Towards Modal Type Theory

Frank Pfenning

Constructivism in Non-Classical Logics and Computer Science
In Memoriam Pierangelo Miglioli
Mantova, Italy
October 2, 2000

Disclaimer: Work in Progress!

Acknowledgments: Alberto Momigliano

Motivation: Programming

- *Intensionality* in programming.
- Logical foundations for
  - run-time code generation [Davies & Pf.'96]
  - partial evaluation [Davies'96]
  - meta-programming [Moggi, Taha, Benaissa & Sheard'99]
- Propositional logic and simple types.

Motivation: Type Theory

- *Reasoning* about programs.
- Specifications via dependent types.
- Extracting programs from constructive proofs
  or reconstructing proof obligations from programs.
- First-order logic and dependent types.

Combining Modal Logic and Type Theory

- Judgmental analysis of modal types and programs.
- Reasoning about programs with intensionality.
- Constructive first-order modal logic and beyond.
- Unexpected result: computational irrelevance.

This Talk

1. Judgmental Reconstruction of Modal Logic (S4)
2. Specifications via Dependent Types
3. Past Worlds and Hidden Variables
4. Dependent Types Revisited
5. Erasure Interpretation
6. Conclusion

Martin-Löf Type Theory

- Judgment — object of knowledge
- Evident judgment — something we know
- Proof — evidence for a judgment
- Proposition — meaning given by rules of verification
- True proposition — possesses verification

Hypothetical Judgments

- Basic judgment: A *true* "A *is true*"
- Hypothetical judgment:

$$\underbrace{A_1 \textit{ true}, \ldots, A_n \textit{ true}}_{\text{assumptions } \Gamma} \vdash A \textit{ true}$$

- Hypothesis rule: $\Gamma_1, A \textit{ true}, \Gamma_2 \vdash A \textit{ true}$
- Substitution property:

  If $\Gamma \vdash A$ *true*
  and Γ, A *true* $\vdash C$ *true*
  then $\Gamma \vdash C$ *true.*

- Implication introduction and elimination as usual (internalizes hypothetical reasoning as a proposition).

Analytic Judgments

- Basic judgment: $M : A$ "*M is a proof term for A*"
  "*M has type A*"
- Parametric and hypothetical judgment:

$$\underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\text{value variables }\Gamma} \vdash M : A$$

- Hypothesis rule: $\Gamma_1, x{:}A, \Gamma_2 \vdash x : A$
- Substitution property:

  If $\Gamma \vdash M : A$
  and $\Gamma, x{:}A \vdash N : C$
  then $\Gamma \vdash [M/x]N : C$.

- λ-abstraction, application, β-reduction as usual.

Categorical Judgments

- Categorical judgment: A *valid* if $\bullet \vdash A$ *true*

- Hypothetical judgment:

$$\underbrace{B_1\ \textit{valid}, \ldots, B_k\ \textit{valid}}_{\text{true in all worlds}}\ ;\ \underbrace{A_1\ \textit{true}, \ldots, A_n\ \textit{true}}_{\text{in current world}} \vdash C\ \textit{true}$$

- Additional hypothesis rule: $(\Delta_1, A\ \textit{valid}, \Delta_2); \Gamma \vdash A\ \textit{true}$

- Additional substitution property:

If $\Delta; \bullet \vdash A$ *true*
and $(\Delta, A\ \textit{valid}); \Gamma \vdash C$ *true*
then $\Delta; \Gamma \vdash C$ *true.*

Necessity

- Proposition: $\Box A$ *prop* "A *is necessarily true*"
- Introduction:

$$\frac{\Delta; \bullet \vdash A \; \textit{true}}{\Delta; \Gamma \vdash \Box A \; \textit{true}} \; \Box I$$

- Elimination:

$$\frac{\Delta; \Gamma \vdash \Box A \; \textit{true} \qquad (\Delta, A \; \textit{valid}); \Gamma \vdash C \; \textit{true}}{\Delta; \Gamma \vdash C \; \textit{true}} \; \Box E$$

- Internalizes validity judgment as proposition.
- No structural rules required.

Expressions

- Categorical judgment: $M :: A$ if $\bullet \vdash M : A$.
- $M :: A$ "M *is an expression of type* A"
- Hypothetical judgment:

$$\underbrace{u_1{::}B_1, \ldots, u_k{::}B_k}_{\text{expression variables}} \; ; \; \underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\text{value variables}} \vdash N : C$$

- Additional hypothesis rule: $(\Delta_1, u{::}A, \Delta_2); \Gamma \vdash u : A$
- Substitution property:

  If $\Delta; \bullet \vdash M : A$
  and $(\Delta, u{::}A); \Gamma \vdash N : C$
  then $\Delta; \Gamma \vdash [M/u]N : C$.

Intensional Types

- Type: $\Box A$ "*terms denoting expressions of type* A"
- Decorate introduction and elimination with proof terms.
- Constructor:

$$\frac{\Delta; \bullet \vdash M : A}{\Delta; \Gamma \vdash \mathsf{box}\, M : \Box A}\ \Box I$$

- Destructor:

$$\frac{\Delta; \Gamma \vdash M : \Box A \qquad (\Delta, u{::}A); \Gamma \vdash N : C}{\Delta; \Gamma \vdash \mathsf{let}\,\mathsf{box}\, u = M\ \mathsf{in}\ N\, \mathsf{end} : C}\ \Box E$$

- Reduction:

$$\mathsf{let}\,\mathsf{box}\, u = \mathsf{box}\, M\ \mathsf{in}\ N\, \mathsf{end} \longrightarrow [M/u]N$$

Some Properties

- Type preservation and progress [Davies & Pf'96].
- Staging interpretation via multiple world semantics.
- Curry-Howard isomorphism with modal logic S4.
- Possible to add possibility ($\Diamond$) [Pf & Davies'00].
- Interpretation of lax logic ($\bigcirc A = \Diamond\Box A$) [Fairtlough & Mendler'97].
- Interpretation of monadic meta-language [Moggi'89].

Programming Language Design Philosophy

- Put the power of staging into the hands of the programmer.
- Simple and declarative.
- Conservative extension of ML based on modal types.
- Safe in the presence of effects.
- Staging errors are type errors.
- Enables, but does not prescribe optimizations.
- Compiler implementation in progress (PML).
- Some experiments in run-time code generation:
  Fabius [Leone & Lee'94'96]
  CCAM [Wickline, Lee & Pf.'98]

Example: Power Function

- Unstaged $\vdash \mathit{power} : \mathsf{nat} \to (\mathsf{nat} \to \mathsf{nat})$

```
fun power 0 b = 1
  | power (s n) b = b * (power n b)
```

- Incorrectly staged $\not\vdash \mathit{power} : \mathsf{nat} \to \Box(\mathsf{nat} \to \mathsf{nat})$

```
fun power 0 = box (λb. 1)
  | power (s n) = box (λb. b * (power n) b)
```

- Correctly staged $\vdash \mathit{power} : \mathsf{nat} \to \Box(\mathsf{nat} \to \mathsf{nat})$

```
fun power 0 = box (λb. 1)
  | power (s n) =
    let box u = power n
    in box (λb. b * u b) end
```

Examples: Laws of S4

- Curry-Howard isomorphism applied to modal logic S4.

- $\vdash \mathit{eval} : \Box A \rightarrow A$

```
fun eval x = let box u = x in u end
```

- $\vdash \mathit{apply} : \Box(A \rightarrow B) \rightarrow \Box A \rightarrow \Box B$

```
fun apply f x =
    let box f' = f
     in let box x' = x
         in box (f' x') end end
```

- $\vdash \mathit{quote} : \Box A \rightarrow \Box\Box A$

```
fun quote x = let box u = x in box (box u) end
```

Specifications via Dependent Types

- Dependent function type $\Pi x{:}A.\, B(x)$ ($\sim A \rightarrow B$) ($\sim \forall x.\, B(x)$)
- Dependent sum type $\Sigma x{:}A.\, B(x)$ ($\sim A \times B$) ($\sim \exists x.\, B(x)$)
- Propositions $m \doteq n$ ($\sim$ proof) as type of proofs.
- Specification of *power* function:

$$power : \Pi n{:}\mathsf{nat}.\, \Pi b{:}\mathsf{nat}.\, \Sigma m{:}\mathsf{nat}.\, m \doteq b^n$$

- Erasing dependencies we obtain:

$$power : \mathsf{nat} \rightarrow \mathsf{nat} \rightarrow (\mathsf{nat} \times \mathsf{proof})$$

- Usually do not compute elements of type proof.

Dependent Function Types

- Constructor:

$$\frac{\Gamma \vdash A : \text{type} \qquad \Gamma, x{:}A \vdash M : B(x)}{\Gamma \vdash \lambda x{:}A.\, M : \Pi x{:}A.\, B(x)}\ \Pi I$$

- Destructor:

$$\frac{\Gamma \vdash M : \Pi x{:}A.\, B(x) \qquad \Gamma \vdash N : A}{\Gamma \vdash M\, N : B(N)}\ \Pi E$$

- Reduction:

$$(\lambda x{:}A.\, M)\, N \quad \longrightarrow \quad [N/x]M$$

Dependent Sum Types

- Constructor:

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B(M)}{\Gamma \vdash \langle M, N \rangle : \Sigma x{:}A.\, B(x)}\ \Sigma I$$

- Destructors:

$$\frac{\Gamma \vdash M : \Sigma x{:}A.\, B(x)}{\Gamma \vdash \pi_1\, M : A}\ \Sigma E_1 \qquad \frac{\Gamma \vdash M : \Sigma x{:}A.\, B(x)}{\Gamma \vdash \pi_2\, M : B(\pi_1\, M)}\ \Sigma E_1$$

- Reductions:

$$\begin{array}{lcl} \pi_1\, \langle M, N \rangle & \longrightarrow & M \\ \pi_2\, \langle M, N \rangle & \longrightarrow & N \end{array}$$

Judgments Revisited

- Well-formed types: $\Gamma \vdash A : \text{type}$
- Definitional equality: $\Gamma \vdash A = B : \text{type}$
- Derived from reduction $M \longrightarrow M'$ and evaluation.
- Type conversion:

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash A = B : \text{type}}{\Gamma \vdash M : B}$$

- Generalized substitution property:

  If $\Gamma \vdash M : A$
  and $\Gamma, x{:}A \vdash N : C(x)$
  then $\Gamma \vdash [M/x]N : C(M)$.

Example Revisited

- Proof objects:

$$p_0 \;:\; \Pi b{:}\mathsf{nat}.\, 1 \doteq b^0$$
$$p_1 \;:\; \Pi b{:}\mathsf{nat}.\, \Pi n{:}\mathsf{nat}.\, \Pi m{:}\mathsf{nat}.\, m \doteq b^n \to b * m \doteq b^{n+1}$$

- Recall: $\mathit{power} : \Pi n{:}\mathsf{nat}.\, \Pi b{:}\mathsf{nat}.\, \Sigma m{:}\mathsf{nat}.\, m \doteq b^n$

- Power function with correctness proof:

```
fun power 0 b = ⟨1, p0 b⟩
  | power (s n) b =
     ⟨b * π1 (power n b),
      p1 b n (π1 (power n b))
         (π2 (power n b))⟩
```

The Problem: Properties of Non-Existent Objects

- Consider the staged version of the power program:

$$
\begin{array}{l}
\texttt{fun power 0 = box } (\lambda\texttt{b. } \langle \texttt{1, p0 b} \rangle) \\
\quad \texttt{| power (s n) =} \\
\quad\quad \texttt{let box u = power n} \\
\quad\quad\; \texttt{in box } (\lambda\texttt{b. } \langle \texttt{b * } (\pi_1 \texttt{ (u b)),} \\
\qquad\qquad\qquad\qquad \texttt{p1 b n } (\pi_1 \texttt{ (u b)) } (\pi_2 \texttt{ (u b))} \rangle \texttt{) end}
\end{array}
$$

- $\not\vdash \mathit{power} : \Pi n{:}\mathsf{nat}.\, \Box(\Pi b{:}\mathsf{nat}.\, \Sigma m{:}\mathsf{nat}.\, m \doteq b^n)$
- $\not\vdash \Pi n{:}\mathsf{nat}.\, \Box(\Pi b{:}\mathsf{nat}.\, \Sigma m{:}\mathsf{nat}.\, m \doteq b^n) : \mathsf{type}$
- Violates staging by accessing n.
- n is used only in the proof object!

Solution: Reasoning about the Past

- Generalize judgment: $\Delta; \Gamma; \Omega \vdash M : A$.
- $\Delta = u_1{::}A_1, \ldots, u_k{::}A_k$ (expression variables), available in all worlds.
- $\Gamma = x_1{:}B_1, \ldots, x_n{:}B_n$ (value variables), available only in current world.
- $\Omega = w_1{\div}C_1, \ldots, w_m{\div}C_m$ (hidden variables), **were available in a past world**.
- **No** Hypothesis rule: $\Delta; \Gamma; (\Omega_1, w{\div}A, \Omega_2) \not\vdash w : A$!

Necessity Revisited

- Constructor:

$$\frac{\Delta; \bullet; (\Gamma, \Omega) \vdash M : A}{\Delta; \Gamma; \Omega \vdash \mathsf{box}\, M : \Box A}\ \Box I$$

- Destructor:

$$\frac{\Delta; \Gamma; \Omega \vdash M : \Box A \qquad (\Delta, u{::}A); \Gamma; \Omega \vdash N : C}{\Delta; \Gamma; \Omega \vdash \mathsf{let}\,\mathsf{box}\, u = M \;\mathsf{in}\; N\, \mathsf{end} : C}\ \Box E$$

- Substitution principle:

If $\Delta; \bullet; (\Gamma, \Omega) \vdash M : A$
and $(\Delta, u{::}A); \Gamma; \Omega \vdash N : C$
then $\Delta; \Gamma; \Omega \vdash N : C$.

Hidden Types

- $M : [A]$ *"M is a term of type A from the past"*
- Constructor:

$$\frac{\Delta; (\Gamma, \Omega); \bullet \vdash M : A}{\Delta; \Gamma; \Omega \vdash [M] : [A]} \; [-]I$$

- Destructor:

$$\frac{\Delta; \Gamma; \Omega \vdash M : [A] \qquad \Delta; \Gamma; (\Omega, w \div A) \vdash N : C}{\Delta; \Gamma; \Omega \vdash \text{let}\, [w] = M \text{ in } N \,\text{end} : C} \; [-]E$$

- Substitution principle:

  If $\Delta; (\Gamma, \Omega); \bullet \vdash M : A$
  and $\Delta; \Gamma; (\Omega, w \div A) \vdash N : C$
  then $\Delta; \Gamma; \Omega \vdash [M/w]N : C$

Example Revisited

- "Hide" proof objects.
- $\vdash \Pi n{:}\mathsf{nat}.\, \Box(\Pi b{:}\mathsf{nat}.\, \Sigma m{:}\mathsf{nat}.\, [m \doteq b^n])$: type
- $\vdash \mathit{power} : \Pi n{:}\mathsf{nat}.\, \Box(\Pi b{:}\mathsf{nat}.\, \Sigma m{:}\mathsf{nat}.\, [m \doteq b^n])$
- Implementation:

```
fun power 0 = box (λb. ⟨1, [p0 b]⟩)
  | power (s n) =
    let box u = power n
     in box (λb. ⟨b * (π1 (u b)),
                  let [w] = π2 (u b)
                  in [p1 b n (π1 (u b)) w] end⟩) end
```

Dependent Types Revisited

- We cannot sort $u{::}A$, $x{:}A$, $w{\div}A$ into zones because of dependencies.
- $\Psi ::= \bullet \mid \Psi, x{::}A \mid \Psi, x{:}A \mid \Psi, x{\div}A$
- $\Psi^{\ominus}$ replaces $x{:}A$ by $x{\div}A$.
- $\Psi^{\oplus}$ replaces $x{\div}A$ by $x{:}A$.
- Valid contexts: $\vdash \Psi\ \mathsf{ctx}$

$$\frac{}{\vdash \bullet\ \mathsf{ctx}} \qquad \frac{\vdash \Psi\ \mathsf{ctx} \quad \Psi \vdash A : \mathsf{type}}{\vdash (\Psi, x{:}A)\ \mathsf{ctx}}$$

$$\frac{\vdash \Psi\ \mathsf{ctx} \quad \Psi^{\ominus} \vdash A : \mathsf{type}}{\vdash (\Psi, x{::}A)\ \mathsf{ctx}} \qquad \frac{\vdash \Psi\ \mathsf{ctx} \quad \Psi^{\oplus} \vdash A : \mathsf{type}}{\vdash (\Psi, x{\div}A)\ \mathsf{ctx}}$$

Dependent Substitution Principles

- For expression variables:

  If $\Psi^{\ominus} \vdash M : A$
  and $\Psi, u{::}A \vdash N : C(u)$
  then $\Psi \vdash [M/u]N : C(M)$.

- For value variables:

  If $\Psi \vdash M : A$
  and $\Psi, x{:}A \vdash N : C(x)$
  Then $\Psi \vdash [M/x]N : C(M)$.

- For hidden variables:

  If $\Psi^{\oplus} \vdash M : A$
  and $\Psi, w{\div}A \vdash N : C(w)$
  then $\Psi \vdash [M/w]N : C(M)$.

Necessity Revisited

- Formation:

$$\frac{\Psi^{\ominus} \vdash A : \mathsf{type}}{\Psi \vdash \Box A : \mathsf{type}}\ \Box F$$

- Constructor:

$$\frac{\Psi^{\ominus} \vdash M : A}{\Psi \vdash \mathsf{box} M : \Box A}\ \Box I$$

- Destructor:

$$\frac{\Psi \vdash M : \Box A \qquad \Psi, u{::}A \vdash N : C}{\Psi \vdash \mathsf{let}\, u = M \ \mathsf{in}\ N\, \mathsf{end} : C}\ \Box E$$

Hidden Types Revisited

- Formation:

$$\frac{\Psi^{\oplus} \vdash A : \mathsf{type}}{\Psi \vdash [A] : \mathsf{type}}\ [-]F$$

- Constructor:

$$\frac{\Psi^{\oplus} \vdash M : A}{\Psi \vdash [M] : [A]}\ [-]I$$

- Destructor:

$$\frac{\Psi \vdash M : [A] \qquad \Psi, w{\div}A \vdash N : C}{\Psi \vdash \mathsf{let}\,[w] = M\ \mathsf{in}\ N\,\mathsf{end} : C}\ [-]E$$

Dependent Function Types Revisited

- Constructor:

$$\frac{\Psi \vdash A : \mathsf{type} \qquad \Psi, x{:}A \vdash M : B(x)}{\Psi \vdash \lambda x{:}A.\, M : \Pi x{:}A.\, B(x)}\ \Pi I$$

- Destructor:

$$\frac{\Psi \vdash M : \Pi x{:}A.\, B(x) \qquad \Psi \vdash N : A}{\Psi \vdash M\, N : B(N)}\ \Pi E$$

- Dependent sums are similarly straightforward.

Application: First-Order Modal Logic

- Define $\forall^e x.\, A(x)$ "*for all ephemeral* x, $A(x)$"
  Quantifies over elements of current world.
- $\forall^e x.\, \Box A(x)$ is not necessarily well-formed.
- Define $\forall^p u.\, A(u)$ "*for all persistent* u, $A(u)$"
  Quantifies over elements existing in all worlds.
- $\Box(\forall^p u.\, A(u)) \supset \forall^p u.\, \Box A(u)$ is true.
- Define $\forall^0 w.\, A[w]$ "*for all ethereal* w, $A[w]$"

Application: Subset Types

- Define $\{x{:}A \mid B(x)\}$ as $\Sigma x{:}A.\,[B(x)]$.
- Derived rules:

$$\frac{\Psi \vdash M : A \qquad \Psi^{\oplus} \vdash N : B(M)}{\Psi \vdash \langle M, N \rangle : \{x{:}A \mid B(x)\}}\,\{-\}I$$

$$\frac{\Psi \vdash M : \{x{:}A \mid B(x)\}}{\Psi \vdash \pi_1\, M : A}\,\{-\}E_1$$

$$\frac{\Psi \vdash M : \{x{:}A \mid B(x)\} \qquad \Psi, w{\div}B(\pi_1\, M) \vdash N : C}{\Psi \vdash \mathsf{let}\,[w] = \pi_2\, M\ \mathsf{in}\ N\ \mathsf{end} : C}\,\{-\}E_2$$

The Erasure Interpretation

- Slogan: "*Forget about the Past!*"
- Type $\odot$ "*terms with no computational contents*"
- Replace $[A]$ by $\odot$ and propagate.
- Replace $M : [A]$ by token $\star$ and propagate.
- Orthogonal to necessity.

Properties of Erasure

- **Theorem:** Erasure commutes with type-checking and evaluation.
- **Theorem:** Hidden variables are *useless* (computationally irrelevant) under erasure interpretation.
- Dependently typed programs are correct and their erasure executes in a well-staged manner.

Definitional Equality Revisited

- $\Box A$ is *intensional* (observable):

  No congruence rule for box$M$*.*

- $[A]$ is *computationally irrelevant* (inobservable):

$$\frac{}{\Psi \vdash M = N : [A]}\ [-]Eq$$

- Other choices are possible.

Future Work on Type Theory

- Decidability of type-checking?
- Variations on rules?
- Variations on definitional equality?
- Application to quotient types?
- Classical reasoning in $[A]$? [Bauer'00]
- Investigate duality between $\Box$ and $[-]$.
- Integrate with DML, where index objects are restricted to (tractable) constraint domains. [Xi & Pf'98] [Xi & Pf'99]
- Develop dependently typed linear logic?

Application: Linear Logic

- Traditional judgment: $\Delta; \Gamma \vdash M : A$
  Δ (unrestricted variables), Γ (linear variables)

- Traditional sample rule:

$$\frac{\Delta; \Gamma_1 \vdash M : A \quad \Delta; \Gamma_2 \vdash N : B}{\Delta; (\Gamma_1, \Gamma_2) \vdash M \otimes N : A \otimes B} \otimes I$$

- New rule (adding hidden variables):

$$\frac{\Delta; \Gamma_1; (\Gamma_2, \Omega) \vdash M : A \quad \Delta; \Gamma_2; (\Gamma_1, \Omega) \vdash N : B}{\Delta; (\Gamma_1, \Gamma_2); \Omega \vdash M \otimes N : A \otimes B} \otimes I$$

- This permits more programs under type assignment:

$$\vdash \lambda x.\, \lambda y.\, x \otimes (\lambda w.\, y)\, x : A \multimap B \multimap A \otimes B$$

Related Work

- Useless variable elimination in functional programming.
- Vacuous and strict variables [Momigliano'00].
- Substructural λ-calculi [Wright'98].
- Hidden variables in Nuprl.
- *Prop* vs *Spec* in Coq.
- Squash types and proof irrelevance [Hofmann'96]

Summary

- Judgmental reconstruction of modal types (S4).
- Dependent types for full specifications.
- References to the past and hidden variables.
- Clear logical solution.
- Admits erasure interpretation.
- Other applications (subset types, linear type theory).