Towards a Type Theory of Contexts

Frank Pfenning

Carnegie Mellon University

Invited Talk

Workshop on Mechanized Reasoning about Languages with Variable Binding (Merλin'05)

Tallinn, Estonia, September 30, 2005

Joint work with Aleks Nanevski and Brigitte Pientka

Work in progress!

Context

- Dictionary definition (Merriam-Webster Online)
  a. *the parts of a discourse that surround a word or passage and can throw light on its meaning*
  b. *the interrelated conditions in which something exists or occurs*
- Of central importance in computer science
  - Computational Linguistics
  - Artificial Intelligence
  - Programming Languages
- Not very well understood

Logic and Type Theory

- *Logic* in this talk always means *intuitionistic logic* and is therefore immediately computational
  - Logical frameworks
  - Functional programming
- *Type theory* makes proof terms explicit
  - Canonical forms
  - Evaluation

Outline

- Goals and Methods
- Validity
- Contextual Validity
- Context and Substitution Variables
- Conclusion

Original Goals

- Understanding *meta-variables*
  - Logical essence
  - Simple and dependent types
  - Reflection?
- In logical frameworks
  - Unification
  - Proof search (subgoals)
- For staged computation
  - Manipulating “open code”
  - Filling holes

Spin-offs

- Understanding *explicit substitutions*
  - Logical foundation
  - Substitution variables, quantification
  - Sequent calculi?
- Understanding *contexts*
  - Logical meaning
  - Context variables
  - Context quantification (?)

Methodology

- Separating judgments from propositions [Martin-Löf'83]
- Categorical judgments [Pf.&Davies'01]
- Other applications of modal type theory
  - Monads [Moggi'88,'91][Pf.&Davies'01]
  - Run-time code generation [Davies&Pf.'96,'01]
  - Partial evaluation [Davies'96]
  - Distributed computation [Murphy,Crary,Harper,Pf.'04]

Some Consequences and Criteria

- Respecting α-conversion
- Computation arises from reduction
- Normalization
- Canonical forms
- Orthogonality of language constructs
  - Adding other propositions, types
  - Modular reasoning

Hypothetical Judgments

- Basic judgment $A\ true$
- Hypothetical judgment

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

- Hypothesis rule

$$\frac{A\ true \in \Gamma}{\Gamma \vdash A\ true}$$

- Substitution principle

  If $\Gamma \vdash A\ true$ and $\Gamma, A\ true \vdash C\ true$ then $\Gamma \vdash C\ true$

Modal Logic of Validity

- Categorical judgment

$$\frac{\bullet \vdash A \; true}{A \; valid}$$

- Generalized hypothetical judgment

$$\underbrace{B_1 \; valid, \ldots, B_m \; valid}_{\Delta}; \underbrace{A_1 \; true, \ldots, A_n \; true}_{\Gamma} \vdash C \; true$$

- Generalized definition of validity

$$\frac{\Delta; \bullet \vdash A \; true}{\Delta; \Gamma \vdash A \; valid}$$

Modal Logic of Validity, Ctd

- New hypothesis rule

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

- New substitution principle

  If $\Delta; \bullet \vdash A\ true$ and $\Delta, A\ valid; \Gamma \vdash C\ true$ then $\Delta; \Gamma \vdash C\ true$

Simple Type Theory of Validity

- Assign proof terms

$$\underbrace{u_1{::}B_1, \ldots, u_m{::}B_m}_{\Delta}; \underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\Gamma} \vdash M : C$$

- Hypothesis rules

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

- Ordinary substitution principle

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

Substitution Operations

- Modal substitution principle

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

- Ordinary substitution $[M/x]N$ as usual
- Modal substitution $[\![M/u]\!]N$ slightly unusual

$$
\begin{array}{rcll}
[M/x](\lambda y.\, N) & = & \lambda y.\, [M/x]N & \text{for } y \notin \text{FV}(M) \\
[\![M/u]\!](\lambda y.\, N) & = & \lambda y.\, [\![M/u]\!]N & \textit{without proviso}
\end{array}
$$

- $[\![M/u]\!]$ can be implemented by “grafting”

Excursion: Higher-Order Unification

- Huet's formulation
- Substituends for meta-variables
  - Have no free ordinary variables
  - May contain other meta-variables

$$\lambda x.\, \lambda y.\, \lambda z.\, u_1\, x\, y \doteq \lambda x.\, \lambda y.\, \lambda z.\, u_2\, y\, z$$

$$u_1 \longleftarrow (\lambda x.\, \lambda y.\, u_3\, y)$$

$$u_2 \longleftarrow (\lambda y.\, \lambda z.\, u_3\, y)$$

- *Precisely modeled by validity*

Internalizing Validity

- Validity is a categorical judgment, not type
- Internalize as modal type constructor
- By Curry-Howard, can be read as proposition

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

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

Computation from Reduction

- Reduce when eliminations follow introductions

$$(\beta) \quad \text{let box } u = \text{box } M \text{ in } N \longrightarrow [\![M/u]\!]N$$

- Expand to create eliminations followed by introductions

$$(\bar{\eta}) \quad M : \Box A \longrightarrow \text{let box } u = M \text{ in box } u$$

- Omit reduction, expansion for $A \rightarrow B$
- Reduction and expansion preserve types
  - Follows from substitution properties

Excursion: Staged Computation

- $\Box A$ is *source code of type A*
- Implement as run-time code generation

$$
\begin{array}{lcl}
\mathit{exp} & : & \mathit{nat} \rightarrow \Box(\mathit{nat} \rightarrow \mathit{nat}) \\
\\
\mathit{exp}\ 0 & = & \text{box}\ (\lambda x.\, 1) \\
\mathit{exp}\ 1 & = & \text{box}\ (\lambda x.\, x) \\
\mathit{exp}\ n & = & \text{let box}\ u = \mathit{exp}\ (n-1)\ \text{in box}\ (\lambda x.\, (u\, x) * x) \\
 & & \text{for}\ n \geq 2 \\
\\
\mathit{exp}\ 2 & \mapsto^* & \text{box}\ (\lambda x_2.\, (\lambda x_1.x_1)\, x_2 * x_2) \\
 & \not\mapsto^* & \text{box}\ (\lambda x_2.\, x_2 * x_2)
\end{array}
$$

Characteristic Laws

- Laws of (intuitionistic) S4

$$\vdash \Box A \to A$$

$$\vdash \Box A \to \Box\Box A$$

$$\vdash \Box(A \to B) \to \Box A \to \Box B$$

$$\nvdash A \to \Box A$$

- Also need necessitation ($\approx \Box I$)
- Kripke interpretation
  - $\Box A$ $true$ means A $true$ in all future worlds
  - Accessibility is reflexive and transitive

Contextual Validity

- $A\ \mathit{valid}[\Psi]$ means A is true in every world where all hypothesis in context Ψ are satisfied
- Generalized hypothetical judgment

$$\underbrace{B_1\ \mathit{valid}[\Psi_1], \ldots, B_m[\Psi_m]\ \mathit{valid}}_{\Delta}; \underbrace{A_1\ \mathit{true}, \ldots, A_n\ \mathit{true}}_{\Gamma} \vdash C\ \mathit{true}$$

- Generalized definition of contextual validity

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

Validity and Contextual Validity

- $A\ valid[\bullet]$ corresponds to $A\ valid$
- $A\ valid[A_1\ true, \ldots, A_n\ true]$ like $(A_1 \rightarrow \cdots \rightarrow A_n \rightarrow A)\ valid$
  - Proof theory and computation is quite different

Proof Terms

- Assign proof terms

$$\underbrace{u_1{::}B_1[\Psi_1], \ldots, u_m{::}B_m[\Psi_m]}_{\Delta};\underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\Gamma} \vdash M : C$$

- Contextual hypothesis rule

$$\frac{u{::}A[\Psi] \in \Delta \quad \Delta;\Gamma \vdash \sigma : \Psi}{\Delta;\Gamma \vdash u[\sigma] : A}$$

- Requires *explicit substitution* σ to establish that all assumptions in Ψ can be realized in Γ

Explicit Substitutions

- Simultaneous substitutions

$$\text{Substitutions} \quad \sigma \quad ::= \quad \bullet \mid \sigma, M/x$$

- Typing judgment $\Delta; \Gamma \vdash \sigma : \Psi$

$$\frac{}{\Delta; \Gamma \vdash \bullet : \bullet} \qquad \frac{\Delta; \Gamma \vdash \sigma : \Psi \quad \Delta; \Gamma \vdash M : A}{\Delta; \Gamma \vdash (\sigma, M/x) : (\Psi, x{:}A)}$$

- Not just a tuple (cf dependencies)

Substitution Principle

- Substitution principles for contextual validity

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

- Also need to substitute into explicit substitutions!

  If $\Delta; \Psi \vdash M : A$ and $\Delta, u{::}A[\Psi]; \Gamma \vdash \tau : \Phi$ then $\Delta; \Gamma \vdash [\![\Psi.M/u]\!]\tau : \Phi$

- Close M over Ψ to preserve α-conversion
  - Not necessary in nameless implementation

Substitution Operations

- $[\![\Psi.M/u]\!]N$ as before, except

$$
\begin{array}{rcll}
[\![\Psi.M/u]\!](w[\sigma]) & = & w[[\![\Psi.M/u]\!]\sigma] & \text{for } u \neq w \\
[\![\Psi.M/u]\!](u[\sigma]) & = & [\sigma'/\Psi](M) & \text{where } \sigma' = [\![\Psi.M/u]\!]\sigma
\end{array}
$$

- σ'/Ψ renames variables in σ' to match Ψ
- Need to awaken the postponed substitution σ
- Different occurrences of u may be under different substitutions

Simultaneous Substitution

- Simultaneous substitution principles

  If $\Delta; \Psi \vdash \sigma : \Gamma$ and $\Delta; \Gamma \vdash M : A$ then $\Delta; \Psi \vdash [\sigma]M : A$

- Definition of $[\sigma]M$ is straightforward, for example

$$\begin{array}{lcll} [\sigma](x) & = & M & \text{where } M/x \in \sigma \\ [\sigma](u[\tau]) & = & u[[\sigma]\tau] & \\ [\sigma](\lambda x.\, M) & = & \lambda x.\, [\sigma, x/x] & \end{array}$$

- Composition of substitutions (also straightforward)

  If $\Delta; \Psi \vdash \sigma : \Gamma$ and $\Delta; \Gamma \vdash \tau : \Phi$ then $\Delta; \Psi \vdash [\sigma]\tau : \Phi$

Excursion: Unification Revisited

- Meta-variables á la Dowek et al. in h.o. unification
- Substituends for meta-variables
  - Have free ordinary variables
  - Occur under explicit substitution

$$u_1{::}i[x{:}i, y{:}i], u_2{::}i[y{:}i, z{:}i]$$

$$\lambda x.\, \lambda y.\, \lambda z.\, u_1[x/x, y/y] \doteq \lambda x.\, \lambda y.\, \lambda z.\, u_2[y/y, z/z]$$

$$u_1 \longleftarrow u_3[y/y]$$

$$u_2 \longleftarrow u_3[y/y]$$

$$\text{for} \quad u_3{::}i[y{:}i]$$

Internalizing Contextual Validity

- New contextual modal operator $[\Psi]A$

$$\frac{\Delta; \Psi \vdash M : A}{\Delta; \Gamma \vdash \mathsf{box}\ (\Psi.M) : [\Psi]A}\ [-]I$$

$$\frac{\Delta; \Gamma \vdash M : [\Psi]A \quad \Delta, u{::}A[\Psi]; \Gamma \vdash N : C}{\Delta; \Gamma \vdash \mathsf{let\,box}\ u = M\ \mathsf{in}\ N : C}\ [-]E$$

- Reduction

$$(\beta) \quad \mathsf{let\,box}\ u = \mathsf{box}\ (\Psi.M)\ \mathsf{in}\ N \longrightarrow [\![(\Psi.M)/u]\!]N$$

Identity Substitutions

- Expansion requires identity substitution

$$(\bar{\eta}) \quad M : [\Psi]A \longrightarrow \mathsf{let\ box}\ u = M\ \mathsf{in\ box}\ (\Psi.u[\mathsf{id}(\Psi)])$$

- Define

$$\begin{aligned} \mathsf{id}(\bullet) &= \bullet \\ \mathsf{id}(\Psi, x{:}A) &= \mathsf{id}(\Psi), x/x \end{aligned}$$

- Reduction and expansion preserve types

Excursion: Staging Revisited

- Avoid creating redexes in code generation
- Manipulate open code
  - $[\Psi]A$ — source with free variables in Ψ

$$
\begin{array}{lcl}
exp & : & nat \rightarrow [x{:}nat]nat \\
 & & \\
exp\ 0 & = & \mathsf{box}\ (x.\,1) \\
exp\ 1 & = & \mathsf{box}\ (x.\,x) \\
exp\ n & = & \mathsf{let\ box}\ u = exp\ (n-1)\ \mathsf{in\ box}\ (x.\,(u[x/x]) * x) \\
 & & \mathsf{for}\ n \geq 2 \\
 & & \\
exp\ 2 & \not\mapsto^* & \mathsf{box}\ (\lambda x_2.\,(\lambda x_1.x_1)\, x_2 * x_2) \\
 & \mapsto^* & \mathsf{box}\ (x_2.\,x_2 * x_2)
\end{array}
$$

Some Sample Theorems

$$\vdash [\bullet]A \longrightarrow A$$

$$\vdash [x{:}B]A \longrightarrow [y{:}C][x{:}B]A$$

$$\vdash [x{:}C](A \longrightarrow B) \longrightarrow [x{:}C]A \longrightarrow [x{:}C]B$$

$$\vdash [x{:}A]A$$

$$\vdash [x{:}B, y{:}B]A \longrightarrow [z{:}B]A$$

$$\vdash [x{:}B]A \longrightarrow [x{:}B, y{:}C]A$$

$$\nvdash [x{:}B]A \longrightarrow A$$

$$\nvdash A \longrightarrow [x{:}B]A$$

Context Variables

- Theorems parametric in *propositions* (A,B,C)

$$\vdash [x{:}B]A \rightarrow [y{:}C][x{:}B]A$$

- Would like theorems parametric in *contexts*
- Write ψ, ϕ, γ for *context variables*

$$\vdash [\psi]A \rightarrow [\phi][\psi]A$$

$$\lambda x{:}[\psi]A.\, \mathsf{let\ box}\ u = x\ \mathsf{in\ box}\ (\phi.\ \mathsf{box}\ (\psi.\ u[\mathsf{id}_\psi]))$$

Identity Substitutions

- Extended language

Contexts	Γ	$::=$	$\bullet \mid \Gamma, x{:}A \mid \gamma$
Substitutions	σ	$::=$	$\bullet \mid \sigma, M/x \mid \mathsf{id}_\gamma$

- Typing

$$\frac{}{\Delta; \gamma, \Gamma \vdash \mathsf{id}_\gamma : \gamma}$$

- Substituting for context variables
  - box $(\Psi.\, M)$ does not bind context variable at head of Ψ
  - $\{\Gamma/\gamma\}(\mathsf{id}_\gamma) = \mathsf{id}(\Gamma)$ expands

Sample Theorems Revisited

$$\vdash [\bullet]A \longrightarrow A$$

$$\vdash [\psi]A \longrightarrow [\phi][\psi]A$$

$$\vdash [\psi](A \longrightarrow B) \longrightarrow [\psi]A \longrightarrow [\psi]B$$

$$\vdash [x{:}A]A$$

$$\vdash [\psi, x{:}B, y{:}B]A \longrightarrow [\psi, z{:}B]A$$

$$\vdash [\psi]A \longrightarrow [\psi, x{:}B]A$$

Excursion: Closures

- Closures in functional languages, $\textsf{clo}(\eta, v)$
- Object language typing

$$\frac{\bullet \vdash \eta : \Psi \qquad \Psi \vdash v : \tau}{\bullet \vdash \textsf{clo}(\eta, v) : \tau}$$

- For direct representation, need context variables and internalized substitutions (here $\psi[\bullet]$)

$$\textsf{clo} : [\bullet]\psi \rightarrow [\psi]A \rightarrow [\bullet]A$$

Substitution Meta-Variables

- Substitution variables are meta-variables
- New substitutions and types

Types	A	$::=$	$a \mid A_1 \rightarrow A_2 \mid [\Psi]A \mid [\Psi]\Phi$
Substitutions	σ	$::=$	$\bullet \mid \sigma, M/x \mid \mathsf{id}_\gamma \mid s[\sigma]$
Meta-Contexts	Δ	$::=$	$\bullet \mid \Delta, u{::}A[\Psi] \mid \Delta, s{::}\Phi[\Psi] \mid \Delta, \gamma\ ctx$

- New hypothesis rule for substitutions

$$\frac{s{::}\Phi[\Psi] \in \Delta \quad \Delta; \Gamma \vdash \tau : \Psi}{\Delta; \Gamma \vdash s[\tau] : \Phi}$$

Extended Term Language

- Proposal for new terms, to be revised

$$\frac{\Delta; \Psi \vdash \sigma : \Phi}{\Delta; \Gamma \vdash \mathsf{sbox}\ (\Psi.\,\sigma) : [\Psi]\Phi} \qquad \frac{\Delta; \Gamma \vdash M : [\Psi]\Phi \quad \Delta, s{::}\Phi[\Psi]; \Gamma \vdash N : C}{\Delta; \Gamma \vdash \mathsf{let\,sbox}\ s = M\ \mathsf{in}\ N : C}$$

- Substitution properties, reduction straightforward
- Example, internalizes substitution property

$$\vdash \lambda x.\,\lambda y.\,\mathsf{let\,sbox}\ s = x\ \mathsf{in}\ \mathsf{let\,box}\ u = y\ \mathsf{in}\ \mathsf{box}\ (\psi.\,u[s[\mathsf{id}_\psi]])$$

$$: [\psi]\phi \to [\phi]A \to [\psi]A$$

Destructing Substitutions

- Problem: there are no destructors for substitutions

$$[\bullet](\phi, x{:}A) \to [\bullet]\phi$$

$$\lambda y.\, \mathsf{let\,sbox}\ s = y\ \mathsf{in\ sbox}\ ??$$

- Solution 1: projections of substitutions (hd, tl_x)

$$\lambda y.\, \mathsf{let\,sbox}\ s = y\ \mathsf{in\ sbox}\ (\mathsf{hd}(s[\mathsf{id}_\phi, x/x]))$$

- Solution 2: pattern matching for substitutions

$$\lambda y.\, \mathsf{let\,sbox}\ s_\phi, u_x = y\ \mathsf{in\ sbox}\ (s[\mathsf{id}_\phi])$$

Pattern Matching Substitutions

- Tentatively adopt solution 2

$$\frac{\Delta;\Gamma\vdash M:[\Psi]\Phi \quad \Delta' = \mathsf{delta}([\Psi]\Phi) \quad \Delta,\Delta';\Gamma\vdash N:C}{\Delta;\Gamma\vdash \mathsf{let\,sbox}\ \Delta' = M\ \mathsf{in}\ N:C}$$

where (subject to renaming)

$$\begin{array}{lcl} \mathsf{delta}([\Psi]\bullet) & = & \bullet \\ \mathsf{delta}([\Psi]\phi) & = & s_\phi \\ \mathsf{delta}([\Psi]\Phi, x{:}A) & = & \mathsf{delta}([\Psi]\Phi), u_x{::}A[\Psi] \end{array}$$

- Substitution for context variables expands patterns

Language Summary So Far

- Contextual modal types and simultaneous substitutions

$$\text{Types} \quad A \quad ::= \quad a \mid A_1 \rightarrow A_2 \mid [\Psi]A \mid [\Psi]\Phi$$

- Context variables
- Substitution principles
- Type preservation for reduction, expansion
- Strong normalization by interpretation (?)

Further Considerations

- Dependent types
- Context quantification
- Contextual modality like $\Diamond A$ ($\langle \Psi \rangle A$)
- Context concatenation ($\gamma\ ctx[\Psi]$)

Dependent Types

- System engineered to permit dependent types
  - Functional version (only sketched)
  - Logical frameworks version [Nanevski,Pf.,Pientka'05]
  - No context variables, internal substitutions so far
- Contexts and substitutions are dependent

$$\frac{\Delta \vdash \Gamma\ ctx \quad \Delta; \Gamma \vdash A : type}{\Delta \vdash \Gamma, x{:}A\ ctx} \qquad \frac{\Delta; \Gamma \vdash \sigma : \Psi \quad \Delta; \Gamma \vdash M : [\sigma]A}{\Delta; \Gamma \vdash (\sigma, M/x) : (\Psi, x{:}A)}$$

- Canonical forms via *hereditary substitutions* [Watkins,Cervesato,Pf.,Walker'02]

Apps: Staged Computation

- Staged computation with "open code"
- Dependent types for reasoning about staged programs
  - Modalities for correct staging
  - Dependent types for functional correctness
- Hypothetical example

$$
\begin{aligned}
exp \quad : \quad & \Pi n{:}nat.\, \Sigma r{:}[x{:}nat]nat. \\
& \quad \Pi k{:}nat.\, \text{let box } r = u \text{ in } u[k/x] \doteq k^n
\end{aligned}
$$

- Not "field-tested"

Apps: Logical Frameworks

- Model meta-variables
  - For unification
  - For proof search
- Efficient implementation via deBruijn indexes
- Logical foundation for Twelf implementation (almost)
  - Wish I had time to rewrite unification

Context Quantification

- Speculative
- $\forall\gamma.\, A$ is highly impredicative
  - May be rejected on philosophical grounds
  - Predicative form (?)
- Inductively defined regular worlds [Schürmann'00]
  - Example

$$W ::= \bullet \mid W, x{:}exp \mid W, t{:}tp$$

$$\forall\gamma \in W.\, A$$

  - Relevant to Delphin [Schürmann'02]?

Selected Other Related Work

- Philosophy, artificial intelligence [. . .many. . .]
  - Generally classical logic
  - No computational interpretation
- Functional programming or staged computation
  - $\lambda\kappa\epsilon$-calculus [Sato,Sakurai,Kameyama'02]
  - Environment classifiers [Taha&Nielson'03]
- Logical frameworks or theorem proving
  - Meta-variables with contexts [Dowek,Hardin,Kirchner'95]
  - Dependently typed in deBruijn form [Muñoz'01]
  - Axiomatic approach [Honsell,Miculan,Scagnetto'01]
  - Meta^n-variables [Sato,Sakurai,Kameyama,Igarashi'03]

Conclusion

- Goal: understanding meta-variables
- Developed contextual modal logic
  - Explicit substitutions inevitable
  - Simply and dependently typed versions
- Staged computation
  - Manipulating open code
  - Reasoning about staged programs
- Logical frameworks
  - Meta-variables for unification, search
  - Basis for efficient implementation

More Information

Contextual Modal Type Theory
Aleksandar Nanevski, Frank Pfenning, and Brigitte Pientka
Submitted, September 2005.
http://www.cs.cmu.edu/~fp/papers/cmtt05.pdf