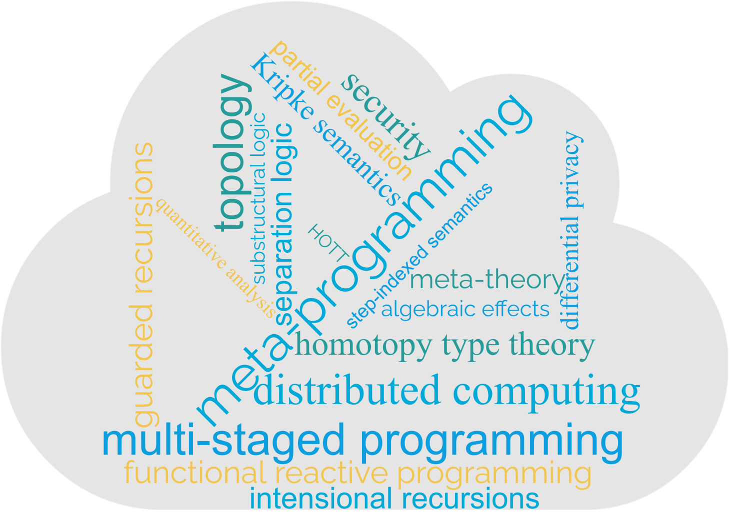
Foundations and Applications of Modal Type Theories

Jason Hu

McGill University

Proposal Examination

Applications of Modalities



We focus on the $\square$ modality.

We focus on the $\Box$ modality.

- Prawitz (1965) proposed a formulation with broken subject reduction.

Foundations of Modal Type Theories

We focus on the $\Box$ modality.

- Prawitz (1965) proposed a formulation with broken subject reduction.
- Formulations of the $\Box$ modality was under intense investigations in the late '90s and the early 2000:

We focus on the $\Box$ modality.

- Prawitz (1965) proposed a formulation with broken subject reduction.
- Formulations of the $\Box$ modality was under intense investigations in the late '90s and the early 2000:
  - $\mathsf{IS4}_{\Box}$ (Bierman and de Paiva, 1996)

Foundations of Modal Type Theories

We focus on the $\Box$ modality.

- Prawitz (1965) proposed a formulation with broken subject reduction.
- Formulations of the $\Box$ modality was under intense investigations in the late '90s and the early 2000:
  - $IS4_{\Box}$ (Bierman and de Paiva, 1996)
  - Modal pure type system (Borghuis, 1994)

Foundations of Modal Type Theories

We focus on the $\Box$ modality.

- Prawitz (1965) proposed a formulation with broken subject reduction.
- Formulations of the $\Box$ modality was under intense investigations in the late '90s and the early 2000:
  - $IS4_{\Box}$ (Bierman and de Paiva, 1996)
  - Modal pure type system (Borghuis, 1994)
  - The dual-context style (Pfenning and Davies, 2001; Davies and Pfenning, 2001)

Foundations of Modal Type Theories

We focus on the $\Box$ modality.

- Prawitz (1965) proposed a formulation with broken subject reduction.
- Formulations of the $\Box$ modality was under intense investigations in the late '90s and the early 2000:
  - $\text{IS4}_\Box$ (Bierman and de Paiva, 1996)
  - Modal pure type system (Borghuis, 1994)
  - The dual-context style (Pfenning and Davies, 2001; Davies and Pfenning, 2001)
  - The Kripke style (Davies and Pfenning, 2001; Pfenning and Wong, 1995)

Foundations of Modal Type Theories

We focus on the $\square$ modality.

- Prawitz (1965) proposed a formulation with broken subject reduction.
- Formulations of the $\square$ modality was under intense investigations in the late '90s and the early 2000:
  - $\mathrm{IS4}_{\square}$ (Bierman and de Paiva, 1996)
  - Modal pure type system (Borghuis, 1994)
  - The dual-context style (Pfenning and Davies, 2001; Davies and Pfenning, 2001)
  - The Kripke style (Davies and Pfenning, 2001; Pfenning and Wong, 1995)
- Foundations of $\square$ still under active investigations (Hu and Pientka, 2022; Valliappan et al., 2022; Gratzer et al., 2019; Gratzer, 2022, etc.).

Goals of Thesis

Study general foundation for modal type theories; then focus on a particular application to meta-programming

Goals of Thesis

Study general foundation for modal type theories; then focus on a particular application to meta-programming

- ✓ Prove directly normalization of the Kripke-style $\lambda^{\to\Box}$ (simply typed) by Davies and Pfenning (2001); Pfenning and Wong (1995) (published in Hu and Pientka (2022) at MFPS)

Goals of Thesis

Study general foundation for modal type theories; then focus on a particular application to meta-programming

- ✓ Prove directly normalization of the Kripke-style $\lambda^{\to\square}$ (simply typed) by Davies and Pfenning (2001); Pfenning and Wong (1995) (published in Hu and Pientka (2022) at MFPS)
- ✓ Extend $\lambda^{\to\square}$ with dependent types, yielding Mint (submitted to JFP)

Goals of Thesis

Study general foundation for modal type theories; then focus on a particular application to meta-programming

- ✓ Prove directly normalization of the Kripke-style $\lambda^{\to\square}$ (simply typed) by Davies and Pfenning (2001); Pfenning and Wong (1995) (published in Hu and Pientka (2022) at MFPS)
- ✓ Extend $\lambda^{\to\square}$ with dependent types, yielding MINT (submitted to JFP)
- ✓ Mechanize normalization proofs of these systems (submitted to JFP)

Goals of Thesis

Study general foundation for modal type theories; then focus on a particular application to meta-programming

- ✓ Prove directly normalization of the Kripke-style $\lambda^{\to\square}$ (simply typed) by Davies and Pfenning (2001); Pfenning and Wong (1995) (published in Hu and Pientka (2022) at MFPS)
- ✓ Extend $\lambda^{\to\square}$ with dependent types, yielding $\textsc{Mint}$ (submitted to JFP)
- ✓ Mechanize normalization proofs of these systems (submitted to JFP)
- ▶ Add pattern matching on code to modal type theory to strengthen its ability to do dependently typed meta-programming

Revisit $\lambda^{\to\square}$

- The Kripke style has multiple advantages:

Revisit $\lambda^{\rightarrow\Box}$

- The Kripke style has multiple advantages:
  - Conceptually simple; direct modeling of the Kripke semantics

Revisit $\lambda^{\to\Box}$

- The Kripke style has multiple advantages:
  - Conceptually simple; direct modeling of the Kripke semantics
  - Captures all four subsystems of $S4$

Revisit $\lambda^{\to\Box}$

- The Kripke style has multiple advantages:
  - Conceptually simple; direct modeling of the Kripke semantics
  - Captures all four subsystems of $S4$
  - Programming paradigm akin to typical meta-programming using quasi-quotes

Revisit $\lambda^{\to\Box}$

- The Kripke style has multiple advantages:
  - Conceptually simple; direct modeling of the Kripke semantics
  - Captures all four subsystems of $S4$
  - Programming paradigm akin to typical meta-programming using quasi-quotes
-

$$\frac{\overrightarrow{\Gamma}; \cdot \vdash t : T}{\overrightarrow{\Gamma} \vdash \texttt{box}\ t : \Box T} \qquad \frac{\overrightarrow{\Gamma} \vdash t : \Box T \qquad |\overrightarrow{\Delta}| = n}{\overrightarrow{\Gamma}; \overrightarrow{\Delta} \vdash \texttt{unbox}_n\ t : T}$$

Revisit $\lambda^{\to\Box}$

- The Kripke style has multiple advantages:
  - Conceptually simple; direct modeling of the Kripke semantics
  - Captures all four subsystems of $S4$
  - Programming paradigm akin to typical meta-programming using quasi-quotes
-

$$\frac{\vec{\Gamma}; \cdot \vdash t : T}{\vec{\Gamma} \vdash \texttt{box}\ t : \Box T} \qquad \frac{\vec{\Gamma} \vdash t : \Box T \qquad |\vec{\Delta}| = n}{\vec{\Gamma}; \vec{\Delta} \vdash \texttt{unbox}_n\ t : T}$$

-
```
K : □ (A → B) → □ A → □ B
K f x = box ((unbox₁ f) (unbox₁ x))
```

Revisit $\lambda^{\to\Box}$

- The Kripke style has multiple advantages:
  - Conceptually simple; direct modeling of the Kripke semantics
  - Captures all four subsystems of $S4$
  - Programming paradigm akin to typical meta-programming using quasi-quotes
-

$$\frac{\vec{\Gamma}; \cdot \vdash t : T}{\vec{\Gamma} \vdash \texttt{box}\ t : \Box T} \qquad \frac{\vec{\Gamma} \vdash t : \Box T \qquad |\vec{\Delta}| = n}{\vec{\Gamma}; \vec{\Delta} \vdash \texttt{unbox}_n\ t : T}$$

-

```
K : □ (A → B) → □ A → □ B
K f x = box ((unbox₁ f) (unbox₁ x))
```

- Challenge: separate reasoning of substitutions and modal transformations leads to complex analyses

Kripke-style Substitutions

- In Hu and Pientka (2022), we propose Kripke-style substitutions as a substitution calculus for $\lambda^{\to\square}$

Kripke-style Substitutions

- In Hu and Pientka (2022), we propose Kripke-style substitutions as a substitution calculus for $\lambda^{\to\Box}$
- K-substitutions: regular simultaneous substitutions extended with modal information

Kripke-style Substitutions

- In Hu and Pientka (2022), we propose Kripke-style substitutions as a substitution calculus for $\lambda^{\to\Box}$
- K-substitutions: regular simultaneous substitutions extended with modal information
- Modularly and uniformly capture the Kripke structure of $\lambda^{\to\Box}$ both syntactically and semantically

Kripke-style Substitutions

- In Hu and Pientka (2022), we propose Kripke-style substitutions as a substitution calculus for $\lambda^{\to\square}$
- K-substitutions: regular simultaneous substitutions extended with modal information
- Modularly and uniformly capture the Kripke structure of $\lambda^{\to\square}$ both syntactically and semantically
- One normalization-by-evaluation proof for all four subsystems

Kripke-style Substitutions

- In Hu and Pientka (2022), we propose Kripke-style substitutions as a substitution calculus for $\lambda^{\to\square}$
- K-substitutions: regular simultaneous substitutions extended with modal information
- Modularly and uniformly capture the Kripke structure of $\lambda^{\to\square}$ both syntactically and semantically
- One normalization-by-evaluation proof for all four subsystems
- Enable a formulation of contextual types (Nanevski et al., 2008) in the Kripke style

MINT: Extensions to Dependent Types

- $\lambda^{\to\Box}$ is simply typed; scales to dependent types naturally

MINT: Extensions to Dependent Types

- $\lambda^{\to\Box}$ is simply typed; scales to dependent types naturally
- MINT, **M**odal **In**tuitionistic **T**ype Theory, is developed:

MINT: Extensions to Dependent Types

- $\lambda^{\rightarrow\square}$ is simply typed; scales to dependent types naturally
- MINT, **M**odal **In**tuitionistic **T**ype Theory, is developed:
  - contains full Martin-Löf type theory

MINT: Extensions to Dependent Types

- $\lambda^{\to\Box}$ is simply typed; scales to dependent types naturally
- MINT, **M**odal **In**tuitionistic **T**ype Theory, is developed:
  - contains full Martin-Löf type theory
  - supports inductive types, large elimination, a full cumulative universe hierarchy

MINT: Extensions to Dependent Types

- $\lambda^{\rightarrow\Box}$ is simply typed; scales to dependent types naturally
- MINT, **M**odal **In**tuitionistic **T**ype Theory, is developed:
  - contains full Martin-Löf type theory
  - supports inductive types, large elimination, a full cumulative universe hierarchy
  - normalization by evaluation

MINT: Extensions to Dependent Types

- $\lambda^{\to\square}$ is simply typed; scales to dependent types naturally
- MINT, **M**odal **Int**uitionistic **T**ype Theory, is developed:
  - contains full Martin-Löf type theory
  - supports inductive types, large elimination, a full cumulative universe hierarchy
  - normalization by evaluation
-

$$\frac{\vec{\Gamma}; \cdot \vdash T : \mathsf{Se}_i}{\vec{\Gamma} \vdash \square T : \mathsf{Se}_i} \qquad \frac{\vec{\Gamma}; \cdot \vdash t : T}{\vec{\Gamma} \vdash \texttt{box}\ t : \square T} \qquad \frac{\vec{\Gamma} \vdash t : \square T \quad \vdash \vec{\Gamma}; \vec{\Delta} \quad |\vec{\Delta}| = n}{\vec{\Gamma}; \vec{\Delta} \vdash \texttt{unbox}_n\ t : T[\vec{I}; \Uparrow^n]}$$

MINT: Extensions to Dependent Types

- $\lambda^{\to\Box}$ is simply typed; scales to dependent types naturally
- MINT, **M**odal **Int**uitionistic **T**ype Theory, is developed:
  - contains full Martin-Löf type theory
  - supports inductive types, large elimination, a full cumulative universe hierarchy
  - normalization by evaluation
-

$$\frac{\vec{\Gamma}; \cdot \vdash T : \text{Se}_i}{\vec{\Gamma} \vdash \Box T : \text{Se}_i} \qquad \frac{\vec{\Gamma}; \cdot \vdash t : T}{\vec{\Gamma} \vdash \texttt{box}\ t : \Box T} \qquad \frac{\vec{\Gamma} \vdash t : \Box T \quad \vdash \vec{\Gamma}; \vec{\Delta} \quad |\vec{\Delta}| = n}{\vec{\Gamma}; \vec{\Delta} \vdash \texttt{unbox}_n\ t : T[\vec{I}; \Uparrow^n]}$$

- Submitted to JFP

Normalization of MINT

- The normalization proof of MINT is a moderate extension of Abel (2013).

Normalization of Mint

- The normalization proof of Mint is a moderate extension of Abel (2013).
  - based on an untyped domain model

Normalization of Mint

- The normalization proof of Mint is a moderate extension of Abel (2013).
  - based on an untyped domain model
  - the algorithm is explicitly given

Normalization of MINT

- The normalization proof of MINT is a moderate extension of Abel (2013).
  - based on an untyped domain model
  - the algorithm is explicitly given
  - use the algebra of truncoid to capture the Kripke structure; inherent the modularity of the normalization proof of $\lambda^{\to\square}$

Normalization of Mint

- The normalization proof of Mint is a moderate extension of Abel (2013).
  - based on an untyped domain model
  - the algorithm is explicitly given
  - use the algebra of truncoid to capture the Kripke structure; inherent the modularity of the normalization proof of $\lambda^{\to\square}$
- Fully mechanized in Agda!

Normalization of MINT

- The normalization proof of MINT is a moderate extension of Abel (2013).
  - based on an untyped domain model
  - the algorithm is explicitly given
  - use the algebra of truncoid to capture the Kripke structure; inherent the modularity of the normalization proof of $\lambda^{\to\square}$
- Fully mechanized in Agda!
  - foundation: MLTT + functional extensionality + induction-recursion; standard extensions

Normalization of Mint

- The normalization proof of Mint is a moderate extension of Abel (2013).
  - based on an untyped domain model
  - the algorithm is explicitly given
  - use the algebra of truncoid to capture the Kripke structure; inherent the modularity of the normalization proof of $\lambda^{\to\Box}$
- Fully mechanized in Agda!
  - foundation: MLTT + functional extensionality + induction-recursion; standard extensions
  - exposes various missing details about modeling universes

Normalization of MINT

- The normalization proof of MINT is a moderate extension of Abel (2013).
  - based on an untyped domain model
  - the algorithm is explicitly given
  - use the algebra of truncoid to capture the Kripke structure; inherent the modularity of the normalization proof of $\lambda^{\to\Box}$
- Fully mechanized in Agda!
  - foundation: MLTT + functional extensionality + induction-recursion; standard extensions
  - exposes various missing details about modeling universes
  - a basis for others to experiment their extensions to MLTT

Normalization of Mint

- The normalization proof of Mint is a moderate extension of Abel (2013).
  - based on an untyped domain model
  - the algorithm is explicitly given
  - use the algebra of truncoid to capture the Kripke structure; inherent the modularity of the normalization proof of $\lambda^{\to\square}$
- Fully mechanized in Agda!
  - foundation: MLTT + functional extensionality + induction-recursion; standard extensions
  - exposes various missing details about modeling universes
  - a basis for others to experiment their extensions to MLTT
  - the algorithm can be run in Haskell after extraction

Pattern Matching on Code

- We focus on applications to meta-programming.

Pattern Matching on Code

- We focus on applications to meta-programming.
  - $\square T$ denotes the type of code representing some T

Pattern Matching on Code

- We focus on applications to meta-programming.
  - $\Box T$ denotes the type of code representing some T
- A missing feature: pattern matching on code

Pattern Matching on Code

- We focus on applications to meta-programming.
  - $\square T$ denotes the type of code representing some T
- A missing feature: pattern matching on code
- Internal analysis of syntactic structure:

```
is-app : □ T → Bool
is-app (box (f x)) = true
is-app _           = false
```

Challenges of Extending Mint

- Philosophical clash: `unbox` in Mint uses $\square$ by projection; while pattern matching on code does case analysis.

Challenges of Extending MINT

- Philosophical clash: `unbox` in MINT uses $\square$ by projection; while pattern matching on code does case analysis.
- Congruence of `box` in MINT breaks confluence:

```
is-app (box ((λ x → x) 0))
```

  There are two distinct reductions:

```
is-app : □ T → Bool
is-app (box (f x)) = true
is-app _           = false
```

Challenges of Extending MINT

- Philosophical clash: `unbox` in MINT uses $\square$ by projection; while pattern matching on code does case analysis.
- Congruence of box in MINT breaks confluence:

  ```
  is-app (box ((λ x → x) 0))
  ```

  There are two distinct reductions:
  - `is-app (box ((λ x → x) 0)) = true`

```
is-app : □ T → Bool
is-app (box (f x)) = true
is-app _           = false
```

Challenges of Extending MINT

- Philosophical clash: `unbox` in MINT uses $\square$ by projection; while pattern matching on code does case analysis.
- Congruence of `box` in MINT breaks confluence:

```
is-app (box ((λ x → x) 0))
```

  There are two distinct reductions:
  - `is-app (box ((λ x → x) 0)) = true`
  - ```
      is-app (box ((λ x → x) 0))
    = is-app (box 0)
    = false
    ```

```
is-app : □ T → Bool
is-app (box (f x)) = true
is-app _           = false
```
```

No Easy Fix for Confluence

Is it possible to only pattern match on normal forms?

No Easy Fix for Confluence

Is it possible to only pattern match on normal forms?

Confluence is still broken:

```
(λ x → is-app (box (unbox₁ x))) (box (F T))
```

No Easy Fix for Confluence

Is it possible to only pattern match on normal forms?
Confluence is still broken:

```
(λ x → is-app (box (unbox₁ x))) (box (F T))
```

- ```
    (λ x → is-app (box (unbox₁ x))) (box (F T))
  = (λ x → false) (box (F T))
  = false
  ```
  ```

No Easy Fix for Confluence

Is it possible to only pattern match on normal forms?
Confluence is still broken:

```
(λ x → is-app (box (unbox₁ x))) (box (F T))
```

- ```
    (λ x → is-app (box (unbox₁ x))) (box (F T))
  = (λ x → false) (box (F T))
  = false
  ```
- ```
    (λ x → is-app (box (unbox₁ x))) (box (F T))
  = is-app (box (F T))
  = true
  ```

Research Plan

- Take away congruence of `box`; move to the dual-context style

Research Plan

- Take away congruence of `box`; move to the dual-context style
- Use the conversion relation instead of NbE; better control over terms

Research Plan

- Take away congruence of `box`; move to the dual-context style
- Use the conversion relation instead of NbE; better control over terms
- [4 months] Follow experience in the Kripke style; start from simple types

Research Plan

- Take away congruence of `box`; move to the dual-context style
- Use the conversion relation instead of NbE; better control over terms
- [4 months] Follow experience in the Kripke style; start from simple types
- [3 months] Scale to dependent types afterwards

Research Plan

- Take away congruence of `box`; move to the dual-context style
- Use the conversion relation instead of NbE; better control over terms
- [4 months] Follow experience in the Kripke style; start from simple types
- [3 months] Scale to dependent types afterwards
- [1 months] Case study

Research Plan

- Take away congruence of `box`; move to the dual-context style
- Use the conversion relation instead of NbE; better control over terms
- [4 months] Follow experience in the Kripke style; start from simple types
- [3 months] Scale to dependent types afterwards
- [1 months] Case study
- Work out a paper proof before mechanization

Research Plan

- Take away congruence of `box`; move to the dual-context style
- Use the conversion relation instead of NbE; better control over terms
- [4 months] Follow experience in the Kripke style; start from simple types
- [3 months] Scale to dependent types afterwards
- [1 months] Case study
- Work out a paper proof before mechanization
- Possible implementation after PhD

Bibliography

Abel, A. (2013). *Normalization by evaluation: dependent types and impredicativity*. Habilitation thesis, Ludwig-Maximilians-Universität München.

Bierman, G. and de Paiva, V. (1996). Intuitionistic necessity revisited. Technical Report CSR–96–10, University of Birmingham.

Borghuis, V. A. J. (1994). *Coming to terms with modal logic : on the interpretation of modalities in typed lambda-calculus*. PhD Thesis, Mathematics and Computer Science.

Davies, R. and Pfenning, F. (2001). A modal analysis of staged computation. *Journal of the ACM*, 48(3):555–604.

Gratzer, D. (2022). Normalization for Multimodal Type Theory. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '22, pages 1–13, New York, NY, USA. Association for Computing Machinery.

Gratzer, D., Sterling, J., and Birkedal, L. (2019). Implementing a modal dependent type theory. *Proceedings of the ACM on Programming Languages*, 3(ICFP):107:1–107:29.

Hu, J. Z. S. and Pientka, B. (2022). A categorical normalization proof for the modal lambda-calculus. In *Proceedings 38th Conference on Mathematical Foundations of Programming Semantics, MFPS 2022*, EPTCS.

Nanevski, A., Pfenning, F., and Pientka, B. (2008). Contextual modal type theory. *ACM Transactions on Computational Logic*, 9(3):23:1–23:49.

Pfenning, F. and Davies, R. (2001). A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11(04).

Pfenning, F. and Wong, H. (1995). On a modal lambda calculus for S4. In Brookes, S. D., Main, M. G., Melton, A., and Mislove, M. W., editors, *Eleventh Annual Conference on Mathematical Foundations of Programming Semantics, MFPS 1995, Tulane University, New Orleans, LA, USA, March 29 - April 1, 1995*, volume 1 of *Electronic Notes in Theoretical Computer Science*, pages 515–534. Elsevier.

Prawitz, D. (1965). *Natural Deduction: A Proof-theoretical Study*. Stockholm.

Valliappan, N., Ruch, F., and Tomé Cortiñas, C. (2022). Normalization for fitch-style modal calculi. *Proc. ACM Program. Lang.*, 6(ICFP):772–798.