Contextual modal type theory: a foundation for meta-variables

Brigitte Pientka

School of Computer Science
McGill University
Montreal, Canada

Joint work with A. Nanevski and F. Pfenning
Outline

• Logical frameworks and certified code
• Contextual modal type theory
• Applications: Higher-order unification
• Conclusion and future work
Logical frameworks

- Meta-languages for deductive systems
  - High-level specification (e.g. logics, type systems)
  - Direct implementations (e.g. proof search, type checking)
  - Meta-reasoning (e.g. cut elim., type preservation)

- Examples:
  - \(\lambda\)Prolog, Twelf, Isabelle

- Other higher-order systems:
  - Coq, PVS, NuPRL, HOL, ...
Foundational proof-carrying code: [Appel, Felty 00]
Temporal-logic proof carrying code [Bernard, Lee02]
Foundational typed assembly language: [Crary 03]
Proof-carrying authentication: [Felten, Appel 99]
Large-scale applications

- Typical code size: 70,000 – 100,000 lines includes data-type definitions and proofs
- Higher-order logic program: 5,000 lines
- Over 600 – 700 clauses
Application: certified code

Special purpose logical frameworks:

- Efficient representation of proofs
  \( \text{LF}_i \) [Necula, Lee’98] (2-level, restricting dependent types)

- Proof checking via
  “higher-order” logic programming
  Oracle-based checking [Necula’01, Fleat’03]
  No higher-order terms
Application: POPLmark Challenge

• Challenge: How can we
  • encode elegantly prog. languages?
  • experiment easily with proposed systems?
  • facilitate interactive proof developments?
  • prove or check meta-properties?
• Goal: Verify mechanically every POPL paper by 2010.
State of the art

• Logical frameworks are widely used.
• Many challenges remain:
  • Higher-order systems are not efficient enough in practice.
  • Complexity of higher-order issues poorly understood.
  • Higher-order systems lack automatic support.
  • …
State of the art

- Logical frameworks are widely used.
- Many challenges remain:
  - Higher-order systems are not efficient enough in practice.
  - Complexity of higher-order issues poorly understood.
  - Higher-order systems lack automatic support.
  - ...
- This talk: Contextual modal logic and type theory
  - Foundation for meta-variables and explicit substitutions
  - Relativized truth and validity
• Logical frameworks and certified code
• Contextual modal type theory
• Applications: Higher-order unification
• Conclusion and future work
Example: Quantifier Manipulation

- Object logic: First-order logic

Formula ::= $P \mid A \supset A \mid \forall x. A \mid \exists x. A \mid \ldots$

- Specifying manipulation of quantifier scope
- Sample rule:

$$\forall x. (A(x) \supset B) \leftrightarrow (\exists x. A(x)) \supset B$$

if $x$ is not free in $B$
Specification in LF

• Based on higher order abstract syntax:

\[
\begin{aligned}
i & : \text{type}. \\
o & : \text{type}. \\
imp & : o \rightarrow o \rightarrow o. \\
all & : (i \rightarrow o) \rightarrow o. \\
exists & : (i \rightarrow o) \rightarrow o. \\
\end{aligned}
\]

• Quantifier manipulation:

\[
\forall x. (A(x) \sqsupset B) \iff (\exists x. A(x)) \sqsupset B
\]

eq : o \rightarrow o \rightarrow \text{type}.

eq_all : eq (\text{all } (\lambda x. \text{imp } (A \ x) B)) \ (\text{imp } (\text{exists } (\lambda x. A \ x)) B).
Meta-variables

- Clause:
  
  \[ \text{eq\_all} : \text{eq} \ (\forall x. \text{imp} \ (A \ x) \ B) \ (\text{imp} \ (\exists x. \ A \ x)) \ B). \]

- A: i → o and B: o are meta-variables
  also sometimes called \textit{existential variables} or \textit{logic variables}

- Unification problem:
  
  \[ \text{eq} \ (\forall y. \text{imp} \ (\text{imp} \ (p \ y) \ (p \ y))) \ q) \ C \]
  
  \[ \equiv \]

  \[ \text{eq} \ (\forall x. \text{imp} \ (A \ x) \ B)) \ (\text{imp} \ (\exists x. \ A \ x)) \ B) \]
Closed instantiation for meta-variables

- Unification problem:
  \[
  \text{eq } (\forall (\lambda y. \text{imp } (\text{imp } (p \ y) (p \ y))) \ q) \ C \\
  = \\
  \text{eq } (\forall (\lambda x. \text{imp } (A \ x) B)) \ (\text{imp } (\exists (\lambda x. A \ x)) B)
  \]

- Solution:
  
  \[
  \begin{align*}
  A &= \lambda z. \text{imp } (p \ z) (p \ z) \\
  B &= q \\
  C &= (\text{imp } (\exists (\lambda x. A \ x)) B) \\
  &= (\text{imp } (\exists (\lambda x. \text{imp } (p \ x) (p \ x))) q)
  \end{align*}
  \]

- Instantiations for meta-variables contain no free ordinary variables![Huet75]
No closed instantiation of meta-variables

- Unification problem

\[
eq (\lambda y. \text{imp } q \ (\text{imp } (p \ y) \ (p \ y))) \ C
\]
\[
\implies
\]
\[
eq (\lambda x. \text{imp } (A \ x) \ B) \ (\text{imp } (\exists x. A \ x)) \ B
\]

- No solution:

\[
A = \lambda z. q
\]
\[
B = (\text{imp } (p \ y) \ (p \ y)) \quad \text{FAILURE}
\]
\[
C = \ldots
\]
Questions concerning meta-variables

1. Which parameters are allowed to occur in a term that instantiates a meta-variable.
Questions concerning meta-variables

1. Which parameters are allowed to occur in a term that instantiates a meta-variable.

2. What constraints does the foundation impose on
   - occurrences of meta-variables in the contexts
   - types of other meta-variables.
Questions concerning meta-variables

1. Which parameters are allowed to occur in a term that instantiates a meta-variable.

2. What constraints does the foundation impose on
   • occurrences of meta-variables in the contexts
   • types of other meta-variables.

3. How do we implement meta-variables and the instantiation operation.
Questions concerning meta-variables

1. Which parameters are allowed to occur in a term that instantiates a meta-variable.

2. What constraints does the foundation impose on
   - occurrences of meta-variables in the contexts
   - types of other meta-variables.

3. How do we implement meta-variables and the instantiation operation.

4. Which algorithm do we use for unification or constraint simplification.
Modal logic and type theory

- Distinguish between truth and validity [Pf, Davies’01]
- Two basic judgments
  \[ A \text{ true} \quad : \quad \text{proposition } A \text{ is true} \]
  \[ A \text{ valid} \quad : \quad \text{proposition } A \text{ is valid} \]
  \[ A \text{ is true in any world} \]
- Two contexts – two kinds of variables:
  \[ \Gamma \quad : \quad x_1:A_1 \text{ true}, \ldots x_n:A_n \text{ true} \]
  \[ \Delta \quad : \quad u_1::B_1 \text{ valid}, \ldots u_k::B_k \text{ valid} \]
Validity and truth

- Hypothetical judgment: \( \Delta; \Gamma \vdash C \) true
  \[ \Delta; \Gamma \vdash C \] valid

- Definition of validity:
  \[ \Delta; \cdot \vdash A \] true
  \[ \Delta; \Gamma \vdash A \] valid

- Hypothesis rule:
  \( (\Delta, u::A \text{ valid}, \Delta'); \Gamma \vdash A \) true

- Substitution principle for validity:
  If \( \Delta; \cdot \vdash A \) true and \( (\Delta, u::A \text{ valid}, \Delta'); \Gamma \vdash C \) true
  then \( (\Delta, \Delta'); \Gamma \vdash C \) true.
Contextual validity

- **Validity**: $A$ valid $\iff A$ is true in any world
- **Contextual validity**: $A$ is valid relative $\Psi$ if $A$ is true in every world in which $\Psi$ is true

$$A \text{ valid}\underbrace{[y_1:B_1 \text{ true}, \ldots, y_n:B_m \text{ true}]}_{\Psi}$$

- **Terminology**: $\Psi$ : context
  $$A[\Psi]$$ : contextual validity
- **Generalization of validity**
Definition of contextual validity

- Definition of contextual validity

\[
\frac{\Delta; \Psi \vdash A \text{ true}}{\Delta; \Gamma \vdash A \text{ valid}[\Psi]}
\]

- Contextual Hypothesis Rule

\[
(\Delta, u :: A \text{ valid}[\Psi], \Delta'); \Gamma \vdash \Psi \quad \text{ctxhyp}_u
\]

\[
(\Delta, u :: A \text{ valid}[\Psi], \Delta'); \Gamma \vdash A \text{ true}
\]

- Contextual Entailment

\[
\frac{\Delta; \Gamma \vdash B_1 \text{ true} \quad \ldots \quad \Delta; \Gamma \vdash B_m \text{ true}}{\Delta; \Gamma \vdash y_1 : B_1 \text{ true}, \ldots, y_m : B_m \text{ true} \quad \text{ctx}}
\]

Contextual modal type theory: – p.17/39
Meta-theoretical Properties

• Contextual substitution principle:
  If $\Delta; \Psi \vdash A$ true and $(\Delta, u::A \text{ valid}[\Psi], \Delta'); \Gamma \vdash C$ true then $(\Delta, \Delta'); \Gamma \vdash C$ true.

• Internalize modality via introduction and elimination rules

• Meta-theoretic properties:
  • Locally sound and complete
  • Cut-elimination for ordinary and contextual cut
Towards a contextual modal type-theory

- Terms $M ::= x \mid u[\sigma] \mid \lambda x. M \mid M_1 \, M_2$

- Ordinary hypothesis rule:

$$\frac{}{\Delta; (\Gamma, x:A, \Gamma') \vdash x : A}^{\text{hyp}}$$

- Contextual hypothesis rule:

$$\frac{\Delta, u::A[\Psi], \Delta' ; \Gamma \vdash \sigma : \Psi}{\Delta, u::A[\Psi], \Delta' ; \Gamma \vdash u[\sigma] : [\sigma]A}^{\text{ctxhyp}}$$
Contextual modal type theory

• Modal variables $u$ defined in a modal context $\Delta$
  – $u :: A[\Psi]$: term which may refer to ordinary variables in $\Psi$
  – modal variables = meta-variable

• Ordinary variables $x$ defined in a context $\Gamma$
  – $x :: A$ stands for any term
  – ordinary variables can be bound by lambda
Modal substitutions

\[ \text{modal substitutions} \]

\[
\begin{align*}
[ M/u ](x) & = x \\
[ M/u ](\lambda y:B.\ N) & = \lambda y:B.\ [ M/u ]N \\
[ M/u ](N_1\ N_2) & = ([ M/u ]N_1)\ ([ M/u ]N_2) \\
[ M/u ](u[\tau]) & = ([ M/u ]\tau)M \\
[ M/u ](v[\tau]) & = v([ M/u ]\tau)
\end{align*}
\]

provided \( v \neq u \)

- No side condition on the \( \lambda \)-rule!
- Modal substitution allows in place update!
$\Delta, u::A[\Psi], \Delta'; \Gamma \vdash \sigma : \Psi$

$\Delta, u::A[\Psi], \Delta'; \Gamma \vdash u[\sigma] : [\sigma]A$

A meta-variable $u$ can depend exactly on the ordinary variables in $\Psi$. 
There must be a linear order for meta-variables.

Clarifies dependencies among meta-variables in the dependently typed case.
Meta-theoretic properties

- Subject reduction and expansion.
- Strong normalization.
- Lowering and raising.
- Bi-directional type-checking decidable.
Related Work: meta-variables

• Explicit substitution calculi
  • Simple types[Dowek’95, Dowek’96], Dependent types[Munoz’01, 00]
  • Associate explicit substitutions with any term
  • Pre-cooking and grafting

• Calculus with meta-variables [Strecker’99]
  • Meta-variables are associated with substitutions
  • No context for meta-variables
  • Decidability of type-checking not obvious.
Related Work

• Incomplete proofs (or Proofs with holes) [Magnusson’ 95, Geuvers’02, Jojgov’02, Bognar’01]
  • Holes in proofs = meta-variables
  • Build on explicit substitution calculi
  • Reduction and instantiation of meta-variables do not commute

• Calculus of meta-variables [Sato’03]
  • Meta-variables are associated with levels
  • Textual substitution with capture
  • Loss of confluence and decidability of type-checking.
• Logical foundation for calculi with meta-variables
  ICML (Nanevski, Pfenning, Pientka’05)
  (earlier version LFM’03)

• Applications:
  • Meta-variables (Theorem proving)
  • Staged computation (Functional programming)
  • Reasoning about different view-points/contexts (AI)
Outline

- Logical frameworks and certified code
- Contextual modal type theory
- Applications: Higher-order unification
- Conclusion and future work
Higher-order unification

- Solving equations in the presence of $\lambda$-abstraction
- Undecidable for second order [Huet’73] [Goldfarb’81]
- Central in higher-order logic and type theory
  - General proof search
  - Logic program execution
  - Type and term reconstruction
  - Partial proofs
Tractable cases

• Pre-unification often practical [Huet’75]
  - Some solvable equations are postponed
  - Non-determinism major drawback in practice

• Pattern unification decidable [Miller’91]
  - Restricting $\beta$-reduction to $\beta_0 : (\lambda x.M)y \rightarrow [y/x]M$
  - Most general unifiers exist.
  - Extends to complex theories [Pf’91]
  - Higher-order patterns as a calculus of variable binding, variable occurrence and renaming
Example revisited

- Unification problem:

\[
\text{eq } (\text{all } (\lambda y. \text{imp } (\text{imp } (p y) (p y))) q) \Rightarrow w[\cdot]
\]

\[
\Rightarrow
\]

\[
\text{eq } (\text{all } (\lambda y. \text{imp } u[y/x] v[\cdot])) (\text{imp } (\text{exists } (\lambda y. u[y/x])) v[\cdot])
\]

- Meta-variables: \( \Delta = u::o[x:i], \ v::o[\cdot], \ w::o[\cdot] \)

- Solve sub-problems:

\[
\Delta ; y:o \vdash u[y/x] \Rightarrow (\text{imp } (p y) (p y))
\]

\[
\Delta ; y:o \vdash v[\cdot] \Rightarrow q
\]

\[
\Delta ; y:o \vdash (\text{imp } (\text{exists } (\lambda y. u[y/x])) v[\cdot]) \Rightarrow w[\cdot]
\]
Solving higher-order patterns

- Higher-order patterns:
  Terms \( M ::= x | u[\sigma] | \lambda x:A.M | M_1 M_2 \)
  Subst. \( \sigma ::= \cdot | \sigma, y/x \)

- Judgment: \( \Delta; \Gamma \vdash M \equiv N/(\theta, \Delta') \)

- To solve: \( \Delta ; y:o \vdash u[y/x] \equiv (\text{imp} \ (\text{p} \ y) \ (\text{p} \ y)) \)

  where \( \Delta = u::o[x:i], \ v::o[, \ w::[.] \)

  - Check for occurrences of \( u \) (occurs-check)
  - Check that \( [y/x]^{-1} \ (\text{imp} \ (\text{p} \ y) \ (\text{p} \ y)) \) exists (pruning)
  - Substitute (with apparent capture): \( [(\text{imp} \ (\text{p} \ x) \ (\text{p} \ x)) / u] \)
Case: \((\Delta_1, \ u::Q[\Psi], \ \Delta_2); \Gamma \vdash u[\sigma] \vdash M/\ldots\)

- \(\Delta; \Gamma \vdash M : Q'\)
- \(\Delta; \Gamma \vdash u[\sigma] : Q'\)
- Can \(u\) occur in \(Q'\)?
Occurs check

Case: \((\Delta_1, \; u::Q[\Psi], \; \Delta_2); \; \Gamma \vdash u[\sigma] \equiv M/\ldots\) 

- \(\Delta; \; \Gamma \vdash M : Q'\)
- \(\Delta; \; \Gamma \vdash u[\sigma] : Q'\)
- \(\Delta; \; \Gamma \vdash u[\sigma] : [\sigma]Q\)
- \(\Delta; \; \Gamma \vdash \sigma : \Psi.\)  
  by rule
Case: $(\Delta_1, \ u::Q[\Psi], \ \Delta_2); \Gamma \vdash u[\sigma] = M/\ldots$

- $\Delta; \Gamma \vdash M : Q'$
- $\Delta; \Gamma \vdash u[\sigma] : Q'$
- $\Delta; \Gamma \vdash u[\sigma] : [\sigma]Q$
- $\Delta; \Gamma \vdash \sigma : \Psi$. \text{ by rule}$
- $Q' = [\sigma]Q$ \text{ by previous lines}
Occurs check

Case: \((\Delta_1, \ u::Q[\Psi], \ \Delta_2); \Gamma \vdash u[\sigma] \equiv M/\ldots\)

- \(\Delta; \Gamma \vdash M : Q'\)
- \(\Delta; \Gamma \vdash u[\sigma] : Q'\)
- \(\Delta; \Gamma \vdash u[\sigma] : [\sigma]Q\)
- \(\Delta; \Gamma \vdash \sigma : \Psi.\) \quad \text{by rule}
- \(Q' = [\sigma]Q\) \quad \text{by previous lines}
- \(\Delta_1; \Psi \vdash Q : \text{type} \quad \text{well-typed}\)
- \(\Delta_1; \Gamma \vdash [\sigma]Q : \text{type since } \sigma \text{ is a pattern substitution}\)
Occurs check

Case: \((\Delta_1, \ u::Q[\Psi], \ \Delta_2); \Gamma \vdash u[\sigma] \doteq M/\ldots\)

- \(\Delta; \Gamma \vdash M : Q'\)
- \(\Delta; \Gamma \vdash u[\sigma] : Q'\)
- \(\Delta; \Gamma \vdash u[\sigma] : [\sigma]Q\)
- \(\Delta; \Gamma \vdash \sigma : \Psi\). \hspace{1cm} \text{by rule}
- \(Q' = [\sigma]Q\) \hspace{1cm} \text{by previous lines}
- \(\Delta_1; \Psi \vdash Q : \text{type}\) \hspace{1cm} \text{well-typed}
- \(\Delta_1; \Gamma \vdash [\sigma]Q : \text{type since } \sigma \text{ is a pattern substitution}\)

No occurs check on \(Q'\) necessary!
Unification with a meta-variable

Case: \( \Delta; \Gamma \vdash u[\sigma] \equiv M/\ldots \)
where \( \Delta = \Delta_1, \ u :: Q[\Psi], \ \Delta_2 \)

Prune \( M \) with respect to \( \sigma \) s.t. \( (u \text{ does not occur in } M) \)

- \( \rho \) is a (modal) pruning substitution s.t. \( [\sigma]^{-1}(\llbracket \rho \rrbracket M) \) exists
- \( \Delta' \vdash \rho : \Delta \)
Case: $\Delta; \Gamma \vdash u[\sigma] \equiv M/\ldots$
where $\Delta = \Delta_1, \ u::Q[\Psi], \Delta_2$

Prune $M$ with respect to $\sigma$ s.t. ($u$ does not occur in $M$)

- $\rho$ is a (modal) pruning substitution s.t. $[\sigma]^{-1}([\rho]M)$ exists
- $\Delta' \vdash \rho : \Delta$ and $\rho = (\rho_1, u/u, \rho_2)$
- $\Delta' = (\Delta'_1, u::[\rho]Q[[\rho]\Psi], \Delta'_2)$
Unification with a meta-variable

Case: $\Delta; \Gamma \vdash u[\sigma] = M/(\theta, \Delta^*)$

where $\Delta = \Delta_1, \ u::Q[\Psi], \Delta_2$

Prune $M$ with respect to $\sigma$ s.t. ($u$ does not occur in $M$)

- $\rho$ is a (modal) pruning substitution s.t. $[\sigma]^{-1} ([\rho] M)$ exists
- $\Delta' \vdash \rho : \Delta$ and $\rho = (\rho_1, u/u, \rho_2)$
- $\Delta' = (\Delta'_1, u::[\rho] Q[[\rho] \Psi], \Delta'_2)$
- $\theta = (\rho_1, [\sigma]^{-1} ([\rho] M)/u, \rho_2)$
  $\quad = (\text{id}_{\Delta'_1}, [\sigma]^{-1} ([\rho] M)/u, \text{id}_{\Delta'_2})\rho$
- $\Delta^* = (\Delta'_1, [\text{id}_{\Delta'_1}, [\sigma]^{-1} ([\rho] M)/u] \Delta'_2)$
Meta-theoretic properties [Pi’03]

- Modal substitutions and pattern substitutions commute.
- Correctness of pruning for dependent types.
  - Pruning of types or context not necessary.
  - Occurs check on types not necessary.
- Correctness of higher-order pattern unification for dependent types.
  - Well-typed simultaneous substitutions for meta-variables
Contributions

- High-level description of higher-order pattern unification for dependent types
  no de Bruijn indices
- Logical foundation based on modal type-theory:
  - Meta-variables = modal variables
  - Strong invariants about modal and ordinary variables.
- Post-hoc justification of implementation in Twelf
- Insights into optimizations:
  - Linearization [Pi’03]
  - Omitting redundant dependent types (current work)
Outline

- Logical frameworks and certified code
- Contextual modal type theory
- Applications: Higher-order unification
- Conclusion and future work
Summary and future work

- Contextual modal type theory
  Foundation for meta-variables and explicit substitutions
- High-level explanation (no de Bruijn indices!)
- Basis for other algorithms:
  - Higher-order term indexing [Pi’03]
  - Proof search [Pie’02]
  - Redundant type elimination (current work)
Future work

- Further development of contextual modal logic
  - Contextual possibility
  - Dependent necessity
  - Internalizing explicit substitutions
- Applications to logical frameworks:
  - Omitting redundant dependent types
  - General higher-order unification
  - Variable definitions