Church and Curry: Combining Intrinsic and Extrinsic Typing

Frank Pfenning

Dedicated to Peter Andrews
on the occasion of his retirement

Department of Computer Science
Carnegie Mellon University

April 5, 2012

Church's Simple Theory of Types

- Language and logic for the formalization of mathematics
  - (Church 1940)
  - Stood the test of time (72 years!)
  - HOL, Isabelle/HOL, TPS, LEO, Satallax, ...

Church's Simple Theory of Types

- Language and logic for the formalization of mathematics
  - (Church 1940)
  - Stood the test of time (72 years!)
  - HOL, Isabelle/HOL, TPS, LEO, Satallax, ...
- Synthesis, simplification, and generalization of
  - Russell and Whitehead's ramified theory of types
  - Church and Rosser's (untyped) λ-calculus

Church's Simple Theory of Types

- Language and logic for the formalization of mathematics
  - (Church 1940)
  - Stood the test of time (72 years!)
  - HOL, Isabelle/HOL, TPS, LEO, Satallax, ...
- Synthesis, simplification, and generalization of
  - Russell and Whitehead's ramified theory of types
  - Church and Rosser's (untyped) λ-calculus
- Some objections

Church's Simple Theory of Types

- Language and logic for the formalization of mathematics
  - (Church 1940)
  - Stood the test of time (72 years!)
  - HOL, Isabelle/HOL, TPS, LEO, Satallax, ...
- Synthesis, simplification, and generalization of
  - Russell and Whitehead's ramified theory of types
  - Church and Rosser's (untyped) λ-calculus
- Some objections
  - As a computer scientist: classical

Church's Simple Theory of Types

- Language and logic for the formalization of mathematics
  - (Church 1940)
  - Stood the test of time (72 years!)
  - HOL, Isabelle/HOL, TPS, LEO, Satallax, ...
- Synthesis, simplification, and generalization of
  - Russell and Whitehead's ramified theory of types
  - Church and Rosser's (untyped) λ-calculus
- Some objections
  - As a computer scientist: classical
  - As a philosopher: impredicative

Church's Simple Theory of Types

- Language and logic for the formalization of mathematics
  - (Church 1940)
  - Stood the test of time (72 years!)
  - HOL, Isabelle/HOL, TPS, LEO, Satallax, . . .
- Synthesis, simplification, and generalization of
  - Russell and Whitehead's ramified theory of types
  - Church and Rosser's (untyped) λ-calculus
- Some objections
  - As a computer scientist: classical
  - As a philosopher: impredicative
- Components

Church's Simple Theory of Types

- Language and logic for the formalization of mathematics
  - (Church 1940)
  - Stood the test of time (72 years!)
  - HOL, Isabelle/HOL, TPS, LEO, Satallax, ...
- Synthesis, simplification, and generalization of
  - Russell and Whitehead's ramified theory of types
  - Church and Rosser's (untyped) λ-calculus
- Some objections
  - As a computer scientist: classical
  - As a philosopher: impredicative
- Components
  - Simply typed λ-calculus (this talk)
  - Logical axioms and inference rules

Simply Typed λ-Calculus

- Church's definitions

Simply Typed λ-Calculus

- Church's definitions
- Types
  1. ι and o are types.
  2. If α and β are types, then $\alpha \to \beta$ is a type.
     (Church wrote $\beta\alpha$)

Simply Typed λ-Calculus

- Church's definitions
- Types
  1. ι and o are types.
  2. If α and β are types, then $\alpha \to \beta$ is a type. (Church wrote $\beta\alpha$)
- Well-formed terms M^α of type α
  1. Any variable x^α or constant c^α is a term.
  2. If x^α is a variable and M^β a term then $(\lambda x.\, M)^{\alpha\to\beta}$ is a term.
  3. If $M_1^{\alpha\to\beta}$ and M_2^α are terms, then $(M_1\, M_2)^\beta$ is a term.

Intrinsic Typing

- Every well-formed term has an intrinsic type, including variables

Intrinsic Typing

- Every well-formed term has an intrinsic type, including variables
- This kind of intrinsic formulation has become rare, but it has a number of advantages

Intrinsic Typing

- Every well-formed term has an intrinsic type, including variables
- This kind of intrinsic formulation has become rare, but it has a number of advantages
- Supports conventions, such as

  > *In the remainder of this [talk] we assume that all terms are well-formed according to the above definition.*

Intrinsic Typing

- Every well-formed term has an intrinsic type, including variables
- This kind of intrinsic formulation has become rare, but it has a number of advantages
- Supports conventions, such as

  > *In the remainder of this [talk] we assume that all terms are well-formed according to the above definition.*

- In a logical framework

```
tp : type.
arrow : tp -> tp -> tp.
tm : tp -> type.
lam : (tm A -> tm B) -> tm (arrow A B).
app : tm (arrow A B) -> tm A -> tm B.
```

Untyped λ-Calculus

- (Church 1932) (Church and Rosser 1936)
- Terms
  1. Any variable x or constant c is a term.
  2. If x is a variable and M a term then $(\lambda x.\, M)$ is a term.
  3. If M_1 and M_2 are terms, then $(M_1\, M_2)$ is a term.

Extrinsic Typing

- (Curry 1934) [for combinators, not λ-terms]
- Types as properties of terms

Extrinsic Typing

- (Curry 1934) [for combinators, not λ-terms]
- Types as properties of terms
- Typing judgments defined by rules

$$\frac{x{:}\alpha \in \Gamma}{\Gamma \vdash x : \alpha} \qquad \frac{c{:}\alpha \in \Sigma}{\Gamma \vdash c : \alpha}$$

$$\frac{\Gamma, x{:}\alpha \vdash M : \beta \quad (x \notin \mathrm{dom}(\Gamma))}{\Gamma \vdash \lambda x.\, M : \alpha \to \beta}$$

$$\frac{\Gamma \vdash M : \alpha \to \beta \quad \Gamma \vdash N : \alpha}{\Gamma \vdash M\, N : \beta}$$

Types as Properties

- A term can have multiple types

$$\cdot \vdash \lambda x.\, x : \iota \to \iota$$
$$\cdot \vdash \lambda x.\, x : (\iota \to \iota) \to (\iota \to \iota)$$

Types as Properties

- A term can have multiple types

$$\cdot \vdash \lambda x.\, x : \iota \to \iota$$
$$\cdot \vdash \lambda x.\, x : (\iota \to \iota) \to (\iota \to \iota)$$

- Express explicitly in the form of a single type

Types as Properties

- A term can have multiple types

$$\cdot \vdash \lambda x.\, x : \iota \to \iota$$
$$\cdot \vdash \lambda x.\, x : (\iota \to \iota) \to (\iota \to \iota)$$

- Express explicitly in the form of a single type
- Parametric polymorphism (universal types)

$$\cdot \vdash \lambda x.\, x : \forall t.\, t \to t$$

Types as Properties

- A term can have multiple types

$$\cdot \vdash \lambda x.\, x : \iota \to \iota$$
$$\cdot \vdash \lambda x.\, x : (\iota \to \iota) \to (\iota \to \iota)$$

- Express explicitly in the form of a single type
- Parametric polymorphism (universal types)

$$\cdot \vdash \lambda x.\, x : \forall t.\, t \to t$$

- Ad hoc polymorphism (intersection types)

$$\cdot \vdash \lambda x.\, x : (\iota \to \iota) \wedge ((\iota \to \iota) \to (\iota \to \iota))$$

Extrinsic Rules

- Parametric polymorphism

$$\frac{\Gamma \vdash M : \beta \quad (t \notin \mathrm{ftv}(\Gamma))}{\Gamma \vdash M : \forall t.\, \beta}\ \forall I \qquad \frac{\Gamma \vdash M : \forall t.\, \alpha}{\Gamma \vdash M : [\beta/t]\alpha}\ \forall E$$

Extrinsic Rules

- Parametric polymorphism

$$\frac{\Gamma \vdash M : \beta \quad (t \notin \mathrm{ftv}(\Gamma))}{\Gamma \vdash M : \forall t.\, \beta}\ \forall I \qquad \frac{\Gamma \vdash M : \forall t.\, \alpha}{\Gamma \vdash M : [\beta/t]\alpha}\ \forall E$$

- Intersection polymorphism

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash M : B}{\Gamma \vdash M : A \wedge B}\ \wedge I$$

$$\frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash M : A}\ \wedge E_1 \qquad \frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash M : B}\ \wedge E_2$$

Types as Properties: The Good News

- Can type terms more generally

$$\cdot \vdash \lambda x.\, x\, x : (\forall \alpha.\, \alpha \to \alpha) \to (\forall \beta.\, \beta \to \beta)$$
$$\cdot \vdash \lambda x.\, x\, x : ((\iota \to \iota) \wedge \iota) \to \iota$$

Types as Properties: The Good News

- Can type terms more generally

$$\cdot \vdash \lambda x.\, x\, x : (\forall \alpha.\, \alpha \rightarrow \alpha) \rightarrow (\forall \beta.\, \beta \rightarrow \beta)$$
$$\cdot \vdash \lambda x.\, x\, x : ((\iota \rightarrow \iota) \wedge \iota) \rightarrow \iota$$

- Can type type terms more accurately

$$\cdot \vdash \lambda x.\, \mathsf{s}(\mathsf{s}(\mathsf{s}\, x)) : (\mathsf{even} \rightarrow \mathsf{odd}) \wedge (\mathsf{odd} \rightarrow \mathsf{even})$$

Types as Properties: The Bad News

- The typing judgment is undecidable

Types as Properties: The Bad News

- The typing judgment is undecidable
- Challenge: generalize to a complete language
  - Practical
  - Useful
  - Philosophically justified
  - Easy to reason about

Layering Type Systems

- Intrinsic simple types for basic consistency
  - Avoiding Russell's paradox

Layering Type Systems

- Intrinsic simple types for basic consistency
  - Avoiding Russell's paradox
- Extrinsic sorts on well-formed terms for precision
  - Circumvent problems with general extrinsic types
  - Achieve pragmatic goals

Layering Type Systems

- Intrinsic simple types for basic consistency
  - Avoiding Russell's paradox
- Extrinsic sorts on well-formed terms for precision
  - Circumvent problems with general extrinsic types
  - Achieve pragmatic goals
- Instances
  - Intersection types to datasort refinement (this talk)
  - Dependent types to index refinements

Layering Type Systems

- Intrinsic simple types for basic consistency
  - Avoiding Russell's paradox
- Extrinsic sorts on well-formed terms for precision
  - Circumvent problems with general extrinsic types
  - Achieve pragmatic goals
- Instances
  - Intersection types to datasort refinement (this talk)
  - Dependent types to index refinements
- Parametric polymorphism is a different story

Representing Data

- Multiple techniques in Church's Type Theory
  - Church numerals
  - Constants and axioms

Representing Data

- Multiple techniques in Church's Type Theory
  - Church numerals
  - Constants and axioms
- We have only basic type ι (o is for truth values)

Representing Data

- Multiple techniques in Church's Type Theory
  - Church numerals
  - Constants and axioms
- We have only basic type ι (o is for truth values)
- Example: natural numbers
  - Constants z^{ι} and $\mathsf{s}^{\iota \to \iota}$ (constructors)
  - Constant $\mathsf{nat}^{\iota \to o}$ (predicate)
  - Axioms

$$\mathsf{nat}(\mathsf{z})$$
$$\forall x^{\iota}.\, \mathsf{nat}(x) \supset \mathsf{nat}(\mathsf{s}(x))$$

Representing Data

- Multiple techniques in Church's Type Theory
  - Church numerals
  - Constants and axioms
- We have only basic type ι (o is for truth values)
- Example: natural numbers
  - Constants z^{ι} and $\mathsf{s}^{\iota \to \iota}$ (constructors)
  - Constant $\mathsf{nat}^{\iota \to o}$ (predicate)
  - Axioms

$$\mathsf{nat}(\mathsf{z})$$
$$\forall x^{\iota}.\, \mathsf{nat}(x) \supset \mathsf{nat}(\mathsf{s}(x))$$

- Lists, trees, etc. all have type ι
  - Sort out using sorts

Simple Sorts

- Example

$$\begin{array}{lll} \mathsf{nat}^{\iota}\ \mathrm{sort} & & \\ \mathsf{z}^{\iota} & : & \mathsf{nat} \\ \mathsf{s}^{\iota\to\iota} & : & \mathsf{nat} \to \mathsf{nat} \end{array}$$

Simple Sorts

- Example

$$\begin{array}{lcl} \mathsf{nat}^{\iota}\ \mathrm{sort} & & \\ \mathsf{z}^{\iota} & : & \mathsf{nat} \\ \mathsf{s}^{\iota\to\iota} & : & \mathsf{nat} \to \mathsf{nat} \end{array}$$

- Define sorts S^{α} refining type α under signature Σ
  1. A base sort Q^{ι} declared in Σ is a simple sort.
  2. If S^{α} and T^{β} are simple sorts, then $(S \to T)^{\alpha\to\beta}$ is a simple sort.

Sorting Judgment

- Context Γ consisting of declarations $x^\alpha : S^\alpha$
- Sorting judgment $\Gamma \vdash M^\alpha : S^\alpha$

Sorting Judgment

- Context Γ consisting of declarations $x^\alpha : S^\alpha$
- Sorting judgment $\Gamma \vdash M^\alpha : S^\alpha$
- Defined only for terms of intrinsic type α and sort refining the same type α!

Sorting Judgment

- Context Γ consisting of declarations $x^\alpha : S^\alpha$
- Sorting judgment $\Gamma \vdash M^\alpha : S^\alpha$
- Defined only for terms of intrinsic type α and sort refining the same type α!
- Rules

$$\frac{x{:}S \in \Gamma}{\Gamma \vdash x : S} \qquad \frac{c{:}S \in \Sigma}{\Gamma \vdash c : S}$$

$$\frac{\Gamma, x{:}S \vdash M : T \quad (x \notin \mathrm{dom}(\Gamma))}{\Gamma \vdash \lambda x.\, M : S \to T}$$

$$\frac{\Gamma \vdash M : S \to T \quad \Gamma \vdash N : S}{\Gamma \vdash M\, N : T}$$

Subsorts

- Subsort declarations $Q_1^\iota \leq Q_2^\iota$
- New rules

$$\frac{}{Q \leq Q} \qquad \frac{Q_1 \leq Q_2 \quad Q_2 \leq Q_3}{Q_1 \leq Q_3}$$

$$\frac{\Gamma \vdash M : Q \quad Q \leq Q'}{\Gamma \vdash M : Q'}$$

- Defined on base types only
- Can derive principles for higher types

Subsorting Example

- Refining natural numbers

$$
\begin{array}{lcl}
\mathsf{zero} & \leq & \mathsf{nat} \\
\mathsf{pos} & \leq & \mathsf{nat} \\
\mathsf{z} & : & \mathsf{zero} \\
\mathsf{s} & : & \mathsf{nat} \to \mathsf{pos}
\end{array}
$$

Subsorting Example

- Refining natural numbers

$$
\begin{array}{lcl}
\mathsf{zero} & \leq & \mathsf{nat} \\
\mathsf{pos} & \leq & \mathsf{nat} \\
\mathsf{z} & : & \mathsf{zero} \\
\mathsf{s} & : & \mathsf{nat} \rightarrow \mathsf{pos}
\end{array}
$$

- Examples

$$
\begin{array}{l}
\cdot \vdash \lambda x.\, x : \mathsf{nat} \rightarrow \mathsf{nat} \\
\cdot \vdash \lambda x.\, x : \mathsf{zero} \rightarrow \mathsf{nat} \\
\cdot \vdash \lambda x.\, \lambda y.\, x\, y : (\mathsf{nat} \rightarrow \mathsf{zero}) \rightarrow (\mathsf{zero} \rightarrow \mathsf{nat}) \\
\cdot \vdash \lambda x.\, s\, x : \mathsf{nat} \rightarrow \mathsf{pos}
\end{array}
$$

Combining Properties

- Want to express and exploit multiple properties of terms

Combining Properties

- Want to express and exploit multiple properties of terms
- Example: even and odd numbers

$$
\begin{array}{lcl}
\mathsf{even} & \leq & \mathsf{nat} \\
\mathsf{odd} & \leq & \mathsf{nat} \\
\mathsf{z} & : & \mathsf{even} \\
\mathsf{s} & : & \mathsf{even} \rightarrow \mathsf{odd} \\
\mathsf{s} & : & \mathsf{odd} \rightarrow \mathsf{even}
\end{array}
$$

Combining Properties

- Want to express and exploit multiple properties of terms
- Example: even and odd numbers

$$
\begin{array}{lcl}
\mathsf{even} & \leq & \mathsf{nat} \\
\mathsf{odd} & \leq & \mathsf{nat} \\
\mathsf{z} & : & \mathsf{even} \\
\mathsf{s} & : & \mathsf{even} \to \mathsf{odd} \\
\mathsf{s} & : & \mathsf{odd} \to \mathsf{even}
\end{array}
$$

- Have no way to express in one sort:

$$
\begin{array}{l}
\cdot \vdash \lambda x^{\iota}.\, \mathsf{s}(\mathsf{s}(\mathsf{s}\, x)) : \mathsf{even} \to \mathsf{odd} \\
\cdot \vdash \lambda x^{\iota}.\, \mathsf{s}(\mathsf{s}(\mathsf{s}\, x)) : \mathsf{odd} \to \mathsf{even}
\end{array}
$$

Intersection Sorts

- Define sorts S^α refining types α (in intrinsic style)
  1. A base sort Q^ι declared in Σ is a sort.
  2. If S^α and T^β are sorts, then $(S \to T)^{\alpha\to\beta}$ is a sort.
  3. If S^α and T^α are sorts then $(S \wedge T)^\alpha$ is a sort.
  4. $\top^\alpha$ is a sort for each type α.

Extended Sorting Judgments

- Recall $\Gamma \vdash M^{\alpha} : S^{\alpha}$

Extended Sorting Judgments

- Recall $\Gamma \vdash M^\alpha : S^\alpha$
- New typing rules

$$\frac{\Gamma \vdash M : S_1 \quad \Gamma \vdash M : S_2}{\Gamma \vdash M : S_1 \wedge S_2} \wedge I$$

$$\frac{\Gamma \vdash M : S_1 \wedge S_2}{\Gamma \vdash M : S_1} \wedge E_1 \quad \frac{\Gamma \vdash M : S_1 \wedge S_2}{\Gamma \vdash M : S_2} \wedge E_2$$

$$\frac{}{\Gamma \vdash M : \top} \top I$$

Extended Sorting Judgments

- Recall $\Gamma \vdash M^\alpha : S^\alpha$
- New typing rules

$$\frac{\Gamma \vdash M : S_1 \quad \Gamma \vdash M : S_2}{\Gamma \vdash M : S_1 \wedge S_2} \wedge I$$

$$\frac{\Gamma \vdash M : S_1 \wedge S_2}{\Gamma \vdash M : S_1} \wedge E_1 \qquad \frac{\Gamma \vdash M : S_1 \wedge S_2}{\Gamma \vdash M : S_2} \wedge E_2$$

$$\frac{}{\Gamma \vdash M : \top} \top I$$

- Philosophically justified in the sense of Dummett/Martin-Löf

Example Revisited

- Can now conjoin properties

$$
\begin{aligned}
\cdot \vdash \lambda x^{\iota}.\, \mathsf{s}(\mathsf{s}(\mathsf{s}\, x)) \quad : \quad & (\mathsf{even} \rightarrow \mathsf{odd}) \\
& \wedge (\mathsf{odd} \rightarrow \mathsf{even}) \\
& \wedge (\mathsf{nat} \rightarrow \mathsf{pos}) \\
& \wedge \ldots
\end{aligned}
$$

Example Revisited

- Can now conjoin properties

$$\cdot \vdash \lambda x^{\iota}.\, \mathsf{s}(\mathsf{s}(\mathsf{s}\, x)) \quad : \quad \begin{array}{l} (\mathsf{even} \to \mathsf{odd}) \\ \wedge\, (\mathsf{odd} \to \mathsf{even}) \\ \wedge\, (\mathsf{nat} \to \mathsf{pos}) \\ \wedge\, \ldots \end{array}$$

- Every (well-formed) term has a principal sort

Some Results

- Sort checking is decidable
  - "Proof:" there are effectively only finitely many refinements of a given type

Some Results

- Sort checking is decidable
  - "Proof:" there are effectively only finitely many refinements of a given type
- Sorting is closed under β-reduction
  - "Proof:" standard substitution property

Some Results

- Sort checking is decidable
  - "Proof:" there are effectively only finitely many refinements of a given type
- Sorting is closed under β-reduction
  - "Proof:" standard substitution property
- Sorting is closed under η-expansion
  - "Proof:" induction over sorts

More Results

- Define $\eta^\alpha(M)$ as η-long form of M^α

More Results

- Define $\eta^\alpha(M)$ as η-long form of M^α
- Define subsorting $S^\alpha \leq T^\alpha$ at higher types

$$S \leq T :\Leftrightarrow x{:}S \vdash \eta^\alpha(x) : T$$

More Results

- Define $\eta^\alpha(M)$ as η-long form of M^α
- Define subsorting $S^\alpha \leq T^\alpha$ at higher types

$$S \leq T :\Leftrightarrow x{:}S \vdash \eta^\alpha(x) : T$$

- Extend subsumption rule

More Results

- Define $\eta^\alpha(M)$ as η-long form of M^α
- Define subsorting $S^\alpha \leq T^\alpha$ at higher types

$$S \leq T :\Leftrightarrow x{:}S \vdash \eta^\alpha(x) : T$$

- Extend subsumption rule
- Sorting is closed under β-expansion
  - "Proof:" intersect all sorts the abstracted term is used at

More Results

- Define $\eta^\alpha(M)$ as η-long form of M^α
- Define subsorting $S^\alpha \leq T^\alpha$ at higher types

$$S \leq T :\Leftrightarrow x{:}S \vdash \eta^\alpha(x) : T$$

- Extend subsumption rule
- Sorting is closed under β-expansion
  - "Proof:" intersect all sorts the abstracted term is used at
- Sorting is closed under η-reduction
  - "Proof:" by subsorting at higher types

More Results

- Define $\eta^\alpha(M)$ as η-long form of M^α
- Define subsorting $S^\alpha \leq T^\alpha$ at higher types

$$S \leq T :\Leftrightarrow x{:}S \vdash \eta^\alpha(x) : T$$

- Extend subsumption rule
- Sorting is closed under β-expansion
  - "Proof:" intersect all sorts the abstracted term is used at
- Sorting is closed under η-reduction
  - "Proof:" by subsorting at higher types
- Conclusion

Extrinsic (Curry) sorting with intersections refining intrinsic (Church) typing is closed under λ-conversion!

Related Developments (my students only)

- Expressive power extends tree automata to higher types
- Canonical ($= \beta$-normal, η-long) terms can by typed bidirectionally
  - In: Festschrift in Honor of Peter B. Andrews on his 70th Birthday, C. Benzmüller, C. Brown, J. Siekmann, and R. Statman, editors
  - Crucial for logical frameworks (Lovas 2010)
- Refinement type inference for ML (Freeman 1994)
- Practical refinements type for SML (Davies 2005)
- Dependent refinements over decidable domains (Xi 1998)
- Unifying sort and dependent refinements (Dunfield 2007)

Conclusion

- Church's original intrinsic formulation of the simply-typed λ-calculus has fallen into disfavor, perhaps unjustly

Conclusion

- Church's original intrinsic formulation of the simply-typed λ-calculus has fallen into disfavor, perhaps unjustly
- It suggests an elegant layering with Curry's extrinsic typing judgment (translated to λ-calculus)

Conclusion

- Church's original intrinsic formulation of the simply-typed λ-calculus has fallen into disfavor, perhaps unjustly
- It suggests an elegant layering with Curry's extrinsic typing judgment (translated to λ-calculus)
- Can be usefully combined with Coppo et al.'s intersection types for high expressiveness, precision, and surprisingly strong metatheoretic results