Lecture Notes on Linear Logic

15-317: Constructive Logic

Frank Pfenning André Platzer

Lecture 24
November 29, 2016

In previous lectures we saw a computational interpretation of constructive proofs as functional programming as well as a computational interpretation of proof search as logical programming. While state-change can be understood indirectly in both paradigms as well, today's lecture develops a direct handle on a logical account of state change, thereby forming the basis of a logical understanding of imperative programming.

In this lecture we will examine ways of writing logic programs for state change (such as in peg solitaire) in a way that treats state logically, rather than as an explicit data structure. In order to allow this we need to generalize the logic to handle state intrinsically, something provided by *linear logic*. We provide an introduction to linear logic as a *sequent calculus*, which generalizes our previous way of specifying truth. The sequent calculus is a bit too general to allow an immediate operational interpretation to obtain a logic programming language, so we postpone this step to the next lecture.

24.1 State-Passing Style

The canonical logic programming way of representing state change is in a separate data structure that is managed manually via some custom definition:

```
activate(A, OldState, NewState) :- ......
deactivate(A, OldState, NewState) :- ......
```

And then an activation of something called `a` with a subsequent deactivation of something called `b` works like this:

```
transformation(InputState, OutputState) :-
  activate(a, InputState, State2),
  deactivate(b, State2, OutputState).
```

This pattern of code is called *state-passing* or *store-passing*.

In functional programming, the related *store-passing* style usually arises in the opposite way: if we want to turn a functional program that uses mutable storage into a pure functional program we can pass the store around as an explicit argument.

24.2 State-Dependent Truth

In a peg solitaire logic program, state would be represented as a list of items `peg(`ij`)` and `hole(`ij`)` for locations ij. Stepping back from this particular representation, it is easy to interpret `peg` and `hole` as *predicates*, and `peg(`ij`)` and `hole(`ij`)` as *propositions*. For example, we say the proposition `peg(`ij`)` is true if there is a peg in location ij on the board.

What makes this somewhat unusual, from the perspective of the logic we have considered so far, is that the notion of truth depends on the state. In some states, `peg(`ij`)` is true, in some it is false. In fact, the state of the board is completely characterized by the `peg` and `hole` propositions.

In mathematical logic, truth is normally invariant and does not depend on state. This is because the mathematical objects we deal with, such as natural numbers, are themselves invariant and considered universal. In philosophical logic, however, the concept of truth depending on the state of the world is central and has been investigated under the name *modal logic*, of which *temporal logic* is a particular branch. In temporal logic, for example, a proposition `peg(`ij`)` may be true now at the current state of the current time but may become false at a state in the future, for example when another peg jumped over it so that it turned into a hole.

Linear logic provides an elegant and logical approach, which does not suffer from the frame problem that some modal or temporal logics may suffer from.

24.3 Linear Logic by a Chemistry Example

Linear logic is a logic of state or a resource-aware logic.[1] Any logic needs to track its current assumptions in a proof carefully (just recall the two-

[1]The term *linear* is connected to its use in algebra, but the connection is not easy to explain. For this lecture just think of "*linear*" as denoting "*must be used exactly once*".

dimensional notation for natural deduction). Unlike in other logics, though, linear logic not just needs to meticulously keep track of what is true but any use of a truth will also consume that truth. It is this consumption of truth that makes it possible for linear logic to change what holds true in a well-defined way.

Before initiating a formal treatment of the connectives of linear logic, we will intuitively develop an example of some of its operators. The most crucial connective is linear implication $A \multimap B$. Just like the implication $A \supset B$ from intuitionistic propositional logic, $A \multimap B$ says something about B being true if A is true. In linear logic, though, the linear implication $A \multimap B$ *consumes* the fact that A is true to produce a fact that B is true. It is, thus, better to think of the linear implication $A \multimap B$ as saying that it can convert a resource A into a resource B, but the resource A is then consumed and gone.

Medieval alchemists were trying to solve the chemical holy grail of converting other substances into gold. For example they were trying to establish ways of turning water into gold, which would have established:

$$\mathsf{water} \multimap \mathsf{gold}$$

That is, if one has a drop of water, the above linear implication would turn it into solid gold. Such a chemical reaction would consume the water and instead produce gold. Modern chemistry discovered that that would be too good to be true, but found other interesting chemical reactions. For example that two hydrogen molecules (H_2) can react with one oxygen molecule (O_2) to yield two water molecules (and energy):

$$H_2 \otimes H_2 \otimes O_2 \multimap \mathsf{water} \otimes \mathsf{water}$$

Representing this needs the simultaneous conjunction $A \otimes B$, which is true if we simultaneously have A and B. The chemical reaction modeled by the above linear logic formula simultaneously needs two hydrogens and one oxygen, consumes all three of them, and turns that into two simultaneous water molecules.

With a nontrivial chemical process (anthraquinone), hydrogen with oxygen can also turn into hydrogen peroxide (H_2O_2):

$$H_2 \otimes O_2 \multimap \mathsf{HydrogenPeroxide}$$

By combining the latter chemical reactions, if we have two hydrogen molecules and an oxygen molecule, we could either get two water molecules or one hydrogen peroxide molecule and retain a hydrogen molecule:

$$H_2 \otimes H_2 \otimes O_2 \multimap (\textsf{water} \otimes \textsf{water}) \mathbin{\&} (\textsf{HydrogenPeroxide} \otimes H_2)$$

The alternative conjunction $A \mathbin{\&} B$ expresses that we can either obtain A or obtain B, but not both, and get to choose which one. While we can choose which side of the $\&$ we want, we cannot choose both, because the oxygen molecule can only either split into two water molecules or turn into one hydrogen peroxide molecule, not both.

The above linear logical formulas with $\&$ models the case where our experimental conditions get to determine whether we want water or hydrogen peroxide. Disjunction $A \oplus B$ would be used to model that A or B are true but we do not get to choose which one, say because chemistry may surprise us:

$$H_2 \otimes H_2 \otimes O_2 \multimap (\textsf{water} \otimes \textsf{water}) \oplus (\textsf{HydrogenPeroxide} \otimes H_2)$$

In the presence of platinum (Pt) as a catalyzer, two hydrogen peroxide molecules will split to two water molecules and an oxygen molecule freeing energy:

$$\textsf{HydrogenPeroxide} \otimes \textsf{HydrogenPeroxide} \otimes \textsf{platinum} \multimap \textsf{water} \oplus \textsf{water} \oplus O_2 \otimes \textsf{platinum}$$

In all of these cases the assumptions on the left of $\multimap$ are consumed and converted into the resources on the right of $\multimap$. Note, however, how the presence of platinum (or other catalyzers) is required to enable this catalytic reaction, but the platinum remains present. In the above linear logical formula, platinum is consumed on the left yet reproduced on the right to model that its presence is required to activate the reaction, but the platinum does not vanish.

If we would like to prevent the above reaction, we would have to get rid of all platinum by using the following explicit disposal

$$\textsf{platinum} \multimap \mathbf{1}$$

The empty truth $\mathbf{1}$ holds only if there are no resources, such that the linear implication $\textsf{platinum} \multimap \mathbf{1}$ can be used to consume one platinum molecule and convert it into no resources, thereby removing one platinum from the world (which is in fact a bit difficult to implement chemically except by having your platinum stolen).

Looking back, the above reactions are of slightly different kinds. Some of them nature provides for free, others require a lot of effort and careful experimental setup to even enable. And in fact, if linear logical formulas ought to be thought of as resources, then the above linear implications themselves are resources as well and could either be available or absent. That is what the exponential connective $!A$ is good for, which means that we can obtain any number of copies of A. With this, the following would indicate that we can always turn two hydrogen and an oxygen into two water molecules:

$$!(H_2 \otimes H_2 \otimes O_2 \multimap \mathsf{water} \otimes \mathsf{water})$$

But the following has no exponential, since we have to invest nontrivial effort to make this reaction possible:

$$H_2 \otimes O_2 \multimap \mathsf{HydrogenPeroxide}$$

The simultaneous conjunction of both would then describe the available linear implication or rewrite rules of chemical reactions in a system of relevance:

$$(H_2 \otimes O_2 \multimap \mathsf{HydrogenPeroxide}) \otimes !(H_2 \otimes H_2 \otimes O_2 \multimap \mathsf{water} \otimes \mathsf{water})$$

It should have become clear at this time that a formal understanding of linear logic should be explicit about understanding propositions as resources that are explicitly tracked in a sequent calculus while enabling explicit ways of consuming and creating resources in accordance to the meaning of the linear logical connectives.

24.4 Linear Logic

Linear logic has been described as a logic of state or a resource-aware logic. Formally, it arises from complementing the usual notion of logical assumption with so-called *linear assumptions* or *linear hypotheses*. Unlike traditional assumptions which may be used many times in a proof, linear assumptions must be used *exactly once* during a proof. Linear assumptions then become (consumable) *resources* in the course of a proof. Because linear assumptions are consumable, they can represent ephemeral truth about the current state, e.g., of a computation, because what is no longer true can be consumed and is then gone. Facts that become true can be made available as resources.

This generalization of the usual mathematical standpoint may seem slight, but as we will see it is quite expressive. We write

$$A_1\ \mathit{res}, \ldots, A_n\ \mathit{res} \Vdash C\ \mathit{true}$$

for a linear hypothetical judgment with resources $A_1, \ldots, A_n$ and goal C. If we can prove this, it means that we can achieve that C is true, given resources A_1 through A_n. Here, all A_i and C are propositions.[2] The version of linear logic defined by this judgment is called *intuitionistic linear logic*, sometimes contrasted with *classical linear logic* in which the sequent calculus has multiple conclusions. While it is possible to develop classical linear logic programming it is more difficult to understand and use.

Hidden in the judgment are other assumptions, usually abbreviated as Γ, which can be used arbitrarily often (including not at all), and are therefore called the *unrestricted assumptions*. If we need to make them explicit in a rule we will write

$$\Gamma; \Delta \Vdash C\ \mathit{true}$$

where Δ abbreviates the resources. As in our development so far, unrestricted assumption are fixed and are carried through from every conclusion to all premisses. Eventually, we will want to generalize this, but not quite yet.

The first rule of linear logic is that if we have a resource P we can achieve goal P, where P is an atomic proposition. It will be a consequence of our definitions that this will be true for arbitrary propositions A, but we need it as a rule only for the atomic case, where the structure of the propositions can not be broken down further.

$$\frac{}{P\ \mathit{res} \Vdash P\ \mathit{true}}\ \text{id}$$

We call this the *identity rule*, it is also sometimes called the *init* rule, and the sequent $P \Vdash P$ is called an *initial sequent*.

24.5 Connectives of Linear Logic

One of the curious phenomena of linear logic is that the ordinary connectives multiply. This is because the presence of linear assumptions allows us to make distinctions we ordinarily could not. The first example of this kind is conjunction. It turns out that linear logic possesses two forms of conjunction.

[2]In the end it will turn out that $A\ \mathit{res}$ and $A\ \mathit{true}$ are interchangeable in that we can go from each one to the other. At this point, however, we do not know this yet, so the judgment we make about our resources is not that they are true, but that they are given resources.

Simultaneous Conjunction ($A \otimes B$). A simultaneous conjunction $A \otimes B$ is true if we can achieve both A and B in the same state. This means we have to subdivide our resources, devoting some of them to achieve A and the others to achieve B.

$$\frac{\Delta = (\Delta_A, \Delta_B) \quad \Delta_A \Vdash A \quad \Delta_B \Vdash B}{\Delta \Vdash A \otimes B} \otimes R$$

The order of linear assumptions is irrelevant, so in $\Delta = (\Delta_A, \Delta_B)$ the comma denotes the multi-set union. In other words, every occurrence of a proposition in Δ will end up in exactly one of Δ_A and Δ_B.

If we name the initial state of peg solitaire Δ_0, then we have $\Delta_0 \Vdash \mathsf{peg}(33) \otimes \mathsf{hole}(03) \otimes \ldots$ for some "$\ldots$" because we can achieve a state with a peg at location 33 and hole at location 03. On the other hand, we cannot prove $\Delta_0 \Vdash \mathsf{peg}(33) \otimes \mathsf{hole}(33) \otimes \ldots$ because we cannot have a peg and an empty hole at location 33 in the same state. We will make the ellipsis "$\ldots$" precise below as consumptive truth $\top$.

In a linear sequent calculus, the right rules show when we can conclude a proposition. The left rules show how we can use a resource. In this case, the resource $A \otimes B$ means that we have A and B simultaneously, so the left rule reads

$$\frac{\Delta, A\ \mathit{res}, B\ \mathit{res} \Vdash C\ \mathit{true}}{\Delta, A \otimes B\ \mathit{res} \Vdash C\ \mathit{true}} \otimes L.$$

Alternative Conjunction ($A \& B$). An alternative conjunction is true if we can achieve both conjuncts, separately, with the current resources. This means if we have a linear assumption $A \& B$ we have to make a choice: either we use A or we use B, but we cannot use them both since $A\ \mathit{true}$ and $B\ \mathit{true}$ are formed from the same resources in $\& R$.

$$\frac{\Delta \Vdash A\ \mathit{true} \quad \Delta \Vdash B\ \mathit{true}}{\Delta \Vdash A \& B\ \mathit{true}} \& R$$

$$\frac{\Delta, A\ \mathit{res} \Vdash C\ \mathit{true}}{\Delta, A \& B\ \mathit{res} \Vdash C\ \mathit{true}} \& L_1 \qquad \frac{\Delta, B\ \mathit{res} \Vdash C\ \mathit{true}}{\Delta, A \& B\ \mathit{res} \Vdash C\ \mathit{true}} \& L_2$$

It looks like the right rule duplicates the assumptions, but this does not violate linearity because in a use of the assumption $A \& B\ \mathit{res}$ we have to commit to one or the other.

Returning to the solitaire example, we have $\Delta_0 \Vdash \mathsf{peg}(33) \otimes \mathsf{hole}(03) \otimes \ldots$ and we also have $\Delta_0 \Vdash \mathsf{hole}(33) \otimes \mathsf{hole}(03) \otimes \ldots$ because we can certainly reach states with these properties. However, we cannot reach a single state with both of these, because the two properties of location 33 clash. If we want to express that both are reachable, we can form their alternative conjunction

$$\Delta_0 \Vdash (\mathsf{peg}(33) \otimes \mathsf{hole}(03) \otimes \ldots) \mathbin{\&} (\mathsf{hole}(33) \otimes \mathsf{hole}(03) \ldots).$$

Consumptive Truth ($\top$). We have seen two forms of conjunction, which are distinguished because of their resource behavior. There are also two truth constants, which correspond to zero-ary conjunctions. The first is *consumptive truth* $\top$. A proof of it consumes all current resources. As such we can extract no information from its presence as an assumption.

$$\frac{}{\Delta \Vdash \top\ \mathit{true}}\ \top R \qquad\qquad \begin{array}{c}\text{no } \top L \text{ rule}\\ \Delta, \top\ \mathit{res} \Vdash C\ \mathit{true}\end{array}$$

Consumptive truth is important in applications where there is an aspect of the state we do not care about, because of the stipulation of linear logic that every linear assumption must be used *exactly once*. In the examples above so far we cared about only two locations, 33 and 03. The state will have a linear assumption for every location, which means we can *not* prove, for example, $\Delta_0 \Vdash \mathsf{peg}(33) \otimes \mathsf{hole}(03)$. However, we *can* prove $\Delta_0 \Vdash \mathsf{peg}(33) \otimes \mathsf{hole}(03) \otimes \top$, because the consumptive truth matches the remaining state.

Consumptive truth is the unit of alternative conjunction in that $A \mathbin{\&} \top$ is equivalent to A.

Empty Truth (1). The other form of truth holds only if there are no resources. If we have this as a linear hypothesis we can transform it into the empty set of resources.

$$\frac{\Delta = (\cdot)}{\Delta \Vdash \mathbf{1}\ \mathit{true}}\ \mathbf{1}R \qquad\qquad \frac{\Delta \Vdash C\ \mathit{true}}{\Delta, \mathbf{1}\ \mathit{res} \Vdash C\ \mathit{true}}\ \mathbf{1}L$$

Empty truth can be useful to dispose explicitly of specific resources (for example $D \multimap \mathbf{1}$ with the linear implication $\multimap$ discussed next would allow the disposal of resource D).

Linear Implication ($A \multimap B$). A linear implication $A \multimap B$ is true if we can achieve B given resource A.

$$\frac{\Delta, A\,\mathit{res} \Vdash B\ \mathit{true}}{\Delta \Vdash A \multimap B\ \mathit{true}} \multimap R$$

Conversely, if we have $A \multimap B$ as a resource, it means that we could transform the resource A into the resource B. We capture this in the following left rule:

$$\frac{\Delta = (\Delta_A, \Delta_B) \quad \Delta_A \Vdash A\ \mathit{true} \quad \Delta_B, B\,\mathit{res} \Vdash C\ \mathit{true}}{\Delta, A \multimap B\,\mathit{res} \Vdash C\ \mathit{true}} \multimap L.$$

An assumption $A \multimap B$ therefore represents a means to transition from a state with A to a state with B.

Unrestricted Assumptions Γ. The left rule for linear implication points at a problem: the linear implication is itself linear and therefore consumed in the application of that rule. If we want to specify via a linear logic program how state may change, we will need to reuse the clauses over and over again. This can be accomplished by a *copy* rule which takes an unrestricted assumption and makes a linear copy of it.[3]

$$\frac{A\,\mathit{ures} \in \Gamma \quad \Gamma; \Delta, A\,\mathit{res} \Vdash C\ \mathit{true}}{\Gamma; \Delta \Vdash C\ \mathit{true}} \text{copy}$$

We label the unrestricted assumptions as unrestricted resources, $A\,\mathit{ures}$. In the logic programming interpretation, the whole program will end up in Γ as unrestricted assumptions, since the program clauses can be used arbitrarily often during a computation.

Resource Independence ($!A$). The proposition $!A$ is true if we can prove A without using any resources. This means we can produce as many copies of A as we need (since it costs nothing) and a linear resource $!A$ licenses us to make the unrestricted assumption A.

$$\frac{\Gamma; \cdot \Vdash A\ \mathit{true}}{\Gamma; \cdot \Vdash\ !A\ \mathit{true}}\ !R \qquad \frac{(\Gamma, A\,\mathit{ures}); \Delta \Vdash C\ \mathit{true}}{\Gamma; \Delta, !A\,\mathit{res} \Vdash C\ \mathit{true}}\ !L$$

[3] It is actually very much like the focusing rule in a focused calculus.

A transition from a state with A to a state with B that is always allowed (as opposed to being consumed upon use) is represented by $!(A \multimap B)$. For example, $!(\mathsf{peg}(22) \multimap \mathsf{hole}(22)) \otimes \mathsf{peg}(22) \multimap \mathsf{hole}(22)$ would represent the (admittedly silly) question whether position 22 could ever have a hole on a broken peg solitaire board that has an actual hole at position 22 through which all pegs could possibly fall through any number of times.

Disjunction ($A \oplus B$). The familiar conjunction from logic was split into two connectives in linear logic: the simultaneous and the alternative conjunction. Disjunction does not split the same way unless we introduce an explicit judgment for falsehood (which we will not pursue). The goal $A \oplus B$ can be achieved if we can achieve either A or B.

$$\frac{\Delta \Vdash A\ \mathit{true}}{\Delta \Vdash A \oplus B\ \mathit{true}}\ \oplus R_1 \qquad\qquad \frac{\Delta \Vdash B\ \mathit{true}}{\Delta \Vdash A \oplus B\ \mathit{true}}\ \oplus R_2$$

Conversely, if we are given $A \oplus B$ as a resource, we do not know which of the two is true, so we have to account for both eventualities. Our proof splits into cases, and we have to show that we can achieve our goal in either case.

$$\frac{\Delta, A\ \mathit{res} \Vdash C\ \mathit{true} \quad \Delta, B\ \mathit{res} \Vdash C\ \mathit{true}}{\Delta, A \oplus B\ \mathit{res} \Vdash C\ \mathit{true}}\ \oplus L$$

Again, it might appear as if linearity is violated due to the duplication of Δ and even C. However, only one of A or B will be true, so only one part of the plan represented by the two premisses really applies, preserving linearity.

Falsehood (0). There is no way to prove falsehood $\mathbf{0}$, so there is no right rule for it. On the other hand, if we have $\mathbf{0}$ as an assumption we know we are really in an impossible state so we are permitted to succeed.

$$\begin{array}{c}\text{no } \mathbf{0}R \text{ rule}\\ \Delta \Vdash \mathbf{0}\ \mathit{true}\end{array} \qquad\qquad \frac{}{\Delta, \mathbf{0}\ \mathit{res} \Vdash C\ \mathit{true}}\ \mathbf{0}L$$

We can also formally think of falsehood as a disjunction between zero alternatives and arrive at the same rule.

24.6 Resource Management

The connectives of linear logic are generally classified into *multiplicative*, *additive*, and *exponential*.[4]

The multiplicative connectives, when their rules are read from conclusion to the premisses, split their resources between the premisses. The connectives $\otimes$, $\mathbf{1}$, and $\multimap$ have this flavor.

The additive connectives, when their rules are read from conclusion to premisses, propagate their resources to all premisses. The connectives $\&$, $\top$, $\oplus$, and $\mathbf{0}$ have this flavor.

The exponential connectives mediate the boundary between linear and non-linear reasoning. The connective ! has this flavor.

During proof search (and therefore in the logic programming setting), a significant question is how to handle the resources. It is clearly impractical, for example, in the rule

$$\frac{\Delta = (\Delta_A, \Delta_B) \quad \Delta_A \Vdash A\ \mathit{true} \quad \Delta_B \Vdash B\ \mathit{true}}{\Delta \Vdash A \otimes B\ \mathit{true}}\ \otimes R$$

to simply enumerate all possibilities and try to prove A and B in each combination until one is found that works for both.

Instead, we pass in all resources Δ into the first subgoal A and keep track which resources are consumed. We then pass the remaining ones to the proof of B. Of course, if B fails we may have to find another proof of A which consumes a different set of resources and then retry B, and so on. In the logic programming setting this is certainly an issue the programmer has to be aware of, just as the programmer has to know which subgoal is solved first, or which clause is tried first.

We will return to this question in the next lecture where we will make resource-passing explicit in the operational semantics.

24.7 Historical Notes

Linear logic in a slightly different form than we present here is due to Girard [2]. He insisted on a classical negation in his formulation, which can get in the way of an elegant logic programming formulation. The judgmental presentation we use here was developed for several courses on *Linear Logic* [3] at CMU. Some additional connectives, and some interesting connections between the two formulations in linear logic are developed by

[4]Again, we will not try to explain the mathematical origins of this terminology.

Chang, Chaudhuri and Pfenning [1]. We'll provide some references on linear logic programming in the next lecture.

24.8 Exercises

Exercise 24.1 *Prove that* $A\ \mathit{res} \Vdash A\ \mathit{true}$ *for any proposition* A.

Exercise 24.2 *For each of the following purely linear entailments, give a proof that they hold or demonstrate that they do not hold because there is no deduction in our system. You do not need to prove formally that no deduction exists.*

i. $A \mathbin{\&} (B \oplus C) \Vdash (A \mathbin{\&} B) \oplus (A \mathbin{\&} C)$

ii. $A \otimes (B \oplus C) \Vdash (A \otimes B) \oplus (A \otimes C)$

iii. $A \oplus (B \mathbin{\&} C) \Vdash (A \oplus B) \mathbin{\&} (A \oplus C)$

iv. $A \oplus (B \otimes C) \Vdash (A \oplus B) \otimes (A \oplus C)$

Exercise 24.3 *Repeat Exercise 24.2 by checking the reverse linear entailments.*

Exercise 24.4 *For each of the following purely linear entailments, give a proof that they hold or demonstrate that they do not hold because there is no deduction in our system. You do not need to prove formally that no deduction exists.*

i. $A \multimap (B \multimap C) \Vdash (A \otimes B) \multimap C$

ii. $(A \otimes B) \multimap C \Vdash A \multimap (B \multimap C)$

iii. $A \multimap (B \mathbin{\&} C) \Vdash (A \multimap B) \mathbin{\&} (A \multimap C)$

iv. $(A \multimap B) \mathbin{\&} (A \multimap C) \Vdash A \multimap (B \mathbin{\&} C)$

v. $(A \oplus B) \multimap C \Vdash (A \multimap C) \mathbin{\&} (A \multimap C)$

vi. $(A \multimap C) \mathbin{\&} (A \multimap C) \Vdash (A \oplus B) \multimap C$

Exercise 24.5 *For each of the following purely linear entailments, give a proof that they hold or demonstrate that they do not hold because there is no deduction in our system. You do not need to prove formally that no deduction exists.*

i. $C \Vdash \mathbf{1} \multimap C$

ii. $\mathbf{1} \multimap C \Vdash C$

iii. $A \multimap \top \Vdash \top$

iv. $\top \Vdash A \multimap \top$

v. $\mathbf{0} \multimap C \Vdash \top$

vi. $\top \Vdash \mathbf{0} \multimap C$

Exercise 24.6 *For each of the following purely linear entailments, give a proof that they hold or demonstrate that they do not hold because there is no deduction in our system. You do not need to prove formally that no deduction exists.*

i. $!(A \otimes B) \Vdash !A \otimes !B$

ii. $!A \otimes !B \Vdash !(A \otimes B)$

iii. $!(A \,\&\, B) \Vdash !A \otimes !B$

iv. $!A \otimes !B \Vdash !(A \,\&\, B)$

v. $!\top \Vdash \mathbf{1}$

vi. $\mathbf{1} \Vdash !\top$

vii. $!\mathbf{1} \Vdash \top$

viii. $\top \Vdash !\mathbf{1}$

ix. $!!A \Vdash !A$

x. $!A \Vdash !!A$

24.9 References

[1] Bor-Yuh Evan Chang, Kaustuv Chaudhuri, and Frank Pfenning. A judgmental analysis of linear logic. Technical Report CMU-CS-03-131R, Carnegie Mellon University, December 2003.

[2] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[3] Frank Pfenning. Linear logic. Lecture Notes for a course at Carnegie Mellon University, 1995. Revised 1998, 2001.

24.10 Appendix: Summary of Intuitionistic Linear Logic

In the rules below, we show the unrestricted assumptions Γ only where affected by the rule. In all other rules it is propagated unchanged from the conclusion to all the premisses. Also recall that the order of hypotheses is irrelevant, and Δ_A, Δ_B stands for the multiset union of two collections of linear assumptions, which are shown here in the simpler notation.

Judgmental Rules

$$\frac{}{P\ \mathit{res} \Vdash P\ \mathit{true}}\ \text{id} \qquad \frac{A\ \mathit{ures} \in \Gamma \quad \Gamma; \Delta, A\ \mathit{res} \Vdash C\ \mathit{true}}{\Gamma; \Delta \Vdash C\ \mathit{true}}\ \text{copy}$$

Multiplicative Connectives

$$\frac{\Delta_A \Vdash A \quad \Delta_B \Vdash B}{\Delta_A, \Delta_B \Vdash A \otimes B}\ {\otimes}R \qquad \frac{\Delta, A\ \mathit{res}, B\ \mathit{res} \Vdash C\ \mathit{true}}{\Delta, A \otimes B\ \mathit{res} \Vdash C\ \mathit{true}}\ {\otimes}L$$

$$\frac{}{\cdot \Vdash \mathbf{1}\ \mathit{true}}\ \mathbf{1}R \qquad \frac{\Delta \Vdash C\ \mathit{true}}{\Delta, \mathbf{1}\ \mathit{res} \Vdash C\ \mathit{true}}\ \mathbf{1}L$$

$$\frac{\Delta, A\ \mathit{res} \Vdash B\ \mathit{true}}{\Delta \Vdash A \multimap B\ \mathit{true}}\ {\multimap}R \qquad \frac{\Delta_A \Vdash A\ \mathit{true} \quad \Delta_B, B\ \mathit{res} \Vdash C\ \mathit{true}}{\Delta_A, \Delta_B, A \multimap B\ \mathit{res} \Vdash C\ \mathit{true}}\ {\multimap}L$$

Additive Connectives

$$\frac{\Delta \Vdash A\ \mathit{true} \quad \Delta \Vdash B\ \mathit{true}}{\Delta \Vdash A \,\&\, B\ \mathit{true}}\ \&R \qquad \frac{\Delta, A\ \mathit{res} \Vdash C\ \mathit{true}}{\Delta, A \,\&\, B\ \mathit{res} \Vdash C\ \mathit{true}}\ \&L_1 \qquad \frac{\Delta, B\ \mathit{res} \Vdash C\ \mathit{true}}{\Delta, A \,\&\, B\ \mathit{res} \Vdash C\ \mathit{true}}\ \&L_2$$

$$\frac{}{\Delta \Vdash \top\ \mathit{true}}\ \top R \qquad \text{no } \top L \text{ rule}$$

$$\frac{\Delta \Vdash A\ \mathit{true}}{\Delta \Vdash A \oplus B\ \mathit{true}}\ {\oplus}R_1 \qquad \frac{\Delta \Vdash B\ \mathit{true}}{\Delta \Vdash A \oplus B\ \mathit{true}}\ {\oplus}R_2 \qquad \frac{\Delta, A\ \mathit{res} \Vdash C\ \mathit{true} \quad \Delta, B\ \mathit{res} \Vdash C\ \mathit{true}}{\Delta, A \oplus B\ \mathit{res} \Vdash C\ \mathit{true}}\ {\oplus}L$$

$$\text{no } \mathbf{0}R \text{ rule} \qquad \frac{}{\Delta, \mathbf{0}\ \mathit{res} \Vdash C\ \mathit{true}}\ \mathbf{0}L$$

Exponential Connective

$$\frac{\Gamma; \cdot \Vdash A\ \mathit{true}}{\Gamma; \cdot \Vdash {!}A\ \mathit{true}}\ {!}R \qquad \frac{(\Gamma, A\ \mathit{ures}); \Delta \Vdash C\ \mathit{true}}{\Gamma; \Delta, {!}A\ \mathit{res} \Vdash C\ \mathit{true}}\ {!}L$$

Figure 1: Intuitionistic Linear Logic