Lecture Notes on Computational Interpretations of Modalities

15-816: Modal Logic
Frank Pfenning

Lecture 4
January 21, 2010

1 Introduction

In this lecture we present the first two of many possible computational interpretations of intuitionistic modal logic. The first interprets the type $\Box A$ as *source code of type A* and relates proof reduction to staged computation [DP01]. For the second we analyze possibility, $\Diamond A$, and relate it to computational effects [PD01].

2 Staged Computation

Consider for the moment the function $\mathsf{exp} : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$ where $\mathsf{exp}\; n\; b = b^n$. We give two different definitions on unary natural numbers.

$$\begin{array}{lcl}
\mathsf{exp}\,(0) & = & \lambda b.\, 1 \\
\mathsf{exp}\,(\mathsf{s}(n)) & = & \lambda b.\, b * \mathsf{exp}\, n\, b \\
\\
\mathsf{exp}'\,(0) & = & \lambda b.\, 1 \\
\mathsf{exp}'\,(\mathsf{s}(n)) & = & \mathbf{let}\ f = \mathsf{exp}'\, n\ \mathbf{in}\ \lambda b.\, b * f\, b
\end{array}$$

These are both correct implementations, but their operational behavior is very different. The first one, when given the argument 2, performs just one step of computation and returns a function. In an operational semantics based on closures, it would return a closure; here we just carry out the substitutions. We write $M \longrightarrow M'$ for computational reduction which is based on local reduction but extended by certain congruences, and $M \Longrightarrow_R M'$ for reductions which can be carried out on any subterm.

$$\mathsf{exp}\,(\mathsf{s}(\mathsf{s}(0))) \quad \longrightarrow^* \quad \lambda b_2.\, b_2 * \mathsf{exp}\,(\mathsf{s}(0))\, b_2$$

The second one actually recurses all the way to 0, returning a function that does not depend on the first argument any more.

$$\begin{array}{lll} \mathsf{exp}'\,(\mathsf{s}(\mathsf{s}(0))) & \longrightarrow^* & \lambda b_2.\, b_2 * (\lambda b_1.\, b_1 * (\lambda b_0.\, 1)\, b_1)\, b_2 \\ & \Longrightarrow_R^* & \lambda b_2.\, b_2 * (b_2 * 1) \end{array}$$

We can see that the second version does a lot more computation than the first. However, if the resulting function is applied many times, to many different bases, then the second can be more efficient, especially if the indicated optimization had been applied.

The difference between the two programs is not the ultimately computed answer, but the way the computation is *staged*. In the second version, all computation depending on the first argument is carried out, while in the first version this is not the case.

We would like to capture the difference between these two functions *in the type system*. The goal is to devise a type system so that we can prescribe a type under which the first program fails to type-check because it does not carry out all computation induced by its first argument, but which allows (essentially) the second version. The computational interpretation of necessity presented in the next section, achieves exactly that!

3 Operational Semantics

In order to show how necessity is related to staged computation, we will be a bit more formal about the operational interpretation of proof terms. A simple definition is a so-called *small-step semantics* which applies local reductions according to a fixed evaluation discipline. We also need the notion of a *value*. For concreteness, we work with a call-by-value functional language although, as mentioned before, the logical underpinnings do not force this.

In the formal treatment we only consider $A \wedge B$, $A \supset B$, and $\top$. This can be extended to $A \vee B$ and $\bot$ (see Exercise 1). First, the definition of value, using a judgment M *value*.

$$\frac{M\ \mathit{value} \quad N\ \mathit{value}}{\langle M, N\rangle\ \mathit{value}}\ \wedge V \qquad \frac{}{\langle\,\rangle\ \mathit{value}}\ \top V \qquad \frac{}{\lambda x{:}A.\, M\ \mathit{value}}\ \supset V$$

The notion of a value is related to the notion of a verification, as can be seen from the cases for conjunction and truth. However, we do not look inside functions — they are treated entirely *extensionally*. We cannot analyze their structure, we can only apply them to arguments and observe their behavior. Unlike verifications, this means that functional values of type $A \supset B$ may refer to types much bigger than A or B. It is therefore a considerable challenge to use an opaque notion of function as the foundation for logic or type theory.

The following reduction and congruence rules allow us to evaluate a closed term to a value by repeated reduction, according to a call-by-value discipline. In the rule names, we use C to indicate a constructor for a type and D a destructor.

$$\frac{M\ \mathit{value}}{(\lambda x{:}A.\,N)\,M \longrightarrow [M/x]N}\ {\supset}DC$$

$$\frac{M\ \mathit{value} \quad N\ \mathit{value}}{\pi_1\,\langle M, N\rangle \longrightarrow M}\ {\wedge}DC_1 \qquad \frac{M\ \mathit{value} \quad N\ \mathit{value}}{\pi_2\,\langle M, N\rangle \longrightarrow N}\ {\wedge}DC_2$$

$$\frac{M \longrightarrow M'}{\langle M, N\rangle \longrightarrow \langle M', N\rangle}\ {\wedge}C_1 \qquad \frac{M\ \mathit{value} \quad N \longrightarrow N'}{\langle M, N\rangle \longrightarrow \langle M, N'\rangle}\ {\wedge}C_2$$

$$\frac{M \longrightarrow M'}{\pi_1\,M \longrightarrow \pi_1\,M'}\ {\wedge}D_1 \qquad \frac{M \longrightarrow M'}{\pi_2\,M \longrightarrow \pi_2\,M'}\ {\wedge}D_2$$

$$\text{no } {\supset}C \text{ rule for } \lambda x.\,M \qquad \frac{M \longrightarrow M'}{M\,N \longrightarrow M'\,N}\ {\supset}D_1 \qquad \frac{M\ \mathit{value} \quad N \longrightarrow N'}{M\,N \longrightarrow M\,N'}\ {\supset}D_2$$

Now we have preservation and progress property. The are formulated only on closed terms because, unlike the process of proof reduction, we only evaluate expressions that are closed.

Theorem 1 (Type Preservation) *If* $\bullet \vdash M : A$ *and* $M \longrightarrow M'$ *then* $\bullet \vdash M' : A$.

Proof: By induction on the derivation of $M \longrightarrow M'$, using subject reduction for the case of a redex. □

Theorem 2 (Progress) *If* $\bullet \vdash M : A$ *then either*

(i) M *value, or*

(ii) $M \longrightarrow M'$ *for some* M'.

Proof: By induction on the structure of M, applying inversion in each case to obtain types for the subterms. □

We also have a termination property for the purely logical language, without recursion. We will not prove this property here.

Theorem 3 (Termination) *If* $\bullet \vdash M : A$ *then there exists a term* V *such that* V *value and* $M \longrightarrow^* V$.

4 Source Expressions

Now we extend the computational interpretation sketched above to encompass the necessity modality $\Box A$. The interpretation will go as follows:

$x{:}A$ *true*	x stands for a value of type A
$u{::}A$ *valid*	u stands for a source expression of type A
$[\![M/u]\!]N$	substitute the source expression M for u in N
$\mathbf{box}\, M$	M is a quoted source expression
$\mathbf{let\,box}\, u = M \;\mathbf{in}\; N$	evaluate M to a quoted source expression $\mathbf{box}\, M'$ and then evaluate $[\![M'/u]\!]N$.

We have one new kind of value: a quoted source expression.

$$\frac{}{\mathbf{box}\, M \ \mathit{value}}\ \Box V$$

We also have one new reduction and one congruence rule.

$$\frac{M \longrightarrow M'}{\mathbf{let\,box}\, u = M \;\mathbf{in}\; N \longrightarrow \mathbf{let\,box}\, u = M' \;\mathbf{in}\; N}\ \Box D$$

$$\frac{}{\mathbf{let\,box}\, u = \mathbf{box}\, M \;\mathbf{in}\; N \longrightarrow [\![M/u]\!]N}\ \Box DC$$

The crucial restriction of the typing rules ensures that in an expression $\mathbf{box}\, M$, the term M does not reference any free variables x that stand for values. It can, however, mention variables u that stand for source expressions. So when we substitute $[\![N/u]\!]\mathbf{box}\, M$ then we are building a larger source expression from two smaller ones, N and M. Conversely, when we substitute a value $[V/x]\mathbf{box}\, M = \mathbf{box}\, M$ the source expression is not affected.

We observe how this plays out in the two versions of the exponential function from Section 2. We now try to enforce proper staging, specifying

$$\mathsf{exp} : \mathsf{nat} \rightarrow \Box(\mathsf{nat} \rightarrow \mathsf{nat})$$

where we have identified implication ($\supset$) with the function type construction ($\rightarrow$). The specification means that the exponential function takes a natural number n as first argument and produces a source expression which computes $\lambda b{:}\mathsf{nat}.\, b^n$. The $\Box$ indicates that this expression can not refer directly to the first argument. Rewriting the first example, we obtain the following implementation

$$\begin{array}{lcl} \mathsf{exp}\,(0) & = & \mathbf{box}\,\lambda b.\, 1 \\ \mathsf{exp}\,(\mathsf{s}(n)) & = & \mathbf{box}\,\lambda b.\, b * \mathsf{exp}\, n\, b \\ \\ \not\vdash \mathsf{exp} & : & \mathsf{nat} \rightarrow \Box(\mathsf{nat} \rightarrow \mathsf{nat}) \end{array}$$

which is *not* well-typed because underneath the **box** in the second line we have a reference to n (which is a value variable, and not an expression variable) which should not be visible. One can also debate if exp itself should be visible or not, but the expression is ill-typed in either case.

The second version, however, can easily be rewritten to have the specified type

$$\begin{array}{lcl} \mathsf{exp}'\,(0) & = & \lambda b.\, 1 \\ \mathsf{exp}'\,(\mathsf{s}(n)) & = & \mathbf{let\,box}\, f = \mathsf{exp}'\, n\ \mathbf{in}\ \lambda b.\, b * f\, b \\ \\ \vdash \mathsf{exp}' & : & \mathsf{nat} \rightarrow \Box(\mathsf{nat} \rightarrow \mathsf{nat}) \end{array}$$

We can now revisit the characteristic axioms and read off their computational interpretation.

$$\begin{array}{lcl} \mathsf{eval} & = & \lambda x{:}\Box A.\, \mathbf{let\,box}\, u = x\ \mathbf{in}\ u \\ & : & \Box A \supset A \end{array}$$

eval M evaluates M, which must denote a quoted source expression $\mathbf{box}\, M'$. It then proceeds to evaluate M'. In a system for runtime code generation, it would first compile M' and then jump to the generated code [WLPD98]. In reality, languages with runtime code generation rarely keep around source expressions. Instead, the type $\mathbf{box}\, M : \Box A$ is compiled to a code generator which, when called, will produce compiled code for M.

$$\begin{array}{rcl} \mathsf{quote} & = & \lambda x{:}\Box A.\, \textbf{let}\,\textbf{box}\,u = x\ \textbf{in}\ \textbf{box}\,\textbf{box}\,u \\ & : & \Box A \supset \Box\Box A \end{array}$$

$\mathsf{quote}\, M$ evaluates M, which must denote a quoted source expression $\textbf{box}\, M'$. It returns that result itself as a quoted source expression ($\textbf{box}\,(\textbf{box}\, M')$).

$$\begin{array}{rcl} \mathsf{apply} & = & \lambda x{:}\Box(A \supset B).\, \lambda y{:}\Box A.\, \textbf{let}\,\textbf{box}\,u = x\ \textbf{in}\ \textbf{let}\,\textbf{box}\,w = y\ \textbf{in}\ \textbf{box}\,u\,w \\ & : & \Box(A \supset B) \supset \Box A \supset \Box B \end{array}$$

$\mathsf{apply}\, M\, N$ evaluates M and N which must denoted quoted expressions $\textbf{box}\, M'$ and $\textbf{box}\, N'$. It then builds and returns a new source expression $\textbf{box}\,(M'\, N')$.

There are a number of common examples that are used to illustrate run-time code generation. Here are the types of some and their meaning in terms of staged computation.

$\mathsf{match} : \mathsf{regexp} \to \Box(\mathsf{string} \to \mathsf{bool})$**.** This specifies a function which, given a regular expression, returns code that will match strings against that particular expression. For example, the match function might construct code implementing a finite automaton from the given regular expression.

$\mathsf{parse} : \mathsf{grammar} \to \Box(\mathsf{string} \to \mathsf{ast})$**.** This specifies a program which, given a grammar, returns code that will parse strings into abstract syntax trees. This is usually called a *parser generator*.

$\mathsf{interpret} : \mathsf{program} \to \Box(\mathsf{input} \to \mathsf{output})$**.** This specifies a function which, given a program, returns code that will run the given program on inputs to produce outputs. Such a function is usually called a *compiler*. In the special case of a self-interpreter, that is, an interpreter for a language written in itself, we can have an input of type $\Box A$, but it is difficult to write the interpreter without further language constructs because there is no way to analyze expressions of type $\Box A$ within the present language.

$\mathsf{mmult} : \mathsf{matrix} \to \Box(\mathsf{matrix} \to \mathsf{matrix})$**.** This specifies a function which, given a matrix, returns code that will multiple the first matrix with a second. This is interesting because with optimizations during the code generation phase, we can achieve the efficiency of specialized algorithms for sparse matrix multiplication using the ordinary program using three nested loops [LL96].

In all these case, by staying within the language itself, we can avoid having to generate textual representations of the program which is not only error-prone, but inefficient since the text has to be parsed, type-checked, compiled, and then executed. The type system presented above influenced the type system for MetaOCaml[1], although the syntax is based on a different presentation of modal logic to which we will return later in the course.

5 Possibility

Most modal logics support at least one other modality besides necessity, namely *possibility*, written $\Diamond A$. In classical modal logics, we can say that A is possible if its negation is not necessary, that is, $\Diamond A = \neg\Box\neg A$. In intuitionistic modal logics this is not the case, because we want explicit evidence for the possibility of A, rather than just checking that its negation is impossible.

From the perspectives of the multiple-worlds interpretation of modal logics, *any* proposition A is possible. For example, if we assume the A is true, then A is clearly possible because it is in fact true in any world satisfying the hypotheses. In order to capture the intuition that anything is possible, we just need to be careful to track the assumptions under which we reason, and we need a new judgment A *poss*. If A is true then it is clearly possible.

$$\frac{\Delta; \Gamma \vdash A\ \mathit{true}}{\Delta; \Gamma \vdash A\ \mathit{poss}}\ \mathsf{poss}$$

This rule is part of the judgmental definition of possibility, so it is a judgmental rule rather than an introduction or elimination rule. No propositional connective is involved.

The substitution principle for possibility is somewhat complex. Assume we have a proof that A *poss*. This does not imply A's truth, so we cannot substitute this proof for an assumption A *true*. However, we know that A must be true in *some* world. So let's place ourselves in that world, assuming A is true but nothing else. Now if we conclude that C is possible from this assumption, we know that C must indeed be possible.

Substitution Principle for Possibility, v.1: If $\Gamma \vdash A\ \mathit{poss}$ and $x{:}A\ \mathit{true} \vdash C\ \mathit{poss}$ then $\Gamma \vdash C\ \mathit{poss}$.

[1] `http://www.metaocaml.org`

It would be incorrect to allow the conclusion $C\ \mathit{true}$ because we only know that A is possible, not that A is true (see Exercise 2). Since we are adding possibility to a system which already has necessity, we have to consider the interaction with assumptions about validity. Hypotheses $A\,\mathit{valid}$ are assumed to be true in all worlds, including the one were A is true. These assumptions therefore carry over and we obtain:

> **Substitution Principle for Possibility, v.2**: If $\Delta; \Gamma \vdash A\ \mathit{poss}$ and $\Delta; x{:}A\,\mathit{true} \vdash C\ \mathit{poss}$ then $\Delta; \Gamma \vdash C\ \mathit{poss}$.

If the first deduction is $\mathcal{D}$ and the second $\mathcal{E}$, we write $\langle\!\langle \mathcal{D}/x \rangle\!\rangle \mathcal{E}$ for the deduction resulting from the application of the substitution principle.

Internalizing possibility as a propositional operator is straightforward, once its judgmental basis has been understood.

$$\dfrac{\Delta; \Gamma \vdash A\ \mathit{poss}}{\Delta; \Gamma \vdash \Diamond A\ \mathit{true}}\ \Diamond I$$

$$\dfrac{\Delta; \Gamma \vdash \Diamond A\ \mathit{true} \qquad \Delta; x{:}A\,\mathit{true} \vdash C\ \mathit{poss}}{\Delta; \Gamma \vdash C\ \mathit{poss}}\ \Diamond E$$

These rules are locally sound and complete.

$$\dfrac{\dfrac{\overset{\mathcal{D}}{\Delta; \Gamma \vdash A\ \mathit{poss}}}{\Delta; \Gamma \vdash \Diamond A\ \mathit{true}}\ \Diamond I \qquad \overset{\mathcal{E}}{\Delta; x{:}A\,\mathit{true} \vdash C\ \mathit{poss}}}{\Delta; \Gamma \vdash C\ \mathit{poss}}\ \Diamond E \quad \Longrightarrow_R \quad \overset{\langle\!\langle \mathcal{D}/x \rangle\!\rangle \mathcal{E}}{\Delta; \Gamma \vdash C\ \mathit{poss}}$$

$$\overset{\mathcal{D}}{\Delta; \Gamma \vdash \Diamond A\ \mathit{true}} \quad \Longrightarrow_E \quad \dfrac{\dfrac{\overset{\mathcal{D}}{\Delta; \Gamma \vdash \Diamond A\ \mathit{true}} \qquad \dfrac{\dfrac{}{\Delta; x{:}A\,\mathit{true} \vdash A\,\mathit{true}}\ x}{\Delta; x{:}A\,\mathit{true} \vdash A\ \mathit{poss}}\ \mathsf{poss}}{\Delta; \Gamma \vdash A\ \mathit{poss}}\ \Diamond E}{\Delta; \Gamma \vdash \Diamond A\ \mathit{true}}\ \Diamond I$$

We can now write out some of the characteristic axioms. We abbreviate assumptions $x{:}A\,\mathit{true}$ as $x{:}A$, and similarly for $u{::}A\,\mathit{valid}$.

$A \supset \Diamond A$. Truth is stronger than possibility. The opposite implication does not hold in general, otherwise truth and possibility would collapse.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\cdot; x{:}A \vdash A\ \mathit{true}}\,x}{\cdot; x{:}A \vdash A\ \mathit{poss}}\,\mathsf{poss}}{\cdot; x{:}A \vdash \Diamond A\ \mathit{true}}\,\Diamond I}{\cdot; \cdot \vdash A \supset \Diamond A\ \mathit{true}}\,{\supset}I$$

$\Diamond\Diamond A \supset \Diamond A$. If we iterate possibility, we obtain an equivalent proposition. In other words, $\Diamond$ is idempotent as a modal operator, just as $\Box$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\cdot; x{:}\Diamond\Diamond A \vdash \Diamond\Diamond A\ \mathit{true}}\,x \quad \dfrac{\dfrac{}{\cdot; y{:}\Diamond A \vdash \Diamond A\ \mathit{true}}\,y \quad \dfrac{\dfrac{}{\cdot; z{:}A \vdash A\ \mathit{true}}\,z}{\cdot; z{:}A \vdash A\ \mathit{poss}}\,\mathsf{poss}}{\cdot; y{:}\Diamond A \vdash A\ \mathit{poss}}\,\Diamond E}{\cdot; x{:}\Diamond\Diamond A \vdash A\ \mathit{poss}}\,\Diamond E}{\cdot; x{:}\Diamond\Diamond A \vdash \Diamond A\ \mathit{true}}\,\Diamond I}{\cdot; \cdot \vdash \Diamond\Diamond A \supset \Diamond A\ \mathit{true}}\,{\supset}I$$

$\Box(A \supset B) \supset \Diamond A \supset \Diamond B$. Finally we have to consider the interaction of necessity and possibility. We cannot prove $(A \supset B) \supset \Diamond A \supset \Diamond B$ because $A \supset B$ may not hold in world at which A is true (and where we hope B would be true). So the implication must be true in all worlds, that is, valid.

$$\dfrac{\dfrac{\dfrac{}{\cdot; x{:}\Box(A \supset B), - \vdash \Box(A \supset B)\ \mathit{true}}\,x \quad \dfrac{\dfrac{\dfrac{}{-; y{:}\Diamond A \vdash \Diamond A\ \mathit{true}}\,y \quad \dfrac{\dfrac{\dfrac{}{u{::}A \supset B; - \vdash A \supset B\ \mathit{true}}\,u \quad \dfrac{}{-; z{:}A \vdash A\ \mathit{true}}\,z}{u{::}A \supset B; z{:}A \vdash B\ \mathit{true}}\,{\supset}E}{u{::}A \supset B; z{:}A \vdash B\ \mathit{poss}}\,\mathsf{poss}}{u{::}A \supset B; y{:}\Diamond A \vdash B\ \mathit{poss}}\,\Diamond E}{u{::}A \supset B; y{:}\Diamond A \vdash \Diamond B\ \mathit{true}}\,\Diamond I}{\cdot; x{:}\Box(A \supset B), y{:}\Diamond A \vdash \Diamond B\ \mathit{true}}\,\Box E}{\cdot; \cdot \vdash \Box(A \supset B) \supset \Diamond A \supset \Diamond B\ \mathit{true}}\,{\supset}I \times 2$$

6 Verifications and Uses

It is easy to generate verifications and uses. We write $A{\cdot}\!\!\uparrow\!\!{\cdot}$ if there is a verification of A *poss*.

$$\frac{\Delta; \Gamma \vdash A \uparrow}{\Delta; \Gamma \vdash A \cdot\!\!\uparrow\!\!\cdot} \text{ poss}$$

$$\frac{\Delta; \Gamma \vdash A \cdot\!\!\uparrow\!\!\cdot}{\Delta; \Gamma \vdash \Diamond A \uparrow} \Diamond I$$

$$\frac{\Delta; \Gamma \vdash \Diamond A \downarrow \quad \Delta; x{:}A\, \mathit{true} \vdash C \cdot\!\!\uparrow\!\!\cdot}{\Delta; \Gamma \vdash C \cdot\!\!\uparrow\!\!\cdot} \Diamond E$$

7 Proof Terms

In order to discuss the computational interpretation, we assign proof terms to possibility. We see that we have a new judgment A *poss* in the conclusion, but not as a hypothesis. For validity, we had a new form of variable, so now we have a new kind of term which is evidence for A *poss*. We write $E \div A$ *poss*, abbreviated as $E \div A$, to indicate that E is a proof term of the possibility of A. We call these *proof expressions* and use letters E and F for proof expressions. First, the judgmental rule to establish possibility from truth.

$$\frac{\Delta; \Gamma \vdash M : A\ \mathit{true}}{\Delta; \Gamma \vdash M \div A\ \mathit{poss}} \text{ poss}$$

This means any proof term M is also a proof expression. We do not write any explicit proof constructor because this rule does not change a proposition, only the judgments.

The introduction and elimination rules:

$$\frac{\Delta; \Gamma \vdash E \div A\ \mathit{poss}}{\Delta; \Gamma \vdash \mathbf{dia}\, E : \Diamond A\ \mathit{true}} \Diamond I$$

$$\frac{\Delta; \Gamma \vdash M : \Diamond A\ \mathit{true} \quad \Delta; x{:}A\, \mathit{true} \vdash E \div C\ \mathit{poss}}{\Delta; \Gamma \vdash \mathbf{let\, dia}\, x = M\ \mathbf{in}\ E \div C\ \mathit{poss}} \Diamond E$$

We see that the new **let dia** $x = M$ **in** E construct is itself a proof expression. Now reconsider the new substitution principle, with proof expressions:

Substitution Principle for Possibility, v.3: If $\Delta;\Gamma \vdash E \div A\ \mathit{poss}$ and $\Delta; x{:}A\ \mathit{true} \vdash F \div C\ \mathit{poss}$ then $\Delta;\Gamma \vdash \langle\!\langle E/x\rangle\!\rangle F \div C\ \mathit{poss}$.

We need to define the substitution operation $\langle\!\langle E/x\rangle\!\rangle F$ on proof expressions. Here we encounter a surprise. Consider the case where F is infered by the possibility rule, so F is actually a term N:

$$\frac{\Delta; x{:}A\ \mathit{true} \vdash N : C\ \mathit{true}}{\Delta; x{:}A\ \mathit{true} \vdash N \div C\ \mathit{poss}}\ \mathsf{poss}$$

In this case we cannot substitute $\langle\!\langle E/x\rangle\!\rangle N$ by traversing N, because N establishes truth, rather than possibility and we have already discussed that such a substitution principle is not correct. Solution: we analyze the structure of E instead of F when defining $\langle\!\langle E/x\rangle\!\rangle F$! Eventually, E will be inferred by the possibility rule, since that is the only way to establish possibility, eventually, and at that point we can revert to an ordinary substitution!

$$\begin{array}{lcl}
\langle\!\langle \mathbf{let\,dia}\, x = M\ \mathbf{in}\ E/x\rangle\!\rangle F & = & \mathbf{let\,dia}\, x = M\ \mathbf{in}\ \langle\!\langle E/x\rangle\!\rangle F \\
\langle\!\langle M/x\rangle\!\rangle F & = & [M/x]^e F \\[4pt]
[M/x]^e(\mathbf{let\,dia}\, y = N\ \mathbf{in}\ F) & = & \mathbf{let\,dia}\, y = [M/x]N\ \mathbf{in}\ F \\
[M/x]^e(N) & = & [M/x]N
\end{array}$$

Here we write $[M/x]^e F$ for an ordinary substitution of a term for a variable in an expression. We may omit the superscript because of the last equation. Note that we do not substitute into the body of a **let dia** construct, because x cannot occur there.

Why does this terminate? The substitution $\langle\!\langle E/x\rangle\!\rangle F$ terminates, because we can first prove that $[M/x]N$ and $[M/x]^e F$ terminate, by induction on N and F, and then apply an induction on the structure of E to show that $\langle\!\langle E/x\rangle\!\rangle F$ terminates.

The substitution $[\![M/u]\!]N$ is extended compositionally to expressions $[\![M/u]\!]^e F$ and has to descend into all subterms and subexpressions since valid variables u may occur anywhere.

Finally, we can express the local reductions and expansions for possibility. For the reduction, we see that we actually reduce one proof expression to another, rather than a proof term.

$$\begin{array}{lcl}
\mathbf{let\,dia}\, x = \mathbf{dia}\, E\ \mathbf{in}\ F & \Longrightarrow_R & \langle\!\langle E/x\rangle\!\rangle F \\
M : \Diamond A & \Longrightarrow_E & \mathbf{dia}\,(\mathbf{let\,dia}\, x = M\ \mathbf{in}\ x)
\end{array}$$

8 Generalized Elimination Rules

When writing the elimination rule for disjunction, falsehood, and later necessity, we wrote, for example,

$$\frac{\Gamma \vdash A \lor B\ \mathit{true} \quad \Gamma, x{:}A\ \mathit{true} \vdash C\ \mathit{true} \quad \Gamma, y{:}B\ \mathit{true} \vdash C\ \mathit{true}}{\Gamma \vdash C\ \mathit{true}} \lor E$$

The justification for limiting the conclusion to $C\ \mathit{true}$ is that we were considering only one judgment at that time. Even in the version with verifications and uses, we are globally interested only in establishing a verification of $C\uparrow$. Even when we introduced validity, we only used it as a hypothesis so we did not need to generalize the elimination rules.

Now, however, we also have the judgment of possibility so we also need

$$\frac{\Delta; \Gamma \vdash A \lor B\ \mathit{true} \quad \Delta; \Gamma, x{:}A\ \mathit{true} \vdash C\ \mathit{poss} \quad \Delta; \Gamma, y{:}B\ \mathit{true} \vdash C\ \mathit{poss}}{\Delta; \Gamma \vdash C\ \mathit{poss}} \lor E$$

and similarly for $\bot E$ and $\Box E$.

In the proof term calculus, this means that we have proof expressions **let box** $u = M$ **in** E, in addition to the proof terms **let box** $u = M$ **in** N (and analogously for **case** and **abort**).

In future systems of natural deduction we will have occasion to recall that the elimination rule really should be

$$\frac{\Delta; \Gamma \vdash A \lor B\ \mathit{true} \quad \Delta; \Gamma, x{:}A\ \mathit{true} \vdash J \quad \Delta; \Gamma, y{:}B\ \mathit{true} \vdash J}{\Delta; \Gamma \vdash J} \lor E$$

except that we do not want to write rule that tries to anticipate all possible judgments. Instead, we analyze which instances of this generalized elimination form is necessary in the inference system at hand.

9 Monads and Computational Effects

The general idea is to think of expressions $E \div A\ \mathit{poss}$ as effectful computations, possibly returning a value of type A. We abstract away from any particular effects, such a non-termination, store, exceptions, etc. and isolate the principles that apply to all effects. Then $\Diamond A$ is a pure term, evaluating a value **dia** E, where E is a potentially effectful computation of type A. We revisit the earlier characteristic axioms in this light.

$A \supset \Diamond A$. The proof term $\lambda x{:}A.\, \textbf{dia}\, x$ simply coerces a value to a trivial computation.

$\Diamond\Diamond A \supset \Diamond A$. The proof term

$$\lambda x{:}\Diamond\Diamond A.\, \textbf{dia}\,(\textbf{let dia}\, y = x \textbf{ in let dia}\, z = y \textbf{ in } z)$$

takes an expression whose evaluation may have an effect and possibly return another expression whose evaluation may have a second effect and sequences the two computations, return the value of the latter.

$\Box(A \supset B) \supset \Diamond A \supset \Diamond B$. The proof term

$$\begin{array}{l} \lambda f{:}\Box(A \supset B).\, \lambda x{:}\Diamond A..\\ \quad \textbf{let box}\, g = f \textbf{ in dia}\,(\textbf{let dia}\, y = x \textbf{ in } g\, y) \end{array}$$

takes a (pure) function and an argument which may have an effect, and returns a computation which first evaluates the argument (possibly executing the effects) and then applies the function to the returned value. The $\Box$ around the function's type guarantees that the function "survives" the computational effect that may occur when the argument is evaluated.

In order to make this somewhat more precise, we extend the operational semantics from earlier in the lecture. On terms, this is quite easy because we consider **dia** as protecting the expression underneath from evaluation.

$$\frac{}{\textbf{dia}\, E \textit{ value}}\ \Diamond V$$

We also have a new evaluation for expressions.

$$\frac{M \longrightarrow M'}{\textbf{let dia}\, x = M \textbf{ in } F \longrightarrow \textbf{let dia}\, x = M' \textbf{ in } F}\ \Diamond D_1$$

$$\frac{M \longrightarrow M'}{\textbf{let box}\, u = M \textbf{ in } F \longrightarrow \textbf{let box}\, u = M' \textbf{ in } F}\ \Box D_1$$

$$\frac{}{\textbf{let dia}\, x = \textbf{dia}\, E \textbf{ in } F \longrightarrow \langle\!\langle E/x \rangle\!\rangle F}\ \Diamond CD$$

To interpret that last rule it is important to keep in mind that the substitution $\langle\!\langle E/x \rangle\!\rangle F$ recurses over the structure of E, thereby incurring any effects in E before those latent in F.

As a particular kind of effect, we consider output. We have a new primitive expression $\mathsf{out}\, n \div \mathsf{nat}$ that prints a natural number and also returns it, and a library function

$$\begin{array}{lcl} \mathsf{write} & : & \mathsf{nat} \rightarrow \Diamond\mathsf{nat} \\ & = & \lambda x{:}\mathsf{nat}.\, \mathbf{dia}\, (\mathsf{out}\, x) \end{array}$$

Then

$$\begin{array}{lcl} \mathsf{upto} & : & \mathsf{nat} \rightarrow \Diamond\mathsf{nat} \\ \mathsf{upto}\,(0) & = & \mathsf{write}\,(0) \\ \mathsf{upto}\,(\mathsf{s}(n)) & = & \mathbf{dia}\,(\mathbf{let\,dia}\, x = \mathsf{upto}\, n\ \mathbf{in}\ \mathbf{let\,dia}\, y = \mathsf{write}\,(\mathsf{s}(x))\ \mathbf{in}\ y) \end{array}$$

This encoding is somewhat awkward, because no ordinary variable can occur in the body of a **let dia** construct. In functional languages we therefore use not just a so-called monad (like $\Diamond$) but a *strong monad*. This integrates into the type system the observation that all values are stable under effects. It is fruitful and interesting to develop the logical foundations of monads using *lax logic* [FM97], but we can also explain in the context of necessity and possibility. In order to embed the ML function space $A \overset{ML}{\rightarrow} B$ we interpret is as $\Box A' \supset \Diamond\Box B'$, where A' and B' are the translations of of A and B, respectively. More details, including an operational interpretation of lax logic in terms of strong monads can be found in [PD01].

10 Rule Summary

Natural Deduction. Here, γ stands for C *true* or C *poss*.

$$\frac{x{:}A\ \mathit{true} \in \Gamma}{\Delta; \Gamma \vdash A\ \mathit{true}}\ x$$

$$\frac{\Delta; \Gamma, x{:}A\ \mathit{true} \vdash B\ \mathit{true}}{\Delta; \Gamma \vdash A \supset B\ \mathit{true}} \supset\! I \qquad \frac{\Delta; \Gamma \vdash A \supset B\ \mathit{true} \quad \Delta; \Gamma \vdash A\ \mathit{true}}{\Delta; \Gamma \vdash B\ \mathit{true}} \supset\! E$$

$$\frac{u{::}A\ \mathit{valid} \in \Delta}{\Delta; \Gamma \vdash A\ \mathit{true}}\ u$$

$$\frac{\Delta; \bullet \vdash A\ \mathit{true}}{\Delta; \Gamma \vdash \Box A\ \mathit{true}} \Box I \qquad \frac{\Delta; \Gamma \vdash \Box A\ \mathit{true} \quad (\Delta, u{::}A\ \mathit{valid}); \Gamma \vdash \gamma}{\Delta; \Gamma \vdash \gamma} \Box E$$

$$\frac{\Delta; \Gamma \vdash A\ \mathit{true}}{\Delta; \Gamma \vdash A\ \mathit{poss}}\ \mathsf{poss}$$

$$\frac{\Delta; \Gamma \vdash A\ \mathit{poss}}{\Delta; \Gamma \vdash \Diamond A\ \mathit{true}} \Diamond I \qquad \frac{\Delta; \Gamma \vdash \Diamond A\ \mathit{true} \quad \Delta; x{:}A\ \mathit{true} \vdash C\ \mathit{poss}}{\Delta; \Gamma \vdash C\ \mathit{poss}} \Diamond E$$

Substitution Operations. The rules for substitution $[M/x]N$, $[M/x]^e F$, and $\langle\!\langle E/x \rangle\!\rangle F$. Substitution for a valid variable $[\![M/u]\!]$ is compositional on all proof terms and expressions, avoiding capture in the body of **let box**.

$$\begin{array}{lcl}
[M/x]x & = & M \\
[M/x]y & = & y \quad \text{for } x \neq y \\
[M/x](\lambda y{:}A.\, N) & = & \lambda y{:}A.\, [M/x]N \quad \text{provided } x \neq y,\, x \notin FV(M) \\
[M/x](N_1\, N_2) & = & ([M/x]N_1)\, ([M/x]N_2) \\
[M/x](\mathbf{box}\, N) & = & \mathbf{box}\, N \\
[M/x](\mathbf{let\, box}\, u = N_1\ \mathbf{in}\ N_2) & = & \mathbf{let\, box}\, u = [M/x]N_1\ \mathbf{in}\ [M/x]N_2 \\
[M/x](\mathbf{dia}\, E) & = & \mathbf{dia}\, ([M/x]^e E) \\
\\
[M/x]^e(\mathbf{let\, dia}\, y = N\ \mathbf{in}\ F) & = & \mathbf{let\, dia}\, y = [M/x]N\ \mathbf{in}\ F \\
[M/x]^e(\mathbf{let\, box}\, u = N\ \mathbf{in}\ F) & = & \mathbf{let\, box}\, u = [M/x]N\ \mathbf{in}\ [M/x]^e F \\
[M/x]^e N & = & [M/x]N \\
\\
\langle\!\langle \mathbf{let\, dia}\, y = N\ \mathbf{in}\ E/x \rangle\!\rangle F & = & \mathbf{let\, dia}\, y = N\ \mathbf{in}\ \langle\!\langle E/x \rangle\!\rangle F \\
\langle\!\langle \mathbf{let\, box}\, u = N\ \mathbf{in}\ E/x \rangle\!\rangle F & = & \mathbf{let\, box}\, u = N\ \mathbf{in}\ \langle\!\langle E/x \rangle\!\rangle F \\
\langle\!\langle M/x \rangle\!\rangle F & = & [M/x]^e F
\end{array}$$

Verifications and Uses. Here, γ stands for $C \uparrow$ or $C \cdot\!\!\uparrow\!\!\cdot$ and P stands for an atomic proposition or propositional variable.

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

$$\frac{x{:}A \downarrow \in \Gamma}{\Delta; \Gamma \vdash A \downarrow} x$$

$$\frac{\Delta; \Gamma, x{:}A \downarrow \vdash B \uparrow}{\Delta; \Gamma \vdash A \supset B \uparrow} \supset\! I \qquad \frac{\Delta; \Gamma \vdash A \supset B \downarrow \quad \Delta; \Gamma \vdash A \uparrow}{\Delta; \Gamma \vdash B \downarrow} \supset\! E$$

$$\frac{u{::}A \Downarrow \in \Delta}{\Delta; \Gamma \vdash A \downarrow} u$$

$$\frac{\Delta; \bullet \vdash A \uparrow}{\Delta; \Gamma \vdash \Box A \uparrow} \Box I \qquad \frac{\Delta; \Gamma \vdash \Box A \downarrow \quad (\Delta, u{::}A \Downarrow); \Gamma \vdash \gamma}{\Delta; \Gamma \vdash \gamma} \Box E$$

$$\frac{\Delta; \Gamma \vdash A \uparrow}{\Delta; \Gamma \vdash A \cdot\!\!\uparrow\!\!\cdot} \mathsf{poss}$$

$$\frac{\Delta; \Gamma \vdash A \cdot\!\!\uparrow\!\!\cdot}{\Delta; \Gamma \vdash \Diamond A \uparrow} \Diamond I \qquad \frac{\Delta; \Gamma \vdash \Diamond A \downarrow \quad \Delta; x{:}A \downarrow \vdash C \cdot\!\!\uparrow\!\!\cdot}{\Delta; \Gamma \vdash C \cdot\!\!\uparrow\!\!\cdot} \Diamond E$$

Exercises

Exercise 1 *Add disjunction and falsehood to the development of possibility in this lecture.*

(i) Give all versions of introduction and elimination rules.

(ii) Show local soundness and completeness.

(iii) Give a proof term assignment and show local reductions and expansions.

(iv) Extend the definition of substitution $\langle\!\langle E/x \rangle\!\rangle F$ *as needed.*

(v) Derive interaction laws for $\Diamond(A \vee B)$ *and* $\Diamond\bot$ *if they exist or argue that they do not. You may assume that verifications are complete for truth and possibility.*

Exercise 2 *Demonstrate that*

> ***Incorrect Substitution Principle:*** *If* $\Delta; \Gamma \vdash A$ *poss and* $\Delta; x{:}A$ *true* $\vdash$ C *true then* $\Delta; \Gamma \vdash C$ *true*

in indeed incorrect. Which property fails? Give a counterexample.

Exercise 3 *An alternative proposal for possibility might be the following pair of rules which avoids the use of a new judgment A poss.*

$$\frac{\Delta; \Gamma \vdash A \textit{ true}}{\Delta; \Gamma \vdash \Diamond A \textit{ true}}\ \Diamond I \qquad \frac{\Delta; \Gamma \vdash \Diamond A \textit{ true} \quad \Delta; x{:}A \textit{ true} \vdash \Diamond C \textit{ true}}{\Delta; \Gamma \vdash \Diamond C \textit{ true}}\ \Diamond E$$

Discuss which desirable properties of the deductive system we have given in this lecture fail for these alternative rules, if any.

Exercise 4 *Characterize the propositions such that* $\Diamond A \equiv A$.

Exercise 5 *Possibility is also a general construction, and we can create weaker and weaker versions of possibility. Define a hierarchy of possibility operators* $\Diamond^n$ *such that* $\Diamond^0 A \equiv A$ *and* $\Diamond^1 A$ *is like the current* $\Diamond A$. *In general,* $\Diamond^{n+1} A$ *should be weaker then* $\Diamond^n A$.

Give a uniform system of natural deduction for iterated posibility, including introduction and eliminations for $\Diamond^n A$ *for arbitrary* n. *State the substitution principle and show local soundness and completeness.*

How do iterated $\Diamond^n$ *operators interact with each other? Is there always a simpler proposition equivalent to* $\Diamond^n \Diamond^m A$?

Exercise 6 *The justification of the elimination rules suggested the extended versions of $\vee E$, $\bot E$ and $\Box E$ with a conclusion of C poss for proofs and C $\cdot\!\!\uparrow\!\!\cdot$ for verifications.*

Investigate the consequences of omitting these rules. Which desirable properties of the resulting system of deduction fail, if any?

Exercise 7 *Both derived judgments of modal logic, A valid and A poss are asymmetric in that the former is only allowed as a hypothesis and the latter only as a conclusion.*

Write out the fragment with implication, necessity, and possibility where the new judgments are allowed on both sides. Which properties of the system presented in this lecture fail, if any? Discuss the trade-offs between the two systems.

Exercise 8 *In (non-modal) intuitionistic logic, if we define two different connectives with the same introduction and elimination rules they will be logically equivalent.*

(i) Demonstrate this by defining a new implication $A \supset' B$ with the same rules as $A \supset B$ and then proving their equivalence.

(ii) Define a second version $\Box' A$ with the same rules as $\Box A$. Can we prove that $\Box' A \equiv \Box A$?

(iii) Does your answer to (ii) change if we define a new judgment A valid$'$ based on the same judgmental principles as A valid and then internalize it as a new connective $\Box' A$?

(iv) Define a second version $\Diamond' A$ with the same rules as $\Diamond A$. Can we prove that $\Diamond' A \equiv \Diamond A$?

(v) Does your answer to (iv) change if we define a new judgment A poss$'$ based on the same judgmental principles as A poss and then internalize it as a new connective $\Diamond' A$?

Exercise 9 *A generalized modality is any string of modal operators $\Box$ and $\Diamond$. In the fragment of intuitionistic modal logic with just $\Box$, there is just one distinct modal operator, because $\Box\Box A \equiv \Box A$. In the fragment with just $\Diamond$, there is also just one distinct modal operator, because $\Diamond\Diamond A \equiv \Diamond A$. How many distinct generalized modalities are there in intuitionistic modal logic with both $\Box$ and $\Diamond$? Prove your answer.*

References

[DP01] Rowan Davies and Frank Pfenning. A modal analysis of staged computation. *Journal of the ACM*, 48(3):555–604, May 2001.

[FM97] M. Fairtlough and M.V. Mendler. Propositional lax logic. *Information and Computation*, 137(1):1–33, August 1997.

[LL96] Peter Lee and Mark Leone. Optimizing ML with run-time code generation. In *Proceedings of the Conference on Programming Language Design and Implementation (PLDI'96)*, pages 137–148, Philadelphia, Pennsylvania, May 1996. ACM SIGPLAN.

[PD01] Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11:511–540, 2001. Notes to an invited talk at the *Workshop on Intuitionistic Modal Logics and Applications* (IMLA'99), Trento, Italy, July 1999.

[WLPD98] Philip Wickline, Peter Lee, Frank Pfenning, and Rowan Davies. Modal types as staging specifications for run-time code generation. *ACM Computing Surveys*, 30(3es), September 1998.