it to the problem of proof search. For this we use a general method of *residuation*. We enrich the judgment $\Gamma; \Delta \overset{-}{\Longrightarrow} A$ by a *residual proposition* F such that

1. if $\Gamma; \Delta \overset{-}{\Longrightarrow} A$ then $\Gamma; \Delta \overset{-}{\Longrightarrow} A \setminus F$ and F is true, and
2. if $\Gamma; \Delta \overset{-}{\Longrightarrow} A \setminus F$ and F is true then $\Gamma; \Delta \overset{-}{\Longrightarrow} A$.

Generally, we cannot prove such properties directly by induction, but we need to generalize them, exhibiting the close relationship between the derivations of the sequents and residual formulas F.

Residual formulas F are amenable to specialized procedures such as unification, since they are drawn from a simpler logic or deductive system than the general propositions A. In practice they are often solved *incrementally* rather than collected throughout a derivation and only solved at the end. This is important for the early detection of failures during proof search. Incremental solution of residual formulas is the topic of Exercise **??**.

What do we need in the residual propositions so that existential choices and equalities between atomic propositions can be expressed? The basic proposition is one of equality between atomic propositions, $P_1 \doteq P_2$. We also have conjunction $F_1 \wedge F_2$, since equalities may be collected from several subgoals, and $\top$ if there are no residual propositions to be proven. Finally, we need the existential quantifier $\exists x.\, F$ to express the scope of existential variables, and $\forall x.\, F$ to express the scope of parameters introduced in a derivation. We add equality between terms, since it is required to describe the unification algorithm itself. We refer to the logic with these connectives as *unification logic*, defined via a deductive system.

$$\textit{Formulas} \quad F \quad ::= \quad P_1 \doteq P_2 \mid t_1 \doteq t_2 \mid F_1 \wedge F_2 \mid \top \mid \exists x.\, F \mid \forall x.\, F$$

The main judgment "F *is valid*", written $\models F$, is defined by the following rules, which are consistent with, but more specialized than the rules for these connectives in intuitionistic natural deduction (see Exercise **??**).

$$\frac{}{\models P \doteq P} \doteq \mathrm{I} \qquad\qquad \frac{}{\models t \doteq t} \doteq \mathrm{I}'$$

$$\frac{\models F_1 \qquad \models F_2}{\models F_1 \wedge F_2} \wedge\mathrm{I} \qquad\qquad \frac{}{\models \top} \top\mathrm{I}$$

$$\frac{\models [t/x]F}{\models \exists x.\, F} \exists\mathrm{I} \qquad\qquad \frac{\models [a/x]F}{\models \forall x.\, F} \forall\mathrm{I}^a$$

The $\forall\mathrm{I}^a$ rule is subject to the usual proviso that a is a new parameter not occurring in $\forall x.\, F$. There are no elimination rules, since we do not need to consider hypotheses of the form $\models F$, which is the primary reason for the simplicity of theorem proving in the unification logic.

We enrich the sequent calculus with residual formulas from the unification logic, postponing all existential choices. Recall that in practice we merge residuation and solution in order to discover unprovable residual formulas as soon as possible. This merging of the phases is not represented in our system.

Hypotheses. Initial sequents residuate an equality between its principal propositions. Any solution to the equation will unify P' and P, which means that this will translate to a correct application of the initial sequent rule in the original system.

$$\frac{}{\Gamma; P' \stackrel{-}{\Longrightarrow} P \setminus P' \doteq P} \mathrm{I} \qquad \frac{(\Gamma, A); (\Delta, A) \stackrel{-}{\Longrightarrow} C \setminus F}{(\Gamma, A); \Delta \stackrel{-}{\Longrightarrow} C \setminus F} \mathrm{DL}$$

Propositional Connectives. We just give a few sample rules for the connectives which do not involve quantifiers, since all of them simply propagate or combine unification formulas, regardless whether they are additive, multiplicative, or exponential.

$$\frac{\Gamma; \Delta, A \stackrel{-}{\Longrightarrow} B \setminus F}{\Gamma; \Delta \stackrel{-}{\Longrightarrow} A \multimap B \setminus F} \multimap\mathrm{R} \qquad \frac{\Gamma; \Delta_1 \stackrel{-}{\Longrightarrow} A \setminus F_1 \qquad \Gamma; \Delta_2, B \stackrel{-}{\Longrightarrow} C \setminus F_2}{\Gamma; \Delta_1 \times \Delta_2, A \multimap B \stackrel{-}{\Longrightarrow} C \setminus F_1 \wedge F_2} \multimap\mathrm{L}$$

$$\frac{}{\Gamma; \cdot \stackrel{=}{\Longrightarrow} \mathbf{1} \setminus \top}\,\mathbf{1}\mathrm{R} \qquad \frac{\Gamma; \Delta \stackrel{=}{\Longrightarrow} C \setminus F}{\Gamma; \Delta, \mathbf{1} \stackrel{=}{\Longrightarrow} C \setminus F}\,\mathbf{1}\mathrm{L}$$

Quantifiers. These are the critical rules. Since we residuate the existential choices entirely, the $\exists$R and $\forall$L rules instantiate a quantifier by a new *parameter*, which is existentially quantified in the residual formula in both cases. Similarly, the $\forall$R and $\exists$L rule introduce a parameter which is universally quantified in the residual formula.

$$\frac{\Gamma; \Delta \stackrel{=}{\Longrightarrow} [a/x]A \setminus [a/x]F}{\Gamma; \Delta \stackrel{=}{\Longrightarrow} \forall x.\ A \setminus \forall x.\ F}\,\forall\mathrm{R}^a \qquad \frac{\Gamma; \Delta, [a/x]A \stackrel{=}{\Longrightarrow} C \setminus [a/x]F}{\Gamma; \Delta, \forall x.\ A \stackrel{=}{\Longrightarrow} C \setminus \exists x.\ F}\,\forall\mathrm{L}^a$$

$$\frac{\Gamma; \Delta \stackrel{=}{\Longrightarrow} [a/x]A \setminus [a/x]F}{\Gamma; \Delta \stackrel{=}{\Longrightarrow} \exists x.\ A \setminus \exists x.\ F}\,\exists\mathrm{R}^a \qquad \frac{\Gamma; \Delta, [a/x]A \stackrel{=}{\Longrightarrow} C \setminus [a/x]F}{\Gamma; \Delta, \exists x.\ A \stackrel{=}{\Longrightarrow} C \setminus \forall x.\ A}\,\exists\mathrm{L}^a$$

The soundness of residuating equalities and existential choices in this manner is straightforward.

Theorem 3.4 (Soundness of Equality Residuation) *If* $\Gamma; \Delta \stackrel{=}{\Longrightarrow} A \setminus F$ *and* $\models F$ *then* $\Gamma; \Delta \stackrel{=}{\Longrightarrow} A$.

Proof: By induction on the structure of $\mathcal{R} :: (\Gamma; \Delta \stackrel{=}{\Longrightarrow} A \setminus F)$. We show the critical cases. Note how in the case of the $\exists$R rule the proof of $\models \exists x.\ F$ provides the essential witness term t.

Case: $\mathcal{R} = \dfrac{}{\Gamma; P' \stackrel{=}{\Longrightarrow} P \setminus P' \doteq P}\,\mathrm{I}.$

We know by assumption that $\models F$ which reads $\models P' \doteq P$. By inversion therefore $P' = P$ (since $\doteq$I is the only rule which applies to this judgment), and $\Gamma; P' \stackrel{=}{\Longrightarrow} P$ is a valid initial sequent.

Case: $\mathcal{R} = \dfrac{\begin{array}{c}\mathcal{R}_1 \\ \Gamma; \Delta \stackrel{=}{\Longrightarrow} [a/x]A_1 \setminus [a/x]F_1\end{array}}{\Gamma; \Delta \stackrel{=}{\Longrightarrow} \exists x.\ A_1 \setminus \exists x.\ F_1}\,\exists\mathrm{R}^a.$

By assumption, we have $\models \exists x.\ F_1$. By inversion, $\models [t/x]F_1$ for some t. By the proviso on the $\exists\mathrm{R}^a$ rule, $\mathcal{R}_1$ is parametric in a, so we can substitute t for a in this derivation an obtain $[t/a]\mathcal{R}_1 :: (\Gamma; \Delta \stackrel{=}{\Longrightarrow} [t/x]A_1 \setminus [t/x]F_1)$. Applying the induction hypothesis to $[t/a]\mathcal{R}_1$ yields a $\mathcal{D}_1$ and we construct

$$\frac{\begin{array}{c}\mathcal{D}_1 \\ \Gamma; \Delta \stackrel{=}{\Longrightarrow} [t/x]A_1\end{array}}{\Gamma; \Delta \stackrel{=}{\Longrightarrow} \exists x.\ A_1}\,\exists\mathrm{R}$$

Case: $$\mathcal{R} = \dfrac{\begin{array}{c}\mathcal{R}_1\\ \Gamma;\Delta \stackrel{-}{\Longrightarrow} [a/x]A_1 \setminus [a/x]F_1\end{array}}{\Gamma;\Delta \stackrel{-}{\Longrightarrow} \forall x.\ A_1 \setminus \forall x.\ F_1}\,\forall\mathrm{R}^a.$$

By assumption, we have $\models \forall x.\ F_1$. By inversion, $\models [b/x]F_1$ for a new parameter b, and therefore also $\models [a/x]F_1$ by substitution. Hence we can apply the induction hypothesis to obtain a $\mathcal{D}_1$ and construct

$$\dfrac{\begin{array}{c}\mathcal{D}_1\\ \Gamma;\Delta \stackrel{-}{\Longrightarrow} [a/x]A_1\end{array}}{\Gamma;\Delta \stackrel{-}{\Longrightarrow} \forall x.\ A_1}\,\forall\mathrm{R}^a$$

□

The opposite direction is more difficult. The desired theorem:

If $\Gamma;\Delta \stackrel{-}{\Longrightarrow} A$ *then* $\Gamma;\Delta \stackrel{-}{\Longrightarrow} A \setminus F$ *for some* F *with* $\models F$

cannot be proved directly by induction, since the premisses of the two derivations are different in the $\exists$R and $\forall$L rules. However, one can be obtained from the other by substituting terms for parameters. Since this must be done simultaneously, we introduce a new notation.

$$\textit{Parameter Substitution} \quad \rho \quad ::= \quad \cdot \mid \rho, t/a$$

We assume all the parameters a substituted for by ρ are distinct to avoid ambiguity. We write $[\rho]A$, $[\rho]F$, and $[\rho]\Gamma$, for the result of applying the substitution ρ to a proposition, formula, or context, respectively.

Lemma 3.5 *If* $\Gamma;\Delta \stackrel{-}{\Longrightarrow} A$ *and* $[\rho]A' = A$, $[\rho]\Delta' = \Delta$, *and* $[\rho]\Gamma' = \Gamma$, *then* $\Gamma';\Delta' \stackrel{-}{\Longrightarrow} A' \setminus F$ *for some* F *and* $\models [\rho]F$.

Proof: The proof proceeds by induction on the structure of $\mathcal{D} :: (\Gamma;\Delta \stackrel{-}{\Longrightarrow} A)$. We show only three cases, the second of which required the generalization of the induction hypothesis.

Case: $$\mathcal{D} = \dfrac{}{\Gamma;(\cdot,P) \stackrel{-}{\Longrightarrow} P}\,I$$

and $[\rho]\Gamma' = \Gamma$, $[\rho]\Delta' = (\cdot,P)$, and $[\rho]A' = P$. Therefore $\Delta' = (\cdot,P'')$ with $[\rho]P'' = P$ and $A' = P'$ with $[\rho]P' = P$ and we construct

$$\dfrac{}{\Gamma';(\cdot,P'') \stackrel{-}{\Longrightarrow} P' \setminus P'' \doteq P'}\,I \qquad \text{and} \qquad \dfrac{}{\models [\rho]P'' \doteq [\rho]P'}\,\doteq\mathrm{I}$$

Case: $$\mathcal{D} = \dfrac{\begin{array}{c}\mathcal{D}_1\\ \Gamma;\Delta \stackrel{-}{\Longrightarrow} [t/x]A_1\end{array}}{\Gamma;\Delta \stackrel{-}{\Longrightarrow} \exists x.\ A_1}\,\exists\mathrm{R}.$$

We assumed $[\rho]A' = \exists x.\ A_1$, so $A' = \exists x.\ A'_1$ and $[\rho, t/a]([a/x]A'_1) = [t/x]A_1$ for a new parameter a. Since a is new, $[\rho, t/a]\Gamma' = [\rho]\Gamma'$ and similarly for Δ', so we can apply the induction hypothesis to $\mathcal{D}_1$ to obtain $\mathcal{R}_1$ and $\mathcal{U}_1$ and construct

$$\dfrac{\begin{array}{c}\mathcal{R}_1\\ \Gamma';\Delta' \stackrel{=}{\Longrightarrow} [a/x]A'_1 \setminus [a/x]F_1\end{array}}{\Gamma';\Delta' \stackrel{=}{\Longrightarrow} \exists x.\ A'_1 \setminus \exists x.\ F_1}\exists\mathrm{R}^a \quad \text{and} \quad \dfrac{\begin{array}{c}\mathcal{U}_1\\ \models [\rho, t/a]([a/x]F_1)\end{array}}{\models [\rho]\exists x.\ F_1}\exists\mathrm{I}.$$

Case: $\mathcal{D} = \dfrac{\begin{array}{c}\mathcal{D}_1\\ \Gamma;\Delta \stackrel{=}{\Longrightarrow} [a/x]A_1\end{array}}{\Gamma;\Delta \stackrel{=}{\Longrightarrow} \forall x.\ A_1}\forall\mathrm{R}^a.$

We assume $[\rho]A' = \forall x.\ A_1$, so $A' = \forall x.\ A'_1$ and $[\rho, a/a']([a'/x]A'_1) = [a/x]A_1$ for an a' new in Γ', Δ' and $\forall x.\ A'_1$. We can then appeal to the induction hypothesis on $\mathcal{D}_1$ to obtain $\mathcal{R}_1$ and $\mathcal{U}_1$ and construct

$$\dfrac{\begin{array}{c}\mathcal{R}_1\\ \Gamma';\Delta' \stackrel{=}{\Longrightarrow} [a'/x]A'_1 \setminus [a'/x]F_1\end{array}}{\Gamma';\Delta' \stackrel{=}{\Longrightarrow} \forall x.\ A'_1 \setminus \forall x.\ F_1}\forall\mathrm{I}^{a'} \quad \text{and} \quad \dfrac{\begin{array}{c}\mathcal{U}_1\\ \models [\rho, a/a']([a'/x]F_1)\end{array}}{\models [\rho]\forall x.\ F_1}\forall\mathrm{I}^a.$$

□

Theorem 3.6 (Completeness of Equality Residuation) *If* $\Gamma;\Delta \stackrel{=}{\Longrightarrow} A$ *then* $\Gamma;\Delta \stackrel{=}{\Longrightarrow} A \setminus F$ *for some* F *and* $\models F$.

Proof: From Lemma 3.5 with $A' = A$, $\Delta' = \Delta$, $\Gamma' = \Gamma$, and ρ the identity substitution on the parameters in Γ, Δ, and A. □

Next we describe an algorithm for proving residuated formulas, that is, an algorithm for unification. We do this in two steps: first we solve the problem in the fragment without parameters and universal quantifiers and then we extend the solution to the general case.

There are numerous ways for describing unification algorithms in the literature. We describe the computation of the algorithm as the bottom-up search for the derivation of a judgment. We restrict the inference rules such that they are essentially deterministic, and the inference rules themselves can be seen as describing an algorithm. This algorithm is in fact quite close to the implementation of it in ML which is available together with these notes.[1]

In order to describe the algorithm in this manner, we need to introduce *existential variables* (often called *meta-variables* or *logic variables*) which are place-holders for the terms to be determined by unification. We use X to stand for existential variables.

[1] http://www.cs.cmu.edu/~fp/courses/linear/code/unif.tar.gz

The second concept we need is a *continuation*, which arises from the introduction rule for conjunction. This rule has two premisses, which leaves the choice on how which premiss to prove first when we work in a bottom-up fashion. Our algorithm commits to do the first conjunct first, but it has remember that the second conjunct remains to be proved. Equational formulas which have been postponed in this way are accumulated in the continuation, which is activated when there are no further equations to be solved. For now, a continuation is simply another formula denoted by S. Initially, we use $\top$ for S. Thus our main judgment describing the algorithm has the form "*F is satisfiable with continuation S*", written as $\models F \;/\; S$.

Continuations. The following rules introduce and manage the continuations.

$$\frac{\models F_1 \;/\; F_2 \wedge S}{\models F_1 \wedge F_2 \;/\; S}\wedge\mathrm{I} \qquad \frac{}{\models \top \;/\; \top}\top\mathrm{I}\top \qquad \frac{\models F \;/\; S}{\models \top \;/\; F \wedge S}\top\mathrm{I}\wedge$$

Existential Quantification. Existential variables are introduced for existential quantifiers. They must be new not only in F but also in S.

$$\frac{\models [X/x]F \;/\; S \quad X \text{ not in } F \text{ or } S}{\models \exists x.\, F \;/\; S}\exists\mathrm{I}$$

Despite the requirement on X to be new, the derivation of the premiss is not parametric in X. That is, we cannot substitute an arbitrary term t for X in a derivation of the permiss and obtain a valid derivations, since the vr, rv, vv, and vv$'$ rules below require one or both sides of the equation to be an existential variable. Substituting for such a variables invalidates the application of these rules.

Predicate and Function Constants. An equation between the same function constant applied to arguments is decomposed into equations between the arguments. Unification fails if different function symbols are compared, but this is only indirectly reflected by an absence of an appropriate rule. Failure can also be explicitly incorporated in the algorithm (see Exercise **??**).

$$\frac{\models t_1 \doteq s_1 \wedge \cdots \wedge t_n \doteq s_n \;/\; S}{\models p(t_1,\ldots,t_n) \doteq p(s_1,\ldots,s_n) \;/\; S}\mathrm{pp} \qquad \frac{\models t_1 \doteq s_1 \wedge \cdots \wedge t_n \doteq s_n \;/\; S}{\models f(t_1,\ldots,t_n) \doteq f(s_1,\ldots,s_n) \;/\; S}\mathrm{rr}$$

These rules violate orthogonality by relying on conjunction in the premisses for the sake of conciseness of the presentation. When f or p have no arguments, the empty conjunction in the premiss should be read as $\top$.

Existential Variables. There are three rules for variables. We write r for terms of the form $f(t_1,\ldots,t_n)$. Existential variables always range over terms (and not propositions), so we do not need rules for equations of the form $X \doteq P$ or $P \doteq X$.

$$\frac{\models \top \;/\; [r/X]S \quad X \text{ not in } r}{\models X \doteq r \;/\; S}\,\mathrm{vr} \qquad \frac{\models \top \;/\; [r/X]S \quad X \text{ not in } r}{\models r \doteq X \;/\; S}\,\mathrm{rv}$$

These two rules come with the proviso that the existential variable X does not occur in the term t. This is necessary to ensure termination of these rules (when viewed as an algorithm) and to recognize formulas such as $\exists x.\, x \doteq f(x)$ as unprovable. This leaves equations of the form $X \doteq Y$ with to existential variables. We write two rules for this case to simplify the analysis.

$$\frac{\models \top \;/\; [Y/X]S}{\models X \doteq Y \;/\; S}\,\mathrm{vv} \qquad \frac{\models \top \;/\; S}{\models X \doteq X \;/\; S}\,\mathrm{vv}'$$

We now analyze these rules when viewed as an algorithm specification. First we observe that all rules have either no or one premiss. Furthermore, for any judgment $\models F \;/\; S$ at most one rule is applicable, and in only one way (the choice of the new existential variable name X is irrelevant). Therefore these rules, when viewed as instructions for construction a derivation of a judgment $\models F \;/\; \top$ are deterministic, but may fail, in which case the formula is not provable.

Furthermore, the bottom-up search for a derivation of $\models F \;/\; S$ in this system will always terminate. The termination ordering involves five measures, ordered lexicographically as follows:

1. the number of free and quantified existential variables,
2. the number of predicate and function symbols,
3. the total number of logical symbols $\wedge$, $\top$, $\exists$ in F and S,
4. the number of logical symbols in F,
5. the number of equations.

This measure decreases in each rule:

$\wedge$I does not change (1)–(3) and decreases (4),

$\top$I$\top$ completes the search,

$\top$I$\wedge$ does not change (1)–(2) and decreases (3),

$\exists$I does not change (1)–(2) and decreases (3),

pp does not change (1) and decreases (2),

rr does not change (1) and decreases (2), vr decreases (1) since X does not occur in r,

rv decreases (1) since X does not occur in r,

vv decreases (1), and

vv$'$ does not change (1)–(4) and decreases (5).

In some of these cases it is also possible that a measure of higher priority decreases (but never increases), preserving the strict decrease along the lexicographic ordering.

We also note that the continuation S is not completely general, but follows the grammar below.

$$\textit{Continuations} \quad S \quad ::= \quad \top \mid F \wedge S$$

In other words, it may be viewed as a stack of formulas. In the ML implementation, this stack is not represented explicitly. Instead we use the call stack of ML itself.

The desired soundness and completess theorems for this algorithm requires some generalizations based on substitutions for existential variables.

$$\textit{Ground Substitutions} \quad \theta \quad ::= \quad \cdot \mid \theta, t/X$$

We always assume that the terms t we assign to variables in substitutions do not contain existential variables. This assumption is reasonable, since we only use substitutions here to connect derivations for $\models F$ (which contains to existential variables) with derivations of $\models F' \;/\; S'$ (which contains existential variables).

Lemma 3.7 (Soundness Lemma for Unification) *If $\models F \;/\; S$ then there exists a ground substitution for the existential variables in F and S such that $\models [\theta]F$ and $\models [\theta]S$.*

Proof: By induction on the structure of $\mathcal{F} :: (\models F \;/\; S)$. □

The soundness theorem follows easily from this lemma.

Theorem 3.8 (Soundness of Unification) *If $\models F \;/\; \top$ and F contains no existential variables, then $\models F$.*

Proof: From Lemma 3.7 for $S = \top$ and $\theta = \cdot$. □

Lemma 3.9 (Completeness Lemma for Unification) *If $\models F$ and $\models S$, then for any formulas F', continuations S' and substitutions θ for the existential variables in F' and S' such that $F = [\theta]F'$ and $S = [\theta]S'$ we have $\models F \;/\; S$.*

Proof: By nested inductions on $\mathcal{F} :: (\models F)$ and $\mathcal{S} :: (\models S)$. This means that when we appeal to the induction hypothesis on a subderivation of $\mathcal{F}$, $\mathcal{S}$ may be larger. We distinguish cases for $\mathcal{F}$.

Case: $\mathcal{F} = \dfrac{}{\models \top}\,\top\mathrm{I}.$

The we distinguish two subcases for $\mathcal{S}$. If $\mathcal{S}$ is $\top$I, the result is trivial by $\top$I$\top$. Otherwise

$$\mathcal{S} = \dfrac{\begin{array}{c}\mathcal{F}_1 \\ \models F_1\end{array} \qquad \begin{array}{c}\mathcal{S}_2 \\ \models S_2\end{array}}{\models F_1 \wedge S_2}\,\wedge\mathrm{I}$$

where $S = F_1 \wedge S_2$ for some F_1 and S_2. Then

$\mathcal{F}_1' :: (\models F_1 \;/\; S_2)$	By ind. hyp. on $\mathcal{F}_1$ and $\mathcal{S}_2$
$\mathcal{F}' :: (\models \top \;/\; F_1 \wedge S_2)$	By $\top\mathrm{I}\wedge$

Case: $\mathcal{F} = \dfrac{\begin{array}{c}\mathcal{F}_1 \\ \models F_1\end{array} \qquad \begin{array}{c}\mathcal{F}_2 \\ \models F_2\end{array}}{\models F_1 \wedge F_2}\,\wedge\mathrm{I}.$

$\mathcal{F}_2' :: (\models F_2' \;/\; S')$	By ind. hyp. on $\mathcal{F}_2$ and $\mathcal{S}$
$\mathcal{S}_2 :: (\models F_2 \wedge S)$	By $\wedge$I from $\mathcal{F}_2$ and $\mathcal{S}$
$\mathcal{F}_1' :: (\models F_1' \;/\; F_2' \wedge S')$	By ind. hyp. on $\mathcal{F}_1$ and $\mathcal{S}_2$
$\mathcal{F}' :: (\models F_1' \wedge F_2' \;/\; S')$	By $\wedge$I from $\mathcal{F}_1'$.

Case: $\mathcal{F} = \dfrac{\begin{array}{c}\mathcal{F}_1 \\ \models [t/x]F_1\end{array}}{\models \exists x.\ F_1}\,\exists\mathrm{I}.$

$F' = \exists x.\ F_1'$ and $[\theta](\exists x.\ F_1') = \exists x.\ F_1$	By assumption
$[\theta, t/X]([X/x]F_1') = [t/x]F_1$ for X not in F' or S'	
$[\theta, t/X]S' = S$	Since X is new
$\mathcal{F}_1' :: (\models [X/x]F_1' \;/\; S')$	By ind. hyp. on $\mathcal{F}_1$ and $\mathcal{S}$
$\mathcal{F} :: (\models \exists x.\ F_1' \;/\; S')$	By $\exists$I

Case: $\mathcal{F} = \dfrac{}{\models t \doteq t}\doteq\mathrm{I}.$

Here we proceed by an auxiliary induction on the structure of t. By assumption $[\theta]F' = (t \doteq t)$, so we have t' and t'' such that $[\theta]t' = [\theta]t'' = t$. We distinguish cases on t' and t'', showing three. The remaining ones are similar.

Subcase: $t' = f(t_1', \ldots, t_n')$ and $t'' = f(t_1'', \ldots, t_n'')$, so also $t = f(t_1, \ldots, t_n)$.

$\models t_n' \doteq t_n'' \;/\; S'$	By ind. hyp. on t_n and $\mathcal{S}$
$\mathcal{S}_n :: (\models t_n \doteq t_n \wedge S)$	By $\wedge$I from $\doteq$I and $\mathcal{S}$
$\models t_{n-1}' \doteq t_{n-1}'' \;/\; t_n' \doteq t_n'' \wedge S'$	By ind. hyp. on t_{n-1} and $\mathcal{S}_n$.
$\models t_1' \doteq t_1'' \;/\; t_2' \doteq t_2'' \wedge \cdots \wedge t_n' \doteq t_n'' \wedge S'$	As above
$\models t_1' \doteq t_1'' \wedge t_2' \doteq t_2'' \wedge \cdots \wedge t_n' \doteq t_n'' \;/\; S'$	by $\wedge$I
$\models f(t_1', \ldots, t_n') \doteq f(t_1'', \ldots, t_n'') \;/\; S'$	by rr.

Subcase: $t' = X$ and $t'' = r$ but contains X. This is impossible, since we assumed $[\theta]t' = [\theta]t'' = t$.

Subcase: $t' = X$ and $t'' = r$ does not contain X. Then $[\theta]([r/X]S') = [\theta]S' = S$ since $[\theta]r = [\theta]X = t$ and θ is a ground substitution. By distinguishing cases for $\mathcal{S}$ as for $F = \top$ above, we conclude

$\models \top \;/\; [r/X]S'$
$\models X \doteq r \;/\; S'$ By rule vr

□

The completeness theorem follows easily from this lemma.

Theorem 3.10 (Completeness of Unification) *If* $\models F$ *(where* F *contains no existential variables) then* $\models F \;/\; \top$.

Proof: From Lemma 3.9 with $S = \top$, $S' = \top$, $F' = F$ and $\theta = \cdot$. □

The generalization of the algorithm above to account for universal quantifiers and parameters is not completely straightforward. The difficulty is that $\forall x.\, \exists y.\, y \doteq x$ is valid, while $\exists y.\, \forall x.\, y \doteq x$ is not. We show an attempt to derive the latter which must be ruled out somehow.

$$\cfrac{\cfrac{\cfrac{\cfrac{}{\models \top \;/\; [a/Y]\top}\;\top I\top}{\models Y \doteq a \;/\; \top}\;\mathrm{vr}}{\models \forall x.\, Y \doteq x \;/\; \top}\;\forall I^a}{\models \exists y.\, \forall x.\, y \doteq x \;/\; \top}\;\exists I$$

In this derivation, the application of $\top I\top$ is correct since $[a/Y]\top = \top$. The problem lies in the fact a is new in the application of the $\forall I^a$ rule, but only because we have not instantiated Y with a yet, which is necessary to complete the derivation.

There are two ways to solve this problem. More or less standard in theorem proving is *Skolemization* which we pursue in Exercise **??**. The dual solution notes for each existential variable which parameters may occur in its substitution term. In the example above, Y was introduced at a point where a did not yet occur, so the substitution of a for Y should be rejected.

In order to describe this concisely, we add a *parameter context* Ψ to the judgment which lists distinct parameters.

$$\textit{Parameter Context} \quad \Psi \quad ::= \quad \cdot \mid \Psi, a$$

This step is analogous to the localization of the hypotheses and should be considered merely a change in notation, not an essential change in the judgment itself. We annotate each judgment with the parameter context and introduce the new judmgnet "*t is closed with respect to* Ψ", written as $\Psi \models t\,\mathrm{term}$. It is defined by the following rules.

$$\frac{}{\Psi_1, a, \Psi_2 \vdash a\,\mathrm{term}}\,\mathrm{parm} \qquad \frac{\Psi \vdash t_1\,\mathrm{term} \quad \cdots \quad \Psi \vdash t_n\,\mathrm{term}}{\Psi \vdash f(t_1, \ldots, t_n)\,\mathrm{term}}\,\mathrm{root}$$

We modify the validity judgment for unification formulas to guarantee this condition.

$$\frac{\Psi \vdash t\,\mathrm{term} \qquad \Psi \models [t/x]F}{\Psi \models \exists x.\ F}\,\exists\mathrm{I} \qquad \frac{\Psi, a \models [a/x]F}{\Psi \models \forall x.\ F}\,\forall\mathrm{I}^a$$

When an existential variable X is introduced during the search for a derivation of a unification formula, we annotate it with the parameter context so we keep track of the admissible substitutions for X.

$$\frac{\Psi \models [X_\Psi/x]F \ /\ S \quad X_\Psi \text{ not in } F \text{ or } S}{\Psi \models \exists x.\ F \ /\ S}\,\exists\mathrm{I}$$

Parameters are introduced in the rule for universal quantifiers as before.

$$\frac{\Psi, a \models [a/x]F \ /\ S}{\Psi \models \forall x.\ F \ /\ S}\,\forall\mathrm{I}^a$$

An equation $X_\Psi \doteq t$ could now be solved immediately, if all parameters of t are contained in Ψ and X does not occur in t. However, there is one tricky case. Consider the judgment

$$a \models X_\cdot \doteq f(Y_a) \wedge Y_a \doteq a \ /\ \top$$

where X cannot depend on any parameters and Y can depend on a. This should have no solution, since $X_\cdot$ would have to be equal to $f(a)$, which is not permissible. On the other hand,

$$a \models X_\cdot \doteq f(Y_a) \wedge Y_a \doteq c \ /\ \top$$

for a constant c has a solution where Y_a is c and $X_\cdot$ is $f(c)$. So when we process an equation $X_\Psi = t$ we need to restrict any variable in t so it can depend only on the parameters in Ψ. In the example above, we would substitute $Y'_\cdot$ for Y_a.

In order to describe the algorithm, we internalize the judgment $\Psi \vdash t\,\mathrm{term}$ as a new formula, written as $t\,|_\Psi$. We define it as follows.

$$\frac{\Psi' \models \top \ /\ S \quad \text{if } a \text{ in } \Psi}{\Psi' \models a\,|_\Psi / \ S}\,|\,\mathrm{a} \qquad \frac{\Psi' \models t_1\,|_\Psi \wedge \cdots \wedge t_n\,|_\Psi / \ S}{\Psi' \models f(t_1, \ldots, t_n)\,|_\Psi / \ S}\,|\,\mathrm{f}$$

$$\frac{\Psi' \models \top \ /\ [Y_{\Psi_2 \cap \Psi_1}/Y_{\Psi_2}]S}{\Psi' \models Y_{\Psi_2}\,|_{\Psi_1} / \ S}\,|\,\mathrm{v}$$

Here, $\Psi_1 \cap \Psi_2$ denotes the intersection of the two contexts. In the rules for variables, this is invoked as follows.

$$\frac{\Psi' \models r \mid_{\Psi} / \; [r/X_\Psi]S \quad \text{where } X_\Psi \text{ not in } r}{\Psi' \models X_\Psi \doteq r \;/\; S}\,\mathrm{vr}$$

$$\frac{\Psi' \models r \mid_{\Psi} / \; [r/X_\Psi]S \quad \text{where } X_\Psi \text{ not in } r}{\Psi' \models r \doteq X_\Psi \;/\; S}\,\mathrm{vr}$$

where r stands for a term $f(t_1, \ldots, t_n)$ or a parameter a. The variable rules are modified similarly.

$$\frac{\Psi' \models Y_{\Psi_2} \mid_{\Psi_1} / \; [Y_{\Psi_2}/X_{\Psi_1}]S}{\Psi' \models X_{\Psi_1} \doteq Y_{\Psi_2} \;/\; S}\,\mathrm{vv} \qquad \frac{\Psi' \models \top \;/\; S}{\Psi' x \models X_\Psi \doteq X_\Psi \;/\; S}\,\mathrm{vv}'$$

The use of continuations introduces on final complication. Consider the case of $(\forall x.\ F_1) \wedge F_2$. Since we linearize bottom-up search the parameter context Ψ will contain the parameter introduced for x when F_2 is finally considered after F_1 has been solved. This introduces spurious dependencies. To prohibit those, we build *closures* consisting of a formula and its parameter context on the continuation stack.

$$\text{Continuations} \quad S \quad ::= \quad \top \mid \{\Psi, F\} \wedge S$$

The rules for continuations are modified as follows.

$$\frac{\Psi \models F_1 \;/\; \{\Psi, F_2\} \wedge S}{\Psi \models F_1 \wedge F_2 \;/\; S}\,\wedge\mathrm{I} \qquad \frac{}{\Psi \models \top \;/\; \top}\,\top\mathrm{I}\top \qquad \frac{\Psi \models F \;/\; S}{\Psi' \models \top \;/\; \{\Psi, F\} \wedge S}\,\top\mathrm{I}\wedge$$

The termination argument is only slightly more difficult, since the restriction operation is a structural recursion over the term r and does not increase the number of variables or equations.

The soundness and completeness theorems from above extend to the problem with parameters, but become more difficult. The principal new notion we need is an *admissible substitution* θ which has the property that for every existential variable X_Ψ, $\Psi \vdash [\theta]X_\Psi$ term (see Exercise **??**).

The ML implementation takes advantage of the fact that whenever a variable must be restricted, one of the two contexts is a prefix of the other. This is because every equation in a formula F lies beneath a path of possibly alternating quantifiers, a so-called *mixed quantifier prefix*. When we apply the rules above algorithmically, we instantiate each existentially quantified variable with a new free existential variable which depends on all parameters which were introduced for the universally quantified variables to its left. Clearly, then, for any two variables in the same equation, one context is a prefix of the other. Our ML implementation does take advantage of this observation by simplifying the intersection operation.

We can take this optimization a step further and only record with an integer (a kind of time stamp), which parameters an existential variable may depend on. This improves the efficiency of the algorithm even further, since we only need to calculate the minimum of two integers instead of intersecting two contexts during restriction. In the ML code for this class, we did not optimize to this extent.

3.3 Resource Management

A form of choice unique to linear logic proof search is *resource management*: in the bottom-up application of the left rule for implication and right rule for tensor, we have to split the linear hypotheses and distribute them to the premisses. We would like to postpone this choice until the further structure of the derivation provides hints which resources might be needed in which subgoals.

To resolve this non-determinism, we use a technique inspired by unification. We pass the complete list of hypotheses to both premisses and maintain constraints which express that each hypothesis must be used in one of the two subderivations, but not both. If one is ever used in one branch, we can propagate this information to the other branch by constraint simplification. This mirrors the way unification propagates substitutions for existential variables between incomplete proof branches.

We annotate each hypothesis with an *occurrence label* b.

$$\textit{Occurrence Labels} \quad b \quad ::= \quad \top \mid \bot \mid o$$

Here, $\top$ labels a hypothesis which is definitely present and must therefore be consumed (in the bottom-up search), $\bot$ labels a hypothesis which is definitely not present and can therefore not be used, and an occurrence variable o labels a hypothesis which may or may not be used, subject to some global constraints. Constraints which arise all have the following forms.

$$\textit{Occurrence Constraints} \quad c \quad ::= \quad b_1 \doteq b_2 \mid b_1 + b_2 \doteq b_3 \mid c_1 \wedge c_2 \mid \mathrm{tt}$$

The validity judgment for constraints, $\models c$, is defined by the following rules.

$$\frac{}{\models \top \doteq \top} \doteq\top \qquad \frac{}{\models \bot \doteq \bot} \doteq\bot$$

$$\frac{}{\models \top + \bot \doteq \top} +\top\bot \qquad \frac{}{\models \bot + \top \doteq \top} +\bot\top$$

$$\textit{no } +\top\top \textit{ rule} \qquad \frac{}{\models \bot + \bot \doteq \bot} +\bot\bot$$

$$\frac{\models c_1 \qquad \models c_2}{\models c_1 \wedge c_2} \wedge\mathrm{i} \qquad \frac{}{\models \mathrm{tt}} \mathrm{tti}$$

We say a constraint c with occurrence variables is *satisfiable* if there is an assignment of $\top$ and $\bot$ to the occurrence variables such that the resulting constraint is valid.

Linear hypotheses, annotated with occurrence labels, have the form

$$\textit{Annotated Contexts} \quad \Delta \quad ::= \quad \cdot \mid \Delta, w^b{:}A$$

It is convenient to abbreviate $w^b{:}A$ as A^b and $\cdot, w_1^{b_1}{:}A_1, \ldots, w_n^{b_n}{:}A_n$ as $\Delta^{\vec{b}}$. The basic sequent now reads $\Gamma; \Delta^{\vec{b}} \Longrightarrow C \setminus c$, where c are the residual constraints. The intuition should be that any satisfying assignment to the occurrence variables in c leads to a valid derivation of $\Gamma; \Delta^* \Longrightarrow C$, where Δ^* retains hypotheses of the form $w^\top{:}A$ and erases hypotheses of the form $w^\bot{:}A$. We now go through the rules, removing resource non-determinism in favor of occurrence constraints. In practice, these constraint should be checked for satisfiability in each step for early detection of failure. To give a more compact presentation of the rules, we further write

$$\begin{array}{rll} \vec{b} \doteq \bot & \text{for} & b_1 \doteq \bot \wedge \cdots \wedge b_n \doteq \bot \quad \text{and} \\ \vec{o'} + \vec{o''} \doteq \vec{b} & \text{for} & o'_1 + o''_1 = b_1 \wedge \cdots \wedge o'_n + o''_n \doteq b_n. \end{array}$$

Hypotheses. Initial sequents change form, since the particular hypothesis we use must be constraint to be present, while all others have to be constrained to be absent. This leaves some residual non-determinism if several available hypotheses match the conclusion.

$$\frac{}{\Gamma; (\Delta^{\vec{b}}, A^d) \Longrightarrow A \setminus d \doteq \top \wedge \vec{b} \doteq \bot}\,\text{I} \qquad \frac{(\Gamma, A); (\Delta, A^\top) \Longrightarrow C}{(\Gamma, A); \Delta \Longrightarrow C}\,\text{DL}$$

Multiplicative Connectives. Multiplicative connectives have to generate constraints as discussed above. New linear hypothesis must be used somewhere, so their initial annotation is $\top$.

$$\frac{\Gamma; \Delta^{\vec{b}}, A^\top \Longrightarrow B \setminus c}{\Gamma; \Delta^{\vec{b}} \Longrightarrow A \multimap B \setminus c}\,{\multimap}\text{R}$$

Whenever a left rule is applied to a hypothesis, its occurrence label is constrained to the $\top$. In addition, since linear implication is multiplicative, we generate *new* occurrence variables $\vec{o'}$ and $\vec{o''}$ and constrain them.

$$\frac{\Gamma; \Delta^{\vec{o'}} \Longrightarrow A \setminus c' \qquad \Gamma; \Delta^{\vec{o''}}, B \Longrightarrow C \setminus c''}{\Gamma; \Delta^{\vec{b}}, (A \multimap B)^d \Longrightarrow C \setminus d \doteq \top \wedge c' \wedge c'' \wedge \vec{o'} + \vec{o''} \doteq \vec{b}}\,{\multimap}\text{L}$$

The tensor rules are similar. Here, too, the occurrence variables $\vec{o'}$ and $\vec{o''}$ must be new.

$$\frac{\Gamma; \Delta^{\vec{o'}} \Longrightarrow A \setminus c' \qquad \Gamma; \Delta^{\vec{o''}} \Longrightarrow B \setminus c''}{\Gamma; \Delta^{\vec{b}} \Longrightarrow A \otimes B \setminus c' \wedge c'' \wedge \vec{o'} + \vec{o''} \doteq \vec{b}}\,\otimes\text{R}$$

$$\frac{\Gamma; \Delta^{\vec{b}}, A^{\top}, B^{\top} \Longrightarrow C \setminus c}{\Gamma; \Delta^{\vec{b}}, (A \otimes B)^d \Longrightarrow C \setminus d \doteq \top \wedge c} \otimes\mathrm{L}$$

The $\mathbf{1}$R rule permits no linear hypotheses, so all of them are constrained to be absent.

$$\frac{}{\Gamma; \Delta^{\vec{b}} \Longrightarrow \mathbf{1} \setminus \vec{b} \doteq \bot} \mathbf{1}\mathrm{R} \qquad \frac{\Gamma; \Delta^{\vec{b}} \Longrightarrow C \setminus c}{\Gamma; \Delta^{\vec{b}}, \mathbf{1}^d \Longrightarrow C \setminus d \doteq \top \wedge c} \mathbf{1}\mathrm{L}$$

Additive Connectives. The additive connective are much simpler and do not affect the occurrence constraints, except that the principal proposition of a left rule must be constrained to be present.

$$\frac{\Gamma; \Delta^{\vec{b}} \Longrightarrow A \setminus c' \qquad \Gamma; \Delta^{\vec{b}} \Longrightarrow B \setminus c''}{\Gamma; \Delta^{\vec{b}} \Longrightarrow A \& B \setminus c' \wedge c''} \&\mathrm{R}$$

$$\frac{\Gamma; \Delta^{\vec{b}}, A^{\top} \Longrightarrow C \setminus c}{\Gamma; \Delta^{\vec{b}}, (A \& B)^d \Longrightarrow C \setminus d \doteq \top \wedge c} \&\mathrm{L}_1 \qquad \frac{\Gamma; \Delta^{\vec{b}}, B^{\top} \Longrightarrow C \setminus c}{\Gamma; \Delta^{\vec{b}}, (A \& B)^d \Longrightarrow C \setminus d \doteq \top \wedge c} \&\mathrm{L}_2$$

$$\frac{}{\Gamma; \Delta^{\vec{b}} \Longrightarrow \top \setminus \mathrm{tt}} \top\mathrm{R} \qquad \textit{No } \top \textit{ left rule}$$

$$\frac{\Gamma; \Delta^{\vec{b}} \Longrightarrow A \setminus c}{\Gamma; \Delta^{\vec{b}} \Longrightarrow A \oplus B \setminus c} \oplus\mathrm{R}_1 \qquad \frac{\Gamma; \Delta^{\vec{b}} \Longrightarrow B \setminus c}{\Gamma; \Delta^{\vec{b}} \Longrightarrow A \oplus B \setminus c} \oplus\mathrm{R}_2$$

$$\frac{\Gamma; \Delta^{\vec{b}}, A^{\top} \Longrightarrow C \setminus c' \qquad \Gamma; \Delta^{\vec{b}}, B^{\top} \Longrightarrow C \setminus c''}{\Gamma; \Delta^{\vec{b}}, (A \oplus B)^d \Longrightarrow C \setminus d \doteq \top \wedge c' \wedge c''} \oplus\mathrm{L}$$

$$\textit{No } \mathbf{0} \textit{ right rule} \qquad \frac{}{\Gamma; \Delta^{\vec{b}}, (\mathbf{0})^d \Longrightarrow C \setminus d \doteq \top} \mathbf{0}\mathrm{L}$$

Quantifiers. The interaction of the quantifiers with resource management is benign, and limited requiring the principal propositions of left rules to occur.

$$\frac{\Gamma; \Delta^{\vec{b}} \Longrightarrow [a/x]A \setminus c}{\Gamma; \Delta^{\vec{b}} \Longrightarrow \forall x.\ A \setminus c} \forall\mathrm{R}^a \qquad \frac{\Gamma; \Delta^{\vec{b}}, ([t/x]A)^{\top} \Longrightarrow C \setminus c}{\Gamma; \Delta^{\vec{b}}, (\forall x.\ A)^d \Longrightarrow C \setminus d \doteq \top \wedge c} \forall\mathrm{L}$$

$$\frac{\Gamma; \Delta^{\vec{b}} \Longrightarrow [t/x]A \setminus c}{\Gamma; \Delta^{\vec{b}} \Longrightarrow \exists x.\ A \setminus c} \exists\mathrm{R} \qquad \frac{\Gamma; \Delta^{\vec{b}}, ([a/x]A)^{\top} \Longrightarrow C \setminus c}{\Gamma; \Delta^{\vec{b}}, (\exists x.\ A)^d \Longrightarrow C \setminus d \doteq \top \wedge c} \exists\mathrm{L}^a$$

Exponentials. There are two natural formulations of the $\supset$L and !R rules: we either do not pass any linear hypotheses to the relevant premiss as shown below, or we pass all linear hypotheses but constrain them not to be used. Depending on the design of the implementation, one or the other might be preferable.

$$\frac{(\Gamma, A); \Delta^{\vec{b}} \Longrightarrow B \setminus c}{\Gamma; \Delta^{\vec{b}} \Longrightarrow A \supset B \setminus c} \supset\mathrm{R} \qquad \frac{\Gamma; \cdot \Longrightarrow A \setminus c' \qquad \Gamma; \Delta^{\vec{b}}, B^{\top} \Longrightarrow C \setminus c''}{\Gamma; \Delta^{\vec{b}}, (A \supset B)^d \Longrightarrow C \setminus d \doteq \top \wedge c' \wedge c''} \supset\mathrm{L}$$

$$\frac{\Gamma; \cdot \Longrightarrow A \setminus c}{\Gamma; \Delta^{\vec{b}} \Longrightarrow {!A} \setminus \vec{b} \doteq \bot \wedge c} !\mathrm{R} \qquad \frac{(\Gamma, A); \Delta^{\vec{b}} \Longrightarrow C \setminus c}{\Gamma; \Delta^{\vec{b}}, (!A)^d \Longrightarrow C \setminus d \doteq \top \wedge c} !\mathrm{L}$$

We write Θ for an assignment of $\top$ or $\bot$ to all occurrence variables. Applying such an assignment to an annotated context of linear hypotheses is defined as

$$\begin{array}{rcll} [\Theta]\cdot & = & \cdot, & \\ [\Theta](\Delta, w^b{:}A) & = & [\Theta]\Delta, w{:}A & \text{if } [\Theta]b = \top, \text{ and} \\ [\Theta](\Delta, w^b{:}A) & = & [\Theta]\Delta & \text{if } [\Theta]b = \bot. \end{array}$$

Applying an assignment of a derivation simply applies it to every sequent in the derivation and erases the occurrence constraints.

We should then have the following soundness and completeness theorems.[2]

Theorem 3.11 (Soundness of Occurrence Constraints) *If* $\mathcal{D} :: (\Gamma; \Delta^{\vec{b}} \Longrightarrow C \setminus c)$ *and* $\models [\Theta]c$ *then* $[\Theta]\mathcal{D} :: (\Gamma; [\Theta]\Delta^{\vec{b}} \Longrightarrow C)$.

Theorem 3.12 (Completeness of Occurrence Constraints) *If* $\mathcal{D} :: (\Gamma; \Delta \Longrightarrow C)$ *then* $\mathcal{D}' :: (\Gamma; \Delta^{\vec{\top}} \Longrightarrow C \setminus c)$ *and there is an assignment* Θ *such that* $\models [\Theta]c$ *and* $\mathcal{D} = [\Theta]\mathcal{D}'$.

One or both of these "theorems" may have to be generalized before they can be proved by induction.

How do we check constraints for satisfiability? We have not fully investigated this issue to date. One possibility (suggested by Harland and Pym [**?**]) is to map them to Boolean constraints, which would make them amenable to standard Boolean constraint solving techniques. It seems, however, that this would lead to an unnecessarily complex procedure. We sketch here a set of of rules which may be used to simplify constraints put them into a normal form which should always have solutions. The rules may be incomplete (and certainly would require additional invariants, as we remark below). The apply to any conjunct of the global constraint c.

[2] [*Warning: at present I have not proven these.*]

$$
\begin{array}{rcl}
\top \doteq \top & \longrightarrow & \mathrm{tt} \\
\bot \doteq \bot & \longrightarrow & \mathrm{tt} \\
\top \doteq \bot & & \text{unsatisfiable} \\
\bot \doteq \top & & \text{unsatisfiable} \\
o \doteq b & \longrightarrow & \mathrm{tt} \quad \text{and substitute } b \text{ for } o \text{ everywhere} \\
\top + \bot \doteq b & \longrightarrow & b \doteq \top \\
\bot + \top \doteq b & \longrightarrow & b \doteq \top \\
\top + \top \doteq b & & \text{unsatisfiable} \\
\bot + \bot \doteq b & \longrightarrow & b \doteq \bot \\
o + \top \doteq b & \longrightarrow & o \doteq \bot \wedge b \doteq \top \\
\top + o \doteq b & \longrightarrow & o \doteq \bot \wedge b \doteq \top \\
o + \bot \doteq b & \longrightarrow & o \doteq b \\
\bot + o \doteq b & \longrightarrow & o \doteq b \\
o_1 + o_2 \doteq \bot & \longrightarrow & o_1 \doteq \bot \wedge o_2 \doteq \bot \\
o_1 + o_2 \doteq \top & & \text{in normal form} \\
o_1 + o_2 \doteq o_3 & & \text{in normal form}
\end{array}
$$

The last two cases of normal forms do not imply satisfiability. For example, $o + o \doteq \top$ is not satisfiable. Similarly, $o + o' \doteq o'$ entails that $o \doteq \bot$, which might be inconsistent with other constraints. However, I believe that there is a natural ordering on occurrence variables (and an induced ordering among atomic constraints) which can guarantee that certain cases of this form can not arise, or arise only in a limited number of circumstances which can be checked easily.

[**Extra Credit Assignment:** *Complete the rules above as needed to guarantee the satisfiability of normal forms for equations which arise from proof search and constraint simplification and proof them correct.*]

3.4 Inversion for Unrestricted Resources

Inversion principles as presented in Section 3.1 reduce don't-know non-deterministic choices by giving us license to always apply strongly invertible rules in the bottom-up search for a derivation. But there is a gap in the analysis in that the dereliction rule is *always* applicable when we have any unrestricted hypotheses, but is not invertible. However, there are a number of cases where we can write out derived or admissible rules that operate directly on unrestricted hypotheses, and which are invertible. We can then limit the use of dereliction to the remaining cases.

The following rules are all admissible and strongly invertible.

$$\frac{(\Gamma, A_1, A_2); \Delta \overset{-}{\Longrightarrow} B}{(\Gamma, A_1 \& A_2); \Delta \overset{-}{\Longrightarrow} B}\ \&\mathrm{L}! \qquad \frac{\Gamma; \Delta \overset{-}{\Longrightarrow} B}{(\Gamma, \top); \Delta \overset{-}{\Longrightarrow} B}\ \top\mathrm{L}!$$

$$\frac{\Gamma; \Delta \overset{-}{\Longrightarrow} B}{(\Gamma, \mathbf{1}); \Delta \overset{-}{\Longrightarrow} B}\ \mathbf{1}\mathrm{L}! \qquad \frac{}{(\Gamma, \mathbf{0}); \Delta \overset{-}{\Longrightarrow} B}\ \mathbf{0}\mathrm{L}!$$

$$\frac{(\Gamma, A); \Delta \overset{-}{\Longrightarrow} B}{(\Gamma, !A); \Delta \overset{-}{\Longrightarrow} B}\ !\mathrm{L}!$$

Theorem 3.13 (Invertibility of Admissible Left! Rules) *The rules* $\&$L!, $\top$L!, $\mathbf{0}$L! *and* !L! *are admissible and invertible. A system with these rules and dereliction restricted to a principal propositions which is of the form* P, $A_1 \multimap A_2$, $\forall x.\ A$, $A_1 \supset A_2$, $A_1 \otimes A_2$, $A_1 \oplus A_2$, *or* $\exists x.\ A$ *is sound and complete.*

Proof: Admissibility and invertibility follows direct calculation in each direction, using the admissibility of Cut! (Theorem **??**) in some cases. Soundness follows easily from admissibility, completeness from invertibility. □

There is also one derivable, weakly invertible rule.

$$\frac{}{(\Gamma, P); \cdot \overset{-}{\Longrightarrow} P}\ \mathrm{I}!$$

None of the remaining connectives admit invertible rules of the kind above (see Exercise **??**). If we want a complete system of rules to replace dereliction DL altogether, we would have to add some non-invertible ones. Here is a possible set of rules.

$$\frac{(\Gamma, A_1 \multimap A_2); \Delta_1 \overset{-}{\Longrightarrow} A_1 \qquad (\Gamma, A_1 \multimap A_2); (\Delta_2, A_2) \overset{-}{\Longrightarrow} B}{(\Gamma, A_1 \multimap A_2); \Delta_1 \times \Delta_2 \overset{-}{\Longrightarrow} B}\ \multimap\mathrm{L}!$$

$$\frac{(\Gamma, A_1 \supset A_2); \cdot \overset{-}{\Longrightarrow} A_1 \qquad (\Gamma, A_2); \Delta \overset{-}{\Longrightarrow} B}{(\Gamma, A_1 \multimap A_2); \Delta \overset{-}{\Longrightarrow} B}\ \multimap\mathrm{L}!$$

$$\frac{(\Gamma, \forall x.\ A); (\Delta, [t/x]A) \overset{-}{\Longrightarrow} B}{(\Gamma, \forall x.\ A); \Delta \overset{-}{\Longrightarrow} B}\ \forall\mathrm{L}! \qquad \frac{(\Gamma, \exists x.\ A); (\Delta, [a/x]A) \overset{-}{\Longrightarrow} B}{(\Gamma, \exists x.\ A); \Delta \overset{-}{\Longrightarrow} B}\ \exists\mathrm{L}!$$

$$\frac{(\Gamma, A_1 \otimes A_2); (\Delta, A_1, A_2) \overset{-}{\Longrightarrow} B}{(\Gamma, A_1 \otimes A_2); \Delta \overset{-}{\Longrightarrow} B}\ \otimes\mathrm{L}!$$

$$\frac{(\Gamma, A_1 \oplus A_2); (\Delta, A_1) \stackrel{-}{\Longrightarrow} B \qquad (\Gamma, A_1 \oplus A_2); (\Delta, A_2) \stackrel{-}{\Longrightarrow} B}{(\Gamma, A_1 \oplus A_2); \Delta \stackrel{-}{\Longrightarrow} B} \oplus\mathrm{L!}$$

In some cases there are other admissible rules, but they are rarely useful. For example, the rule

$$\frac{(\Gamma, \forall x.\ A, [t/x]A); \Delta \stackrel{-}{\Longrightarrow} B}{(\Gamma, \forall x.\ A); \Delta \stackrel{-}{\Longrightarrow} B} \forall\mathrm{L!}'$$

is certainly admissible and even invertible, but it cannot be applied eagerly, since it would lead to non-termination. Instead, we can simply reuse $\forall x.\ A$ if we need another copy of $[t/x]A$.

3.5 Another Example: Arithmetic

Because linear hypotheses must be used exactly once, we can encode arithmetic problems as propositions in linear logic. We map a set of linear equations over the natural numbers into a proposition of linear logic, such that any proof of the proposition corresponds to a solution to the set of equations. When the proposition has not proof, the linear equations have no solutions.

We first represent natural numbers using a new (uninterpreted) atomic proposition p.

$$\begin{array}{rcl} \ulcorner 0 \urcorner & = & \mathbf{1} \\ \ulcorner n+1 \urcorner & = & p \otimes \ulcorner n \urcorner \end{array}$$

Since $p \otimes \mathbf{1} \dashv\vdash p$ we omit the trailing $\mathbf{1}$ in the examples, and also sometimes abbreviate $\ulcorner n \urcorner$ as p^n.

Addition is then easily represented by the multiplicative conjunction, and equality by linear implication. We use e to range over arithmetic expressions (which are not yet completely defined).

$$\begin{array}{rcl} \ulcorner e_1 + e_2 \urcorner & = & \ulcorner e_1 \urcorner \otimes \ulcorner e_2 \urcorner \\ \ulcorner e_1 = e_2 \urcorner & = & \ulcorner e_1 \urcorner \multimap \ulcorner e_2 \urcorner \end{array}$$

For example, the equation $3 + 2 = 1 + 4$ would be represented as

$$(p \otimes p \otimes p \otimes \mathbf{1}) \otimes (p \otimes p \otimes \mathbf{1}) \multimap (p \otimes \mathbf{1}) \otimes (p \otimes p \otimes p \otimes p \otimes \mathbf{1})$$

which is clearly true. It is also easy to see that an equation between different numbers will be an unprovable linear implication.

For every variable x in the left-hand side of an equation we have a hypothesis $!p$. If the variable x is instantiated by a number n, the corresponding derivation will use this hypothesis n times, creating a linear copy of p each time. For example (omitting $\mathbf{1}$s):

$$\ulcorner x + y + 1 = 3 \urcorner = \,!p \otimes !p \otimes p \multimap p \otimes p \otimes p$$

If a variable x occurs more than once, or multiplied by a constant, we collect common terms and think of $kx = x + x + \cdots + x$. So the representation of $3x$ is $!(p \otimes p \otimes p)$. For example,

$$\ulcorner 3x + 2y = 7 \urcorner = !(p^3) \otimes !(p^2) \multimap p^7$$

Representing several simultaneous equations is a bit more difficult, because we must make sure that a variable is instantiated to the same number in all equations. We achieve this by using a different representation of the natural numbers in each equation (say p_i for equation number i), and let each variable generate the appropriate number of p_i's for equation i. The right-hand sides of the equations are then combined with $\otimes$. For example,

$$\begin{array}{ll}
\ulcorner x + y + 1 = 4 \wedge 2x + 3y = 6 \urcorner = & \\
!(p_1 \otimes p_2^2) & x \text{ and } 2x \\
\otimes !(p_1 \otimes p_2^3) & y \text{ and } 3y \\
\multimap & \\
(p_1 \multimap p_1^4) & (x + y) + 1 = 4 \\
\otimes p_2^6 & \text{and } (2x + 3y) = 6
\end{array}$$

The multiplicative conjunction in the conclusion forces all hypotheses regarding p_1 to the first conjunct, and all hypothesis regarding p_2 to the second conjunct. Since the exponential ! operator is outside the tensor for the variables in the different equations, the number of uses of this unrestricted assumption determines the instantiation for the variable.

Note that this example requires only a very small fragment, the so-called *multiplicative exponential linear logic.*

$$\textit{Multiplicative Exponential} \quad M ::= \quad P \mid M_1 \otimes M_2 \mid \mathbf{1} \mid M_1 \multimap M_2 \mid !M$$

Actually, in the propositions above a linear implication never appears on the left-hand side of a linear implication, and an exponential never appears on the right-hand side of a linear implication, which is a significant further restriction.[3]

One can add negative numbers and stay within the multiplicative exponential fragment although it seems one left-nested implication is now necessary. For each equation i we add a new propositional constant q_i representing -1, and the hypothesis

$$!((p_i \otimes q_i) \multimap \mathbf{1})$$

expressing that $1 + (-1) = 0$. An occurrence of a variable on the right-hand side of an equation is implemented as a negative occurrence on the left. For

[3] [*add a note on what is known about the complexity of these two fragments*]

example,

$$
\begin{array}{ll}
\ulcorner x + 1 = 2y \wedge 2x + y - 2 = 3 \urcorner = & \\
!(p_1 \otimes q_1 \multimap \mathbf{1}) & \text{1+(-1)=0} \\
\otimes !(p_2 \otimes q_2 \multimap \mathbf{1}) & \text{1+(-1)=0} \\
\otimes !(p_1 \otimes p_2^2) & x \text{ and } 2x \\
\otimes !(q_1^2 \otimes p_2) & -2y \text{ and } y \\
\multimap & \\
(p_1 \multimap \mathbf{1}) & (x - 2y) + 1 = 0 \\
\otimes (q_2^2 \multimap p_2^3) & \text{and } (2x + y) - 2 = 3
\end{array}
$$

3.6 Weakly Uniform Derivations

In the preceding sections we have dealt with universal, existential, and resource non-determinism. The inversion principles in Sections 3.1 and 3.4 reduced *disjunctive non-determinism*, since invertible rules can always be applied eagerly in the bottom-up search for a derivation without losing completeness. In this section we extend this analysis in order to reduce disjunctive non-determinism even when rules are not necessarily invertible, resulting in a notion of *weakly uniform derivations.*

> [***Warning:*** *The material in this section is highly speculative. While the soundness of the procedure is clear, its completeness has not yet been proven and quite possibly fails.*]

As in the section for unification, we describe the search procedure as a deductive system. We do not explicitly deal with universal, existential, or resource non-determinism, which can easily be incorporated in the system following the ideas in the previous two sections. Unlike unification, however, there remains some residual disjunctive non-determinism which reflects the remaining difficult choices. This can be treated, for example, by backtracking in an iterative deepening implementation.

The basic structure of sequents in the procedure is

$$\Gamma; \Delta \longrightarrow A$$

where Γ contains only propositions whose corresponding unrestricted left rule is not invertible and Δ contains only propositions whose corresponding left rules are not invertible. If a right rule is invertible in this situation, we apply it, until we are in a situation where neither the left rules for the principal connectives on the hypotheses, nor the right rule for the conclusion is invertible. Then we have to make a choice and pick one of the hypotheses or the conclusion.

The surprising[4] observation is that we can now *focus* on this hypothesis or conclusion and break it down, applying a sequence of left or right rules as long as the corresponding rules for the principal connective are *not* invertible. We

[4] *[and perhaps false]*

call the structure of the resulting derivation *weakly focussed*. In the strongly focussed case which is relevant to logic programming, we can always reduce a focussed hypothesis or conclusion to an atomic proposition; here we may have to stop when we encounter other propositions.

We now go through various classes of inference rules. We omit the cases for intuitionistic implication (which is left to Exercise **??**), since it is easily defined by $A_1 \supset A_2 = (!A_1) \multimap A_2$.

Invertible Right Rules. Whenever the right rule for the principal connective of the succedent is applicable, we must apply it.

$$\frac{\Gamma; \Delta \longrightarrow A_1 \qquad \Gamma; \Delta \longrightarrow A_2}{\Gamma; \Delta \longrightarrow A_1 \& A_2}\,\&\mathrm{R} \qquad \frac{}{\Gamma; \Delta \longrightarrow \top}\,\top\mathrm{R}$$

The right rule for linear implication is invertible, but it requires a new auxiliary judgment. When applying the right rule to $A_1 \multimap A_2$ we cannot add A_1 directly to Δ, since its principal connective may have an invertible left rule, violating our invariant. Instead, we employ an auxiliary judgment $\Gamma; (\Delta \mid \Delta') \longrightarrow A$. Interpreted algorithmically, it will apply all the invertible left rules to the propositions in Δ' and merge the remaining ones into Δ

$$\frac{\Gamma; (\Delta \mid A_1) \longrightarrow A_2}{\Gamma; \Delta \Longrightarrow A_1 \multimap A_2}\multimap\mathrm{R} \qquad \frac{\Gamma; \Delta \longrightarrow [a/x]A}{\Gamma; \Delta \longrightarrow \forall x.\, A}\,\forall\mathrm{R}^a$$

Invertible Left Rules. The invertible left rules require yet another additional judgment to merge unrestricted hypotheses into Γ in a way that eliminates all unrestricted invertible rules, written as $(\Gamma \mid \Gamma'); (\Delta \mid \Delta') \longrightarrow B$.

$$\frac{\Gamma; (\Delta \mid A_1, A_2, \Delta') \longrightarrow B}{\Gamma; (\Delta \mid A_1 \otimes A_2, \Delta') \longrightarrow B}\,\otimes\mathrm{L} \qquad \frac{\Gamma; (\Delta \mid \Delta') \longrightarrow B}{\Gamma; (\Delta \mid \mathbf{1}, \Delta') \longrightarrow B}\,\mathbf{1}\mathrm{L}$$

$$\frac{\Gamma; (\Delta \mid A_1, \Delta') \longrightarrow B \qquad \Gamma; (\Delta \mid A_2, \Delta') \longrightarrow B}{\Gamma; (\Delta \mid A_1 \oplus A_2, \Delta') \longrightarrow B}\,\oplus\mathrm{L} \qquad \frac{}{\Gamma; (\Delta \mid \mathbf{0}, \Delta') \longrightarrow B}\,\mathbf{0}\mathrm{L}$$

$$\frac{(\Gamma \mid A); (\Delta \mid \Delta') \longrightarrow B}{\Gamma; (\Delta \mid !A, \Delta') \longrightarrow B}\,!\mathrm{L} \qquad \frac{\Gamma; (\Delta \mid [a/x]A, \Delta') \longrightarrow B}{\Gamma; (\Delta \mid \exists x.\, A, \Delta')}\,\exists\mathrm{L}^a$$

There is also one rule which allows us to merge remaining propositions into Δ, and one rule which allows us to go back to work on the succedent when all hypotheses have been decomposed and merged into Δ.

$$\frac{\Gamma; (\Delta, D \mid \Delta') \longrightarrow B}{\Gamma; (\Delta \mid D, \Delta') \longrightarrow B}\,\mathrm{md} \qquad \frac{\Gamma; \Delta \longrightarrow B}{\Gamma; (\Delta \mid \cdot) \longrightarrow B}\,\mathrm{me}$$

In the md rule, we use D to stand for a proposition whose left rule is non-invertible. We we call them *left-critical propositions.*

$$\textit{Left-Critical Propositions} \quad D \quad ::= \quad P \mid A_1 \& A_2 \mid \top \mid A_1 \multimap A_2 \mid \forall x.\, A$$

Invertible Unrestricted Left Rules. Next we show the rules which eagerly apply invertible left rules to unrestricted hypotheses.

$$\frac{(\Gamma \mid A_1, A_2, \Gamma'); (\Delta \mid \Delta') \longrightarrow B}{(\Gamma \mid A_1 \& A_2, \Gamma'); (\Delta \mid \Delta') \longrightarrow B}\,\&\mathrm{L!} \qquad \frac{(\Gamma \mid \Gamma'); (\Delta \mid \Delta') \longrightarrow B}{(\Gamma \mid \top, \Gamma'); (\Delta \mid \Delta') \longrightarrow B}\,\top\mathrm{L!}$$

$$\frac{(\Gamma \mid \Gamma'); (\Delta \mid \Delta') \longrightarrow B}{(\Gamma \mid \mathbf{1}, \Gamma'); (\Delta \mid \Delta') \longrightarrow B}\,\mathbf{1}\mathrm{L!} \qquad \frac{}{(\Gamma \mid \mathbf{0}, \Gamma'); (\Delta \mid \Delta') \longrightarrow B}\,\mathbf{0}\mathrm{L!}$$

$$\frac{(\Gamma \mid A, \Gamma'); (\Delta \mid \Delta') \longrightarrow B}{(\Gamma \mid\, !A, \Gamma'); (\Delta \mid \Delta') \longrightarrow B}\,!\mathrm{L!}$$

$$\frac{(\Gamma, E \mid \Gamma'); (\Delta \mid \Delta') \longrightarrow B}{(\Gamma \mid E, \Gamma'); (\Delta \mid \Delta') \longrightarrow B}\,\mathrm{md!} \qquad \frac{\Gamma; (\Delta \mid \Delta') \longrightarrow B}{(\Gamma \mid \cdot); (\Delta \mid \Delta') \longrightarrow B}\,\mathrm{me!}$$

In the md! rule we use E to stand for those propositions whose principal connective is not invertible when it appears among the unrestricted resources.

$$\begin{array}{lcl} \textit{Left!-Critical Propositions} \quad E & ::= & P \mid A_1 \multimap A_2 \mid \forall x.\, A \\ & & \mid A_1 \otimes A_2 \mid A_1 \oplus A_2 \mid \exists x.\, A \end{array}$$

Focussing Rules. With the rules above we arrive at a situation where Γ consists entirely of left!-critical propositions, Δ of left-critical propositions, and the succedent is a *right-critical proposition* C, defined as a proposition whose principal connective does not have an invertible right rule.

$$\textit{Right-Critical Propositions} \quad C \quad ::= \quad P \mid A_1 \otimes A_2 \mid \mathbf{1} \mid A_1 \oplus A_2 \mid \mathbf{0} \mid\, !A \mid \exists x.\, A$$

We also use C^* to stand for a non-atomic right-critical proposition. The idea of focussing is to pick either the succedent of the sequent or one of the linear or unrestricted hypothesis and apply a sequence of either left or right rules to this one distinguished proposition. Thus we have two judgments,

$$\begin{array}{ll} \Gamma; \Delta \longrightarrow\gg A & A \text{ has a right-focussed derivation, and} \\ \Gamma; \Delta \longrightarrow A \gg C & C \text{ has a derivation left-focussed on } A. \end{array}$$

They arise in a derivation when a sequent consists entirely of critical propositions.

$$\frac{\Gamma; \Delta \longrightarrow\gg C^*}{\Gamma; \Delta \longrightarrow C^*}\gg\mathrm{R}$$

$$\frac{\Gamma; \Delta \longrightarrow D \gg C}{\Gamma; (\Delta, D) \longrightarrow C} \gg \mathrm{L} \qquad \frac{(\Gamma, E); \Delta \longrightarrow E \gg C}{(\Gamma, E); \Delta \longrightarrow C} \gg \mathrm{L!}$$

Here, we restrict the $\gg$ R rule to *non-atomic* critical propositions C^*. Note that these three rules represent a don't-know non-deterministic choice. In the presence of resource non-determinism, some of the hypotheses may only be potentially available—choosing them forces them to occur, possibly compromising other pending subgoals. This is also why, for example, the **1**R rule below is not trivial: in practice it constitutes a commitment that none of the potential hypotheses are used in this branch of the derivation.

The rules for right-focussed sequents and immediate entailment are once again just the right and left rules, until the focus proposition is no longer critical, at which point a new critical sequent may arise.

$$\frac{\Gamma; \Delta_1 \longrightarrow\gg A_1 \qquad \Gamma; \Delta_2 \longrightarrow\gg A_2}{\Gamma; \Delta_1 \times \Delta_2 \longrightarrow\gg A_1 \otimes A_2} \otimes\mathrm{R} \qquad \frac{}{\Gamma; \cdot \longrightarrow\gg \mathbf{1}} \mathbf{1}\mathrm{R}$$

$$\frac{\Gamma; \Delta \longrightarrow\gg A_1}{\Gamma; \Delta \longrightarrow\gg A_1 \oplus A_2} \oplus\mathrm{R}_1 \qquad \frac{\Gamma; \Delta \longrightarrow\gg A_2}{\Gamma; \Delta \longrightarrow\gg A_1 \oplus A_2} \oplus\mathrm{R}_2$$

No right rule for $\mathbf{0}$

$$\frac{\Gamma; \cdot \longrightarrow\gg A}{\Gamma; \cdot \longrightarrow\gg\, !A} !\mathrm{R}$$

$$\frac{\Gamma; \Delta \longrightarrow\gg [t/x]A}{\Gamma; \Delta \longrightarrow\gg \exists x.\, A} \exists\mathrm{R} \qquad \frac{\Gamma; \Delta \longrightarrow \overline{C^*}}{\Gamma; \Delta \longrightarrow\gg \overline{C^*}} \mathrm{ur}$$

In the last rule $\overline{C^*}$ is a proposition which is either atomic or not right critical.

$$\frac{\Gamma; \Delta \longrightarrow A_1 \gg C}{\Gamma; \Delta \longrightarrow A_1 \& A_2 \gg C} \&\mathrm{L}_1 \qquad \frac{\Gamma; \Delta \longrightarrow A_2 \gg C}{\Gamma; \Delta \longrightarrow A_1 \& A_2 \gg C} \&\mathrm{L}_2$$

No left rule for $\top$

$$\frac{\Gamma; \Delta_2 \longrightarrow A_2 \gg C \qquad \Gamma; \Delta_1 \longrightarrow A_1}{\Gamma; \Delta_1 \times \Delta_2 \longrightarrow A_1 \multimap A_2 \gg C} \multimap \mathrm{L}$$

$$\frac{\Gamma; \Delta \longrightarrow [t/x]A \gg C}{\Gamma; \Delta \longrightarrow \forall x.\, A \gg C} \forall\mathrm{L} \qquad \frac{}{\Gamma; \cdot \longrightarrow P \gg P} \mathrm{I}$$

$$\frac{\Gamma; (\Delta \mid D) \longrightarrow C}{\Gamma; \Delta \longrightarrow D \gg C} \mathrm{ul}$$

Note that the premisses of the $\multimap$ L rule are another immediate entailment and a general, weakly uniform derivation of the antecedent of the linear implication.

The soundness of this inference system can be seen rather easily by induction, but we must generalize the statement to include all auxiliary judgments.

Theorem 3.14 (Soundness of Weakly Uniform Derivations)

1. *If* $\Gamma; \Delta \longrightarrow A$ *then* $\Gamma; \Delta \overset{-}{\Longrightarrow} A$.
2. *If* $\Gamma; (\Delta \mid \Delta') \longrightarrow A$ *then* $\Gamma; (\Delta, \Delta') \overset{-}{\Longrightarrow} A$.
3. *If* $(\Gamma \mid \Gamma'); (\Delta \mid \Delta') \longrightarrow A$ *then* $(\Gamma, \Gamma'); (\Delta, \Delta') \overset{-}{\Longrightarrow} A$.
4. *If* $\Gamma; \Delta \longrightarrow\gg A$ *then* $\Gamma; \Delta \overset{-}{\Longrightarrow} A$.
5. *If* $\Gamma; \Delta \longrightarrow A \gg C$ *then* $\Gamma; (\Delta, A) \overset{-}{\Longrightarrow} A$.

Proof: By a straightforward simultaneous induction on the given derivations. Each left and right rule in the weakly uniform derivation corresponds directly to a left and right rule on the sequent calculus. For the rules $\&$L!, $\top$L!, **1**L!, **0**L!, !L! we use their admissibility (Theorem 3.13). The remaining rules disappear in the translation, since the premiss and conclusion sequent are intpreted identically. $\square$

The completeness is open at present. One appropriate proof technique would be to use the *permutability* of inference rules to show explicitly how to transform a derivation of $\Gamma; \Delta \overset{-}{\Longrightarrow} A$ into a weakly uniform one. The only difficult in this proof is the explosive number of cases which must be checked.

Chapter 4

Linear λ-Calculus

In intuitionistic logic, proofs are related to functional programs via the *Curry-Howard isomorphism* [CF58, How69]. Howard observed that there is a bijective correspondence between proofs in intuitionistic propositional natural deduction and simply-typed λ-terms. A related observation on proof in combinatory logic had been made previously by Curry.

A generalization of this observation to include quantifiers later gives rise to the rich field of type theory, which we will analyze in Chapter **??**. Here we study the basic correspondence, extended to the case of linear logic.

A linear λ-calculus of proof terms will be useful for us in various circumstances. First of all, it gives a compact and faithful representation of proofs as terms. Proof checking is reduced to type-checking in a λ-calculus. For example, if we do not trust the implementation of our theorem prover, we can instrument it to generate proof terms which can be verified independently. Secondly, the terms in the λ-calculus provide the core of a functional language with an expressive type system, in which statements such as "*this function will use its argument exactly once*" can be formally expressed and checked. Thirdly, linear λ-terms can serve as an expressive representation language within a *logical framework*, a general meta-language for the formalization of deductive systems.

4.1 Proof Terms

We now assign proof terms to the system of linear natural deduction. Our main criterion for the design of the proof term language is that the proof terms should reflect the structure of the deduction as closely as possible. Moreover, we would like every valid proof term to uniquely determine a natural deduction. Because of the presence of $\top$, this strong property will fail, but a slightly weaker and, from the practical point of view, sufficient property holds. Under the Curry-Howard isomorphism, a proposition corresponds to a type in the proof term calculus. We will there call a proof term *well-typed* if it represents a deduction.

The proof term assignment is defined via the judgment $\Gamma; \Delta \vdash M : A$, where each formula in Γ and Δ is labelled. We also use $M \longrightarrow_\beta M'$ for the local reduction and $M : A \longrightarrow_\eta M'$ for the local expansion, both expressed on proof terms. The type on the left-hand side of the expansion reminds is a reminder that this rule only applies to term of the given type (contexts are elided here).

Hypotheses. We use the label of the hypotheses as the name for a variable in the proof terms. There are no reductions or expansions specific to variables, although variables of non-atomic type may be expanded by the later rules.

$$\frac{}{\Gamma; (\cdot, w{:}A) \vdash w : A}\, w \qquad \frac{}{(\Gamma_1, u{:}A, \Gamma_2); \cdot \vdash u : A}\, u$$

Multiplicative Connectives. Linear implication corresponds to a *linear function types* with corresponding linear abstraction and application. We distinguish them from unrestricted abstraction and application by a "hat". In certain circumstances, this may be unnecessary, but here we want to reflect the proof structure as directly as possible.

$$\frac{\Gamma; (\Delta, w{:}A) \vdash M : B}{\Gamma; \Delta \vdash \hat{\lambda} w{:}A.\ M : A \multimap B} \multimap \mathrm{I}^w$$

$$\frac{\Gamma; \Delta \vdash M : A \multimap B \qquad \Gamma; \Delta' \vdash N : A}{\Gamma; (\Delta \times \Delta') \vdash M \,\hat{}\, N : B} \multimap \mathrm{E}$$

$$\begin{array}{rcl} (\hat{\lambda} w{:}A.\ M) \,\hat{}\, N & \longrightarrow_\beta & [N/w]M \\ M : A \multimap B & \longrightarrow_\eta & \hat{\lambda} w{:}A.\ M \,\hat{}\, w \end{array}$$

In the rules for the simultaneous conjunction, the proof term for the elimination inference is a **let** form which deconstructs a pair, naming the components. The linearity of the two new hypotheses means that the variables must both be used in M.

$$\frac{\Gamma; \Delta_1 \vdash M : A \qquad \Gamma; \Delta_2 \vdash N : B}{\Gamma; (\Delta_1 \times \Delta_2) \vdash M \otimes N : A \otimes B} \otimes \mathrm{I}$$

$$\frac{\Gamma; \Delta \vdash M : A \otimes B \quad \Gamma; (\Delta', w_1{:}A, w_2{:}B) \vdash N : C}{\Gamma; (\Delta' \times \Delta) \vdash \mathbf{let}\ w_1 \otimes w_2 = M\ \mathbf{in}\ N : C} \otimes \mathrm{E}^{w_1, w_2}$$

The reduction and expansion mirror the local reduction and expansion for deduction as the level of proof terms. We do not reiterate them here, but simply give the proof term reduction.

$$\begin{array}{rcl} \mathbf{let}\ w_1 \otimes w_2 = M_1 \otimes M_2\ \mathbf{in}\ N & \longrightarrow_\beta & [M_1/w_1, M_2/w_2]N \\ M : A \otimes B & \longrightarrow_\eta & \mathbf{let}\ w_1 \otimes w_2 = M\ \mathbf{in}\ w_1 \otimes w_2 \end{array}$$

The unit type allows us to consume linear hypotheses without introducing new linear ones.

$$\frac{}{\Gamma; \cdot \vdash \star : \mathbf{1}}\,\mathbf{1}\mathrm{I} \qquad \frac{\Gamma; \Delta \vdash M : \mathbf{1} \qquad \Gamma; \Delta' \vdash N : C}{\Gamma; (\Delta' \times \Delta) \vdash \mathbf{let}\ \star = M\ \mathbf{in}\ N : C}\,\mathbf{1}\mathrm{E}$$

$$\begin{array}{rcl} \mathbf{let}\ \star = M\ \mathbf{in}\ N & \longrightarrow_\beta & N \\ M : \star & \longrightarrow_\eta & \mathbf{let}\ \star = M\ \mathbf{in}\ \star \end{array}$$

Additive Connectives. As we have seen from the embedding of intuitionistic in linear logic, the simultaneous conjunction represents products from the simply-typed λ-calculus.

$$\frac{\Gamma; \Delta \vdash M : A \qquad \Gamma; \Delta \vdash N : B}{\Gamma; \Delta \vdash \langle M, N \rangle : A \& B}\,\&\mathrm{I}$$

$$\frac{\Gamma; \Delta \vdash M : A \& B}{\Gamma; \Delta \vdash \mathrm{fst}\, M : A}\,\&\mathrm{E_L} \qquad \frac{\Gamma; \Delta \vdash M : A \& B}{\Gamma; \Delta \vdash \mathrm{snd}\, M : B}\,\&\mathrm{E_R}$$

The local reduction are also the familiar ones.

$$\begin{array}{rcl} \mathrm{fst}\,\langle M_1, M_2 \rangle & \longrightarrow_\beta & M_1 \\ \mathrm{snd}\,\langle M_1, M_2 \rangle & \longrightarrow_\beta & M_2 \\ M : A \& B & \longrightarrow_\eta & \langle \mathrm{fst}\, M, \mathrm{snd}\, M \rangle \end{array}$$

The additive unit corresponds to a unit type with no operations on it.

$$\frac{}{\Gamma; \Delta \vdash \langle\,\rangle : \top}\,\top\mathrm{I} \qquad \textit{No } \top \textit{ elimination}$$

The additive unit has no elimination and therefore no reduction. However, it still admits an expansion, which witnesses the local completeness of the rules.

$$M : \top \quad \longrightarrow_\eta \quad \langle\,\rangle$$

The disjunction (or *disjoint sum* when viewed as a type) uses injection and case as constructor and destructor forms, respectively. We annotated the injections with a type to preserve the property that any well-typed term has a unique type.

$$\frac{\Gamma; \Delta \vdash M : A}{\Gamma; \Delta \vdash \mathrm{inl}^B : A \oplus B}\,\oplus\mathrm{I_L} \qquad \frac{\Gamma; \Delta \vdash M : B}{\Gamma; \Delta \vdash \mathrm{inr}^A : A \oplus B}\,\oplus\mathrm{I_R}$$

$$\frac{\Gamma; \Delta \vdash M : A \oplus B \quad \Gamma; (\Delta', w_1{:}A) \vdash N_1 : C \quad \Gamma; (\Delta', w_2{:}B) \vdash N_2 : C}{\Gamma; (\Delta' \times \Delta) \vdash \mathbf{case}\ M\ \mathbf{of}\ \mathrm{inl}\, w_1 \Rightarrow N_1 \mid \mathrm{inr}\, w_2 \Rightarrow N_2 : C}\,\oplus\mathrm{E}^{w_1, w_2}$$

The reductions are just like the ones for disjoint sums in the simply-typed λ-caclulus.

$$\begin{array}{lcl}
\mathbf{case}\ \mathrm{inl}^B\, M\ \mathbf{of}\ \mathrm{inl}\, w_1 \Rightarrow N_1 \mid \mathrm{inr}\, w_2 \Rightarrow N_2 & \longrightarrow_\beta & [M/w_1]N_1 \\
\mathbf{case}\ \mathrm{inr}^A\, M\ \mathbf{of}\ \mathrm{inl}\, w_1 \Rightarrow N_1 \mid \mathrm{inr}\, w_2 \Rightarrow N_2 & \longrightarrow_\beta & [M/w_2]N_2
\end{array}$$

$$M : A \oplus B \quad \longrightarrow_\eta \quad \mathbf{case}\ M\ \mathbf{of}\ \mathrm{inl}\, w_1 \Rightarrow \mathrm{inl}^B\, w_1 \mid \mathrm{inr}\, w_2 \Rightarrow \mathrm{inr}^A\, w_2$$

For the additive falsehood, there is no introduction rule. It corresponds to a *void type* without any values. Consequently, there is no reduction. Once again we annotate the abort constructor in order to guarantee uniqueness of types.

$$\textit{No } \mathbf{0} \textit{ introduction} \qquad \frac{\Gamma; \Delta \vdash M : \mathbf{0}}{\Gamma; (\Delta' \times \Delta) \vdash \mathrm{abort}^C\, M : C}\,\mathbf{0}\mathrm{E}$$

$$M : \mathbf{0} \quad \longrightarrow_\eta \quad \mathrm{abort}^{\mathbf{0}}\, M$$

Exponentials. Unrestricted implication corresponds to the usual *function type* from the simply-typed λ-calculus. For consistency, we will still write $A \supset B$ instead of $A \to B$, which is more common in λ-calculus. Note that the argument of an unrestricted application may not mention any linear variables.

$$\frac{(\Gamma, u{:}A); \Delta \vdash M : B}{\Gamma; \Delta \vdash \lambda u{:}A.\ M : A \supset B}\,{\supset}\mathrm{I}^u$$

$$\frac{\Gamma; \Delta \vdash M : A \supset B \qquad \Gamma; \cdot \vdash N : A}{\Gamma; \Delta \vdash M\, N : B}\,{\supset}\mathrm{E}$$

The reduction and expansion are the origin of the β and η rules names due to Church [Chu41].

$$\begin{array}{rcl}
(\lambda u{:}A.\ M)\, N & \longrightarrow_\beta & [N/u]M \\
M : A \supset B & \longrightarrow_\eta & \lambda u{:}A.\ M\, u
\end{array}$$

The rules for the *of course* operator allow us to name term of type $!A$ and use it freely in further computation.

$$\frac{\Gamma; \cdot \vdash M : A}{\Gamma; \cdot \vdash !M : !A}\,!\mathrm{I} \qquad \frac{\Gamma; \Delta \vdash M : !A \qquad (\Gamma, u{:}A); \Delta' \vdash N : C}{\Gamma; (\Delta' \times \Delta) \vdash \mathbf{let}\ !u = M\ \mathbf{in}\ N : C}\,!\mathrm{E}^u$$

$$\begin{array}{rcl}
\mathbf{let}\ !u = !M\ \mathbf{in}\ N & \longrightarrow_\beta & [M/u]N \\
M : !A & \longrightarrow_\eta & \mathbf{let}\ !u = M\ \mathbf{in}\ !u
\end{array}$$

Below is a summary of the linear λ-calculus with the β-reduction and η-expansion rules.

$$
\begin{array}{lll}
M & ::= & w & \textit{Linear Variables} \\
 & & \mid \hat{\lambda} w{:}A.\ M \mid M_1 \hat{\ } M_2 & A \multimap B \\
 & & \mid M_1 \otimes M_2 \mid \mathbf{let}\ w_1 \otimes w_2 = M\ \mathbf{in}\ M' & A \otimes B \\
 & & \mid \star \mid \mathbf{let}\ \star = M\ \mathbf{in}\ M' & \mathbf{1} \\
 & & \mid \langle M_1, M_2 \rangle \mid \mathrm{fst}\, M_1 \mid \mathrm{snd}\, M_2 & A \& B \\
 & & \mid \langle\, \rangle & \top \\
 & & \mid \mathrm{inl}^B\, M \mid \mathrm{inr}^A\, M & A \oplus B \\
 & & \quad \mid (\mathbf{case}\ M\ \mathbf{of}\ \mathrm{inl}\, w_1 \Rightarrow M_1 \mid \mathrm{inr}\, w_2 \Rightarrow M_2) & \\
 & & \mid \mathrm{abort}^C\, M & \mathbf{0} \\
 & & \mid u & \textit{Unrestricted Variables} \\
 & & \mid \lambda u{:}A.\ M \mid M_1\, M_2 & A \supset B \\
 & & \mid\ !M \mid \mathbf{let}\ u = M\ \mathbf{in}\ M' & !A
\end{array}
$$

Below is a summary of the β-reduction rules, which correspond to local reductions of natural deductions.

$$
\begin{array}{rcll}
(\hat{\lambda} w{:}A.\ M) \hat{\ } N & \longrightarrow_\beta & [N/w]M & A \multimap B \\
\mathbf{let}\ w_1 \otimes w_2 = M_1 \otimes M_2\ \mathbf{in}\ N & \longrightarrow_\beta & [M_1/w_1, M_2/w_2]N & A \otimes B \\
\mathbf{let}\ \star = M\ \mathbf{in}\ N & \longrightarrow_\beta & N & \mathbf{1} \\
\mathrm{fst}\, \langle M_1, M_2 \rangle & \longrightarrow_\beta & M_1 & A \& B \\
\mathrm{snd}\, \langle M_1, M_2 \rangle & \longrightarrow_\beta & M_2 & \\
\textit{No } \top \textit{ reduction} & & & \\
\mathbf{case}\ \mathrm{inl}^B\, M\ \mathbf{of}\ \mathrm{inl}\, w_1 \Rightarrow N_1 \mid \mathrm{inr}\, w_2 \Rightarrow N_2 & \longrightarrow_\beta & [M/w_1]N_1 & A \oplus B \\
\mathbf{case}\ \mathrm{inr}^A\, M\ \mathbf{of}\ \mathrm{inl}\, w_1 \Rightarrow N_1 \mid \mathrm{inr}\, w_2 \Rightarrow N_2 & \longrightarrow_\beta & [M/w_1]N_2 & \\
\textit{No } \mathbf{0} \textit{ reduction} & & & \\
(\lambda u{:}A.\ M)\, N & \longrightarrow_\beta & [N/u]M & A \supset B \\
\mathbf{let}\ !u = !M\ \mathbf{in}\ N & \longrightarrow_\beta & [M/u]N & !A
\end{array}
$$

The substitution $[M/w]N$ and $[M/u]N$ assumes that there are no free variables in M which would be captured by a variables binding in N. We nonethless consider it a total function, since the capturing variable can always be renamed to avoid a conflict (see Exercise 4.3).

Next is a summary of the η-expansion rules, which correspond to local expansions of natural deductions.

$$
\begin{array}{rcl}
M : A \multimap B & \longrightarrow_\eta & \hat{\lambda} w{:}A.\ M \hat{\ } w \\
M : A \otimes B & \longrightarrow_\eta & \mathbf{let}\ w_1 \otimes w_2 = M\ \mathbf{in}\ w_1 \otimes w_2 \\
M : \star & \longrightarrow_\eta & \mathbf{let}\ \star = M\ \mathbf{in}\ \star \\
M : A \& B & \longrightarrow_\eta & \langle \mathrm{fst}\, M, \mathrm{snd}\, M \rangle \\
M : \top & \longrightarrow_\eta & \langle\, \rangle \\
M : A \oplus B & \longrightarrow_\eta & \mathbf{case}\ M\ \mathbf{of}\ \mathrm{inl}\, w_1 \Rightarrow \mathrm{inl}^B\, w_1 \mid \mathrm{inr}\, w_2 \Rightarrow \mathrm{inr}^A\, w_2 \\
M : \mathbf{0} & \longrightarrow_\eta & \mathrm{abort}^{\mathbf{0}}\, M \\
M : A \supset B & \longrightarrow_\eta & \lambda u{:}A.\ M\, u \\
M : !A & \longrightarrow_\eta & \mathbf{let}\ !u = M\ \mathbf{in}\ !u
\end{array}
$$

Note that there is an implicit assumption that the variables w and u in the cases for $A \multimap B$ and $A \supset B$ do not already occur in M: they are chosen to be new.

We have the following fundamental properties. Uniqueness, where claimed, holds only up to renaming of bound variables.

Theorem 4.1 (Properties of Proof Terms)

1. *If* $\Gamma; \Delta \vdash A$ *then* $\Gamma; \Delta \vdash M : A$ *for a unique* M.
2. *If* $\Gamma; \Delta \vdash M : A$ *then* $\Gamma; \Delta \vdash A$.

Proof: By straightforward inductions over the given derivations. □

Types are also unique for well-typed terms (see Exercise 4.1). Uniqueness of derivations fails, that is, a proof term does not uniquely determine its derivation, even under identical contexts. A simple counterexample is provided by the following two derivations (with the empty unrestricted context elided).

$$\dfrac{\dfrac{}{w{:}\top \vdash \langle\,\rangle : \top}\top I \qquad \dfrac{}{\cdot \vdash \langle\,\rangle : \top}\top I}{w{:}\top \vdash \langle\,\rangle \otimes \langle\,\rangle : \top \otimes \top}\otimes I \qquad \dfrac{\dfrac{}{\cdot \vdash \langle\,\rangle : \top}\top I \qquad \dfrac{}{w{:}\top \vdash \langle\,\rangle : \top}\top I}{w{:}\top \vdash \langle\,\rangle \otimes \langle\,\rangle : \top \otimes \top}\otimes I$$

It can be shown that linear hypotheses which are absorbed by $\top I$ are the only source of only ambiguity in the derivation. A similar ambiguity already exists in the sense that any proof term remains valid under weakening in the intuitionistic context: whenever $\Gamma; \Delta \vdash M : A$ then $(\Gamma, \Gamma'); \Delta \vdash M : A$. So this phenomenon is not new to the linear λ-calculus, and is in fact a useful identification of derivations which differ in "irrelevant" details, that is, unused or absorbed hypotheses.

The substitution principles on natural deductions can be expressed on proof terms. This is because the translations from natural deductions to proof terms and *vice versa* are *compositional*: uses of a hypothesis labelled w in natural deduction corresponds to an occurrence of a variable w in the proof term.

Lemma 4.2 (Substitution on Proof Terms)

1. *If* $\Gamma; (\Delta, w{:}A) \vdash N{:}C$ *and* $\Gamma; \Delta' \vdash M : A$*, then* $\Gamma; (\Delta \times \Delta') \vdash [M/w]N : C$.
2. *If* $(\Gamma, u{:}A); \Delta \vdash N{:}C$ *and* $\Gamma; \cdot \vdash M : A$*, then* $\Gamma; \Delta \vdash [M/u]N : C$.

Proof: By induction on the structure of the first given derivation, using the property of exchange. □

We also have the property of weakening for unrestricted hypotheses. The substitution properties are the critical ingredient for the important *subject reduction* properties, which guarantee that the result of β-reducing a well-typed term will again be well-typed. The expansion rules also preserve types when invoked properly.

Theorem 4.3 (Subject Reduction and Expansion)

1. *If* $\Gamma; \Delta \vdash M : A$ *and* $M \longrightarrow_\beta M'$ *then* $\Gamma; \Delta \vdash M' : A$.
2. *If* $\Gamma; \Delta \vdash M : A$ *and* $M : A \longrightarrow_\eta M'$ *then* $\Gamma; \Delta \vdash M' : A$.

Proof: For subject reduction we examine each possible reduction rule, applying inversion to obtain the shape of the typing derivation. From this we either directly construct the typing derivation of M' or we appeal to the substitution lemma.

For subject expansion we directly construct the typing derivation for M' from the typing derivation of M. □

Note that the opposite of subject reduction does not hold: there are well-typed terms M' such that $M \longrightarrow_\beta M'$ and M is not well-typed (see Exercise 4.4).

4.2 Example: A Small Imperative Language

[*to be filled in*]

4.3 Term Assignment for the Sequent Calculus

Writing an efficient theorem prover is an arduous and error-prone task, since one must carefully optimize at low and high levels of abstraction. This means that it is difficult to trust the correctness of a theorem prover. In order to alleviate this problem, we can follow two strategies. The first goes back to the design ML [GMW79] where we use the strong typing and the data abstraction mechanisms of the implementation language to reduce the correctness of a larger implementation to the correctness of a small core. Unfortunately, having to always go through primitive rules of inference during search is a severe practical restriction and prohibits many efficient implementation techniques (see, for example, a discussion in [**?**]). Another is to require the prover to be able to generate *proof terms*. If the proof is written in a simple and concise language, we can write an external (and hopefully much simpler) checker, which we may trust much more readily than a complicated prover manipulating constraints, unification, using indexing schemes for fast retrieval of logical assumptions, etc. Unless we can find a way to include admissible and derived rules of inference, we still have to go through primitive inference rules, but we now have an external manifestation of the deduction.

For linear logic, the proof term calculus developed in Section 4.1 is an ideal candidate. Its definition is relatively simple, directly translating the rules of natural deduction. Compare this to the complexity of unification with parameters and manipulating occurrence constraints. In the next section we will also see that type-checking proof terms is certainly decidable and actually not a difficult task.

What remains is to bridge the gap between the rules of the sequent calculus and proof terms for natural deduction. Actually, we have already established the connection between sequent calculus and natural deduction in both directions. Of interest here is soundness, since the constructive soundness proof gives an explicit method for translating a sequent derivation into a natural deduction (see Theorem 2.9). We now need to make this construction explicit. This can be done by defining a judgment which relates a sequent derivation to a proof term, or perhaps to a natural deduction which includes a proof term. Such higher-level judgments which relate derivations quickly become unmanageable, so we write out one judgment which may be thought of as a sequent derivation annotated by a proof term, written as $\Gamma; \Delta \Longrightarrow M : A$. We are writing it in such a way that, if $\Gamma; \Delta \Longrightarrow A$ then there is an annotation $\Gamma; \Delta \Longrightarrow M : A$ and $\Gamma; \Delta \vdash M : A$.

Since variable occurrences in proof terms are critical, we now make the hypothesis labels explicit. However, we will still allow implicit exchange, so that $\Delta, w{:}A$ matches any hypotheses of the form $\Delta_1, w{:}A, \Delta_2$.

Hypotheses. The use of a hypothesis is just translated into the corresponding variable. The dereliction rule requires a substitution.

$$\frac{}{\Gamma; w{:}A \Longrightarrow w : A}\,\mathrm{I} \qquad \frac{(\Gamma, u{:}A); (\Delta, w{:}A) \Longrightarrow M : C}{(\Gamma, u{:}A); \Delta \Longrightarrow [u/w]M : C}\,\mathrm{DL}$$

Why is the substitution in the dereliction rule valid? In the correctness proof of the proof term assignment we need to show that $(\Gamma, u{:}A); \Delta \vdash [u/w]M : C$, given that $(\Gamma, u{:}A); (\Delta, w{:}A) \vdash M : C$. But this follows from the substitution property for proof terms (Lemma 4.2), since $(\Gamma, u{:}A); \cdot \vdash u : A$. Many other cases follow a similar pattern.

Multiplicative Connectives. In general, the right rules of the sequent calculus match the introduction rules of natural deduction. Therefore, the proof term assignment for the left rules is quite straightforward. On the other side, the left rules decompose a proposition in bottom-up search, while the elimination rules work top-down. We therefore substitute a small piece of a derivation which applies the corresponding elimination for the hypothesis in the premiss.

$$\frac{\Gamma; \Delta, w{:}A \Longrightarrow M : B}{\Gamma; \Delta \Longrightarrow \hat{\lambda} w{:}A.\ M : A \multimap B}\,{\multimap}\mathrm{R}$$

$$\frac{\Gamma; \Delta_1 \Longrightarrow M : A \qquad \Gamma; \Delta_2, w_2{:}B \Longrightarrow N : C}{\Gamma; \Delta_2 \times \Delta_1, w{:}A \multimap B \Longrightarrow [(w\,\hat{}\,M)/w_2]N : C}\,{\multimap}\mathrm{L}$$

$$\frac{\Gamma;\Delta_1 \Longrightarrow M_1 : A \qquad \Gamma;\Delta_2 \Longrightarrow M_2 : B}{\Gamma;\Delta_1 \times \Delta_2 \Longrightarrow M_1 \otimes M_2 : A \otimes B}\otimes\mathrm{R}$$

$$\frac{\Gamma;\Delta, w_1{:}A, w_2{:}B \Longrightarrow N : C}{\Gamma;\Delta, w{:}A \otimes B \Longrightarrow \mathbf{let}\ w_1 \otimes w_2 = w\ \mathbf{in}\ N : C}\otimes\mathrm{L}$$

$$\frac{}{\Gamma;\cdot \Longrightarrow \star : \mathbf{1}}\mathbf{1}\mathrm{R} \qquad \frac{\Gamma;\Delta \Longrightarrow N : C}{\Gamma;\Delta, w{:}\mathbf{1} \Longrightarrow \mathbf{let}\ \star = w\ \mathbf{in}\ N : C}\mathbf{1}\mathrm{L}$$

Additive Connectives. The additive connectives do not introduce any complications.

$$\frac{\Gamma;\Delta \Longrightarrow M : A \qquad \Gamma;\Delta \Longrightarrow N : B}{\Gamma;\Delta \Longrightarrow \langle M, N\rangle A \& B}\&\mathrm{R}$$

$$\frac{\Gamma;\Delta, w_1{:}A \Longrightarrow N : C}{\Gamma;\Delta, w{:}A\&B \Longrightarrow [(\mathrm{fst}\, w)/w_1]N : C}\&\mathrm{L}_1 \qquad \frac{\Gamma;\Delta, w_2{:}B \Longrightarrow N : C}{\Gamma;\Delta, w{:}A\&B \Longrightarrow [(\mathrm{snd}\, w)/w_1]N : C}\&\mathrm{L}_2$$

$$\frac{}{\Gamma;\Delta \Longrightarrow \langle\,\rangle\top}\top\mathrm{R} \qquad \textit{No } \top \textit{ left rule}$$

$$\frac{\Gamma;\Delta \Longrightarrow M : A}{\Gamma;\Delta \Longrightarrow \mathrm{inl}^B\, M : A \oplus B}\oplus\mathrm{R}_1 \qquad \frac{\Gamma;\Delta \Longrightarrow M : B}{\Gamma;\Delta \Longrightarrow \mathrm{inr}^A\, M : A \oplus B}\oplus\mathrm{R}_2$$

$$\frac{\Gamma;\Delta, w_1{:}A \Longrightarrow N_1 : C \qquad \Gamma;\Delta, w_2{:}B \Longrightarrow N_1 : C}{\Gamma;\Delta, w{:}A \oplus B \Longrightarrow \mathbf{case}\ w\ \mathbf{of}\ \mathrm{inl}\, w_1 \Rightarrow N_1 \mid \mathrm{inr}\, w_2 \Rightarrow N_2 : C}\oplus\mathrm{L}$$

$$\textit{No } \mathbf{0} \textit{ right rule} \qquad \frac{}{\Gamma;\Delta, w{:}\mathbf{0} \Longrightarrow \mathrm{abort}^C\, w : C}\mathbf{0}\mathrm{L}$$

Exponentials. Surprisingly, we do not need any explicit substitution for unrestricted variables, since the !L rule introduces a **let**-expression.

$$\frac{(\Gamma, u{:}A);\Delta \Longrightarrow M : B}{\Gamma;\Delta \Longrightarrow \lambda u{:}A.\ M : A \supset B}\supset\mathrm{R}$$

$$\frac{\Gamma;\cdot \Longrightarrow M : A \qquad \Gamma;\Delta, w_2{:}B \Longrightarrow N : C}{\Gamma;\Delta, w{:}A \supset B \Longrightarrow [(w\, M)/w_2]N : C}\supset\mathrm{L}$$

$$\frac{\Gamma;\cdot \Longrightarrow M : A}{\Gamma;\cdot \Longrightarrow !M : !A}!\mathrm{R}$$

$$\frac{(\Gamma, u{:}A);\Delta \Longrightarrow N : C}{\Gamma;\Delta, w{:}!A \Longrightarrow \mathbf{let}\ !u = w\ \mathbf{in}\ N : C}!\mathrm{L}$$

Cut. It is easy to check that the proof terms assigned with this system have the right type. Moreover, the resulting natural deduction terms are always normal. As can be expected from Theorem 2.14, this term assignment can be extended to derivations with cut, except that the result may no longer be normal. The assignment for the judgment $\Gamma; \Delta \overset{+}{\Longrightarrow} M : A$ is as above, with the following two additional rules.

$$\frac{\Gamma; \Delta \overset{+}{\Longrightarrow} M : A \qquad \Gamma; (\Delta', w{:}A) \overset{+}{\Longrightarrow} N : C}{\Gamma; \Delta' \times \Delta \overset{+}{\Longrightarrow} [M/w]N : C}\ \mathit{Cut}$$

$$\frac{\Gamma; \cdot \overset{+}{\Longrightarrow} M : A \qquad (\Gamma, u{:}A); \Delta' \overset{+}{\Longrightarrow} N : C}{\Gamma; \Delta' \overset{+}{\Longrightarrow} [M/u]N : C}\ \mathit{Cut!}$$

The following theorem summarizes the main properties of the term assignment system.

Theorem 4.4 (Term Assignment for Sequent Calculus)

1. *If* $\Gamma; \Delta \Longrightarrow A$ *then* $\Gamma; \Delta \Longrightarrow M : A$ *for a unique* M.
2. *If* $\Gamma; \Delta \Longrightarrow M : A$ *then* $\Gamma; \Delta \vdash M : A \uparrow$.
3. *If* $\Gamma; \Delta \overset{+}{\Longrightarrow} A$ *then* $\Gamma; \Delta \overset{+}{\Longrightarrow} M : A$ *for a unique* M.
4. *If* $\Gamma; \Delta \overset{+}{\Longrightarrow} M : A$ *then* $\Gamma; \Delta \vdash M : A$.

Proof: All by straightforward inductions over the structure of the given derivations, appealing to the substitution lemma 4.2 when necessary. □

Our proof term assignment was purposely designed to generate proof terms for the natural deduction system. This means the proof terms do not faithfully record the structure of the derivation and we cannot uniquely reconstruct a sequent derivation from a proof term. It is also possible to write out a proof term assignment which is faithful and then relate them to natural deduction proof terms (see Exercise 4.5).

4.4 Linear Type Checking

The typing rules for the linear λ-calculus are *syntax-directed* in that the principal term constructor determines the typing rule which must be used. Nonetheless, the typing rules are not immediately suitable for an efficient type-checking algorithm since we would have to guess how the linear hypotheses are to be split between the hypothesis in a number of rules.

The occurrence constraints introduced in Section 3.3 would be sufficient to avoid this choice, but they are rather complex, jeopardizing our goal of designing a simple procedure which is easy to trust. Fortunately, we have significantly more information here, since the proof term is given to us. This determines the amount of work we have to do in each branch of a derivation, and we can resolve the don't-care non-determinism directly.

Instead of guessing a split of the linear hypotheses between two premisses of a rule, we pass all linear variables to the first premiss. Checking the corresponding subterm will consume some of these variables, and we pass the remaining ones one to check the second subterms. This idea requires a judgment

$$\Gamma; \Delta_I \setminus \Delta_O \vdash M : A$$

where Δ_I represents the available linear hypotheses and $\Delta_O \subseteq \Delta_I$ the linear hypotheses not used in M. For example, the rules for the simultaneous conjunction and unit would be

$$\frac{\Gamma; \Delta_I \setminus \Delta' \vdash M : A \qquad \Gamma; \Delta' \setminus \Delta_O \vdash N : B}{\Gamma; \Delta_I \setminus \Delta_O \vdash M \otimes N : A \otimes B}\otimes\mathrm{I}$$

$$\frac{}{\Gamma; \Delta_I \setminus \Delta_I \vdash \star : A}\mathbf{1}\mathrm{I}.$$

Unfortunately, this idea breaks down when we encounter the additive unit (and only then!). Since we do not know which of the linear hypotheses might be used in a different branch of the derivation, it would have to read

$$\frac{\Delta_I \supseteq \Delta_O}{\Gamma; \Delta_I \setminus \Delta_O \vdash \langle\,\rangle : \top}\top\mathrm{I}$$

which introduces undesirable non-determinism if we were to guess which subset of Δ_I to return. In order to cirumvent this problem we return all of Δ_I, but flag it to indicate that it may not be exact, but that some of these linear hypotheses may be absorbed if necessary. In other words, in the judgment

$$\Gamma; \Delta_I \setminus \Delta_O \vdash_1 M : A$$

any of the remaining hypotheses in Δ_O need not be consumed in the other branches of the typing derivation. On the other hand, the judgment

$$\Gamma; \Delta_I \setminus \Delta_O \vdash_0 M : A$$

indicates the M uses exactly the variables in $\Delta_I - \Delta_O$.

When we think of the judgment $\Gamma; \Delta_I \setminus \Delta_O \vdash_i M : A$ as describing an algorithm, we think of Γ, Δ_I and M as given, and Δ_O and the slack indicator i as part of the result of the computation. The type A may or may not be given—in one case it is synthesized, in the other case checked. This refines our view as computation being described as the bottom-up construction of a derivation to include parts of the judgment in different roles (as input, output, or bidirectional components). In logic programming, which is based on the notion of computation-as-proof-search, these roles of the syntactic constituents of a judgment are called *modes*. When writing a deductive system to describe an algorithm, we have to be careful to respect the modes. We discuss this further when we come to the individual rules.

Hypotheses. The two variable rules leave no slack, since besides the hypothesis itself, no assumptions are consumed.

$$\frac{}{\Gamma; (\Delta_I, w{:}A) \setminus \Delta_I \vdash_0 w : A}\, w \qquad \frac{}{(\Gamma, u{:}A); \Delta_I \setminus \Delta_I \vdash_0 u : A}\, u$$

Multiplicative Connectives. For linear implication, we must make sure that the hypothesis introduced by $\multimap$I actually was used and is not part of the residual hypothesis Δ_O. If there is slack, we can simply erase it.

$$\frac{\Gamma; (\Delta_I, w{:}A) \setminus \Delta_O \vdash_i M : B \quad \text{where } i = 1 \text{ or } w \text{ not in } \Delta_O}{\Gamma; \Delta_I \setminus (\Delta_O - w{:}A) \vdash_i \hat{\lambda} w{:}A.\ M : A \multimap B}\, {\multimap}\mathrm{I}^w$$

$$\frac{\Gamma; \Delta_I \setminus \Delta' \vdash_i M : A \multimap B \qquad \Gamma; \Delta' \setminus \Delta_O \vdash_k N : A}{\Gamma; (\Delta_I \setminus \Delta_O) \vdash_{i \vee k} M\,\hat{}\,N : B}\, {\multimap}\mathrm{E}$$

Here $i \vee k = 1$ if $i = 1$ or $k = 1$, and $i \vee k = 0$ otherwise. This means we have slack in the result, if either of the two premisses permits slack.

$$\frac{\Gamma; \Delta_I \setminus \Delta' \vdash_i M : A \qquad \Gamma; \Delta' \setminus \Delta_O \vdash_k N : B}{\Gamma; \Delta_I \setminus \Delta_O \vdash_{i \vee k} M \otimes N : A \otimes B}\, \otimes\mathrm{I}$$

$$\frac{\begin{array}{l}\Gamma; \Delta_I \setminus \Delta' \vdash_i M : A \otimes B \\ \Gamma; (\Delta', w_1{:}A, w_2{:}B) \setminus \Delta_O \vdash_k N : C \\ \text{where } k = 1 \text{ or } w_1 \text{ and } w_2 \text{ not in } \Delta_O\end{array}}{\Gamma; \Delta_I \setminus (\Delta_O - w_1{:}A - w_2{:}B) \vdash_{i \vee k} \mathbf{let}\ w_1 \otimes w_2 = M\ \mathbf{in}\ N : C}\, \otimes\mathrm{E}^{w_1, w_2}$$

In the $\otimes$E rule we stack the premisses on top of each other since they are too long to fit on one line. The unit type permits no slack.

$$\frac{}{\Gamma; \Delta_I \setminus \Delta_I \vdash_0 \star : \mathbf{1}}\, \mathbf{1}\mathrm{I}$$

$$\frac{\Gamma; \Delta_I \setminus \Delta' \vdash_i M : \mathbf{1} \qquad \Gamma; \Delta' \setminus \Delta_O \vdash_k N : C}{\Gamma; \Delta_I \setminus \Delta_O \vdash_{i \vee k} \mathbf{let}\ \star = M\ \mathbf{in}\ N : C}\, \mathbf{1}\mathrm{E}$$

Additive Connectives. The mechanism of passing and consuming resources was designed to eliminate unwanted non-determinism in the multiplicative connectives. This introduces complications in the additives, since we have to force premisses to consume exactly the same resources. We write out four version of the &I rule.

$$\frac{\Gamma; \Delta_I \setminus \Delta'_O \vdash_0 M : A \quad \Gamma; \Delta_I \setminus \Delta''_O \vdash_0 N : B \quad \Delta'_O = \Delta''_O}{\Gamma; \Delta_I \setminus (\Delta'_O \cap \Delta''_O) \vdash_0 \langle M, N\rangle : A \& B} \&\mathrm{I}_{00}$$

$$\frac{\Gamma; \Delta_I \setminus \Delta'_O \vdash_0 M : A \quad \Gamma; \Delta_I \setminus \Delta''_O \vdash_1 N : B \quad \Delta'_O \subseteq \Delta''_O}{\Gamma; \Delta_I \setminus (\Delta'_O \cap \Delta''_O) \vdash_0 \langle M, N\rangle : A \& B} \&\mathrm{I}_{10}$$

$$\frac{\Gamma; \Delta_I \setminus \Delta'_O \vdash_1 M : A \quad \Gamma; \Delta_I \setminus \Delta''_O \vdash_0 N : B \quad \Delta'_O \supseteq \Delta''_O}{\Gamma; \Delta_I \setminus (\Delta'_O \cap \Delta''_O) \vdash_0 \langle M, N\rangle : A \& B} \&\mathrm{I}_{01}$$

$$\frac{\Gamma; \Delta_I \setminus \Delta'_O \vdash_1 M : A \quad \Gamma; \Delta_I \setminus \Delta''_O \vdash_1 N : B}{\Gamma; \Delta_I \setminus (\Delta'_O \cap \Delta''_O) \vdash_1 \langle M, N\rangle : A \& B} \&\mathrm{I}_{11}$$

Note that in $\&\mathrm{I}_{00}$, $\Delta'_O \cap \Delta''_O = \Delta'_O = \Delta''_O$ by the condition in the premiss. Similarly for the other rules. We chose to present the rules in a uniform way despite this redundancy to highlight the similarities. Only if both premisses permit slack do we have slack overall.

$$\frac{\Gamma; \Delta \setminus \Delta_O \vdash_i M : A \& B}{\Gamma; \Delta_I \setminus \Delta_O \vdash_i \mathrm{fst}\, M : A} \&\mathrm{E_L} \qquad \frac{\Gamma; \Delta_I \setminus \Delta_O \vdash_i M : A \& B}{\Gamma; \Delta_I \setminus \Delta_O \vdash_i \mathrm{snd}\, M : B} \&\mathrm{E_R}$$

Finally, we come to the reason for the slack indicator.

$$\frac{}{\Gamma; \Delta_I \setminus \Delta_I \vdash_1 \langle\,\rangle : \top} \top\mathrm{I} \qquad \textit{No } \top \textit{ elimination}$$

The introduction rules for disjunction are direct.

$$\frac{\Gamma; \Delta_I \setminus \Delta_O \vdash_i M : A}{\Gamma; \Delta_I \setminus \Delta_O \vdash_i \mathrm{inl}^B : A \oplus B} \oplus\mathrm{I_L} \qquad \frac{\Gamma; \Delta_I \setminus \Delta_O \vdash_i M : B}{\Gamma; \Delta_I \setminus \Delta_O \vdash_i \mathrm{inr}^A : A \oplus B} \oplus\mathrm{I_R}$$

The elimination rule for disjunction combines resource propagation (as for multiplicatives) introduction of hypothesis, and resource coordination (as for additives) and is therefore somewhat tedious. It is left to Exercise 4.6. The **0**E rule permits slack, no matter whether the derivation of the premiss permits slack.

$$\textit{No } \mathbf{0} \textit{ introduction} \qquad \frac{\Gamma; \Delta_I \setminus \Delta_O \vdash_i M : \mathbf{0}}{\Gamma; \Delta_I \setminus \Delta_O \vdash_1 \mathrm{abort}^C\, M : C} \mathbf{0}\mathrm{E}$$

Exponentials. Here we can enforce the emptiness of the linear context directly.

$$\frac{(\Gamma, u{:}A); \Delta_I \setminus \Delta_O \vdash_i M : B}{\Gamma; \Delta_I \setminus \Delta_O \vdash_i \lambda u{:}A.\ M : A \supset B} \supset\!\mathrm{I}^u$$

$$\frac{\Gamma; \Delta_I \setminus \Delta_O \vdash_i M : A \supset B \qquad \Gamma; \cdot \setminus \Delta^* \vdash_k N : A}{\Gamma; \Delta_I \setminus \Delta_O \vdash_i M\, N : B} \supset\!\mathrm{E}$$

Here Δ^* will always have to be $\cdot$ (since it must be a subset of $\cdot$) and k is irrelevant. The same is true in the next rule.

$$\frac{\Gamma; \cdot \setminus \Delta^* \vdash_i M : A}{\Gamma; \Delta_I \setminus \Delta_I \vdash_0 !M : !A} !\mathrm{I}$$

$$\frac{\Gamma; \Delta_I \setminus \Delta' \vdash_i M : !A \qquad (\Gamma, u{:}A); \Delta' \setminus \Delta_O \vdash_k N : C}{\Gamma; \Delta_I \setminus \Delta_O \vdash_{i \vee j} \mathbf{let}\ !u = M\ \mathbf{in}\ N : C} !\mathrm{E}^u$$

The desired soundness and completeness theorem for the algorithmic typing judgment must first be generalized before it can be proved by induction. For this generalization, the mode (input and output) of the constituents of the judgment is a useful guide. For example, in the completness direction (3), we can expect to distinguish cases based on the slack indicator which might be returned when we ask the question if there are Δ_O and i such that $\Gamma; \Delta \setminus \Delta_O \vdash_i M : A$ for the given Γ, Δ, M and A.

Lemma 4.5 (Properties of Algorithmic Type Checking)

1. *If* $\Gamma; \Delta_I \setminus \Delta_O \vdash_0 M : A$ *then* $\Delta_I \supseteq \Delta_O$ *and* $\Gamma; \Delta_I - \Delta_O \vdash M : A$.
2. *If* $\Gamma; \Delta_I \setminus \Delta_O \vdash_1 M : A$ *then* $\Delta_I \supseteq \Delta_O$ *and for any* Δ *such that* $\Delta_I \supseteq \Delta \supseteq \Delta_I - \Delta_O$ *we have* $\Gamma; \Delta \vdash M : A$.
3. *If* $\Gamma; \Delta \vdash M : A$ *then either*
   (a) $\Gamma; (\Delta' \times \Delta) \setminus \Delta' \vdash_0 M : A$ *for any* Δ', *or*
   (b) $\Gamma; (\Delta' \times \Delta) \setminus (\Delta' \times \Delta_O) \vdash_1 M : A$ *for all* Δ' *and some* $\Delta_O \subseteq \Delta$.

Proof: By inductions on the structure of the given derivations.[1] Items (1) and (2) must be proven simultaneously. □

From this lemma, the soundness and completeness of algorithmic type checking follow directly.

Theorem 4.6 (Algorithmic Type Checking)
$\Gamma; \Delta \vdash M : A$ *if and only if either*

[1] *[check]*

1. $\Gamma; \Delta \setminus \cdot \vdash_0 M : A$, *or*

2. $\Gamma; \Delta \setminus \Delta' \vdash_1 M : A$ *for some* Δ'.

Proof: Directly from Lemma 4.5 $\square$

4.5 Pure Linear Functional Programming

The linear λ-calculus developed in the preceding sections can serve as the basis for a programming language. The step from λ-calculus to programming language can be rather complex, depending on how realistic one wants to make the resulting language. The first step is to decide on *observable types* and a language of *values* and then define an *evaluation judgment*. This is the subject of this section. Given the purely logical view we have taken, this language still lacks datatypes and recursion. In order to remedy this, we introduce *recursive types* and *recursive terms* in the next section.

Our operational semantics follows the intuition that we should not evaluate expressions whose value may not be needed for the result. Expressions whose value will definitely be used, can be evaluated eagerly. There is a slight mismatch in that the linear λ-calculus can identifies expressions whose value will be needed *exactly once*. However, we can derive other potential benefits from the stronger restriction at the lower levels of an implementation such as improved garbage collection or update-in-place. These benefits also have their price, and at this time the trade-offs are not clear. For the *strict λ-calculus* which captures the idea of definite use of the value of an expression, see Exercise 4.2.

We organize the functional language strictly along the types, discussing observability, values, and evaluation rules for each. We have two main judgments, M *Value* (M is a value), and $M \hookrightarrow v$ (M evaluates to v). In general we use v for terms which are legal values. For both of these we assume that M is *closed* and *well-typed*, that is, $\cdot; \cdot \vdash M : A$.

Linear Implication. An important difference between a general λ-calculus and a functional language is that the structure of functions in a programming language is *not* observable. Instead, functions are compiled to code. Their behavior can be observed by applying functions to arguments, but their definition cannot be seen. Thus, strictly speaking, it is incorrect to say that functions are first-class. This holds equally for so-called lazy functional languages such as Haskell [?] and eager functional languages such as ML [?]. Thus, any expression of the form $\hat{\lambda} w{:}A.\ M$ is a possible value.

$$\frac{}{\hat{\lambda} w{:}A.\ M\ \mathit{Value}} \multimap \mathrm{val}$$

Evaluation of a $\hat{\lambda}$-abstraction returns itself immediately.

$$\frac{}{\hat{\lambda} w{:}A.\ M \hookrightarrow \hat{\lambda} w{:}A.\ M} \multimap \mathrm{Iv}$$

Since a linear parameter to a function is definitely used (in fact, used exactly once), we can evaluate the argument without doing unnecessary work and substitute it for the bound variable during the evaluation of an application.

$$\frac{M_1 \hookrightarrow \hat{\lambda} w{:}A_2.\ M_1' \qquad M_2 \hookrightarrow v_2 \qquad [v_2/w]M_1' \hookrightarrow v}{M_1 \hat{\ } M_2 \hookrightarrow v}\ {\multimap}\mathrm{Ev}$$

Note that after we substitute the value of argument v_2 for the formal parameter w in the function, we have to evaluate the body of the function.

Simultaneous Pairs. The multiplicative conjunction $A \otimes B$ corresponds to the type of pairs where both elements must be used exactly once. Thus we can evaluate the components (they will be used!) and the pairs are observable. The elimination form is evaluated by creating the pair and then deconstructing it.

$$\frac{M_1\ \mathit{Value} \qquad M_2\ \mathit{Value}}{M_1 \otimes M_2\ \mathit{Value}}\ {\otimes}\mathrm{val}$$

$$\frac{M_1 \hookrightarrow v_1 \qquad M_2 \hookrightarrow v_2}{M_1 \otimes M_2 \hookrightarrow v_1 \otimes v_2}\ {\otimes}\mathrm{Iv} \qquad \frac{M \hookrightarrow v_1 \otimes v_2 \qquad [v_1/w_1, v_2/w_2]N \hookrightarrow v}{\mathbf{let}\ w_1 \otimes w_2 = M\ \mathbf{in}\ N \hookrightarrow v}\ {\otimes}\mathrm{Ev}$$

Multiplicative Unit. The multiplicative unit $\mathbf{1}$ is observable and contains exactly one value $\star$. Its elimination rule explicitly evaluates a term and ignores its result (which must be $\star$).

$$\frac{}{\star\ \mathit{Value}}\ \mathbf{1}\mathrm{val}$$

$$\frac{}{\star \hookrightarrow \star}\ \mathbf{1}\mathrm{Iv} \qquad \frac{M \hookrightarrow \star \qquad N \hookrightarrow v}{\mathbf{let}\ \star = M\ \mathbf{in}\ N \hookrightarrow v}\ \mathbf{1}\mathrm{Ev}$$

Alternative Pairs. Alternative pairs of type $A \& B$ are such that we can only use one of the two components. Since we may not be able to predict which one, we should not evaluate the components. Thus pairs $\langle M_1, M_2 \rangle$ are *lazy*, not observable and any pair of this form is a value. When we extract a component, we then have to evaluate the corresponding term to obtain a value.

$$\frac{}{\langle M_1, M_2 \rangle\ \mathit{Value}}\ \&\mathrm{val}$$

$$\frac{}{\langle M_1, M_2 \rangle \hookrightarrow \langle M_1, M_2 \rangle}\ \&\mathrm{Iv}$$

$$\frac{M \hookrightarrow \langle M_1, M_2 \rangle \qquad M_1 \hookrightarrow v_1}{\mathrm{fst}\, M \hookrightarrow v_1}\ \&\mathrm{Ev}_1 \qquad \frac{M \hookrightarrow \langle M_1, M_2 \rangle \qquad M_2 \hookrightarrow v_2}{\mathrm{snd}\, M \hookrightarrow v_2}\ \&\mathrm{Ev}_2$$

Additive Unit. By analogy, the additive unit $\top$ is not observable. Since there is no elimination rule, we can never do anything interesting with a value of this type, except embed it in larger values.

$$\frac{}{\langle\,\rangle\ \mathit{Value}}\ \top\mathrm{val}$$

$$\frac{}{\langle\,\rangle \hookrightarrow \langle\,\rangle}\ \top\mathrm{Iv}$$

This rule does not express the full operational intuition behind $\top$ which "garbage collects" all linear resources. However, we can only fully appreciate this when we define evaluation under environments (see Section **??**).

Disjoint Sum. The values of a disjoint sum type are guaranteed to be used (no matter whether it is of the form $\mathrm{inl}^B\, M$ or $\mathrm{inr}^A\, M$). Thus we can require values to be built up from injections of values, and the structure of sum values is observable. There are two rules for evaluation, depending on whether the subject of a **case**-expression is a left injection or right injection into the sum type.

$$\frac{M\ \mathit{Value}}{\mathrm{inl}^B\, M\ \mathit{Value}}\ \oplus\mathrm{val}_1 \qquad \frac{M\ \mathit{Value}}{\mathrm{inr}^A\, M\ \mathit{Value}}\ \oplus\mathrm{val}_2$$

$$\frac{M \hookrightarrow v}{\mathrm{inl}^B\, M \hookrightarrow \mathrm{inl}^B\, v}\ \oplus\mathrm{Iv}_1 \qquad \frac{M \hookrightarrow v}{\mathrm{inr}^A\, M \hookrightarrow \mathrm{inr}^A\, v}\ \oplus\mathrm{Iv}_2$$

$$\frac{M \hookrightarrow \mathrm{inl}^B\, v_1 \qquad [v_1/w_1]N_1 \hookrightarrow v}{\mathbf{case}\ M\ \mathbf{of}\ \mathrm{inl}\, w_1 \Rightarrow N_1 \mid \mathrm{inr}\, w_2 \Rightarrow N_2 \hookrightarrow v}\ \oplus\mathrm{Ev}_1$$

$$\frac{M \hookrightarrow \mathrm{inr}^A\, v_2 \qquad [v_2/w_2]N_2 \hookrightarrow v}{\mathbf{case}\ M\ \mathbf{of}\ \mathrm{inl}\, w_1 \Rightarrow N_1 \mid \mathrm{inr}\, w_2 \Rightarrow N_2 \hookrightarrow v}\ \oplus\mathrm{Ev}_2$$

Void Type. The void type $\mathbf{0}$ contains no value. In analogy with the disjoint sum type it is observable, although this is not helpful in practice. There are no evaluation rules for this type: since there are no introduction rules there are no constructor rules, and the elimination rule distinguishes between zero possible cases (in other words, is impossible). We called this $\mathrm{abort}^A\, M$, since it may be viewed as a global program abort.

Unrestricted Function Type. The unrestricted function type $A \supset B$ (also written as $A \to B$ in accordance with the usual practice in functional programming) may or may not use its argument. Therefore, the argument is not evaluated, but simply substituted for the bound variable. This is referred to as a *call-by-name* semantics. It is usually implemented by *lazy evaluation*, which means that first time the argument is evaluated, this value is memoized to avoid re-evaluation. This is not represented at this level of semantic description. Values of functional type are not observable, as in the linear case.

$$\frac{}{\lambda u{:}A.\ M\ \mathit{Value}} \multimap \text{val}$$

$$\frac{}{\lambda u{:}A.\ M \hookrightarrow \lambda u{:}A.\ M} \rightarrow \text{Iv}$$

$$\frac{M_1 \hookrightarrow \lambda u{:}A_2.\ M_1' \qquad [M_2/u]M_1' \hookrightarrow v}{M_1\, M_2 \hookrightarrow v} \rightarrow \text{Ev}$$

Modal Type. A linear variable of type $!A$ must be used, but the embedded expression of type A may *not* be used since it is unrestricted. Therefore, terms $!M$ are values and "!" is like a quotation of its argument M, protecting it from evaluation.

$$\frac{}{!M\ \mathit{Value}} \text{!val}$$

$$\frac{}{!M \hookrightarrow !M} \text{!Iv} \qquad \frac{M \hookrightarrow !M' \qquad [M'/u]N \hookrightarrow v}{\mathbf{let}\ !u = M\ \mathbf{in}\ N \hookrightarrow v} \text{!Ev}$$

We abbreviate the value judgment from above in the form of a grammar.

Values	v	$::=$	$\hat{\lambda} w{:}A.\ M$	$A \multimap B$	not observable
		$\mid$	$v_1 \otimes v_2$	$A_1 \otimes A_2$	observable
		$\mid$	$\star$	$\mathbf{1}$	observable
		$\mid$	$\langle M_1, M_2 \rangle$	$A_1 \& A_2$	not observable
		$\mid$	$\langle\,\rangle$	$\top$	not observable
		$\mid$	$\mathrm{inl}^B\, v \mid \mathrm{inr}^A\, v$	$A \oplus B$	observable
			No values	$\mathbf{0}$	observable
		$\mid$	$\lambda u{:}A.\ M$	$A \rightarrow B$	not observable
		$\mid$	$!M$	$!A$	not observable

In the absence of datatypes, we cannot write many interesting programs. As a first example we consider the representation of the Booleans with two values, true and false, and a conditional as an elimination construct.

$$\begin{array}{rcl}
\text{bool} & = & \mathbf{1} \oplus \mathbf{1} \\
\text{true} & = & \mathrm{inl}^{\mathbf{1}} \star \\
\text{false} & = & \mathrm{inr}^{\mathbf{1}} \star \\
\mathbf{if}\ M\ \mathbf{then}\ N_1\ \mathbf{else}\ N_2 & = & \mathbf{case}\ M \\
& & \quad \mathbf{of}\ \mathrm{inl}^{\mathbf{1}}\, w_1 \Rightarrow \mathbf{let}\ \star = w_1\ \mathbf{in}\ N_1 \\
& & \quad\ \ \mid \mathrm{inr}^{\mathbf{1}}\, w_2 \Rightarrow \mathbf{let}\ \star = w_2\ \mathbf{in}\ N_2
\end{array}$$

The elimination of $\star$ in the definition of the conditional is necessary, because a branch $\mathrm{inl}^{\mathbf{1}}\, w_1 \Rightarrow N_1$ would not be well-typed: w_1 is a linear variable not used in its scope. Destructuring a value in several stages is a common idiom and it is helpful for the examples to introduce some syntactic sugar. We allow patterns which nest the elimination forms which appear in a **let** or **case**. Not all combination of these are legal, but it is not difficult to describe the legal pattern and match expressions (see Exercise 4.7).

$$\begin{array}{llcl} \textit{Patterns} & p & ::= & w \mid p_1 \otimes p_2 \mid \star \mid \mathrm{inl}\, p \mid \mathrm{inr}\, p \mid u \mid !p \\ \textit{Matches} & m & ::= & p \Rightarrow M \mid (m_1 \mid m_2) \mid \cdot \end{array}$$

An *extended case expression* has the form **case** M **of** m.

In the example of Booleans above, we gave a uniform definition for conditionals in terms of **case**. But can we define a function cond with arguments M, N_1 and N_2 which behaves like $\mathbf{if}\, M\, \mathbf{then}\, N_1\, \mathbf{else}\, N_2$? The first difficulty is that the type of branches is generic. In order to avoid the complications of polymorphism, we uniformly define a whole family of functions cond_C types C. We go through some candidate types for cond_C and discuss why they may or may not be possible.

$\mathrm{cond}_C : \mathbf{1} \oplus \mathbf{1} \multimap C \multimap C \multimap C$**.** This type means that both branches of the conditional (second and third argument) would be evaluated before being substituted in the definition of cond_C. Moreover, both must be used during the evaluation of the body, while intuitively only one branch should be used.

$\mathrm{cond}_C : \mathbf{1} \oplus \mathbf{1} \multimap (!C) \multimap (!C) \multimap C$**.** This avoids evaluation of the branches, since they now can have the form $!N_1$ and $!N_2$, which are values. However, N_1 and N_2 can now no longer use linear variables.

$\mathrm{cond}_C : \mathbf{1} \oplus \mathbf{1} \multimap C \rightarrow C \rightarrow C$**.** This is equivalent to the previous type and undesirable for the same reason.

$\mathrm{cond}_C : \mathbf{1} \oplus \mathbf{1} \multimap (C \& C) \multimap C$**.** This type expresses that the second argument of type $C \& C$ is a pair $\langle N_1, N_2 \rangle$ such that exactly one component of this pair will be used. This expresses precisely the expected behavior and we define

$$\begin{array}{lcl} \mathrm{cond}_C & : & \mathbf{1} \oplus \mathbf{1} \multimap (C \& C) \multimap C \\ & = & \hat{\lambda} b{:}\mathbf{1} \oplus \mathbf{1}.\ \hat{\lambda} n{:}C \& C. \\ & & \quad \mathbf{case}\ b \\ & & \qquad \mathbf{of}\ \mathrm{inl}\, \star \Rightarrow \mathrm{fst}\, n \\ & & \qquad\quad \mid \mathrm{inr}\, \star \Rightarrow \mathrm{snd}\, n \end{array}$$

which is linearly well-typed: b is used as the subject of the **case** and n is used in both branches of the **case** expression (which is additive).

As a first property of evaluation, we show that it is a strategy for β-reductions. That is, if $M \hookrightarrow v$ then M reduces to v in some number of β-reduction steps (possibly none), but not *vice versa*. For this we need a new judgment $M \longrightarrow_\beta^* M'$ is the congruent, reflexive, and transitive closure of the $M \longrightarrow_\beta M'$

relation. In other words, we extend β-reduction so it can be applied to an arbitrary subterm of M and then allow arbitrary sequences of reductions. The subject reduction property holds for this judgment as well.

Theorem 4.7 (Generalized Subject Reduction) *If* $\Gamma;\Delta \vdash M : A$ *and* $M \longrightarrow_\beta^* M'$ *then* $\Gamma;\Delta \vdash M' : A$.

Proof: See Exercise 4.8 $\square$

Evaluation is related to β-reduction in that an expression reduces to its value.

Theorem 4.8 *If* $M \hookrightarrow v$ *then* $M \longrightarrow_\beta^* v$.

Proof: By induction on the structure of the derivation of $M \hookrightarrow v$. In each case we directly combine results obtained by appealing to the induction hypothesis using transitivity and congruence. $\square$

The opposite is clearly false. For example,

$$\langle(\hat{\lambda}w{:}\mathbf{1}.\ w)\hat{\ }\star, \star\rangle \longrightarrow_\beta^* \langle\star, \star\rangle,$$

but

$$\langle(\hat{\lambda}w{:}\mathbf{1}.\ w)\hat{\ }\star, \star\rangle \hookrightarrow \langle(\hat{\lambda}w{:}\mathbf{1}.\ w)\hat{\ }\star, \star\rangle$$

and this is the only evaluation for the pair. However, if we limit the congruence rules to the components of $\otimes$, inl, inr, and all elimination constructs, the correspondence is exact (see Exercise 4.9). Type preservation is a simple consequence of the previous two theorems. See Exercise 4.10 for a direct proof.

Theorem 4.9 (Type Preservation) *If* $\cdot;\cdot \vdash M : A$ *and* $M \hookrightarrow v$ *then* $\cdot;\cdot \vdash v : A$.

Proof: By Theorem 4.8, $M \longrightarrow_\beta^* v$. Then the result follows by generalized subject reduction (Theorem 4.7). $\square$

The final theorem of this section establishes the uniqueness of values.

Theorem 4.10 (Determinacy) *If* $M \hookrightarrow v$ *and* $M \hookrightarrow v'$ *then* $v = v'$.

Proof: By straightforward simultaneous induction on the structure of the two given derivations. For each for of M except case expressions there is exactly one inference rule which could be applied. For **case** we use the uniqueness of the value of the case subject to determine that the same rule must have been used in both derivations. $\square$

We can also prove that evaluation of any closed, well-typed term M terminates in this fragment. We postpone the proof of this (Theorem 4.13) until we have seen further, more realistic, examples.

4.6 Recursive Types

The language so far lacks basic data types, such as natural numbers, integers, lists, trees, etc. Moreover, except for finitary ones such as booleans, they are not definable with the mechanism at our disposal so far. At this point we can follow two paths: one is to define each new data type in the same way we defined the logical connectives, that is, by introduction and elimination rules, carefully checking their local soundness and completeness. The other is to enrich the language with a general mechanism for defining such new types. Again, this can be done in different ways, using either *inductive types* which allow us to maintain a clean connection between propositions and types, or *recursive types* which are more general, but break the correspondence to logic. Since we are mostly interested in programming here, we chose the latter path.

Recall that we defined the booleans as $\mathbf{1} \oplus \mathbf{1}$. It is easy to show by the definition of values, that there are exactly two values of this type, to which we can arbitrarily assign true and false. A finite type with n values can be defined as the disjoint sum of n observable singleton types, $\mathbf{1} \oplus \cdots \oplus \mathbf{1}$. The natural numbers would be $\mathbf{1} \oplus \mathbf{1} \oplus \cdots$, except that this type is infinite. We can express it finitely as a recursive type $\mu\alpha.\ \mathbf{1} \oplus \alpha$. Intuitively, the meaning of this type should be invariant under unrolling of the recursion. That is,

$$\begin{array}{rcl}
\mathrm{nat} & = & \mu\alpha.\ \mathbf{1} \oplus \alpha \\
& \cong & [(\mu\alpha.\ \mathbf{1} \oplus \alpha)/\alpha]\mathbf{1} \oplus \alpha \\
& = & \mathbf{1} \oplus \mu\alpha.\ \mathbf{1} \oplus \alpha \\
& = & \mathbf{1} \oplus \mathrm{nat}
\end{array}$$

which is the expected recursive definition for the type of natural numbers.

In functional languages such as ML or Haskell, recursive type definitions are not directly available, but the results of elaborating syntactically more pleaseant definitions. In addition, recursive type definitions are *generative*, that is, they generate new constructors and types every time they are invoked. This is of great practical value, but the underlying type theory can be seen as simple recursive types combined with a mechanism for generativity. Here, we will only treat the issue of recursive types.

Even though recursive types do not admit a logical interpretation as propositions, we can still define a term calculus using introduction and elimination rules, including local reduction and expansions. In order maintain the property that a term has a unique type, we annotate the introduction constant fold with the recursive type itself.

$$\frac{\Gamma; \Delta \vdash M : [\mu\alpha.\ A/\alpha]A}{\Gamma; \Delta \vdash \mathrm{fold}^{\mu\alpha.\ A} M : \mu\alpha.\ A}\,\mu\mathrm{I} \qquad \frac{\Gamma; \Delta \vdash M : \mu\alpha.\ A}{\Gamma; \Delta \vdash \mathrm{unfold}\, M : [\mu\alpha.\ A/\alpha]A}\,\mu\mathrm{E}$$

The local reduction and expansions, expressed on the terms.

$$\begin{array}{rcl}
\mathrm{unfold}\, \mathrm{fold}^{\mu\alpha.\ A}\, M & \longrightarrow_\beta & M \\
M : \mu\alpha.\ A & \longrightarrow_\eta & \mathrm{fold}^{\mu\alpha.\ A}\, (\mathrm{unfold}\, M)
\end{array}$$

It is easy to see that uniquess of types and subject reduction remain valid properties (see Exercise 4.11). There are also formulation of recursive types where the term M in the premiss and conclusion is the same, that is, there are no explicit constructor and destructors for recursive types. This leads to more concise programs, but significantly more complicated type-checking (see Exercise 4.12).

We would like recursive types to represent data types. Therefore the values of recursive type must be of the form $\mathrm{fold}^{\mu\alpha.\,A}\,v$ for values v—otherwise data values would not be observable.

$$\frac{M\ \mathit{Value}}{\mathrm{fold}^{\mu\alpha.\,A}\,M\ \mathit{Value}}\,\mu\mathrm{val}$$

$$\frac{M \hookrightarrow v}{\mathrm{fold}^{\mu\alpha.\,A}\,M \hookrightarrow \mathrm{fold}^{\mu\alpha.\,A}\,v}\,\mu\mathrm{Iv} \qquad \frac{M \hookrightarrow \mathrm{fold}^{\mu\alpha.\,A}\,v}{\mathrm{unfold}\,M \hookrightarrow v}\,\mu\mathrm{Ev}$$

In order to write interesting programs simply, it is useful to have a general recursion operator $\mathbf{fix}\,u{:}A.\ M$ at the level of terms. It is not associated with an type constructor and simply unrolls its definition once when executed. In the typing rule we have to be careful: since the number on unrollings generally unpredictable, no linear variables are permitted to occur free in the body of a recursive definition. Moreover, the recursive function itself may be called arbitrarily many times—one of the characteristics of recursion. Therefore its uses are unrestricted.

$$\frac{(\Gamma, u{:}A); \cdot \vdash M : A}{\Gamma; \cdot \vdash \mathbf{fix}\,u{:}A.\ M : A}\,\mathbf{fix}$$

The operator does not introduce any new values, and one new evaluation rules which unrolls the recursion.

$$\frac{[\mathbf{fix}\,u{:}A.\ M/u]M \hookrightarrow v}{\mathbf{fix}\,u{:}A.\ M \hookrightarrow v}\,\mathbf{fix}\mathrm{v}$$

In order to guarantee subject reduction, the type of whole expression, the body M of the fixpoint expression, and the bound variable u must all have the same type A. This is enforced in the typing rules.

We now consider a few examples of recursive types and some example programs.

Natural Numbers.

$$\begin{array}{rcl}
\mathrm{nat} & = & \mu\alpha.\ \mathbf{1} \oplus \alpha \\
\mathrm{zero} & : & \mathrm{nat} \\
 & = & \mathrm{fold}^{\mathrm{nat}}\,(\mathrm{inl}^{\mathrm{nat}}\,\star) \\
\mathrm{succ} & : & \mathrm{nat} \multimap \mathrm{nat} \\
 & = & \hat{\lambda} x{:}\mathrm{nat}.\ \mathrm{fold}^{\mathrm{nat}}\,(\mathrm{inr}^{\mathbf{1}}\,x)
\end{array}$$

With this definition, the addition function for natural numbers is linear in both argument.

$$
\begin{array}{lcl}
\mathrm{plus} & : & \mathrm{nat} \multimap \mathrm{nat} \multimap \mathrm{nat} \\
 & = & \mathbf{fix}\, p{:}\mathrm{nat} \multimap \mathrm{nat} \multimap \mathrm{nat}. \\
 & & \quad \hat{\lambda} x{:}\mathrm{nat}.\ \hat{\lambda} y{:}\mathrm{nat}.\ \mathbf{case}\ \mathrm{unfold}\, x \\
 & & \qquad\qquad\qquad\qquad\quad \mathbf{of}\ \mathrm{inl}\, \star \Rightarrow y \\
 & & \qquad\qquad\qquad\qquad\quad\ \mid \mathrm{inr}\, x' \Rightarrow \mathrm{succ}\,\hat{}\,(p\,\hat{}\,x'\,\hat{}\,y)
\end{array}
$$

It is easy to ascertain that this definition is well-typed: x occurs as the case subject, y in both branches, and x' in the recursive call to p. On the other hand, the natural definition for multiplication is *not* linear, since the second argument is used twice in one branch of the case statement and not at all in the other.

$$
\begin{array}{lcl}
\mathrm{mult} & : & \mathrm{nat} \multimap \mathrm{nat} \to \mathrm{nat} \\
 & = & \mathbf{fix}\, m{:}\mathrm{nat} \multimap \mathrm{nat} \to \mathrm{nat} \\
 & & \quad \hat{\lambda} x{:}\mathrm{nat}.\ \lambda y{:}\mathrm{nat}.\ \mathbf{case}\ \mathrm{unfold}\, x \\
 & & \qquad\qquad\qquad\qquad\quad \mathbf{of}\ \mathrm{inl}\, \star \Rightarrow \mathrm{zero} \\
 & & \qquad\qquad\qquad\qquad\quad\ \mid \mathrm{inr}\, x' \Rightarrow \mathrm{plus}\,\hat{}\,(m\,\hat{}\,x'\,y)\,\hat{}\,y
\end{array}
$$

Interestingly, there is also a linear definition of mult (see Exercise 4.13), but its operational behavior is quite different. This is because we can explicitly copy and delete natural numbers, and even make them available in an unrestricted way.

$$
\begin{array}{lcl}
\mathrm{copy} & : & \mathrm{nat} \multimap \mathrm{nat} \otimes \mathrm{nat} \\
 & = & \mathbf{fix}\, c{:}\mathrm{nat} \multimap \mathrm{nat} \otimes \mathrm{nat} \\
 & & \quad \hat{\lambda} x{:}\mathrm{nat}.\ \mathbf{case}\ \mathrm{unfold}\, x \\
 & & \qquad\qquad\quad \mathbf{of}\ \mathrm{inl}\, \star \Rightarrow \mathrm{zero} \otimes \mathrm{zero} \\
 & & \qquad\qquad\quad\ \mid \mathrm{inr}\, x' \Rightarrow \mathbf{let}\ x'_1 \otimes x'_2 = c\,\hat{}\,x'\ \mathbf{in}\ (\mathrm{succ}\,\hat{}\,x'_1) \otimes (\mathrm{succ}\,\hat{}\,x'_2) \\
\mathrm{delete} & : & \mathrm{nat} \multimap \mathbf{1} \\
 & = & \mathbf{fix}\, d{:}\mathrm{nat} \multimap \mathbf{1} \\
 & & \quad \hat{\lambda} x{:}\mathrm{nat}.\ \mathbf{case}\ \mathrm{unfold}\, x \\
 & & \qquad\qquad\quad \mathbf{of}\ \mathrm{inl}\, \star \Rightarrow \mathbf{1} \\
 & & \qquad\qquad\quad\ \mid \mathrm{inr}\, x' \Rightarrow \mathbf{let}\ \star = d\,\hat{}\,x'\ \mathbf{in}\ \mathbf{1} \\
\mathrm{promote} & : & \mathrm{nat} \multimap {!\mathrm{nat}} \\
 & = & \mathbf{fix}\, p{:}\mathrm{nat} \multimap {!\mathrm{nat}} \\
 & & \quad \hat{\lambda} x{:}\mathrm{nat}.\ \mathbf{case}\ \mathrm{unfold}\, x \\
 & & \qquad\qquad\quad \mathbf{of}\ \mathrm{inl}\, \star \Rightarrow {!\mathrm{zero}} \\
 & & \qquad\qquad\quad\ \mid \mathrm{inr}\, x' \Rightarrow \mathbf{let}\ !u' = p\,\hat{}\,x'\ \mathbf{in}\ !(\mathrm{succ}\, u')
\end{array}
$$

Lazy Natural Numbers. Lazy natural numbers are a simple example of *lazy data types* which contain unevaluated expressions. Lazy data types are useful in applications with potentially infinite data such as streams. We encode such lazy data types by using the $!A$ type constructor.

$$
\begin{array}{rcl}
\mathrm{lnat} & = & \mu\alpha.\,!(\mathbf{1} \oplus \alpha) \\
\mathrm{lzero} & : & \mathrm{lnat} \\
 & = & \mathrm{fold}^{\mathrm{lnat}}\,!\,(\mathrm{inl}^{\mathrm{lnat}}\,\star) \\
\mathrm{lsucc} & : & \mathrm{lnat} \to \mathrm{lnat} \\
 & = & \lambda u{:}\mathrm{lnat}.\ \mathrm{fold}^{\mathrm{lnat}}\,!\,(\mathrm{inr}^{\mathbf{1}}\,u)
\end{array}
$$

There is also a linear version of successor of type, $\mathrm{lnat} \multimap \mathrm{lnat}$, but it is not as natural since it evaluates its argument just to build another lazy natural number.

$$
\begin{array}{rcl}
\mathrm{lsucc}' & : & \mathrm{lnat} \multimap \mathrm{lnat} \\
 & = & \hat{\lambda} x{:}\mathrm{lnat}.\ \mathbf{let}\ !u = \mathrm{unfold}\,x\ \mathbf{in}\ \mathrm{fold}^{\mathrm{lnat}}\,(!\,\mathrm{inr}^{\mathbf{1}}\,(\mathrm{fold}^{\mathrm{lnat}}\,(!u)))
\end{array}
$$

The "infinite" number number ω can be defined by using the fixpoint operator. We can either use lsucc as defined above, or define it directly.

$$
\begin{array}{rcl}
\omega & : & \mathrm{lnat} \\
 & = & \mathbf{fix}\,u{:}\mathrm{lnat}.\ \mathrm{lsucc}\,u \\
 & \cong & \mathbf{fix}\,u{:}\mathrm{lnat}.\ \mathrm{fold}^{\mathrm{lnat}}\,!\,(\mathrm{inr}^{\mathbf{1}}\,u)
\end{array}
$$

Note that lazy natural numbers are not directly observable (except for the $\mathrm{fold}^{\mathrm{lnat}}$), so we have to decompose and examine the structure of a lazy natural number successor by successor, or we can convert it to an observable natural number (which might not terminate).

$$
\begin{array}{rcl}
\mathrm{toNat} & : & \mathrm{lnat} \multimap \mathrm{nat} \\
 & = & \mathbf{fix}\,t{:}\mathrm{lnat} \multimap \mathrm{nat} \\
 & & \quad \hat{\lambda} x{:}\mathrm{lnat}.\ \mathbf{case}\ \mathrm{unfold}\,x \\
 & & \qquad\qquad \mathbf{of}\ !\mathrm{inl}^{\mathrm{lnat}}\,\star \Rightarrow \mathrm{zero} \\
 & & \qquad\qquad \mid\ !\mathrm{inr}^{\mathbf{1}}\,x' \Rightarrow \mathrm{succ}\,\hat{}\,(t\,\hat{}\,x')
\end{array}
$$

Lists. To avoid issues of polymorphism, we define a family of data types list_A for an arbitrary type A.

$$
\begin{array}{rcl}
\mathrm{list}_A & = & \mu\alpha.\,\mathbf{1} \oplus (A \otimes \alpha) \\
\mathrm{nil}_A & : & \mathrm{list}_A \\
 & = & \mathrm{fold}^{\mathrm{list}_A}\,(\mathrm{inl}^{\mathrm{list}_A}\,\star) \\
\mathrm{cons}_A & : & A \otimes \mathrm{list}_A \multimap \mathrm{list}_A \\
 & = & \hat{\lambda} p{:}A \otimes \mathrm{list}_A.\ \mathrm{fold}^{\mathrm{list}_A}\,(\mathrm{inr}^{\mathbf{1}}\,p)
\end{array}
$$

We can easily program simple functions such as append and reverse which are linear in their arguments. We show here reverse; for other examples see Exercise 4.14. we define an auxiliary tail-recursive function rev which moves element from it first argument to its second.

$$
\begin{array}{rcl}
\mathrm{rev}_A & : & \mathrm{list}_A \multimap \mathrm{list}_A \multimap \mathrm{list}_A \\
& = & \mathbf{fix}\, r{:}\mathrm{list}_A \multimap \mathrm{list}_A \multimap \mathrm{list}_A \\
& & \quad \hat{\lambda} l{:}\mathrm{list}_A.\ \hat{\lambda} k{:}\mathrm{list}_A. \\
& & \qquad \mathbf{case}\ \mathrm{unfold}\, l \\
& & \qquad\quad \mathbf{of}\ \mathrm{inl}^{A \otimes \mathrm{list}_A} \star \Rightarrow k \\
& & \qquad\quad\ \mid \mathrm{inr}^{\mathbf{1}}\, (x \otimes l') \Rightarrow r \,\hat{}\, l' \,\hat{}\, (\mathrm{cons}_A\, (x \otimes k)) \\
\mathrm{reverse}_A & : & \mathrm{list}_A \multimap \mathrm{list}_A \\
& = & \hat{\lambda} l{:}\mathrm{list}_A.\ \mathrm{rev} \,\hat{}\, l \,\hat{}\, \mathrm{nil}_A
\end{array}
$$

To make definitions like this a bit easier, we can also define a case for lists, in analogy with the conditional for booleans. It is a family indexed by the type of list elements A and the type of the result of the conditional C.

$$
\begin{array}{rcl}
\mathrm{listCase}_{A,C} & : & \mathrm{list}_A \multimap (C \& (A \otimes \mathrm{list}_A \multimap C)) \multimap C \\
& = & \hat{\lambda} l{:}\mathrm{list}_{\mathrm{A}}.\ \hat{\lambda} n{:}C \& (A \otimes \mathrm{list}_A \multimap C). \\
& & \qquad \mathbf{case}\ \mathrm{unfold} l \\
& & \qquad\quad \mathbf{of}\ \mathrm{inl}^{A \otimes \mathrm{list}_A} \star \Rightarrow \mathrm{fst}\, n \\
& & \qquad\quad\ \mid \mathrm{inr}^{\mathbf{1}}\, p \Rightarrow (\mathrm{snd}\, n) \,\hat{}\, p
\end{array}
$$

Lazy Lists. There are various forms of lazy lists, depending of which evaluation is postponed.

$\mathrm{llist}^1_A = \mu\alpha.\ !(\mathbf{1} \oplus (A \otimes \alpha))$**.** This is perhaps the canonical lazy lists, in which we can observe neither head nor tail.

$\mathrm{llist}^2_A = \mu\alpha.\ \mathbf{1} \oplus !(A \otimes \alpha)$**.** Here we can observe directly if the list is empty or not, but not the head or tail which remains unevaluated.

$\mathrm{llist}^3_A = \mu\alpha.\ \mathbf{1} \oplus (A \otimes !\alpha)$**.** Here we can observe directly if the list is empty or not, and the head of the list is non-empty. However, we cannot see the tail.

$\mathrm{llist}^4_A = \mu\alpha.\ \mathbf{1} \oplus (!A \otimes \alpha)$**.** Here the list is always eager, but the elements are lazy. This is the same as $\mathrm{list}_{!A}$.

$\mathrm{llist}^5_A = \mu\alpha.\ \mathbf{1} \oplus (A \& \alpha)$**.** Here we can see if the list is empty or not, but we can access only either the head or tail of list, but not both.

$\mathrm{infStream}_A = \mu\alpha.\ !(A \otimes \alpha)$**.** This is the type of infinite streams, that is, lazy lists with no nil constructor.

Functions such as append, map, etc. can also be written for lazy lists (see Exercise 4.15).

Other types, such as trees of various kinds, are also easily represented using similar ideas. However, the recursive types (even without the presence of the fixpoint operator on terms) introduce terms which have no normal form. In the pure, untyped λ-calculus, the classical examples of a term with no normal form is $(\lambda x.\ x\,x)\,(\lambda x.\ x\,x)$ which β-reduces to itself in one step. In the our typed λ-calculus (linear or intuitionistic) this cannot be assigned a type, because x is used as an argument to itself. However, with recursive types (and the fold and unfold constructors) we can give a type to a version of this term which β-reduces to itself in two steps.

$$\begin{array}{rcl} \Omega & = & \mu\alpha.\ \alpha \to \alpha \\ \omega & : & \Omega \to \Omega \\ & = & \lambda x{:}\Omega.\ (\mathrm{unfold}\, x)\, x \end{array}$$

Then

$$\begin{array}{l} \omega\,(\mathrm{fold}^{\Omega}\,\omega) \\ \quad \longrightarrow_{\beta} (\mathrm{unfold}\,(\mathrm{fold}^{\Omega}\,\omega))\,(\mathrm{fold}^{\Omega}\,\omega) \\ \quad \longrightarrow_{\beta} \omega\,(\mathrm{fold}^{\Omega}\,\omega). \end{array}$$

At teach step we applied the only possible β-reduction and therefore the term can have no normal form. An attempt to evaluate this term will also fail, resulting in an infinite regression (see Exercise 4.16).

4.7 Termination

As the example at the end of the previous section shows, unrestricted recursive types destroy the normalization property. This also means it is impossible to give all recursive types a logical interpretation. When we examine the inference rules we notice that recursive types are *impredicative*: the binder $\mu\alpha$ in $\mu\alpha.\ A$ ranges over the whole type. This means in the introduction rule, the type in the premiss $[\mu\alpha.\ A/\alpha]A$ generally will be larger than the type $\mu\alpha.\ A$ in the conclusion. That alone is not responsible for non-termination: there are other type disciplines such as the polymorphic λ-calculus which retain a logical interpretation and termination, yet are impredicative.

In this section we focus on the property that all well-typed terms in the linear λ-calculus *without* recursive types and fixpoint operators evaluate to a value. This is related to the normalization theorem for natural deductions (Theorem 2.19): if $\Gamma; \Delta \vdash A$ then $\Gamma; \Delta \vdash A\ \ \Uparrow$. We proved this by a rather circuitous route: unrestricted natural deductions can be translated to sequent derivations with cut from which we can eliminate cut and translate the result cut-free derivation back to a noraml natural deduction.

Here, we prove directly that every term evaluates using the proof technique of *logical relations* also called *Tait's method.* Because of the importance of this technique, we spend some time motivating its form. Our ultimate goal is to prove:

If $\cdot;\cdot \vdash M : A$ *then* $M \hookrightarrow v$ *for some value* v.

The first natural attempt would be to prove this by induction on the typing derivation. Surprisingly, case for $\multimap$I works, even though we cannot apply the inductive hypothesis, since every linear λ-abstraction immediately evaluates to itself.

In the case for $\multimap$E, however, we find that we cannot complete the proof. Let us examine why.

Case: $$\mathcal{D} = \dfrac{\begin{array}{c}\mathcal{D}_1 \\ \cdot;\cdot \vdash M_1 : A_2 \multimap A_1\end{array} \qquad \begin{array}{c}\mathcal{D}_2 \\ \cdot;\cdot \vdash M_2 : A_2\end{array}}{\cdot;\cdot \vdash M_1 \hat{\ } M_2 : A_1} \multimap\mathrm{E}.$$

We can make the following inferences.

$M_1 \hookrightarrow v_1$ for some v_1	By ind. hyp. on $\mathcal{D}_1$
$v_1 = \hat{\lambda} w{:}A_2.\ M_1'$	By type preservation and inversion
$M_2 \hookrightarrow v_2$ for some v_2	By ind. hyp. on $\mathcal{D}_2$

At this point we cannot proceed: we need a derivation of

$$[v_2/w]M_1' \hookrightarrow v \textit{ for some } v$$

to complete the derivation of $M_1\, M_2 \hookrightarrow v$. Unfortunately, the induction hypothesis does not tell us anything about $[v_2/w]M_1'$. Basically, we need to extend it so it makes a statement about the *result* of evaluation ($\hat{\lambda} w{:}A_2.\ M_1'$, in this case).

Sticking to the case of linear application for the moment, we call a term M "good" if it evaluates to a "good" value v. A value v is "good" if it is a function $\hat{\lambda} w{:}A_2.\ M_1'$ and if substituting a "good" value v_2 for w in M_1' results in a "good" term. Note that this is not a proper definition, since to see if v is "good" we may need to substitute any "good" value v_2 into it, possibly including v itself. We can make this definition inductive if we observe that the value v_2 will be of type A_2, while the value v we are testing has type $A_2 \multimap A_1$, and that the resulting term as type A_1. That is, we can fashion a definition which is inductive on the structure of the *type*. Instead of saying "good" we say $M \in \|A\|$ and $v \in |A|$. Still restricting ourselves to linear implication only, we define:

$$\begin{array}{rll} M \in \|A\| & \text{iff} & M \hookrightarrow v \text{ and } v \in |A| \\ M \in |A_2 \multimap A_1| & \text{iff} & M = \hat{\lambda} w{:}A_2.\ M_1 \text{ and } [v_2/w]M_1 \in \|A_1\| \text{ for any } v_2 \in |A_2| \end{array}$$

From $M \in \|A\|$ we can immediately infer $M \hookrightarrow v$, so when proving that $\cdot;\cdot \vdash M : A$ implies $M \in \|A\|$ we do indeed have a much stronger induction hypothesis.

While the case for application now goes through, the case for linear λ-abstraction fails, since we cannot prove the stronger property for the value.

Case: $$\mathcal{D} = \dfrac{\begin{array}{c}\mathcal{D}_1 \\ \cdot;w{:}A_2 \vdash M_1 : A_1\end{array}}{\cdot;\cdot \vdash \hat{\lambda} w{:}A_2.\ M_1 : A_2 \multimap A_1} \multimap\mathrm{I}.$$

Then $\hat{\lambda} w{:}A_2.\ M_1 \hookrightarrow \hat{\lambda} w{:}A_2.\ M_1$ and it remains to show that for every $v_2 \in |A_2|$, $[v_2/w]M_2 \in \|A_1\|$.

This last statement should follow from the induction hypothesis, but presently it is too weak since it only allows for closed terms. The generalization which suggests itself from this case (ignoring the unrestricted context for now) is:

> If $\Delta \vdash M : A$, then for any substitution θ which maps the linear variables $w{:}A$ in Δ to values $v \in |A|$, $[\theta]M \in \|A\|$.

This generalization indeed works after we also account for the unrestricted context. During evaluation we substitute values for linear variables and expressions for unrestricted variables. Therefore, the substitutions we must consider for the induction hypothesis have to behave accordingly.

$$\begin{array}{rrcl} \textit{Unrestricted Substitution} & \eta & ::= & \cdot \mid \eta, M/u \\ \textit{Linear Substitution} & \theta & ::= & \cdot \mid \theta, v/w \end{array}$$

We write $[\eta; \theta]M$ for the simultaneous application of the substitutions η and θ to M. For our purposes here, the values and terms in the substitutions are always closed, but we do not need to postulate this explicitly. Instead, we only deal with substitution satisfying the property necessary for the generalization of the induction hypothesis.

$$\begin{array}{rcl} \theta \in |\Delta| & \text{iff} & [\theta]w \in |A| \text{ for every } w{:}A \text{ in } \Delta \\ \eta \in \|\Gamma\| & \text{iff} & [\eta]u \in \|A\| \text{ for every } u{:}A \text{ in } \Gamma \end{array}$$

We need just one more lemma, namely that values evaluate to themselves.

Lemma 4.11 (Value Evaluation) *For any value v, $v \hookrightarrow v$*

Proof: See Exercise 4.18. □

Now we have all ingredients to state the main lemma in the proof of termination, the so called *logical relations lemma* [**?**]. The "logical relations" are $\|A\|$ and $|A|$, seen as unary relations, that is, predicates, on terms and values, respectively. They are "logical" since they are defined by induction on the structure of the type A, which corresponds to a proposition under the Curry-Howard isomorphism.

Lemma 4.12 (Logical Relations) *If $\Gamma; \Delta \vdash M : A$, $\eta \in \|\Gamma\|$ and $\theta \in |\Delta|$ then $[\eta; \theta]M \in \|A\|$.*

Before showing the proof, we extend the definition of the logical relations to all the types we have been considering.

$$
\begin{array}{rll}
M \in \|A\| & \text{iff} & M \hookrightarrow v \text{ and } v \in |A| \\
v \in |A_2 \multimap A_1| & \text{iff} & v = \hat{\lambda} w{:}A_2.\ M_1 \text{ and } [v_2/w]M_1 \in \|A_1\| \text{ for any } v_2 \in |A_2| \\
v \in |A_1 \otimes A_2| & \text{iff} & v = v_1 \otimes v_2 \text{ where } v_1 \in |A_1| \text{ and } v_2 \in |A_2| \\
v \in |\mathbf{1}| & \text{iff} & v = \star \\
v \in |A_1 \& A_2| & \text{iff} & v = \langle M_1, M_2 \rangle \text{ where } M_1 \in \|A_1\| \text{ and } M_2 \in \|A_2\| \\
v \in |\top| & \text{iff} & v = \langle\,\rangle \\
v \in |A_1 \& A_2| & \text{iff} & \text{either } v = \mathrm{inl}^{A_2}\, v_1 \text{ and } v_1 \in |A_1|, \\
 & & \text{or } v = \mathrm{inr}^{A_1}\, v_2 \text{ and } v_2 \in |A_2| \\
v \in |\mathbf{0}| & & \text{never} \\
v \in |!A| & \text{iff} & v = {!M} \text{ and } M \in \|A\| \\
v \in |A_2 \to A_1| & \text{iff} & v = \lambda u{:}A_2.\ M_1 \text{ and } [M_2/u]M_1 \in \|A_1\| \text{ for any } M_2 \in \|A_2\|
\end{array}
$$

These definitions are motivated directly from the form of values in the language. One can easily see that it is indeed inductive on the structure of the type. If we tried to add recursive types in a similar way, the proof below would still go through, except that the definition of the logical relation would no longer be well-founded.

Proof: (of the logical relations lemma 4.12). The proof proceeds by induction on the structure of the typing derivation $\mathcal{D} :: (\Gamma; \Delta \vdash M : A)$. We show three cases—all others are similar.

$$
\textbf{Case: } \mathcal{D} = \dfrac{\begin{array}{c}\mathcal{D}_1 \\ \Gamma; \Delta \vdash M_1 : A_2 \multimap A_1\end{array} \qquad \begin{array}{c}\mathcal{D}_2 \\ \Gamma; \Delta \vdash M_2 : A_2\end{array}}{\Gamma; \Delta \vdash M_1 \,\hat{}\, M_2 : A_1} \multimap \mathrm{E}.
$$

$$
\begin{array}{ll}
\eta \in \|\Gamma\| & \text{by assumption} \\
\theta \in |\Delta| & \text{by assumption} \\
[\eta;\theta]M_1 \in \|A_2 \multimap A_1\| & \text{by ind. hyp. on } \mathcal{D}_1 \\
\mathcal{E}_1 :: ([\eta;\theta]M_1 \hookrightarrow v_1) \text{ and } v_1 \in |A_2 \multimap A_1| & \text{by definition of } \|A_2 \multimap A_1\| \\
v_1 = \hat{\lambda} w{:}A_1.\ M_1' \text{ and } [v_2/w]M_1' \in \|A_1\| \text{ for any } v_2 \in |A_2| & \text{by definition of } |A_2 \multimap A_1| \\
[\eta;\theta]M_2 \in \|A_2\| & \text{by ind. hyp. on } \mathcal{D}_2 \\
\mathcal{E}_2 :: ([\eta;\theta]M_2 \hookrightarrow v_2) \text{ and } v_2 \in |A_2| & \text{by definition of } \|A_2\| \\
[v_2/w]M_1' \in \|A_1\| & \text{since } v_2 \in |A_2| \\
\mathcal{E}_3 :: ([v_2/w]M_1' \hookrightarrow v) \text{ and } v \in |A_1| & \text{by definition of } \|A_1\| \\
\mathcal{E} :: ([\eta;\theta](M_1 \,\hat{}\, M_2) \hookrightarrow v) & \text{by } \multimap \mathrm{Ev} \text{ from } \mathcal{E}_1,\ \mathcal{E}_2, \text{ and } \mathcal{E}_3 \\
[\eta;\theta](M_1 \,\hat{}\, M_2) \in \|A_1\| & \text{by definition of } \|A_1\|
\end{array}
$$

$$
\textbf{Case: } \mathcal{D} = \dfrac{\begin{array}{c}\mathcal{D}_1 \\ \Gamma; (\Delta, w{:}A_2) \vdash M_1 : A_1\end{array}}{\Gamma; \Delta \vdash \hat{\lambda} w{:}A_2.\ M_1 : A_2 \multimap A_1} \multimap \mathrm{I}.
$$

$$
\begin{array}{ll}
\eta \in \|\Gamma\| & \text{by assumption}
\end{array}
$$

$\theta \in \lvert\Delta\rvert$	by assumption
$\mathcal{E} :: ([\eta;\theta](\hat{\lambda}w{:}A_2.\ M_1) \hookrightarrow [\eta;\theta](\hat{\lambda}w{:}A_2.\ M_1))$	by $\multimap$Iv
$v_2 \in \lvert A_2\rvert$	assumption
$(\theta, v_2/w) \in \lvert\Delta, w{:}A_2\rvert$	by definition of $\lvert\Delta\rvert$
$[\eta;(\theta, v_2/w)]M_1 \in \lVert A_1\rVert$	by ind. hyp. on $\mathcal{D}_1$
$[v_2/w]([\eta;(\theta, w/w)]M_1) \in \lVert A_1\rVert$	by properties of substitution
$(\hat{\lambda}w{:}A_2.\ [\eta;(\theta, w/w)]M_1) \in \lvert A_2 \multimap A_1\rvert$	by definition of $\lvert A_2 \multimap A_1\rvert$
$[\eta;\theta](\hat{\lambda}w{:}A_2.\ M_1) \in \lvert A_2 \multimap A_1\rvert$	by properties of substitution
$[\eta;\theta](\hat{\lambda}w{:}A_2.\ M_1) \in \lVert A_2 \multimap A_1\rVert$	by definition of $\lVert A_2 \multimap A_1\rVert$

Case: $\mathcal{D} = \dfrac{}{\Gamma;(\cdot, w{:}A) \vdash w : A}\, w.$

$\theta \in \lvert\cdot, w{:}A\rvert$	by assumption
$[\theta]w \in \lvert A\rvert$	by definition of $\lvert\cdot, w{:}A\rvert$
$\mathcal{E} :: ([\eta;\theta]w \hookrightarrow [\eta;\theta]w)$	by Lemma 4.11
$[\eta;\theta]w \in \lVert A\rVert$	by definition of $\lVert A\rVert$

□

The termination theorem follows directly from the logical relations lemma. Note that the theorem excludes recursive types and the fixpoint operator by a general assumption for this section.

Theorem 4.13 (Termination) *If* $\cdot;\cdot \vdash M : A$ *then* $M \hookrightarrow v$ *for some* v.

Proof: We have $\cdot \in \lVert\cdot\rVert$ and $\cdot \in \lvert\cdot\rvert$ since the conditions are vacuously satisfied. Therefore, by the logical relations lemma 4.12, $[\cdot;\cdot]M \in \lVert A\rVert$. By the definition of $\lVert A\rVert$ and the observation that $[\cdot;\cdot]M = M$, we conclude that $M \hookrightarrow v$ for some v. □

4.8 Exercises

Exercise 4.1 Prove that if $\Gamma;\Delta \vdash M : A$ and $\Gamma;\Delta \vdash M : A'$ then $A = A'$.

Exercise 4.2 A function in a functional programming language is called *strict* if it is guaranteed to use its argument. Strictness is an important concept in the implementation of lazy functional languages, since a strict function can evaluate its argument eagerly, avoiding the overhead of postponing its evaluation and later memoizing its result.

In this exercise we design a λ-calculus suitable as the core of a functional language which makes strictness explicit at the level of types. Your calculus should contain an unrestricted function type $A \to B$, a strict function type $A \twoheadrightarrow B$, a vacuous function type $A \dashrightarrow B$, a full complement of operators refining product and disjoint sum types as for the linear λ-calculus, and a modal operator to internalize the notion of closed term as in the linear λ-calculus. Your calculus should not contain quantifiers.

1. Show the introduction and elimination rules for all types, including their proof terms.
2. Given the reduction and expansions on the proof terms.
3. State (without proof) the valid substitution principles.
4. If possible, give a translation from types and terms in the strict λ-calculus to types and terms in the linear λ-calculus such that a strict term is well-typed if and only if its linear translation is well-typed (in an appropriately translated context).
5. Either sketch the correctness proof for your translation in each direction by giving the generalization (if necessary) and a few representative cases, or give an informal argument why such a translation is not possible.

Exercise 4.3 Give an example which shows that the substitution $[M/w]N$ must be capture-avoiding in order to be meaningful. *Variable capture* is a situation where a bound variable w' in N occurs free in M, and w occurs in the scope of w'. A similar definition applies to unrestricted variables.

Exercise 4.4 Give a counterexample to the conjecture that if $M \longrightarrow_\beta M'$ and $\Gamma; \Delta \vdash M' : A$ then $\Gamma; \Delta \vdash M : A$. Also, either prove or find a counterexample to the claim that if $M \longrightarrow_\eta M'$ and $\Gamma; \Delta \vdash M' : A$ then $\Gamma; \Delta \vdash M : A$.

Exercise 4.5 The proof term assignment for sequent calculus identifies many distinct derivations, mapping them to the same natural deduction proof terms. Design an alternative system of proof terms from which the sequent derivation can be reconstructed uniquely (up to weakening of unrestricted hypotheses and absorption of linear hypotheses in the $\top$R rule).

1. Write out the term assignment rules for all propositional connectives.
2. Give a calculus of reductions which corresponds to the initial and principal reductions in the proof of admissibility of cut.
3. Show the reduction rule for the dereliction cut.
4. Show the reduction rules for the left and right commutative cuts.
5. Sketch the proof of the subject reduction properties for your reduction rules, giving a few critical cases.
6. Write a translation judgment $S \Longrightarrow M$ from faithful sequent calculus terms to natural deduction terms.
7. Sketch the proof of type preservation for your translation, showing a few critical cases.

Exercise 4.6 Supply the missing rules for $\oplus$E in the definition of the judgment $\Gamma; \Delta_I \setminus \Delta_O \vdash_i M : A$ and show the corresponding cases in the proof of Lemma 4.5.

Exercise 4.7 In this exercise we explore the syntactic expansion of *extended case expressions* of the form **case** M **of** m.

1. Define a judgment which checks if an extended case expression is valid. This is likely to require some auxiliary judgments. You must verify that the cases are exhaustive, circumscribe the legal patterns, and check that the overall expression is linearly well-typed.

2. Define a judgment which relates an extended case expression to its expansion in terms of the primitive **let** , **case** , and abort constructs in the linear λ-calculus.

3. Prove that an extended case expression which is valid according to your criteria can be expanded to a well-typed linear λ-term.

4. Define an operational semantics directly on extended case expressions.

5. Prove that your direct operational semantics is correct on valid patterns with respect to the translational semantics from questions 2.

Exercise 4.8 Define the judgment $M \longrightarrow_\beta^* M'$ via inference rules. The rules should directly express that it is the congruent, reflexive and transitive closure of the β-reduction judgment $M \longrightarrow_\beta M'$. Then prove the generalized subject reduction theorem 4.7 for your judgment. You do not need to show all cases, but you should carefully state your induction hypothesis in sufficient generality and give a few critical parts of the proof.

Exercise 4.9 Define *weak β-reduction* as allows simple β-reduction under $\otimes$, inl, and inr constructs and in all components of the elimination form. Show that if M weakly reduces to a value v then $M \hookrightarrow v$.

Exercise 4.10 Prove type preservation (Theorem 4.9) directly by induction on the structure of the evaluation derivation, using the substitution lemma 4.2 as necessary, but without appeal to subject reduction.

Exercise 4.11 Prove the subject reduction and expansion properties for recursive type computation rules.

Exercise 4.12 [*An exercise exploring the use of type conversion rules without explicit term constructors.*]

Exercise 4.13 Define a linear multiplication function mult : $\mathrm{nat} \multimap \mathrm{nat} \multimap \mathrm{nat}$ using the functions copy and delete.

Exercise 4.14 Defined the following functions on lists. Always explicitly state the type, which should be the most natural type of the function.

1. append to append two lists.

2. concat to append all the lists in a list of lists.

3. map to map a function f over the elements of a list. The result of mapping f over the list $x_1, x_2, \ldots, x_n$ should be the list $f(x_1), f(x_2), \ldots f(x_n)$, where you should decide if the application of f to its argument should be linear or not.

4. foldr to reduce a list by a function f. The result of folding f over a list $x_1, x_2, \ldots x_n$ should be the list $f(x_1, f(x_2, \ldots, f(x_n, \mathit{init})))$, where *init* is an initial value given as argument to foldr. You should decide if the application of f to its argument should be linear or not.

5. copy, delete, and promote.

Exercise 4.15 For one of the form of lazy lists on Page 107, define the functions from Exercise 4.14 plus a function toList which converts the lazy to an eager list (and may therefore not terminate if the given lazy lists is infinite). Make sure that your functions exhibit the correct amount of laziness. For example, a map function applied to a lazy list should not carry out any non-trivial computation until the result is examined.

Further for your choice of lazy list, define the infinite lazy list of eager natural numbers $0, 1, 2, \ldots$.

Exercise 4.16 Prove that there is no term v such that $\omega\,(\mathrm{fold}^{\Omega}\,\omega) \hookrightarrow v$.

Exercise 4.17 [*An exercise about the definability of fixpoint operators at various type.*]

Exercise 4.18 Prove Lemma 4.11 which states that all values evaluate to themselves.

Chapter 5

A Linear Logical Framework

[*a lot of stuff omitted here for now which is easily accessible in published papers*]

5.1 Representation of Meta-Theory

The features which make the linear logical framework so suitable for the representation of deductive systems with linear or imperative features, also make it a good candidate to represent proofs of properties of such deductive systems. If one follows the natural line of development, however, it turns out that proofs of meta-theorems are representable, but that their validity cannot be guaranteed by linear type-checking alone. Presently, there exist no good tools to verify the necessary additional properties, but their development is the subject of ongoing research.

In this section, we will examine two relatively simple examples of meta-theoretic properties and their proofs in the context of LLF encodings. We will pay particular attention to the additional checks required to verify the proofs, since they must currently be carried out by the user. The examples are soundness and completeness of sequent derivations (including cut) with respect to natural deductions. The proofs we give here is somewhat different from the ones in Chapter 2 since here we are not interested in the fine-grained analysis which relates cut-free sequent derivations to normal deductions.

We present all the LLF encodings in the concrete syntax of linear Twelf introduced in prior lectures and documented in [CP97] and Section 5.2. In particular, we will take full advantage of term reconstruction to recover the types of all free variables, which are implicitly Π-quantified over each declaration.

We begin with the encoding of the intuitionistic, propositional fragment of linear logic. Here we omit propositions $\bot$, $\neg A$, and $?A$ which require a second judgment of "possible truth".

```
%%% Propositions
o : type.                   %name o A

%%% Multiplicatives
lolli  : o -> o -> o.       % A -o B
tensor : o -> o -> o.       % A * B
one    : o.                 % 1

%%% Additives
with : o -> o -> o.         % A & B
top : o.                    % T
plus : o -> o -> o.         % A + B
zero : o.                   % 0

%%% Exponentials
imp : o -> o -> o.          % A -> B
bang : o -> o.              % ! A
```

We encode the main judgment of linear logic, $\Gamma; \Delta \vdash A$ using the standard encoding technique for hypothetical and linear hypothetical judgments. A natural deduction $\mathcal{D}$ of $\Gamma; \Delta \vdash A$ will therefore be represented as a canonical LLF object M such that $\ulcorner\Gamma\urcorner; \ulcorner\Delta\urcorner \vdash^{LF} M : \mathrm{nd}\ulcorner A\urcorner$ where

$$\begin{array}{rcl} \ulcorner . \urcorner & = & . \\ \ulcorner \Gamma, u{:}A \urcorner & = & \ulcorner\Gamma\urcorner, u{:}\mathrm{nd}\ulcorner A\urcorner \\ \ulcorner \Delta, x{:}A \urcorner & = & \ulcorner\Delta\urcorner, x{:}\mathrm{nd}\ulcorner A\urcorner \end{array}$$

With this idea it is almost possible to achieve a perfect bijection between canonical LLF objects of type $\mathrm{nd}\ulcorner A\urcorner$ and natural deductions of A.[1]

```
nd : o -> type.             %name nd D

%%% Multiplicatives

% A -o B
lolliI : (nd A -o nd B) -o nd (lolli A B).
lolliE : nd (lolli A B) -o nd A -o nd B.

% A * B
tensorI : nd A -o nd B -o nd (tensor A B).
tensorE : nd (tensor A B)
           -o (nd A -o nd B -o nd C)
           -o nd C.

% 1
```

[1]For the exception consider the two derivations of $A \multimap (\top \otimes \top)$ and their representation.

```
oneI : nd (one).
oneE : nd (one)
        -o nd C
        -o nd C.

%%% Additives

% A & B
withI : nd A & nd B -o nd (with A B).
withE1 : nd (with A B) -o nd A.
withE2 : nd (with A B) -o nd B.

% T
topI : <T> -o nd (top).
% no topE

% A + B
plusI1 : nd A -o nd (plus A B).
plusI2 : nd B -o nd (plus A B).
plusE : nd (plus A B)
         -o ((nd A -o nd C) & (nd B -o nd C))
         -o nd C.

% 0
% no zeroI
zeroE : nd (zero)
         -o <T>
         -o nd C.

%%% Exponentials

% A -> B
impI : (nd A -> nd B) -o nd (imp A B).
impE : nd (imp A B) -o nd A -> nd B.

% ! A
bangI : nd A -> nd (bang A).
bangE : nd (bang A)
         -o (nd A -> nd C)
         -o nd C.
```

Representation of the sequent calculus is discussed in some detail in the literature [CP97]. Briefly, it arises by considering two basic judgments, "*A is a hypothesis*" and "*A is the conclusion*". We represent this via two corresponding type families, left$\ulcorner A \urcorner$ for hypotheses and right$\ulcorner A \urcorner$ for the conclusion. So a sequent derivation of $\Gamma; \Delta \Longrightarrow A$ is represented by a canonical LLF object M

such that $\ulcorner\Gamma\urcorner;\ulcorner\Delta\urcorner \stackrel{LF}{\vdash} M : \mathrm{right}\ulcorner A\urcorner$, where

$$\begin{array}{rcl} \ulcorner\cdot\urcorner & = & \cdot \\ \ulcorner\Gamma, u{:}A\urcorner & = & \ulcorner\Gamma\urcorner, u{:}\mathrm{left}\ulcorner A\urcorner \\ \ulcorner\Delta, x{:}A\urcorner & = & \ulcorner\Delta\urcorner, x{:}\mathrm{left}\ulcorner A\urcorner \end{array}$$

Unlike natural deduction, we now need to explicitly connect the left and right judgments, which is done in two dual ways. Initial sequents allow us to conclude right$\ulcorner A\urcorner$ from left$\ulcorner A\urcorner$. The rules of Cut allow us to substitute a derivation of right$\ulcorner A\urcorner$ for a hypothesis left$\ulcorner A\urcorner$. There are two forms of cut, since we might replace a linear or unrestricted hypothesis.

```
left  : o -> type.          %name left L
right : o -> type.          %name right R

%%% Initial sequents
init : (left A -o right A).

%%% Cuts
cut : right A
       -o (left A -o right C)
       -o right C.

cut! : right A
        -> (left A -> right C)
        -o right C.
```

The renaming rules are the left and right rules for each connective, cast into LLF.

```
%%% Multiplicatives

% A -o B
lolliR : (left A -o right B) -o right (lolli A B).
lolliL : right A
          -o (left B -o right C)
          -o (left (lolli A B) -o right C).

% A * B
tensorR : right A -o right B -o right (tensor A B).
tensorL : (left A -o left B -o right C)
           -o (left (tensor A B) -o right C).

% 1
oneR : right (one).
oneL : (right C)
        -o (left (one) -o right C).
```

```
%%% Additives

% A & B
withR : right A & right B -o right (with A B).
withL1 : (left A -o right C)
          -o (left (with A B) -o right C).
withL2 : (left B -o right C)
          -o (left (with A B) -o right C).

% T
topR : <T> -o right (top).
% no topL

% A + B
plusR1 : right A -o right (plus A B).
plusR2 : right B -o right (plus A B).
plusL : (left A -o right C) & (left B -o right C)
        -o (left (plus A B) -o right C).

% 0
% no zeroR
zeroL : <T> -o (left (zero) -o right C).

%%% Exponentials

% A -> B
impR : (left A -> right C) -o right (imp A C).
impL : right A
        -> (left B -> right C)
        -o (left (imp A B) -o right C).

% ! A
bangR : right A -> right (bang A).
bangL : (left A -> right C)
         -o (left (bang A) -o right C).
```

Note that LLF allows a one-by-one representation of the rules without any auxiliary judgment forms. It is this directness of encoding which makes it feasible to write out the proof of soundness and completeness of the sequent calculus with respect to natural deduction in a similar manner. We begin with the slightly easier direction of soundness.

Theorem 5.1 (Soundness of Sequent Calculus with Cut)
If $\Gamma; \Delta \Longrightarrow A$ *then* $\Gamma; \Delta \vdash A$.

Proof: By induction over the structure of the given sequent derivation. This constructive proof contains a method by which a natural deduction $\mathcal{D}$ can be constructed from a sequent derivation $\mathcal{R}$. Under the Curry-Howard isomorphism, one might expect this method to be represented as a dependently typed function

$$\mathrm{sd} : \Pi A{:}o.\ \mathrm{right}\, A \to \mathrm{nd}\, A.$$

Unfortunately, such a function cannot be defined within LLF, since it lacks the means to define functions recursively, or distinguish cases based on the possible input derivations.[2]

Instead, we represent the proof as a *higher-level judgment* sd relating the derivations $\mathcal{D}$ and $\mathcal{R}$. We do not give the awkward informal presentation of this relation, only its implementation in linear Twelf. Since hypotheses on the left of the sequent are represented by a different judgment, we also need to explicitly relate the hypotheses in the two judgments, using sd$'$.

```
sd : right A -> nd A -> type.
sd' : left A -> nd A -> type.
```

Each case in the proof corresponds to an inference rule defining this higher-level judgment.

Case: $\mathcal{R} = \dfrac{}{\Gamma; x{:}A \Longrightarrow A}\,\mathrm{I}.$

Then

$$\dfrac{}{\Gamma; x{:}A \vdash A}\,x$$

is the corresponding derivation. In the formalization, the labels of the hypothesis A are different, but related by sd$'$. Thus we declare:

```
initSD : sd (init ^ L) X
          o- sd' L X.
```

where L stands for label of the sequent hypothesis (of type left$\ulcorner A \urcorner$) and X stands for the label of the natural deduction hypothesis (of type nd$\ulcorner A \urcorner$).

Case: $\mathcal{R} = \dfrac{\begin{array}{c}\mathcal{R}_1\\ \Gamma; \Delta_1 \Longrightarrow A\end{array} \qquad \begin{array}{c}\mathcal{R}_2\\ \Gamma; \Delta_2 \Longrightarrow B\end{array}}{\Gamma; (\Delta_1 \times \Delta_2) \Longrightarrow A \otimes B}\,\otimes\mathrm{R}.$

Then we reason:

$\mathcal{D}_1 :: (\Gamma; \Delta_1 \vdash A)$	By i.h. on $\mathcal{R}_1$
$\mathcal{D}_2 :: (\Gamma; \Delta_2 \vdash B)$	By i.h. on $\mathcal{R}_2$
$\mathcal{D} :: (\Gamma; (\Delta_1 \times \Delta_2) \vdash A \otimes B)$	By $\otimes$I from $\mathcal{D}_1$ and $\mathcal{D}_2$

[2]The lack of definition by cases and recursion is essential to obtain adequate encodings (see [SDP97])

The appeals to the induction hypothesis are implemented by "recursive calls" in the sd judgment.

```
tensorRSD : sd (tensorR ^ R1 ^ R2) (tensorI ^ D1 ^ D2)
             o- sd R1 D1
             o- sd R2 D2.
```

Case: $\mathcal{R} = \dfrac{\begin{array}{c}\mathcal{R}_1\\ \Gamma;(\Delta_1, x_1{:}A, x_2{:}B) \Longrightarrow C\end{array}}{\Gamma;(\Delta_1, x{:}A\otimes B) \Longrightarrow C}\otimes\mathrm{L}.$

In this case we reason:

$\mathcal{D}_1 :: (\Gamma;(\Delta_1, x_1{:}A, x_2{:}B) \vdash C)$ By i.h. on $\mathcal{R}_1$
$\mathcal{X} :: (\Gamma; x{:}A\otimes B \vdash A\otimes B)$ By hypothesis x
$\mathcal{D} :: (\Gamma;(\Delta_1, x{:}A\otimes B) \vdash C)$ By $\otimes$E from $\mathcal{X}$ and $\mathcal{D}_1$

In the implementation, we have to take care to introduce the new hypotheses in the premisses, that is, the parameters x_1 and x_2. Further, we need to relate the hypotheses in the two different judgments (as indicated already in the case for initial sequents above), by using the sd′ judgment. We label the hypotheses in the sequent calculus with l, l_1, and l_2.

```
tensorLSD : sd (tensorL ^ R1 ^ L) (tensorE ^ X ^ D1)
             o- ({l1:left A} {l2:left B} {x1:nd A} {x2:nd B}
                   sd' l1 x1
                   -o sd' l2 x2
                   -o sd (R1 ^ l1 ^ l2) (D1 ^ x1 ^ x2))
             o- sd' L X.
```

Case: $\mathcal{R} = \dfrac{\begin{array}{c}\mathcal{R}_1\\ \Gamma;\Delta \Longrightarrow A\end{array} \qquad \begin{array}{c}\mathcal{R}_2\\ \Gamma;\Delta \Longrightarrow B\end{array}}{\Gamma;\Delta \Longrightarrow A\&B}\&\mathrm{R}.$

$\mathcal{D}_1 :: (\Gamma;\Delta \vdash A)$ By i.h. on $\mathcal{R}_1$
$\mathcal{D}_2 :: (\Gamma;\Delta \vdash B)$ By i.h. on $\mathcal{R}_2$
$\mathcal{D} :: (\Gamma;\Delta \vdash A\&B)$ By &I from $\mathcal{D}_1$ and $\mathcal{D}_2$

In the formalization of this case, we have to be careful to use the alternative conjunction at the meta-level as well, since the assumptions about the connection between hypotheses left$\ulcorner A\urcorner$ and nd$\ulcorner A\urcorner$ are linear and may be needed in both branches.

```
withRSD : sd (withR ^ (R1, R2)) (withI ^ (D1, D2))
           o- sd R1 D1 & sd R2 D2.
```

Case: $\mathcal{R} = \dfrac{\begin{array}{c}\mathcal{R}_1 \\ \Gamma; (\Delta_1, x_1{:}A) \Longrightarrow C\end{array}}{\Gamma; (\Delta_1, x{:}A \& B) \Longrightarrow C}\ \&\mathrm{L}_1.$

This case is slightly more complicated than the previous ones, since we need to appeal to the substitution lemma.

$\mathcal{D}_1 :: (\Gamma; \Delta_1, x_1{:}A \vdash C)$	By i.h. on $\mathcal{R}_1$
$\mathcal{D}_2 :: (\Gamma; x{:}A\&B \vdash A)$	By $\&\mathrm{E}_1$ from hypothesis x
$\mathcal{D} :: (\Gamma; (\Delta_1, x{:}A\&B) \vdash C)$	By the substitution lemma from $\mathcal{D}_1$ and $\mathcal{D}_2$

The implementation exploits the compositionality of the representation to model the appeal to the substitution lemma by application in the LLF. We have (in the LLF representation):

$$\begin{array}{rcl}
\ulcorner \Delta_1 \urcorner & \vdash^{LF} & (\hat{\lambda} x_1{:}\mathrm{nd}\,\ulcorner A \urcorner.\ \ulcorner \mathcal{D}_1 \urcorner) : \mathrm{nd}\,\ulcorner A \urcorner \multimap \mathrm{nd}\,\ulcorner C \urcorner \\
x{:}\mathrm{nd}\,(\mathrm{with}\,\ulcorner A \urcorner \ulcorner B \urcorner) & \vdash^{LF} & \mathrm{withE}_1 \,\hat{}\, x : \mathrm{nd}\,\ulcorner A \urcorner \\
\ulcorner \Delta_1, x{:}A\&B \urcorner & \vdash^{LF} & (\hat{\lambda} x_1{:}\mathrm{nd}\,\ulcorner A \urcorner.\ \ulcorner \mathcal{D}_1 \urcorner) \,\hat{}\, (\mathrm{withE}_1 \,\hat{}\, x) : \mathrm{nd}\,\ulcorner C \urcorner \\
\ulcorner \Delta_1, x{:}A\&B \urcorner & \vdash^{LF} & [(\mathrm{withE}_1 \,\hat{}\, x)/x_1] \ulcorner \mathcal{D}_1 \urcorner : \mathrm{nd}\,\ulcorner C \urcorner
\end{array}$$

In the concrete code, `D1` will be bound to $(\hat{\lambda} x_1{:}\mathrm{nd}\,\ulcorner A \urcorner.\ \ulcorner \mathcal{D}_1 \urcorner)$, so the application in the second to the last line is written as `D1 ^ (withE1 ^ X)`.

```
withL1SD : sd (withL1 ^ R1 ^ L) (D1 ^ (withE1 ^ X))
            o- ({l1:left A} {x1:nd A}
                  sd' l1 x1 -o sd (R1 ^ l1) (D1 ^ x1))
            o- sd' L X.
```

Case: $\mathcal{R} = \dfrac{\begin{array}{c}\mathcal{R}_1 \\ \Gamma; \Delta_1 \Longrightarrow A\end{array} \qquad \begin{array}{c}\mathcal{R}_2 \\ \Gamma; (\Delta_2, x{:}A) \Longrightarrow C\end{array}}{\Gamma; (\Delta_1 \times \Delta_2) \Longrightarrow C}\ \mathrm{Cut}.$

$\mathcal{D}_1 :: (\Gamma; \Delta_1 \vdash A)$	By i.h. on $\mathcal{R}_1$
$\mathcal{D}_2 :: (\Gamma; (\Delta_2, x{:}A) \vdash C)$	By i.h. on $\mathcal{R}_2$
$\mathcal{D} :: (\Gamma; (\Delta_1 \times \Delta_2) \vdash C)$	By the substitution lemma from $\mathcal{D}_1$ and $\mathcal{D}_2$

The representation uses the same technique of meta-level application as the case for $\&\mathrm{R}_1$ above.

```
cutSD : sd (cut ^ R1 ^ R2) (D2 ^ D1)
         o- sd R1 D1
         o- ({l:left A} {x:nd A}
               sd' l x -o sd (R2 ^ l) (D2 ^ x)).
```

□

Among the remaining cases, the exponentials and the cut of an unrestricted hypothesis require some additional case to make sure the meta-level hypothesis are also unrestricted. Similarly, in the case for $\top$R, care must be taken so the linearity of the meta-reasoning is not violated by allowing weakening with the $\top$ of LLF. We leave it to the reader to write out these cases and relate them to the complete code given below.

```
sd : right A -> nd A -> type.
sd' : left A -> nd A -> type.

%%% Initial sequents
initSD : sd (init ^ L) X
          o- sd' L X.

%%% Cuts
cutSD : sd (cut ^ R1 ^ R2) (D2 ^ D1)
         o- sd R1 D1
         o- ({l:left A} {x:nd A}
               sd' l x -o sd (R2 ^ l) (D2 ^ x)).

cut!SD : sd (cut! R1 ^ R2) (D2 D1)
          <- sd R1 D1
          o- ({l:left A} {u:nd A}
                sd' l u -> sd (R2 l) (D2 u)).

%%% Multiplicatives

% A -o B
lolliRSD : sd (lolliR ^ R1) (lolliI ^ D1)
            o- ({l:left A} {x:nd A}
                  sd' l x -o sd (R1 ^ l) (D1 ^ x)).

lolliLSD : sd (lolliL ^ R1 ^ R2 ^ L) (D2 ^ (lolliE ^ X ^ D1))
            o- sd R1 D1
            o- ({l:left B} {x:nd B}
                  sd' l x -o sd (R2 ^ l) (D2 ^ x))
            o- sd' L X.

% A * B
tensorRSD : sd (tensorR ^ R1 ^ R2) (tensorI ^ D1 ^ D2)
             o- sd R1 D1
             o- sd R2 D2.

tensorLSD : sd (tensorL ^ R1 ^ L) (tensorE ^ X ^ D1)
             o- ({l1:left A} {l2:left B} {x1:nd A} {x2:nd B}
                   sd' l1 x1
                   -o sd' l2 x2
                   -o sd (R1 ^ l1 ^ l2) (D1 ^ x1 ^ x2))
             o- sd' L X.
```

```
% 1
oneRSD : sd (oneR) (oneI).

oneLSD : sd (oneL ^ R1 ^ L) (oneE ^ X ^ D1)
          o- sd R1 D1
          o- sd' L X.

%%% Additives

% A & B
withRSD : sd (withR ^ (R1, R2)) (withI ^ (D1, D2))
           o- sd R1 D1 & sd R2 D2.

withL1SD : sd (withL1 ^ R1 ^ L) (D1 ^ (withE1 ^ X))
            o- ({l1:left A} {x1:nd A}
                  sd' l1 x1 -o sd (R1 ^ l1) (D1 ^ x1))
            o- sd' L X.

withL2SD : sd (withL2 ^ R2 ^ L) (D2 ^ (withE2 ^ X))
            o- ({l2:left B} {x2:nd B}
                  sd' l2 x2 -o sd (R2 ^ l2) (D2 ^ x2))
            o- sd' L X.

% T
topRSD : sd (topR ^ ()) (topI ^ ())
          o- <T>.
% no topL

% A + B
plusR1SD : sd (plusR1 ^ R1) (plusI1 ^ D1)
            o- sd R1 D1.

plusR2SD : sd (plusR2 ^ R2) (plusI2 ^ D2)
            o- sd R2 D2.

plusLSD : sd (plusL ^ (R1, R2) ^ L) (plusE ^ X ^ (D1, D2))
           o- (({l1:left A} {x1:nd A}
                  sd' l1 x1 -o sd (R1 ^ l1) (D1 ^ x1))
                 & ({l2:left B} {x2:nd B}
                      sd' l2 x2 -o sd (R2 ^ l2) (D2 ^ x2)))
           o- sd' L X.

% 0
% no zeroR
zeroLSD : sd (zeroL ^ () ^ L) (zeroE ^ X ^ ())
           o- <T>
           o- sd' L X.

%%% Exponentials
```

```
% A -> B
impRSD : sd (impR ^ R1) (impI ^ D1)
          o- ({l:left A} {u:nd A}
                sd' l u -> sd (R1 l) (D1 u)).

impLSD : sd (impL R1 ^ R2 ^ L) (D2 (impE ^ X D1))
          <- sd R1 D1
          o- ({l:left B} {u:nd B}
                sd' l u -> sd (R2 l) (D2 u))
          o- sd' L X.

% ! A
bangRSD : sd (bangR R1) (bangI D1)
           <- sd R1 D1.

bangLSD : sd (bangL ^ R1 ^ L) (bangE ^ X ^ D2)
           o- ({l:left A} {u:nd A}
                 sd' l u -> sd (R1 l) (D2 u))
           o- sd' L X.
```

The signature above is type-correct in linear Twelf. But what does this establish? For example, assume that we had forgotten the last clause—the signature would still have been correctly typed, but it would no longer represent a proof, since not all possible cases have been covered. So we need to verify the following properties *in addition to the type correctness* in order to be sure that the signature represents a proof.

1. The signature is *well-moded*. This means that when we appeal to the induction hypothesis we have actually constructed an of the appropriate type, and in the end we have fully constructed the object whose existence is postulated. In concrete terms, when we make a recursive call, we need to make sure the input arguments to the type family are fully instantiated, and before we return the output arguments to the type family are also fully instantiated.

   For example, the first argument of the type family `sd` is an input argument, the second an output argument. We then reason as follows:

   ```
   tensorRSD : sd (tensorR ^ R1 ^ R2) (tensorI ^ D1 ^ D2)
                o- sd R1 D1
                o- sd R2 D2.
   ```

   we reason as follows:

   > When we use this clause, the first argument to `sd` is given,
   > so `R1` and `R2` are ground.
   > Therefore, the input arguments to both recursive calls are ground.
   > They yield ground outputs `D1` and `D2`.
   > Therefore the output `(tensorI ^ D1 ^ D2)` will be ground.

Failure of mode-correctness are often simple typographical mistakes which leave the clause well-typed, such as in the following cases.

```
tensorRSD : sd (tensorR ^ R1 ^ R2) (tensorI ^ D1 ^ D2)
             o- sd R1 D1
             o- sd R3 D2.

tensorRSD : sd (tensorR ^ R1 ^ R2) (tensorI ^ D1 ^ D3)
             o- sd R1 D1
             o- sd R2 D2.
```

In the first, `R3` is not ground, in the second `D3` is not ground.

2. The signatures is *terminating*. This means all recursive calls are carried out on smaller terms. These will just be proper subterms if the informal induction argument is over the structure of a derivation. There is one slight complication in that we must allow instantiation of bound variables by parameters so that, for example,

```
tensorLSD : sd (tensorL ^ R1 ^ L) (tensorE ^ X ^ D1)
             o- ({l1:left A} {l2:left B}
                 {x1:nd A} {x2:nd B}
                   sd' l1 x1
                   -o sd' l2 x2
                   -o sd (R1 ^ l1 ^ l2) (D1 ^ x1 ^ x2))
             o- sd' L X.
```

is terminating since `(R1 ^ l1 ^ l2)` is considered a subterm of the input argument `(tensorL ^ R1 ^ L)` because `l1` and `l2` are parameters.

3. The signature *covers* all possible cases. This is the most error-prone property which must be verified, and failures can be subtle. There are three principles classes of failures: A case for a constructor has been omitted, a case for a parameter is missing, or a case is given, but not general enough. For example, a missing case for a parameter arises if we omit the assumption `sd' l1 x1` from the declaration of `tensorLSD` above:

```
tensorLSD : sd (tensorL ^ R1 ^ L) (tensorE ^ X ^ D1)
             o- ({l1:left A} {l2:left B}
                 {x1:nd A} {x2:nd B}
                   sd' l2 x2
                   -o sd (R1 ^ l1 ^ l2) (D1 ^ x1 ^ x2))
             o- sd' L X.
```

When `l1` is encountered in an initial sequence or the principal proposition of a left rule, it will be impossible to proceed with the translation, since `l1` has not been related to the natural deduction hypothesis `x2`.

The more subtle case of insufficient generality is exhibited by the following two examples, both of which mean that perfectly valid translations cannot be carried out. The first fails to allow weakening when translating instances of the $\top$R rule.

```
topRSD : sd (topR ^ ()) (topI ^ ())
          o- <T>.
```

The second applies only to instances where the conclusion of the $\otimes$R rule has the form $A \otimes A$.

```
tensorRSD : sd (tensorR ^ R1 ^ R2) (tensorI ^ D2 ^ D1)
             o- sd R1 D1
             o- sd R2 D2.
```

This fact is only apparent when we look at the reconstructed form of this clause.

```
tensorRSD : {A1:o} {R2:right A1} {D2:nd A1} {R1:right A1} {D1:nd A1}
               sd R2 D2 -o sd R1 D1
                  -o sd (tensorR ^ R1 ^ R2) (tensorI ^ D2 ^ D1).
```

Both R1 and R2 are derivations of sequents with conclusion A1. The reconstruction of the correct clause is

```
tensorRSD : {A1:o} {R2:right A1} {D2:nd A1} {A2:o}
            {R1:right A2} {D1:nd A2}
               sd R2 D2 -o sd R1 D1
                  -o sd (tensorR ^ R1 ^ R2) (tensorI ^ D1 ^ D2).
```

From the above examples one can see that bugs in proof representations can be subtle. Some may be caught by running examples, other by inspection, but the need for a reliable verification procedure should be clear.

The second example is the completeness property: whenever A can be derived by natural deduction, it can also be derived in the sequent calculus. The proof is very direct, but makes rather heavy use of the cut rule (in contrast to the proof in Chapter 2).

Theorem 5.2 (Completeness of Sequent Calculus with Cut)
If $\Gamma; \Delta \vdash A$ *then* $\Gamma; \Delta \Longrightarrow A$.

Proof: By induction of the structure over the given natural deduction. The main insight is that all introduction rules can be translated straightforwardly to right rules, while all elimination rules require one or more uses of the cut rule in the sequent calculus.[3] □

[3] [*show a few cases*]

The representation of this proof is straightforward along the lines of the soundness proof. This time, we need only one meta-level judgment since we can relate hypotheses in natural deduction directly to initial sequent derivations.

```
cp : nd A -> right A -> type.

%%% Multiplicatives

% A -o B
lolliICP : cp (lolliI ^ D1) (lolliR ^ R1)
            o- ({x:nd A} {l:left A}
                  cp x (init ^ l) -o cp (D1 ^ x) (R1 ^ l)).

lolliECP : cp (lolliE ^ D1 ^ D2)
            (cut ^ R2
               ^ ([l^left A] cut ^ R1
                    ^ ([k^left (lolli A B)]
                         lolliL ^ (init ^ l) ^ ([r^left B] init ^ r) ^ k)))
            o- cp D1 R1
            o- cp D2 R2.

% A * B
tensorICP : cp (tensorI ^ D1 ^ D2) (tensorR ^ R1 ^ R2)
             o- cp D1 R1
             o- cp D2 R2.

tensorECP : cp (tensorE ^ D1 ^ D2)
             (cut ^ R1
                ^ [l^left (tensor A B)] tensorL ^ R2 ^ l)
             o- cp D1 R1
             o- ({x:nd A} {l:left A} {y:nd B} {k:left B}
                   cp x (init ^ l)
                   -o cp y (init ^ k)
                   -o cp (D2 ^ x ^ y) (R2 ^ l ^ k)).
% 1
oneICP : cp (oneI) (oneR).

oneECP : cp (oneE ^ D1 ^ D2) (cut ^ R1 ^ ([l^left (one)] oneL ^ R2 ^ l))
          o- cp D1 R1
          o- cp D2 R2.

%%% Additives

% A & B

withICP : cp (withI ^ (D1, D2)) (withR ^ (R1, R2))
           o- cp D1 R1 & cp D2 R2.

withE1CP : cp (withE1 ^ D1)
            (cut ^ R1
```

```
             ^ ([l^left (with A B)] withL1 ^ ([k^left A] init ^ k) ^ l))
          o- cp D1 R1.

withE2CP : cp (withE2 ^ D1)
          (cut ^ R1
             ^ ([l^left (with A B)] withL2 ^ ([k^left B] init ^ k) ^ l))
          o- cp D1 R1.

% T
topICP : cp (topI ^ ()) (topR ^ ())
          o- <T>.

% no topE

% A + B
plusI1CP : cp (plusI1 ^ D1) (plusR1 ^ R1)
            o- cp D1 R1.

plusI2CP : cp (plusI2 ^ D2) (plusR2 ^ R2)
            o- cp D2 R2.

plusECP : cp (plusE ^ D1 ^ (D2 , D3))
           (cut ^ R1
              ^ ([l^left (plus A B)] plusL ^ (R2, R3) ^ l))
           o- cp D1 R1
           o- ({x:nd A} {l:left A}
                 cp x (init ^ l)
                 -o cp (D2 ^ x) (R2 ^ l))
              & ({y:nd B} {k:left B}
                   cp y (init ^ k)
                   -o cp (D3 ^ y) (R3 ^ k)).

% 0
% no zeroI
zeroECP : cp (zeroE ^ D1 ^ ())
           (cut ^ R1 ^ ([l^left (zero)] zeroL ^ () ^ l))
           o- cp D1 R1
           o- <T>.

%%% Exponentials

% A -> B
impICP : cp (impI ^ D1) (impR ^ R1)
          o- ({u:nd A} {l:left A}
                cp u (init ^ l) -> cp (D1 u) (R1 l)).

impECP : cp (impE ^ D1 D2)
          (cut! R2
             ^ ([l:left A] cut ^ R1
                  ^ ([k^left (imp A B)]
```

```
                     impL (init ^ l) ^ ([r:left B] init ^ r) ^ k)))
         o- cp D1 R1
         <- cp D2 R2.

% ! A
bangICP : cp (bangI D1) (bangR R1)
           <- cp D1 R1.

bangECP : cp (bangE ^ D1 ^ D2)
          (cut ^ R1 ^ ([l^left (bang A)] bangL ^ R2 ^ l))
          o- cp D1 R1
          o- ({u:nd A} {l:left A}
                cp u (init ^ l) -o cp (D2 u) (R2 l)).
```

5.2 Concrete Syntax of Linear Twelf

In this section, we extend the concrete syntax of *Elf* [Pfe94] to express the linear operators of *LLF*. In doing so, we want to fulfill two constraints: first of all, existing *Elf* programs should not undergo any syntactic alteration (unless they declare some of the reserved identifiers that we will introduce) if we were to execute them in an implementation of *LLF* relying on the new syntax. In other words, the extension we propose should be conservative with respect to the syntax of *Elf*. Second, we want to avoid a proliferation of operators: keeping their number as small as possible will make future extensions easier to accommodate if their inclusion appears beneficial.

The set of special characters of *Elf* consists of `% : . ) ( ] [ } {`. We extend these with two symbols: `,` and `^`. LLF object and type family constants are consequently represented as identifiers consisting of any non-empty string that does not contain spaces or the characters `% : . ) ( ] [ } { , ^`. As in *Elf*, identifiers must be separated from each other by whitespace (i.e., blanks, tabs, and new lines) or special characters. We augment the set of reserved identifiers of *Elf* (`type`, `->` and `<-`) with `<T>`, `&`, `-o`, `o-`, `<fst>` and `<snd>`. Although not properly an identifier, the symbol `()` is also reserved; this string is forbidden in *Elf*.

The following table associates every $\lambda^{\Pi,\multimap,\&,\top}$ operator to its concrete representation. Terms in the λ^{Π} sublanguage of *LLF* are mapped to the syntax of *Elf*. This language offers the convenience of writing `->` as `<-` with the arguments reversed in order to give a more operational reading to a program, when desired: under this perspective, we read the expression A `<-` B as "A *if* B". We extend this possibility to linear implication, `-o`. Clearly, when we use `o-`, the arguments should be swapped: A `o-` B is syntactic sugar for B `-o` A.

	Abstract syntax	*Concrete syntax*
Kinds	type	type
	$\Pi x{:}A.\ K$	{x:A}K A -> K K <- A
Types	$P\,M$	P M
	$\top$	<T>
	$A \& B$	A & B
	$A \multimap B$	A -o B B o- A
	$\Pi x{:}A.\ B$	{x:A}B A -> B B <- A
Objects	$\langle\,\rangle$	()
	$\langle M, N\rangle$	M,N
	fst M	<fst> M
	snd M	<snd> M
	$\hat{\lambda} x{:}A.\ M$	[x^A]M
	$M\,\hat{}\,N$	M ^ N
	$\lambda x{:}A.\ M$	[x:A]M
	$M\,N$	M N

The next table gives the relative precedence and associativity of these operators. Parentheses are available to override these behaviors. Note that -o, ->, o-, and <- all have the same precedence.

Precedence	*Operator*	*Position*
highest	<fst> _ <snd> _	left prefix
	_ _ _ ^ _	left associative
	_ & _	right associative
	_ -o _ _ -> _	right associative
	_ o- _ _ <- _	left associative
	_,_	right associative
	_:_	left associative
lowest	{_:_}_ [_:_]_ [_^_]_	left prefix

As in *Elf*, a signature declaration $c : A$ is represented by the program clause:

$$c\ \texttt{:}\ A\texttt{.}$$

Type family constants are declared similarly. For practical purposes, it is convenient to provide a means of declaring linear assumptions. Indeed, whenever the object formalism we want to represent requires numerous linear hypotheses, it is simpler to write them as program clauses than to rely on some initialization routine that assumes them in the context during its execution. To this end, we permit declarations of the form

$$c\ \texttt{\^{}}\ A\texttt{.}$$

with the intent that this declaration should be inserted in the context as a linear assumption.[4]

We retain from *Elf* the use of `%` for comments and interpreter directives. Delimited comments have the form `%{... }%`, where embedded delimited comments must be nested properly. We adopt the conventions available in that language in order to enhance the readability of *LLF* programs [Pfe91]. In particular, we permit keeping the type of bound variables implicit whenever they can be effectively reconstructed by means techniques akin to those currently implemented in *Elf*.

We write `{`x`}`B, `[`x`]`B and `[`x`^]`B when maintaining implicit the type A of the variable x in `{`x`:`A`}`B, `[`x`:`A`]`B and `[`x`^`A`]`B, respectively. Similar conventions apply to dependent kinds. As in *Elf*, the binders for variables quantified at the head of a clause can be omitted altogether if we write these variables with identifiers starting with a capital letter. Moreover, the arguments instantiating them can be kept implicit when using these declarations.

Finally, we relax the requirement of writing *LLF* declarations only in η-long form. With sufficient typing information it is always possible to transform a signature to that format.

[4] [*currently, this is unimplemented*]

Chapter 6

Non-Commutative Linear Logic

[*warning: this chapter is even more tentative some most of the other material in these lecture notes.*]

The goal of this chapter is to develop a system of pure natural deduction which encompasses the (ordinary) intuitionistic simply-typed λ-calculus, the intuitionistic linear λ-calculus, and new constructs for a non-commutative linear λ-calculus. It is important that this calculus be conservative over the intuitionistic and linear fragments, so that we do not lose any expressive power and the new features can be introduced gently into the intended application domains.

The system has applications in functional languages, logic programming languages, and logical frameworks. In functional languages, the non-commutative type system allows us to capture strong stackability properties, thereby, for example, giving a logical and general foundation for observations made about terms in continuation-passing style and monadic style [DP95, ?]. In logic programming languages, it allows us to remove some uses of cut which arise from don't-care non-determinism in languages based on linear logic such as Lolli [HM94, CHP97]. In logical frameworks, non-commutative connectives allow us drastically simplify the representations of problems involving stacks or languages such as the ones above.

We start with the simplest fragment which fixes the basic concepts and auxiliary definitions and then add other connectives and modalities incrementally.

Various formulations of non-commutative linear logic have been considered, both in their classical [?, Roo92, Abr95] and intuitionistic [BG91, Abr90a, Abr90b] variants, including various modal operators, analyzed in particular depth in [?]. Except for a brief mention in [Abr90a], we are not aware of any systematic study of natural deduction, the Curry-Howard isomorphism, and the computational consequences of non-commutativity in the λ-calculus. The material in this chapter may grow to eventually fill this gap in the literature and sketch some applications of non-commutativity in the area of logic programming, logical frameworks, and functional programming, complementing Ruet's investigation of concurrent constraint programming from the point of view of mixed classical non-commutative linear logic [?].

6.1 The Implicational Fragment

In this first section we present the pure implicational fragment, containing only the intuitionistic implication ($\rightarrow$), the linear implication ($\multimap$), ordered right implication ($\twoheadrightarrow$) and ordered left implication ($\rightarrowtail$).

We use a formulation of the main judgment using multiple zones: one for *intuitionistic assumptions*, one for *linear assumptions*, and one for *ordered assumptions*. While this may not be the best formulation for all purposes, it is the one we found most easy to understand. We will also freely go back and forth between propositions and types, using the well-known Curry-Howard correspondence.

$$
\begin{array}{lllll}
\textit{Types} & A & ::= & P & \text{atomic types} \\
 & & | & A_1 \rightarrow A_2 & \text{intuitionistic implication} \\
 & & | & A_1 \multimap A_2 & \text{linear implication} \\
 & & | & A_1 \twoheadrightarrow A_2 & \text{ordered right implication} \\
 & & | & A_1 \rightarrowtail A_2 & \text{ordered left implication}
\end{array}
$$

Objects of the λ-calculus (or proof terms for the underlying logic) are defined in a straightforward fashion. We do not formally distinguish different kinds of variables, although we later use the convention that x stands for intuitionistic assumptions, y for linear assumptions, and z for ordered assumptions.

$$
\begin{array}{lllll}
\textit{Objects} & M & ::= & x & \text{variables} \\
 & & | & \lambda x{:}A.\ M \mid M_1\, M_2 & \text{intuitionistic functions} \\
 & & | & \hat{\lambda} x{:}A.\ M \mid M_1 \,\hat{}\, M_2 & \text{linear functions} \\
 & & | & \overset{>}{\lambda} x{:}A.\ M \mid M_1 \,^{>}\, M_2 & \text{right ordered functions} \\
 & & | & \overset{<}{\lambda} x{:}A.\ M \mid M_1 \,^{<}\, M_2 & \text{left ordered functions}
\end{array}
$$

Contexts are simply lists of assumptions $x{:}A$ with distinct variables x. In a triple of contexts $\Gamma;\Delta;\Omega$ needed for the typing judgment, we also assume that no variable occurs more than once.

$$\textit{Contexts} \quad \Gamma \quad ::= \quad \cdot \mid \Gamma, x{:}A$$

We use the convention that Γ stands for an intuitionistic context, Δ for a linear context, and Ω for an ordered context. We abbreviate $\cdot, x{:}A$ as $x{:}A$.

In order to describe the inference rules, we need some auxiliary operations on contexts, *context join* Ω, Ω' and *context merge* $\Delta \times \Delta'$. Context join preserves the order of the assumption, while the non-deterministic merge allows any interleaving of assumption.

$$
\begin{array}{llcl}
\textit{Context Join} & \Omega, \cdot & = & \Omega \\
& \Omega, (\Omega', x{:}A) & = & (\Omega, \Omega'), x{:}A \\
\textit{Context Merge} & \cdot \times \cdot & = & \cdot \\
& (\Delta, x{:}A) \times \Delta' & = & (\Delta \times \Delta'), x{:}A \\
& \Delta \times (\Delta', x{:}A) & = & (\Delta \times \Delta'), x{:}A
\end{array}
$$

The typing rules below are perhaps most easily understood when reading them from the conclusion to the premises, as rules for the construction of a typing derivation for a term. We have designed the language of objects so that the rules are completely syntax directed, and that every well-typed object has a unique type (but not necessarily a unique typing derivation).

When viewing a derivation bottom-up, we think of context join Ω_1, Ω_2 as *ordered context split* and context merge $\Delta_1 \times \Delta_2$ as *context split*. Both of these are non-deterministic when read in this way, that is, there may be many way to split a context $\Omega = \Omega_1, \Omega_2$ or $\Delta = \Delta_1 \times \Delta_2$.

The typing judgment has the form

$$\Gamma; \Delta; \Omega \vdash M : A$$

where Γ is the context of intuitistic assumptions, Δ is the context of linear assumptions, and Ω is the context of ordered assumptions.

Intuitionistic Functions $A {\rightarrow} B$.

$$\frac{}{(\Gamma_1, x{:}A, \Gamma_2); \cdot; \cdot \vdash x : A}\,\textbf{ivar}$$

$$\frac{(\Gamma, x{:}A); \Delta; \Omega \vdash M : B}{\Gamma; \Delta; \Omega \vdash \lambda x{:}A.\ M : A \rightarrow B}\,{\rightarrow}I$$

$$\frac{\Gamma; \Delta; \Omega \vdash M : A \rightarrow B \qquad \Gamma; \cdot; \cdot \vdash N : A}{\Gamma; \Delta; \Omega \vdash M\,N : B}\,{\rightarrow}E$$

Linear Functions $A \multimap B$.

$$\frac{}{\Gamma; y{:}A; \cdot \vdash y : A}\,\textbf{lvar}$$

$$\frac{\Gamma; (\Delta, y{:}A); \Omega \vdash M : B}{\Gamma; \Delta; \Omega \vdash \hat{\lambda} y{:}A.\ M : A \multimap B}\,{\multimap}I$$

$$\frac{\Gamma; \Delta_1; \Omega \vdash M : A \multimap B \qquad \Gamma; \Delta_2; \cdot \vdash N : A}{\Gamma; (\Delta_1 \times \Delta_2); \Omega \vdash M\,\hat{}\,N : B}\,{\multimap}E$$

Ordered Variables.

$$\frac{}{\Gamma; \cdot; z{:}A \vdash z : A}\,\mathbf{ovar}$$

Right Ordered Functions $A \twoheadrightarrow B$**.**

$$\frac{\Gamma; \Delta; (\Omega, z{:}A) \vdash M : B}{\Gamma; \Delta; \Omega \vdash \stackrel{>}{\lambda} z{:}A.\ M : A \twoheadrightarrow B}\,{\twoheadrightarrow}I$$

$$\frac{\Gamma; \Delta_1; \Omega_1 \vdash M : A \twoheadrightarrow B \qquad \Gamma; \Delta_2; \Omega_2 \vdash N : A}{\Gamma; (\Delta_1 \times \Delta_2); (\Omega_1, \Omega_2) \vdash M^{>} N : B}\,{\twoheadrightarrow}E$$

Left Ordered Functions $A \rightarrowtail B$**.**

$$\frac{\Gamma; \Delta; (z{:}A, \Omega) \vdash M : B}{\Gamma; \Delta; \Omega \vdash \stackrel{<}{\lambda} z{:}A.\ M : A \rightarrowtail B}\,{\rightarrowtail}I$$

$$\frac{\Gamma; \Delta_2; \Omega_2 \vdash M : A \rightarrowtail B \qquad \Gamma; \Delta_1; \Omega_1 \vdash N : A}{\Gamma; (\Delta_1 \times \Delta_2); (\Omega_1, \Omega_2) \vdash M^{<} N : B}\,{\rightarrowtail}E$$

These rules enforce linearity and ordering constraints on assumptions through the restrictions placed upon contexts.

In the three variable rules **ivar**, **lvar**, and **ovar**, the linear and ordered contexts must either be empty or contain only the subject variable, while the intuitionistic context is unrestricted. This forces linear and ordered assumptions made in the ${\multimap}I$ and ${\twoheadrightarrow}I$ rules to be appear at least once in a term.

In the ${\multimap}E$ and ${\twoheadrightarrow}E$ rules, the linear context is split into two disjoint parts (when reading from the bottom up), which means that each assumption can be used at most once. In the ${\rightarrow}E$ rules, all linear assumption propagate to the left premise. These observations together show that each linear variable is used at most once. Since it is also used at least once by the observation made about the variable rules, linear assumptions occur exactly once.

In the ${\twoheadrightarrow}E$ rules, the ordered context is split in an order-preserving way, with the leftmost assumptions Ω_1 going to the left premise and the rightmost assumptions Ω_2 going to the right premise. In the ${\multimap}E$ and ${\rightarrow}E$ rules the whole ordered context Ω goes to the left premise. These observations, together with the observation on the variable rules, show that ordered assumptions occur exactly once and in the order they were made.

As we will see, the emptiness restrictions on the linear and ordered contexts in the ${\multimap}E$ and ${\rightarrow}E$ rules are necessary to guarantee subject reduction. The reduction rules, of course, are simply β-reduction for all three kinds of functions. We will later also consider a form of η-expansion.

Reduction Rules.

$$
\begin{array}{rcl}
(\lambda x.\ M)\,N & \Longrightarrow & [N/x]M \\
(\hat{\lambda} x.\ M)\,\hat{}\,N & \Longrightarrow & [N/x]M \\
(\overset{>}{\lambda} x.\ M)^{>}\,N & \Longrightarrow & [N/x]M \\
(\overset{<}{\lambda} x.\ M)^{<}\,N & \Longrightarrow & [N/x]M
\end{array}
$$

In order to prove subject reduction we proceed to establish the expected structural properties for contexts and then verify the expected substitution lemmas.

Lemma 6.1 *The following structural properties hold for derivations in the implicational fragment of INCLL.*

1. *(Intuitionistic Exchange)*
If $(\Gamma_1, x{:}A, x'{:}A', \Gamma_2); \Delta; \Omega \vdash M : B$ *then* $(\Gamma_1, x'{:}A', x{:}A, \Gamma_2); \Delta; \Omega \vdash M : B$.

2. *(Intuitionistic Weakening)*
If $(\Gamma_1, \Gamma_2); \Delta; \Omega \vdash M : B$ *then* $(\Gamma_1, x{:}A, \Gamma_2); \Delta; \Omega \vdash M : B$.

3. *(Intuitionistic Contraction)*
If $(\Gamma_1, x{:}A, \Gamma_2, x'{:}A, \Gamma_3); \Delta; \Omega \vdash M : B$ *then* $(\Gamma_1, x{:}A, \Gamma_2, \Gamma_3); \Delta; \Omega \vdash [x/x']M : B$.

4. *(Linear Exchange)*
If $\Gamma; (\Delta_1, y{:}A, y'{:}A', \Delta_2); \Omega M : B$ *then* $\Gamma; (\Delta_1, y'{:}A', y{:}A, \Delta_2); \Omega \vdash M : B$.

Proof: By straightforward induction on the structure of the given derivations. □

It is easy to construct counterexamples to the missing properties such as "linear contraction" or "ordered exchange". With these properties we can now establish the critical substitution lemmas.

Lemma 6.2 *The following substitution properties hold for the implicational fragment of INCLL.*

1. *(Intuitionistic Substitution)*
If $(\Gamma_1, x{:}A, \Gamma_2); \Delta; \Omega \vdash M : B$ *and* $\Gamma_1; \cdot; \cdot \vdash N : A$ *then* $(\Gamma_1, \Gamma_2); \Delta; \Omega \vdash [N/x]M : B$.

2. *(Linear Substitution)*
If $\Gamma; (\Delta_1, y{:}A, \Delta_2); \Omega \vdash M : B$ *and* $\Gamma; \Delta'; \cdot \vdash N : A$ *then* $\Gamma; (\Delta_1, \Delta', \Delta_2); \Omega \vdash [N/x]M : B$.

3. *(Ordered Substitution)*
If $\Gamma; \Delta; (\Omega_1, x{:}A, \Omega_2) \vdash M : B$ *and* $\Gamma; \Delta'; \Omega' \vdash N : A$ *then* $\Gamma; (\Delta \times \Delta'); (\Omega_1, \Omega', \Omega_2) \vdash [N/x]M : B$.

Proof: By induction over the structure of the given typing derivation for M in each case, using Lemma 6.1. □

Subject reduction now follows immediately.

Theorem 6.3 (Subject Reduction) *If $M \Longrightarrow M'$ and $\Gamma; \Delta; \Omega \vdash M : A$ then $\Gamma; \Delta; \Omega \vdash M' : A$.*

Proof: For each reduction, we apply inversion to the giving typing derivation and then use the substitution lemma 6.2 to obtain the typing derivation for the conclusion. □

We also have three forms of η-expansion.

Theorem 6.4 (Subject Expansion) *The following η-expansion properties hold for the implication fragment of INCLL.*

1. *(Intuitionistic Expansion) If $\Gamma; \Delta; \Omega \vdash M : A \to B$ then $\Gamma; \Delta; \Omega \vdash \lambda x{:}A.\ M\, x : A \to B$.*
2. *(Linear Expansion) If $\Gamma; \Delta; \Omega \vdash M : A \multimap B$ then $\Gamma; \Delta; \Omega \vdash \hat{\lambda} y{:}A.\ M\,\hat{}\, y : A \multimap B$.*
3. *(Right Ordered Expansion) If $\Gamma; \Delta; \Omega \vdash M : A \twoheadrightarrow B$ then $\Gamma; \Delta; \Omega \vdash \overset{>}{\lambda} z{:}A.\ M^{>} z : A \twoheadrightarrow B$.*
4. *(Left Ordered Expansion) If $\Gamma; \Delta; \Omega \vdash M : A \rightarrowtail B$ then $\Gamma; \Delta; \Omega \vdash \overset{<}{\lambda} z{:}A.\ M^{<} z : A \rightarrowtail B$.*

Proof: By a straightforward derivation in each case, using intuitionistic weakening for intuitionistic expansion. □

We also believe that our calculus satisfies the normalization and Church-Rosser properties, and that canonical form (that is, long $\beta\eta$-normal forms) exist for well-typed objects. We have not checked all details for the above formulation, but it appears that these properties can be established by straightforward logical relations arguments.

6.2 Other Logical Connectives

Putting off the ordered left implication for the moment, there are other multiplicative and additive connectives. However, there is no explosion in the number of connectives, since the link between linear and ordered hypotheses rules out certain possibilities. The coupling between the linear and ordered hypotheses arises from a desire to look at linear hypotheses as a form of intuitionistic hypotheses whose use is restricted, and at ordered hypotheses as a form of linear hypotheses whose use is even further restricted. Extension of our core so far should therefore preserve the following property of *demotion*.

Lemma 6.5 *The following structural properties hold for derivations in the right implicational fragment of INCLL.*

1. *(Linear Demotion)*
   If $(\Gamma_1, \Gamma_2); (\Delta_1, y{:}A, \Delta_2); \Omega \vdash M : B$ *then* $(\Gamma_1, x{:}A, \Gamma_2); (\Delta_1, \Delta_2); \Omega \vdash [x/y]M : B$.

2. *(Ordered Demotion)*
   If $\Gamma; (\Delta_1, \Delta_2); (\Omega_1, z{:}A, \Omega_2) \vdash M : B$ *then* $\Gamma; (\Delta_1, y{:}A, \Delta_2); (\Omega_1, \Omega_2) \vdash [y/z]M : B$.

Proof: In both cases by induction on the structure of the given derivation. □

Preserving this property means that we cannot have a connective which, for example, behaves multiplicatively on the linear context and additively on the ordered context. Some other connectives which we do not show below are definable through use of the modal operators in a way which even preserves the structure of proofs.

Tensor $A \otimes B$**.** This is adjoint to the right ordered implication, that is, $(A \otimes B) \twoheadrightarrow C$ iff $A \twoheadrightarrow (B \twoheadrightarrow C)$. Its rules introduce commutative conversions into the proof term calculus, and canonical forms no longer exist.

$$\frac{\Gamma; \Delta_1; \Omega_1 \vdash M{:}A \qquad \Gamma; \Delta_2; \Omega_2 \vdash N{:}B}{\Gamma; (\Delta_1 \times \Delta_2); (\Omega_1, \Omega_2) \vdash M \otimes N : A \otimes B}\otimes\mathrm{I}$$

$$\frac{\Gamma; \Delta_2; \Omega_2 \vdash M : A \otimes B \qquad \Gamma; \Delta_1; (\Omega_1, z{:}A, z'{:}B, \Omega_3) \vdash N : C}{\Gamma; (\Delta_1 \times \Delta_2); (\Omega_1, \Omega_2, \Omega_3) \vdash \mathbf{let}\, z \otimes z' = M \,\mathbf{in}\, N : C}\otimes\mathrm{E}$$

Besides destroying the existence of canonical forms, this connective also complicates the simple functional interpretation of the ordered context Ω as describing a stack. The problem is foreshadowed in the substitution lemma, where we also have to allow ordered variables to the left and right of the variable to be substituted. The new reduction rule is rather straightforward.[1]

$$\mathbf{let}\, z \otimes z' = M \otimes M' \,\mathbf{in}\, N \Longrightarrow [M/z, M'/z']N$$

Multiplicative Unit 1. The is the right and left unit element for the tensor connective. We have $\mathbf{1} \twoheadrightarrow C$ iff C off $\mathbf{1} \rightarrowtail C$, and $A \otimes \mathbf{1}$ iff A iff $\mathbf{1} \otimes A$. The

[1]It seems plausible that the restriction of this rule to $\Omega_3 = \cdot$ is also sound and complete, and that the general form is admissible in the system with the restricted rule. This would form a much better basis for functional language applications of this calculus, since the stack-like nature of accesses to the ordered context is preserved. Similar remarks may hold for the other elimination rules of a similar shape.

introduction rule shows why there is only one multiplicative unit.

$$\frac{}{\Gamma; \cdot; \cdot \vdash \star : \mathbf{1}}\ \mathbf{1}\mathrm{I}$$

$$\frac{\Gamma; \Delta_2; \Omega_2 \vdash M : \mathbf{1} \qquad \Gamma; \Delta_1; (\Omega_1, \Omega_3) \vdash N : C}{\Gamma; (\Delta_1 \times \Delta_2); (\Omega_1, \Omega_2, \Omega_3) \vdash \mathbf{let}\, \star = M \,\mathbf{in}\, N : C}\ \mathbf{1}\mathrm{E}$$

The reduction rule is straightforward.

$$\mathbf{let}\, \star = \star \,\mathbf{in}\, N \Longrightarrow N$$

Additive Conjunction $A\&B$. This is additive on both the linear an ordered contexts, in order to preserve demotion.

$$\frac{\Gamma; \Delta; \Omega \vdash M : A \qquad \Gamma; \Delta; \Omega \vdash N : B}{\Gamma; \Delta; \Omega \vdash \langle M, N \rangle : A\&B}\ \&\mathrm{I}$$

$$\frac{\Gamma; \Delta; \Omega \vdash M : A\&B}{\Gamma; \Delta; \Omega \vdash \mathbf{fst}\, M : A}\ \&\mathrm{E}_1 \quad \frac{\Gamma; \Delta; \Omega \vdash M : A\&B}{\Gamma; \Delta; \Omega \vdash \mathbf{snd}\, M : B}\ \&\mathrm{E}_2$$

$$\begin{array}{rcl} \mathbf{fst}\, \langle M, N \rangle & \Longrightarrow & M \\ \mathbf{snd}\, \langle M, N \rangle & \Longrightarrow & N \end{array}$$

Additive Unit $\top$. Because it is additive, the left and right units for $\&$ coincide.

$$\frac{}{\Gamma; \Delta; \Omega \vdash \langle\,\rangle : \top}\ \top I$$

(no $\top E$ rule)

Since there is no elimination rule, there are no reduction for the additive unit.

Disjunction $\oplus$. The disjunction in intuitionistic linear logic and its non-commutative refinement is additive. Therefore the connective does not split into left and right disjunction.

$$\frac{\Gamma; \Delta; \Omega \vdash M : A}{\Gamma; \Delta; \Omega \vdash \mathbf{inl}^B\, M : A \oplus B}\ \oplus\mathrm{I}_1 \quad \frac{\Gamma; \Delta; \Omega \vdash M : A}{\Gamma; \Delta; \Omega \vdash \mathbf{inr}^A\, M : A \oplus B}\ \oplus\mathrm{I}_2$$

$$\frac{\Gamma; \Delta_2; \Omega_2 \vdash M : A \oplus B \qquad \Gamma; \Delta_1; (\Omega_1, z{:}A, \Omega_3) \vdash N : C \qquad \Gamma; \Delta_1; (\Omega_1, z'{:}B, \Omega_3) \vdash N' : C}{\Gamma; (\Delta_1 \times \Delta_2); (\Omega_1, \Omega_2, \Omega_3) \vdash \mathbf{case}\, M \,\mathbf{of}\, \mathbf{inl}\, z \Rightarrow N \mid \mathbf{inr}\, z' \Rightarrow N' : C}\ \oplus\mathrm{E}$$

$$\begin{array}{rcl}
\mathbf{case}\,\mathbf{inl}^B\,M\;\mathbf{of}\;\mathbf{inl}\,z \Rightarrow N \mid \mathbf{inr}\,z' \Rightarrow N' & \Longrightarrow & [M/z]N \\
\mathbf{case}\,\mathbf{inr}^A\,M'\;\mathbf{of}\;\mathbf{inl}\,z \Rightarrow N \mid \mathbf{inr}\,z' \Rightarrow N' & \Longrightarrow & [M'/z]N'
\end{array}$$

Additive Falsehood 0. This is the unit for disjunction. Since it is additive, it does not split into left and right versions.

$$\textit{(no } \mathbf{0} \textit{ introduction rule)}$$

$$\frac{\Gamma; \Delta_2; \Omega_2 \vdash M : \mathbf{0}}{\Gamma; (\Delta_1 \times \Delta_2); (\Omega_1, \Omega_2, \Omega_3) \vdash \mathbf{abort}^C\, M : C}\,\mathbf{0}\mathrm{E}$$

Since there is no introduction rule for $\mathbf{0}$, there are no new reductions.

In analogy with linear logic, we have two modal operators: one allows an ordered assumption to become mobile (while it must remain linear), another one allows a linear assumption to become intuitionistic.

Mobility Modal $¡A$.

$$\frac{\Gamma; \Delta; \cdot \vdash M : A}{\Gamma; \Delta; \cdot \vdash ¡M : ¡A}\,¡I$$

$$\frac{\Gamma; \Delta_2; \Omega_2 \vdash M : ¡A \qquad \Gamma; (\Delta_1, y{:}A); (\Omega_1, \Omega_3) \vdash N : C}{\Gamma; (\Delta_1 \times \Delta_2); (\Omega_1, \Omega_2, \Omega_3) \vdash \mathbf{let}\; ¡y = M \;\mathbf{in}\; N : C}\,¡E$$

$$\mathbf{let}\; ¡y = ¡M \;\mathbf{in}\; N \quad\Longrightarrow\quad [M/y]N$$

Linear Exponential $!A$.

$$\frac{\Gamma; \cdot; \cdot \vdash M : A}{\Gamma; \cdot; \cdot \vdash !M : !A}\,!I$$

$$\frac{\Gamma; \Delta_2; \Omega_2 \vdash M : !A \qquad (\Gamma, x{:}A); \Delta_1; (\Omega_1, \Omega_3) \vdash N : C}{\Gamma; (\Delta_1 \times \Delta_2); (\Omega_1, \Omega_2, \Omega_3) \vdash \mathbf{let}\; !x = M \;\mathbf{in}\; N : C}\,!E$$

$$\mathbf{let}\; !x = !M \;\mathbf{in}\; N \quad\Longrightarrow\quad [M/x]N$$

Bibliography

[ABCJ94] D. Albrecht, F. Bäuerle, J. N. Crossley, and J. S. Jeavons. Curry-Howard terms for linear logic. In ??, editor, *Logic Colloquium '94*, pages ??–?? ??, 1994.

[Abr90a] V. M. Abrusci. A comparison between Lambek syntactic calculus and intuitionistic linear propositional logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 36:11–15, 1990.

[Abr90b] V. M. Abrusci. Non-commutative intuitionistic linear propositional logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 36:297–318, 1990.

[Abr93] Samson Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111:3–57, 1993.

[Abr95] V. M. Abrusci. Non-commutative proof nets. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 271–296. Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.

[Bar96] Andrew Barber. Dual intuitionistic linear logic. Draft manuscript, March 1996.

[BG91] C. Brown and D. Gurr. Relations and non-commutative linear logic. Technical Report DAIMI PB-372, Computer Science Department, Aarhus University, November 1991.

[Bie94] G. Bierman. On intuitionistic linear logic. Technical Report 346, University of Cambridge, Computer Laboratory, August 1994. Revised version of PhD thesis.

[CF58] H. B. Curry and R. Feys. *Combinatory Logic*. North-Holland, Amsterdam, 1958.

[CHP97] Iliano Cervesato, Joshua S. Hodas, and Frank Pfenning. Efficient resource management for linear logic proof search. Submitted to a special issue of TCS on Proof Search in Type-Theoretic Languages. Revised version of paper in the Proceedings of the 5th International

Workshop on Extensions of Logic Programming, Leipzig, Germany, March 1996, 1997.

[Chu41] Alonzo Church. *The Calculi of Lambda-Conversion*. Princeton University Press, Princeton, New Jersey, 1941.

[CP97] Iliano Cervesato and Frank Pfenning. A linear spine calculus. Technical Report CMU-CS-97-125, Department of Computer Science, Carnegie Mellon University, April 1997.

[Cur30] H.B. Curry. Grundlagen der kombinatorischen Logik. *American Journal of Mathematics*, 52:509–536, 789–834, 1930.

[Doš93] Kosta Došen. A historical introduction to substructural logics. In Peter Schroeder-Heister and Kosta Došen, editors, *Substructural Logics*, pages 1–30. Clarendon Press, Oxford, England, 1993.

[DP95] Olivier Danvy and Frank Pfenning. The occurrence of continuation parameters in CPS terms. Technical Report CMU-CS-95-121, Department of Computer Science, Carnegie Mellon University, February 1995.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935. Translated under the title *Investigations into Logical Deductions* in [Sza69].

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[GMW79] Michael J. Gordon, Robin Milner, and Christopher P. Wadsworth. *Edinburgh LCF*. Springer-Verlag LNCS 78, 1979.

[Her30] Jacques Herbrand. Recherches sur la théorie de la démonstration. *Travaux de la Société des Sciences et de Lettres de Varsovic*, 33, 1930.

[Her95] Hugo Herbelin. *Séquents qu'on calcule*. PhD thesis, Universite Paris 7, January 1995.

[Hil22] David Hilbert. Neubegründung der Mathematik (erste Mitteilung). In *Abhandlungen aus dem mathematischen Seminar der Hamburgischen Universität*, pages 157–177, 1922. Reprinted in [Hil35].

[Hil35] David Hilbert. *Gesammelte Abhandlungen*, volume 3. Springer-Verlag, Berlin, 1935.

[HM94] Joshua Hodas and Dale Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327–365, 1994. A preliminary version appeared in the Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science, pages 32–42, Amsterdam, The Netherlands, July 1991.

[How69] W. A. Howard. The formulae-as-types notion of construction. Unpublished manuscript, 1969. Reprinted in To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, 1980.

[Hue76] Gérard Huet. *Résolution d'équations dans des langages d'ordre* $1, 2, \ldots, \omega$. PhD thesis, Université Paris VII, September 1976.

[Kni89] Kevin Knight. Unification: A multi-disciplinary survey. *ACM Computing Surveys*, 2(1):93–124, March 1989.

[Lin92] P. Lincoln. Linear logic. *ACM SIGACT Notices*, 23(2):29–37, Spring 1992.

[LS86] Joachim Lambek and Philip J. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge University Press, Cambridge, England, 1986.

[ML85a] Per Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. Technical Report 2, Scuola di Specializzazione in Logica Matematica, Dipartimento di Matematica, Università di Siena, 1985.

[ML85b] Per Martin-Löf. Truth of a proposition, evidence of a judgement, validity of a proof. Notes to a talk given at the workshop *Theory of Meaning*, Centro Fiorentino di Storia e Filosofia della Scienza, June 1985.

[ML94] Per Martin-Löf. Analytic and synthetic judgements in type theory. In Paolo Parrini, editor, *Kant and Contemporary Epistemology*, pages 87–99. Kluwer Academic Publishers, 1994.

[MM76] Alberto Martelli and Ugo Montanari. Unification in linear time and space: A structured presentation. Internal Report B76-16, Ist. di Elaborazione delle Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy, July 1976.

[MM82] Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, April 1982.

[MOM91] N. Martí-Oliet and J. Meseguer. From Petri nets to linear logic through categories: A survey. *Journal on Foundations of Computer Science*, 2(4):297–399, December 1991.

[MPP92] Dale Miller, Gordon Plotkin, and David Pym. A relevant analysis of natural deduction. Talk given at the workshop on *Types for Proofs and Programs*, Båstad, Sweden, June 1992.

[Par92] Michel Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In A. Voronkov, editor, *Proceedings of the International Conference on Logic Programming and Automated Reasoning*, pages 190–201, St. Petersburg, Russia, July 1992. Springer-Verlag LNCS 624.

[Pfe91] Frank Pfenning. Logic programming in the LF logical framework. In Gérard Huet and Gordon Plotkin, editors, *Logical Frameworks*, pages 149–181. Cambridge University Press, 1991.

[Pfe94] Frank Pfenning. Elf: A meta-language for deductive systems. In A. Bundy, editor, *Proceedings of the 12th International Conference on Automated Deduction*, pages 811–815, Nancy, France, June 1994. Springer-Verlag LNAI 814. System abstract.

[Pfe95] Frank Pfenning. Structural cut elimination. In D. Kozen, editor, *Proceedings of the Tenth Annual Symposium on Logic in Computer Science*, pages 156–166, San Diego, California, June 1995. IEEE Computer Society Press.

[Pra65] Dag Prawitz. *Natural Deduction.* Almquist & Wiksell, Stockholm, 1965.

[PW78] M. S. Paterson and M. N. Wegman. Linear unification. *Journal of Computer and System Sciences*, 16(2):158–167, April 1978.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[Rob71] J. A. Robinson. Computational logic: The unification computation. *Machine Intelligence*, 6:63–72, 1971.

[Roo92] D. Roorda. Proof nets for Lambek calculus. *Journal of Logic and Computation*, 2:211–231, 1992.

[Sce93] A. Scedrov. A brief guide to linear logic. In G. Rozenberg and A. Salomaa, editors, *Current Trends in Theoretical Computer Science*, pages 377–394. World Scientific Publishing Company, 1993. Also in Bulletin of the European Association for Theoretical Computer Science, volume 41, pages 154–165.

[SDP97] Carsten Schürmann, Jöelle Despeyroux, and Frank Pfenning. Primitive recursion for higher-order abstract syntax. Submitted to TCS. Revised version of Technical Report CMU-CS-96-172, December 1997.

[SHD93] Peter Schroeder-Heister and Kosta Došen, editors. *Substructural Logics.* Number 2 in Studies in Logic and Computation. Clarendon Press, Oxford, England, 1993.

[Sza69] M. E. Szabo, editor. *The Collected Papers of Gerhard Gentzen.* North-Holland Publishing Co., Amsterdam, 1969.

[Tro92] A. S. Troelstra. *Lectures on Linear Logic.* CSLI Lecture Notes 29, Center for the Study of Language and Information, Stanford, California, 1992.

[Tro93] A. S. Troelstra. Natural deduction for intuitionistic linear logic. Unpublished manuscript, May 1993.