Type Theories from Barendregt's Cube for Theorem Provers

Jonathan P. Seldin
Department of Mathematics and Computer Science
University of Lethbridge
Lethbridge, Alberta, Canada
jonathan.seldin@uleth.ca
http://home.uleth.ca/~jonathan.seldin

July 11, 2001

Theorem provers here are for *verification*

Must be *trusted*. Need

- Consistency

- Confidence in implementation

In some cases, must be *small*

- Example: Proof Carrying Code

Barendregt's λ cube

Syntax: $M \longrightarrow x|c|\mathsf{Prop}|\mathsf{Type}|(MM)|(\lambda x : M\ .\ M)|(\forall x : M)M$

Prop and Type are *sorts*. s, s', s_i are sorts (It is common to use * for Prop, $\Box$ for Type. I formerly referred to sorts as *kinds*.)

Conversion is β-conversion, $M =_\beta N$, generated by $(\lambda x : A\ .\ M)N \rhd [N/x]M$

Judgements are of the form $\Gamma \vdash M : A$, where Γ is

$$x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$$

These systems all have the same axiom:

$$\vdash \mathsf{Prop} : \mathsf{Type}$$

General Rules

(start) *If* $x \notin FV(\Gamma)$

$$\frac{\Gamma \;\vdash\; A : s}{\Gamma, x : A \;\vdash\; x : A}$$

(weakening) *If* $x \notin FV(\Gamma)$

$$\frac{\Gamma \;\vdash\; A : B \qquad \Gamma \;\vdash\; C : s}{\Gamma, x : C \;\vdash\; A : B}$$

(application)

$$\frac{\Gamma \;\vdash\; M : (\forall x : A)B \qquad \Gamma \;\vdash\; N : A}{\Gamma \;\vdash\; MN : [N/x]B}$$

(abstraction) *If* $x \notin FV(\Gamma)$

$$\frac{\Gamma, x : A \vdash M : B \quad \Gamma \vdash (\forall x : A)B : s}{\Gamma \vdash \lambda x : A \,.\, M : (\forall x : A)B}$$

(conversion)

$$\frac{\Gamma \vdash A : B \quad \Gamma \vdash B' : s \quad B =_\beta B'}{\Gamma \vdash A : B'}$$

Specific rules

(ss' rule) *If* $x \notin FV(\Gamma)$

$$\frac{\Gamma \vdash A : s \quad \Gamma, x : A \vdash B : s'}{\Gamma \vdash (\forall x : A)B : s'}$$

System depends on possible values of s and s' in these specific rules

Some examples

- $\lambda\!\rightarrow$, related to simple type assignment: Both s and s' must be Prop

- $\lambda 2$, related to Second order typed λ-calculus: s' must be Prop

- λP, related to AUT-QE and LF: s must be Prop

- $\lambda\omega$, related to Girard's $F\omega$: If s is Prop, so is s'

- λC, Calculus of constructions: s and s' can both be either kind

HOL, (Church 1940), which is not in the λ-cube, is a subsystem of λC.

Advantages of these systems

- They are all consistent by strong normalization
- All have small number of primitive postulates (easier to trust implementation)

Disadvantage

- All are impredicative

But now many non-logicians have ever heard of predicativity?

Representing logic with equality

Use $A \rightarrow B$ for $(\forall x : A)B$ if x does not occur free in B

If $A : \mathsf{Prop}$ and $B : \mathsf{Prop}$, use $A \supset B$ for this

If $A : \mathsf{Prop}$ and $B : \mathsf{Prop}$, use $A \wedge B$ for $(\forall w : \mathsf{Prop})((A \supset B \supset w) \supset w)$

Terms of type $A \wedge B$ are pairs with projections

If $A : \mathsf{Prop}$ and $B : \mathsf{Prop}$ use $A \vee B$ for

$$(\forall w : \mathsf{Prop})((A \supset w) \supset ((B \supset w) \supset w))$$

Terms of type $A \vee B$ are disjoint unions with injections

Define $\mathsf{void} \equiv \bot \equiv (\forall x : \mathsf{Prop})x$

If $A : \mathsf{Prop}$, use $\neg A$ for $A \supset \bot$

If $A : \mathsf{Prop}$ and $x : A \vdash B : \mathsf{Prop}$, then use $(\exists x : A)B$ for

$$(\forall w : \mathsf{Prop})((\forall x : A)(B \to w) \to w)$$

Terms of type $(\exists x : A)B$ are pairs (differently typed from those of type $A \wedge B$) with a left projection but (for technical reasons) no right projection

All this gives us the standard rules for *intuitionistic logic*

If $A : s$, $M : A$ and $N : A$, use $M =_A N$ for

$$(\forall z : A \to \mathsf{Prop})(zM \supset zN)$$

This is called *Leibniz equality*

I am **not** assuming extensionality for this as S. Berardi does

Boolean type

$$\mathsf{Bool} \equiv (\forall u : \mathsf{Prop})(u \to u \to u)$$

$$\mathsf{T} \equiv \lambda u : \mathsf{Prop} \,.\, \lambda x : u \,.\, \lambda y : u \,.\, x$$

$$\mathsf{F} \equiv \lambda u : \mathsf{Prop} \,.\, \lambda x : u \,.\, \lambda y : u \,.\, y$$

Adding assumptions

To make A an assumption (for A : Prop), add as a new hypothesis $c : A$, where c is a constant.

By strong normalization, the underlying system is consistent. There is no term M such that $\vdash M : \bot$.

However, assuming

$$c_1 : \mathsf{Prop}, c_2 : c_1, c_3 : \neg c_1$$

we get an inconsistency.

A set Γ of assumptions is *consistent* if there is no term M such that $\Gamma \vdash M : \bot$. This is equivalent to: there is no term N such that $\Gamma, x : \mathsf{Prop} \vdash N : x$, where $x \notin \mathsf{FV}(\Gamma)$..

Goal: Prove certain sets Γ of assumptions consistent

Method: Get consistency results in calculus of constructions

Results will hold for entire λ-cube and HOL. If a given construct cannot be typed in a weaker system, the results for calculus of constructions will justify certain additional assumptions.

For example, in $\lambda\!\rightarrow$, the assumptions for conjunction would be

- $\mathsf{\Lambda}$: Prop $\rightarrow$ Prop $\rightarrow$ Prop (here $A \wedge B$ is to be an abbreviation for $\mathsf{\Lambda}AB$)
- and.in : λx : Prop . λy : Prop . $x \rightarrow y \rightarrow \mathsf{\Lambda}xy$
- and.left : λx : Prop . λy : Prop . $\mathsf{\Lambda}xy \rightarrow x$
- and.right : λx : Prop . λy : Prop . $\mathsf{\Lambda}xy \rightarrow y$

The justification for these assumptions in $\lambda\!\rightarrow$ is that they can be interpreted by terms in the calculus of constructions which can be proved to satisfy these typings in an environment that has been proved consistent

Deduction Normalization A deduction of the form

$$\dfrac{\dfrac{\dfrac{\Gamma, x : A \ \vdash \ M : B \quad \Gamma \ \vdash \ (\forall x : A)B : s}{\Gamma \ \vdash \ \lambda x : A \,.\, M : (\forall x : A)B}\text{(abstraction)}}{\Gamma \ \vdash \ \lambda x : A \,.\, M : (\forall x : C)D}\text{(conversion)} \qquad \Gamma \ \vdash \ N : C}{\Gamma \ \vdash \ (\lambda x : A \,.\, M)N : [N/x]D}\text{(application)}$$

where $x \notin \mathsf{FV}(\Gamma, A)$, $A =_\beta C$, and $B =_\beta D$, reduces to

$$\dfrac{\dfrac{\Gamma, x : A \ \vdash \ M : B \qquad \dfrac{\Gamma \ \vdash \ N : C}{\Gamma \ \vdash \ N : A}\text{(conversion)}}{\Gamma \ \vdash \ [N/x]M : [N/x]B}\text{(substitution lemma)}}{\Gamma \ \vdash \ [N/x]M : [N/x]D}\text{(conversion)}$$

Strong normalization holds for deductions

Derivation to avoid: Let Γ be

$$A : \mathsf{Prop}, w : (\forall z : A \to \mathsf{Prop})(zN), x : \mathsf{Prop}$$

We want to avoid

$$\dfrac{\dfrac{\Gamma \ \vdash \ w : (\forall z : A \to \mathsf{Prop})(zN) \qquad \Gamma \ \vdash \ \lambda y : A \ \overset{\vdots}{.}\ x : A \to \mathsf{Prop}}{\Gamma \ \vdash \ w(\lambda y : A \, . \, x) : (\lambda y : A \, . \, x)N}\,\text{(application)}}{\Gamma \ \vdash \ w(\lambda y : A \, . \, x) : x}\,\text{(conversion)}$$

A *strongly consistent* environment (well-formed) is defined to exclude this. It does not allow any types of the form $A \wedge B$, $A \vee B$, $(\exists x : A)B$, $\bot$, $\neg A$, or $M =_A N$.

Every strongly consistent environment is consistent.

There are consistent environments which allow negations of formulas.

Important Result

Let Γ_1 be a well-formed environment in which each type is the negation of an equation between terms with distinct normal forms, and let Γ_2 be strongly consistent. Then if, for $B : s$ and a closed term R,

$$\Gamma_1, \Gamma_2 \vdash R : M =_B N,$$

then $M =_\beta N$.

The proof is to assume a shortest deduction (in normal form) for any Γ_2 and prove that there must be in the deduction an inference from zM to zN, from which $M =_\beta N$ follows.

This proves the consistency of Γ_1, Γ_2 and identifies Leibniz equality with conversion.

Example: Γ_1 is $\mathsf{bool} : \neg\ \mathsf{T} =_{\mathsf{Bool}} \mathsf{F}$ and Γ_2 is empty

Arithmetic (example of recursive datatype)

Define:

1. $\mathsf{N} \equiv (\forall A : \mathsf{Prop})((A \to A) \to (A \to A))$

2. $\mathbf{0} \equiv \lambda A : \mathsf{Prop}\,.\, \lambda x : A \to A\,.\, \lambda y : A\,.\, y$

3. $\boldsymbol{\sigma} \equiv \lambda u : \mathsf{N}\,.\, \lambda A : \mathsf{Prop}\,.\, \lambda x : A \to A\,.\, \lambda y : A\,.\, x(uAxy)$

Here

$$\mathsf{n} =_\beta \lambda A : \mathsf{Prop}\,.\, \lambda x : A \to A\,.\, \lambda y : A\,.\, \underbrace{x(x(\ldots(x}_{n}\,y)\ldots))$$

It is possible to define $\boldsymbol{\pi}$ so that

$$\begin{aligned} \boldsymbol{\pi}\mathbf{0} &=_\beta \mathbf{0} \\ \boldsymbol{\pi}(\boldsymbol{\sigma}\mathsf{n}) &=_\beta \mathsf{n} \end{aligned}$$

Using this $\boldsymbol{\pi}$, it is possible to define R so that if $A : \mathsf{Prop}$, $M : A$, and $N : \mathsf{N} \rightarrow A \rightarrow A$,

$$\begin{aligned} \mathsf{R}MN\mathbf{0} &=_\beta M \\ \mathsf{R}MN(\boldsymbol{\sigma}\mathsf{n}) &=_\beta N\mathsf{n}(\mathsf{R}MN\mathsf{n}) \end{aligned}$$

We can prove

$$\begin{aligned} &\vdash \mathsf{N} : \mathsf{Prop} \\ &\vdash \mathbf{0} : \mathsf{N} \\ &\vdash \boldsymbol{\sigma} : \mathsf{N} \rightarrow \mathsf{N} \end{aligned}$$

But what about mathematical induction?

There is a non-numeral in N, namely $\lambda A : \mathsf{Prop} \,.\, \lambda x : A \to A \,.\, x$. This is η-convertible to a numeral, but not β-convertible

η-reduction: If x does not occur free in $U : (\forall x : A)B$

$$\lambda x : A \,.\, Ux \triangleright U$$

(I am avoiding η-conversion for technical reasons)

Pfenning and Paulin-Mohring (1989) give an example of a recursive datatype represented this way in which there is a term in the type which does not β- or η-convert to anything constructed from the constructors of the datatype.

We should not **expect** mathematical induction to hold for N.

To get mathematical induction, define (Dedekind's definition of natural number, 1887)

$$\mathcal{N} \equiv \lambda n : \mathsf{N} \,.\, (\forall A : \mathsf{N} \to \mathsf{Prop})((\forall m : \mathsf{N})(Am \supset A(\boldsymbol{\sigma} m)) \supset A\mathbf{0} \supset An)$$

We can prove

$$\begin{array}{ll}
\vdash & \mathcal{N} : \mathsf{N} \to \mathsf{Prop} \\
\vdash & \mathcal{N}\mathbf{0} \\
\vdash & (\forall n : \mathsf{N})(\mathcal{N}n \supset \mathcal{N}(\boldsymbol{\sigma} n)) \\
\vdash & (\forall A : \mathsf{N} \to \mathsf{Prop})((\forall m : \mathsf{N})(Am \supset A(\boldsymbol{\sigma} m)) \supset A\mathbf{0} \supset \\
& \qquad (\forall n : \mathsf{N})(\mathcal{N}n \supset An))
\end{array}$$

Then, using $\boldsymbol{\pi}$ we can prove

$$\vdash (\forall n : \mathsf{N})(\forall m : \mathsf{N})(\mathcal{N}n \supset \mathcal{N}m \supset \boldsymbol{\sigma} n =_{\mathsf{N}} \boldsymbol{\sigma} m \supset n =_{\mathsf{N}} m)$$

Also, using $\mathsf{Bool} : \mathsf{Prop}$, $\mathsf{T} : \mathsf{Bool}$, and $\lambda n : \mathsf{N} \,.\, \lambda x : \mathsf{Bool} \,.\, \mathsf{F} : \mathsf{N} \to \mathsf{Bool} \to \mathsf{Bool}$, we can define

$$\mathsf{Iszero} \equiv \mathsf{RT}(\lambda n : \mathsf{N} \,.\, \lambda x : \mathsf{Bool} \,.\, \mathsf{F})$$

Then for $\mathtt{n} : \mathsf{N}$, since $\vdash \mathcal{N}\mathtt{n}$,

$$\begin{aligned} \mathsf{Iszero}\ \mathbf{0} \quad &=_\beta \quad \mathsf{T} \\ \mathsf{Iszero}\ (\boldsymbol{\sigma}\mathsf{n}) \quad &=_\beta \quad \mathsf{F} \end{aligned}$$

Hence, we can prove

$$\mathsf{bool} : \neg\ \mathsf{T} =_{\mathsf{Bool}} \mathsf{F} \vdash\ (\forall n : N)(\mathcal{N} \supset \neg\ \boldsymbol{\sigma} n =_{\mathsf{N}} \mathbf{0})$$

This means that *arithmetic in a typed system is consistent*

Classical Logic

Add assumption $\mathsf{cl} : (\forall u : \mathsf{Prop})(\neg\ \neg\ u \supset u)$

Classical arithmetic can be proved consistent by using a variation of the $\neg\neg$-translation.

Abstract recursively defined data types

In a recent paper (Seldin, 2000), this is extended to a large class of abstract recursively defined data types.

(But not all. Only those for which the type of each constructor has the form

$$A_1 \to (A_2 \to \ldots (A_n \to D) \ldots)$$

where D is the type of the database and each A_i is either D or is the type of another such database or is a variable of type Prop.)

Sets as predicates

Huet's (1987) idea:

Let U be any small type: i.e., $U : \mathsf{Prop}$

Then $\mathsf{Set}_U \equiv U \to \mathsf{Prop}$

If $A : \mathsf{Set}_U$, then $x \in A$ is Ax

For $P : \mathsf{Prop}$, $\{x : U \mid P\} \equiv \lambda x : U \,.\, P$

$A \subseteq B \equiv (\forall x : U)(x \in A \supset x \in B)$

$A =_{\mathsf{ex}} B \equiv (A \subseteq B) \wedge (B \subseteq A)$

Further definitions:

$\emptyset \equiv \{x : U \mid \bot\}$

$A \cap B \equiv \{x : U \mid x \in A \wedge x \in B\}$

$A \cup B \equiv \{x : U \mid x \in A \vee x \in B\}$

$\sim A \equiv \{x : U \mid \neg\, x \in A\}$

$\mathcal{P} A \equiv \lambda B : \mathsf{Set}_U \,.\, B \subseteq A$

Here, $\mathcal{P} A : \mathsf{Set}_U \rightarrow \mathsf{Prop}$

$\mathsf{Class}_U \equiv \mathsf{Set}_U \rightarrow \mathsf{Prop}$

Functions

The set of functions from set A to set B is

$$\lambda f : U \to U \;.\; (\forall x : U)(x \in A \supset fx \in B)$$

We get much of set theory, but not all:

Seldin (1997) shows that essentially all the axioms of Intuitionistic Zermelo-Frankel set theory (formulation of Beeson 1985) are provable *except* the axioms of power set and $\in$-induction.

$\in$-induction prevents infinite descending $\in$-chains; here they are prevented by the type structure.

So the important missing axiom is that of power set. We can get any finite number of power set operations.

Note 1 Representing this part of set theory involves *no* new assumptions, only definitions.

Note 2 If $\Gamma \vdash M : A$ and if Γ is consistent, then, $\Gamma, c : A$ is consistent (c a new constant). Useful if M is large.

If this A is $A_1 \rightarrow (A_2 \rightarrow (\ldots(A_n \rightarrow A_0)\ldots))$, then the new assumption represents the derived rule

$$\frac{M_1 : A_1 \quad M_2 : A_2 \quad \ldots \quad M_n : A_n}{M_1 M_2 \ldots M_n : A_0}$$

Conclusion

This approach leads to

- small systems
- few assumptions
- provably consistent
- sufficient for a lot of mathematical reasoning