In this lecture we will introduce a general approach to combining logics, using the special case of ordered and linear logic. Other examples of this construction are monads in the sense used in functional programming to integrate effects [Mog89] and LNL [Ben94] and the modal logic S4 which gives rise to a type system for quotation [DP01, PD01]. A systematic logical study was initiated by Reed [Ree09] and subsequently extended and applied to concurrency [PG15].

1 Connecting Two Layers with Shifts

We start by writing down ordered and linear logic in a way that exhibits the correspondence between the propositions. We use a subscript $O$ to indicate an ordered mode and $L$ to indicate linear mode. In a later lecture we will introduce other modes. $p_O$ and $p_L$ stand for ordered and linear propositional variables, respectively.

**Linear**

\[
A_L, B_L ::= p_L \mid A_L \oplus B_L \mid A_L \& B_L \mid 1 \mid A_L \multimap B_L \mid A_L \otimes B_L
\]

**Ordered**

\[
A_O, B_O ::= p_O \mid A_O \oplus B_O \mid A_O \& B_O \mid 1 \mid A_O \multimap B_O \mid A_O \setminus B_O \mid A_O \mathbin{/} B_O \mid A_O \bullet B_O \mid A_O \circ B_O
\]

Strictly speaking, there are also two versions of internal and external choice and the unit $1$, but since their rules and behavior are independent of whether we have ordered or unordered antecedents, so we write them the same way. At this point there is no overlap, so the two logics are not connected in any way. To allow ordered proposition to mention linear ones and vice versa, we add two *shift* operators. Since we view linear propositions as
more powerful antecedents (they can move around, after all, while the ordered ones are locked into place) we think of \( L \) as stronger than \( O \) (written \( L > O \)) and we use an uparrow to go from ordered to linear and a downarrow to go from linear to ordered. Officially, this arrow is annotated with the two modes. We will omit them in today’s lecture since \( O \) and \( L \) will be the only modes.

\[
\begin{align*}
\text{Linear} & : A_L, B_L ::= p_L \mid A_L \oplus B_L \mid A_L \& B_L \mid 1 \mid A_L \rightarrow B_L \mid A_L \otimes B_L \mid \uparrow_o A_0 \\
\text{Ordered} & : A_0, B_0 ::= p_o \mid A_0 \oplus B_0 \mid A_0 \& B_0 \mid 1 \mid A_0 \setminus B_0 \mid B_0 / A_0 \mid A_0 \cdot B_0 \mid A_0 \circ B_0 \mid \downarrow_o A_L
\end{align*}
\]

We read \( \uparrow_o A_0 \) as from ordered \( A_0 \) up to linear or just up \( A_0 \), and \( \downarrow_o A_L \) as from linear \( A_L \) down to ordered or just down \( A_L \).

## 2 Counterexample: Swap

We refer to unary logical connectives that mediate between different modes of truth as modal operators or modalities. In traditional philosophical logic such modes could be knowledge, belief, necessity, or possibility, among many others that have been investigated. Here the mode determines whether we are considering ordered or linear logic.

We need some principled understanding of the shift modalities before we can write correct left and right rules to define their meaning. I hesitate to write down wrong rules, lest they burn themselves into your memory, but it is worth playing the what if game for a little to understand the restrictions we will eventually impose. Consider the judgment

\[ \Delta_L ; \Omega_o \vdash C_m \]

where \( \Delta_L \) collects all linear antecedents, \( \Omega_o \) collects all ordered antecedents, and \( C_m \) is the succedent with the mode \( m \) either \( O \) or \( L \). The first attempt at the rules, which will be shown to be wrong, would be to just strip the modality in all four cases. We take care to place each proposition in its appropriate antecedent zone.

\[
\begin{align*}
\Delta ; \Omega \vdash A_0 & \quad \Delta ; \Omega \vdash \uparrow A_0 & \quad \Delta ; \Omega_1 A_0 \Omega_2 \vdash C_m & \quad \Delta, \uparrow A_0 ; \Omega_1 \Omega_2 \vdash C_m \\
\Delta ; \Omega \vdash A_L & \quad \Delta ; \Omega \vdash \downarrow A_L & \quad \Delta, A_L ; \Omega_1 \Omega_2 \vdash C_m & \quad \Delta ; \Omega_1 (\downarrow A_L) \Omega_2 \vdash C_m
\end{align*}
\]
Next comes the counterexample which shows that this combined logic does not make sense in the way we intended.

\[
\begin{array}{c}
\cdot \vdash \mathbf{A} \\
\cdot \vdash \mathbf{A} \uparrow R ?? \\
\cdot \vdash \mathbf{A} \mathbf{B} \vdash \mathbf{B} \cdot \mathbf{A} \uparrow L ?? \\
\cdot \vdash \mathbf{A} \cdot \mathbf{B} \vdash \mathbf{B} \cdot \mathbf{A} \cdot L
\end{array}
\]

Why is this troublesome? The conclusion proves that fuse is commutative, which means that the ordered connectives no longer have the meaning we expect. Moreover, something must fail in cut elimination: our conclusion contains only ordered propositions, and in ordered logic it is certainly not the case that fuse is commutative. There can be no cut-free proof of the endsequent in the combined logic, so cut elimination must fail.

The issue is that \( \uparrow A_0 \) is a linear proposition, which is therefore “mobile” in the sense that it can move around arbitrarily among the antecedents. If we could indeed prove \( \cdot \vdash \mathbf{A} \uparrow A_0 \) this means we could make any ordered proposition mobile, place it wherever we like (the use of \( \uparrow L ?? \) in the counterexample) when we make it ordered again. This “trick” was exploited to illegally move \( A_0 \) from the left of \( B_0 \) to its right.

Our analysis is that the proof of linear proposition \( \uparrow A_0 \) should not be allowed to depend on the ordered proposition \( A_0 \), because the linear succedent can be used in ways not justified by the ordered antecedent. This would rule out the very judgment \( \cdot \vdash \mathbf{A} \uparrow A_0 \) so the cut rule cannot be correct, nor can the application of \( \uparrow R ?? \). On the other hand, \( \uparrow L ?? \) looks defensible.

We can also take a look at what the result of a hypothetical cut reduction would be. It is a principal case, since the cut formula \( \uparrow A_0 \) is the principal formula of both inferences. We obtain:

\[
\begin{array}{c}
\cdot \vdash \mathbf{A} \\
\cdot \vdash \mathbf{A} \uparrow R ?? \\
\cdot \vdash \mathbf{A} \mathbf{B} \vdash \mathbf{B} \cdot \mathbf{A} \uparrow L ?? \\
\cdot \vdash \mathbf{A} \cdot \mathbf{B} \vdash \mathbf{B} \cdot \mathbf{A} \cdot L
\end{array}
\]
We see that the conclusion of the cut would have antecedents $B_0 A_0$ when before the cut reduction it was $A_0 B_0$. Thus, the cut reduction fails.

3 The Declaration of Independence

The considerations in the previous section lead us to the following principle of independence.

**Independence Principle:** The proof of a linear succedent cannot depend on ordered antecedents.

The general form of this principle in later lectures will be: The proof of a succedent with mode $m$ can only depend on antecedents with modes equal to or stronger than $m$. Here we consider a mode stronger if it supports more structural properties such as exchange, weakening, or contraction.

This means we really have only the following two judgment forms.

$$
\Delta_L \Omega \vdash A_0 \\
\Delta \vdash A_L
$$

Ordered succedents $A_0$ can depend on linear and ordered antecedents, while linear succedents $A_L$ can only depend on linear antecedents. We will later unify these with others into a single judgment form with some presuppositions to ensure they are meaningful.

Now we can go through our rules and subject them to the independence principle to arrive at the following collection of right and left rules.

$$
\frac{\Delta \vdash A_0}{\Delta \vdash \uparrow A_0} \uparrow R \\
\frac{\Delta, \uparrow A_0 \Omega_0 \Omega_2 \vdash C_0}{\Delta \Omega_1 \Omega_2 \vdash C_0} \uparrow L
$$

$$
\frac{\Delta \vdash A_L}{\Delta ; \cdot \vdash \downarrow A_L} \downarrow R \\
\frac{\Delta, A_L \Omega_1 \Omega_2 \vdash C_0}{\Delta ; \Omega_1 \left(\downarrow A_L\right) \Omega_2 \vdash C_0} \downarrow L
$$

Please make sure you go through these and understand how they arise from the previous dubious rules simply by heeding the independence principle. As an example, with consider $\downarrow R$. We had proposed

$$
\frac{\Delta \Omega \vdash A_L}{\Delta \Omega \vdash \downarrow A_L} \downarrow R??$$
In the premise, there cannot be any ordered antecedents, leading to a better approximation:

$$\Delta \vdash A_L \quad \Downarrow R$$

But every antecedent in $\Omega$ must be used—we cannot simply drop them at this rule which would be tantamount to weakening. So we must require there to be no ordered antecedents and we arrive at the correct rule.

$$\Delta \vdash A_L \quad \Downarrow R$$

4 Cut and Identity

The rules for cut and identity can now be derived from the same considerations of independence. Identity is particularly straightforward.

$$\cdot \vdash A_0 \quad \text{id}_0 \\
A_L \vdash A_L \quad \text{id}_L$$

For cut, it turns out there are three rules because independence rules out one of the four combinations of the judgment forms.

$$\Delta' ; \Omega' \vdash A_0 \quad \Delta ; \Omega_1 \ A_0 \ \Omega_2 \vdash C_0 \\
\Delta, \Delta' ; \Omega_1' \ \Omega' \ \Omega_2' \vdash C_0 \quad \text{cut}_{oo}$$

$$\Delta' \vdash A_L \quad \Delta, A_L \ ; \Omega \vdash C_0 \\
\Delta, \Delta' ; \Omega \vdash C_0 \quad \text{cut}_{lo}$$

$$\Delta' \vdash A_L \quad \Delta, A_L \vdash C_L \\
\Delta, \Delta' \vdash C_L \quad \text{cut}_{ll}$$

5 Other Propositional Rules

The other propositional rules are carried over or straightforwardly adapted from ordered or linear logic. We show only two examples: the rules for $A_0 \setminus B_0$ and $A_L \rightarrow B_L$. The $\rightarrow L$ rules split into two, for similar reasons why there are two version of the cut rule for linear propositions.

$$\Delta ; A_0 \ \Omega \vdash B_0 \\
\Delta ; \Omega \vdash A_0 \ \setminus R$$

$$\Delta' ; \Omega' \vdash A_0 \quad \Delta ; \Omega_1 \ B_0 \ \Omega_2 \vdash C_0 \\
\Delta, \Delta' ; \Omega_1' \ (A_0 \ \setminus B_0) \ \Omega_2 \vdash C_0 \quad \setminus L$$

$$\Delta, A_L \vdash B_L \\
\Delta \vdash A_L \rightarrow B_L \ \rightarrow R$$

$$\Delta' \vdash A_L \quad \Delta, B_L \vdash C_L \\
\Delta, \Delta', A_L \rightarrow B_L \vdash C_L \ \rightarrow L_L$$

$$\Delta' \vdash A_L \quad \Delta, B_L ; \Omega \vdash C_0 \\
\Delta, \Delta', A_L \rightarrow B_L ; \Omega \vdash C_0 \ \rightarrow L_O$$
6 Roundtrips

Including shifts in the language of propositions allows us to consider round trips from ordered to linear logic and back, and from ordered to linear and back. How does ↓↑\(A_0\) compare to \(A_0\)? Conversely, how does ↑↓\(A_L\) compare to \(A_L\)? We just try construct a cut-free derivation, bottom up, for each of these proposition to see which hold and which do not.

\[
\begin{align*}
\text{no rule applicable} & \quad \cdot; A_0 \vdash ↓↑A_0 \\
\cdot; A_0 \vdash ↑↓A_0 & \quad \frac{\cdot; A_0 \vdash ↑R}{↑A_0; \cdot \vdash A_0 \vdash ↓L}
\end{align*}
\]

\[
\begin{align*}
\frac{A_L \vdash A_L \vdash ↑R}{↑↓A_L \vdash ↑R} & \quad \frac{↑↓A_L \vdash ↑R}{↑↓A_L \vdash ↑R} \quad \frac{\cdot; A_0 \vdash A_0 \vdash ↑R}{↑A_0; \cdot \vdash A_0 \vdash ↓L}
\end{align*}
\]

In two sequents we get stuck immediately due to the restrictions imposed by the independence principle.

So if we know \(A_L\) we can conclude ↑↓\(A_L\) via a roundtrip to an ordered proposition but not the other way around. Conversely, if we are trying to prove \(A_0\) it is sufficient to prove ↓↑\(A_0\).

The observations

\[
\begin{align*}
A_0 & \not\vdash ↓↑A_0 \\
↓↑A_0 & \vdash A_0 \\
\not\vdash ↑↓A_L \\
↑↑A_L & \not\vdash A_L
\end{align*}
\]

suggest that ↓↑\(A_0\) ⊍ \(A_0\) behaves like the modality of necessity, which can be characterized categorically as a comonad, while ↑↓\(A_0\) ⊌ \(A_L\) behaves like a form of possibility, which can be characterized categorically as a strong monad.

We pursue this conjecture a little further to see if the shifts distribute over implications. They do. While we distribute the shift, linear implications will turn into over/under and vice versa. We prefer \(A \setminus B\) here since the order of its arguments is consistent with linear implication, but this
choice is arbitrary. We start

\[
\begin{align*}
\vdash \downarrow (A \rightarrow B) & \quad \downarrow B \\
\vdash \downarrow (A \rightarrow B) & \quad \downarrow A \quad \downarrow B \quad R \times 2
\end{align*}
\]

We pause here to note that the \( \downarrow R \) rule is not applicable since the ordered context is not empty. So we apply the \( \downarrow L \) rules twice, to empty it out. The remainder is straightforward.

\[
\begin{align*}
A, A \rightarrow B, B \quad \vdash \downarrow B \\
A, A \rightarrow B, B \quad \vdash \downarrow B \\
\vdash A \quad \vdash B \\
\vdash A \quad \vdash B
\end{align*}
\]

Similarly, \( \uparrow \) distributes over ordered implication \( A \rightarrow B \). We pause after two steps.

\[
\begin{align*}
\vdash \uparrow (A \rightarrow B) \quad \uparrow A \rightarrow \uparrow B \\
\vdash \uparrow (A \rightarrow B) \quad \uparrow A \rightarrow \uparrow B
\end{align*}
\]

Now we need to apply \( \uparrow R \) so we can then apply \( \uparrow L \) twice to place the two antecedents in the ordered zone in the correct order.

\[
\begin{align*}
A \vdash A \\
B \vdash B \\
\vdash A \rightarrow B \\
\vdash A \rightarrow B
\end{align*}
\]

The observations

\[
\begin{align*}
\vdash \downarrow (A \rightarrow B) \downarrow A \downarrow B \\
\vdash \uparrow (A \rightarrow B) \uparrow A \uparrow B
\end{align*}
\]

provide further evidence for our conjectures of the modal nature of \( \downarrow \uparrow \) and \( \uparrow \downarrow \) and necessity and strong possibility (see Exercises 1 and 2).
7 Operational Interpretation of Shifts

When we assign processes as proof terms we mark channels ordered or linear, written as $x_0$ or $x_l$. This doesn’t affect the identity of a channel, only how it is used.

\[(x_{1l} : A_{1l}, \ldots, x_{nl} : A_{nl}) ; (y_{1o} : B_{1o}, \ldots, y_{ko} : B_{ko}) \vdash P :: (z_0 : C_0)\]
\[(x_{1l} : A_{1l}, \ldots, x_{nl} : A_{nl}) \vdash P :: (z_l : C_l)\]

As we remarked several times, the operational interpretation of ordered and their corresponding linear connectives is exactly the same. Order only imposes a restriction on the use of ordered channels. Therefore we could assume that the shifts have no essential operational purpose. Indeed, looking at the rules it seems that a message of shift corresponds to communicating a mutual agreement that a channel changes status, from ordered to linear or vice versa. We have to determine which of the rules carries information and which does not.

\[\Delta ; \cdot \vdash A_0 \quad \Delta \vdash \uparrow A_0 \quad \uparrow R\]
\[\Delta \vdash A_0 \quad \Delta, \uparrow A_0 \vdash C_0 \quad \Delta \vdash \uparrow L\]

We see that $\uparrow R$ can always be applied and thus carries no information, while an antecedent $\uparrow A_0$ may have to wait until the succedent is ordered. It also implicitly carries the information on where in the context to insert $A_0$. This means $\uparrow R$ will receive and $\uparrow L$ will send.

\[\Delta ; \cdot \vdash P :: (x_0 : A_0) \quad \Delta \vdash \text{shift} \leftarrow \text{recv } x_l ; P :: (x_l : \uparrow A_0) \quad \uparrow R\]
\[\Delta, x_l : \uparrow A_0 ; \Omega_1, \Omega_2 \vdash Q :: (z_0 : C_0) \quad \Delta \vdash \text{send } x_l \text{ shift} ; Q :: (z_0 : C_0) \quad \uparrow L\]

For the down shift modality, the roles are reversed. $\downarrow R$ does not always apply since there may be ordered antecedents. On the other hand, $\downarrow L$ always applies (read bottom-up, of course) and therefore it carries no information and will receive.

\[\Delta \vdash P :: (x_l : A_l) \quad \Delta ; \cdot \vdash \text{send } x_0 \text{ shift} ; P :: (x_0 : \downarrow A_l) \quad \downarrow R\]
\[\Delta, x_l : A_l ; \Omega_1, \Omega_2 \vdash Q :: (z_0 : C_0) \quad \Delta, x_l : \downarrow A_l ; \Omega_1, \Omega_2 \vdash \text{recv } x_0 ; Q :: (z_0 : C_0) \quad \downarrow L\]
As with internal/external choice and ordered implication/fuse, the up and down shifts form dual pairs and therefore use the same program construct. We can always tell from the situation which of the two is meant.

The operational rules are straightforward.

\[
\begin{array}{c}
\frac{\text{proc}(x_l, \text{shift} \leftarrow \text{recv } x_l ; P) \quad \text{proc}(z_0, \text{send } x_l \text{ shift} ; Q)}{	ext{proc}(x_l, P) \quad \text{proc}(z_0, Q)} \quad \uparrow C \\
\frac{\text{proc}(x_0, \text{send } x_0 \text{ shift} ; P) \quad \text{proc}(z_0, \text{shift} \leftarrow \text{recv } x_0 ; Q)}{	ext{proc}(x_0, P) \quad \text{proc}(z_0, Q)} \quad \downarrow C
\end{array}
\]

8 Example: A Linear Client Using an Ordered Service

By the independence principle, a process that provides along a linear channel cannot use an ordered channel. But if we have an ordered process that uses linear channels internally, we can use an ordered storage server (say, a stack) to store the linear channel \(x_l:A_l\). In order to allow this we have to coerce the linear channel to be ordered, of type \(\downarrow A_l\), so we define

\[
x_l:A_l ; \vdash \text{down :: (} y_0 : \downarrow A_l \text{)}
\]

\[
y_0 \leftarrow \text{down} \leftarrow x_l =
\text{send } y_0 \text{ shift ;}
\]

\[
y_l \leftarrow x_l
\]

The we assume we have ordered type

\[
\text{stack}_A = \&\{ \text{ins} : A \setminus \text{stack}_A \\
\quad \text{del} : \oplus\{ \text{none} : 1, \text{some} : A \bullet \text{stack} \}
\}
\]

with the usual ordered implementation. Now assume we are in a situation where we own a channel of type \(A_l\) and also a stack with elements of type \(\downarrow A_l\)

\[
(x_l:A_l) ; (s:\text{stack}_l\downarrow A_l) \vdash (z_0 : C_0)
\]

In order to push \(x_l\) onto the stack we first lower it to be ordered. Upon retrieval, we do the opposite. In the types we omit the succedent since it plays no role in this code.

\[
w_0 \leftarrow \text{down} \leftarrow x_l =
\text{send } w_0 \text{ shift ;}
\]

\[
s.\text{ins} ; \text{send } s \ w_0 ;
\]

\[
\ldots \text{ other operations} \ldots
\]
Combining Logics L12.10

$s\text{.del} ; \% \cdot s : \oplus \{\text{none} : 1, \text{some} : \downarrow A_L \cdot \text{stack}_L\} \vdash \ldots$
case $s$ (none $\Rightarrow$ wait $s$ ; \ldots
| some $\Rightarrow u_0 \leftarrow \text{recv} s$ ; \% \cdot (u_0 : \downarrow A_L) (s : \text{stack}_L) \vdash \ldots$
  shift $\leftarrow \text{recv} u_0$ ; \% \cdot (u_L : A_L) ; (s : \text{stack}_L) \vdash \ldots$

9 Identity Expansion
To verify harmony we should check cut reduction and identity expansion. We begin with the simpler identity expansion.

$\cdot ; A_0 \vdash A_0 \quad \text{id}_o$
$\downarrow A_0 ; \cdot \vdash A_0 \quad \text{id}_l$
$\downarrow A_0 \vdash \downarrow A_0 \quad \text{cut}_L$
$\uparrow A_0 \vdash \uparrow A_0 \quad \text{cut}_R$

10 Cut Reduction
Cut reduction usually has the key insight in the proof of admissibility of cut and therefore ultimately cut elimination. We only show one case of cut reduction (for $\uparrow A_0$) and elide other cases as well as admissibility of cut (see also Exercise 3).
Assume the first premise of a cut ends in $\uparrow R$ rule.

$\Delta', \cdot \vdash A_0 \quad \text{cut}_{L}$
$\Delta', \uparrow A_0 \quad \Delta, \uparrow A_0 ; \Omega \vdash C_0$
$\Delta, \Delta' ; \Omega \vdash C_0 \quad \text{cut}_{iO}$

The $\uparrow L$ rule has the form

$\Delta ; \Omega_1 A_0 \Omega_2 \vdash C_0 \quad \uparrow L$
$\Delta, \uparrow A_0 ; \Omega_1 \Omega_2 \vdash C_0 \quad \text{cut}_{iL}$

so only the first of these two cases will result in a cut reduction. The second case above will lead to some commuting conversions, if $A_0$ is a side formula
of inferences in $\mathcal{E}$. The only case for reduction then has the form

$\frac{\mathcal{D}' \vdash A_0}{\Delta' \vdash \uparrow A_0} \uparrow R \quad \frac{\mathcal{E}' \vdash \Delta ; \Omega_1 A_0 \Omega_2 \vdash C_0}{\Delta, \Delta' ; \Omega \vdash \uparrow C_0} \uparrow L$

$\text{cut}_{\text{io}}$

which reduces to

$\frac{\mathcal{D}' \vdash A_0 \quad \mathcal{E}' \vdash \Delta ; \Omega_1 A_0 \Omega_2 \vdash C_0}{\Delta, \Delta' ; \Omega \vdash \uparrow C_0} \text{cut}_{\text{oo}}$

**Exercises**

**Exercise 1** Define $\Box A_0 = \downarrow \uparrow A_0$. Taking the small liberty of omitting the linear zone when it is empty, we have already seen

1. $\Box A_0 \vdash A_0$
2. $A_0 \not\vdash \Box A_0$

Prove or refute each of the following:

1. $\Box A_0 \vdash \Box \Box A_0$
2. $\Box (A_0 \setminus B_0) (\Box A_0) \vdash \Box B_0$

**Exercise 2** Define $\Diamond A_l = \uparrow \downarrow A_l$. We have already seen

1. $A_l \vdash \Diamond A_l$
2. $\Diamond A_l \not\vdash A_l$

Proof or refute each of the following

1. $\Diamond \Diamond A_l \vdash \Diamond A_l$
2. $\Diamond (A_l \rightarrow B_l), \Diamond A_l \vdash \Diamond B_l$

**Exercise 3** Show all cases of cut reduction for $\downarrow A_l$. 

**Lecture Notes** October 6, 2016
Exercise 4 Recall

\[ x : A \vdash \text{down} :: (y : \downarrow A) \]

from Section 8. A priori, there are four interesting possibilities for such coercions. For each, show that it cannot exist or write a process that implements it.

References


