Pure Type Systems: Extensions and Restrictions

Fairouz Kamareddine
Heriot-Watt University
Edinburgh, UK

Brasilia, Brasil, 24 May 2017

Types and sets existed since antiquity

- Euclid's *Elements* (circa 325 B.C.) begins with:
  1. A *point* is that which has no part;
  2. A *line* is breadthless length.

     ⋮

  15. A *circle* is a plane figure contained by one line such that all the straight lines falling upon it from one point among those lying within the figure are equal to one another.
- 1..15 *define* points, lines, and circles which Euclid *distinguished*.
- Euclid always mentioned to which *class* (points, lines, etc.) an object belonged.
- In Euclid's Elements (Book IX, Proposition 20):
  Consider any finite list of prime numbers $p_1, p_2, ..., p_n$. At least one additional prime number not in this list exists.

Paradoxes in the 20th century

- Paradoxes in the 20th century led to the creation of explicit theories of sets and types (and this was in parallel).
- Researchers in set theory and type theory don't collaborate often despite the huge overlap and complementarity of their work.
- However, an impressive body of work exists to explain the strengths and weaknesses of theories and this work is a promising avenue in both areas.

The calculus and the paradoxes of motion

- Zeno of Elea, C. 590-525 B.C.E., devised some paradoxical arguments against the possibility of motion.
- Since the calculus was developed partly to deal with motion, these paradoxical arguments are important for the foundations of analysis.
- Three of the most important of these are in Aristotle in his Physics.
  - *Dichotomy. There is no motion, because what moves must arrive at the middle of its course before it reaches the end.* In other words, to leave the room, you first have to get halfway to the door, then you have to get halfway from that point to the door, etc. No matter how close you are to the door, you have to go half the remaining distance before proceeding.
  - *Arrow. The flying arrow is at rest,* because a thing is at rest when occupying its own space at a given time, as the arrow does at every instant of its alleged flight.

A short history of numbers

- *natural numbers* like 0, 1, 2, which were used to count (sheep for example);
- *integers* like 0, 1, -1, 2, -2, etc. which were also used to count;
- *rational numbers* which are the quotients or fractions of integers like $2/3$ and which were used to measure (the ancients used *anthyphairesis/Alternated substitution* to evaluate Ratios);
  - $2/3 = 0.6666666...$ where 6 repeats over and over
  - $41/333 = 0.123123123123...$ where 123 repeats over and over.
- *irrational numbers* like $\sqrt{2}$, $\sqrt{3}$, π;
  - $\pi = 3.14159265358979323846264338...$
  - $\sqrt{2} = 1.4142135623730950488016887242096980785696718753769 4...$

It started with incommensurability

- According to Webster's dictionary, *commensurability* is *divisibility without remainder by a common unit.*
- Hence 6 and 9 are commensurable (since they are both divisible by 3).
- Attempts to find the unit which measures exactly the side and diagonal of a square led to the proof of the *incommensurability* of the side and diagonal of a square.
- This result on incommensurability implies that $\sqrt{2}$ *is not a rational number*. That is, it cannot be represented as the quotient of two integers.

With incommensurability, number was no longer everything

- The discovery of the incommensurability of the side and diagonal of a square showed that the Pythagorean idea that *number is everything would not work.*
- The Pythagoreans needed to treat quantities which are not numbers.
- For them, numbers are rationals and quantities are the incommensurable (which we call *real numbers*).
- The Greeks constructed geometric figures (recall Euclid), but took numbers as given. They separated numbers, which are discrete, from continuous magnitudes (quantities/real numbers).
- They did not use fractions to approximate continuous magnitudes.
- They did not construct the reals, nor multiply them nor divide them, etc.

Euclid used anthyphairesis to find the greatest common divisor of two numbers

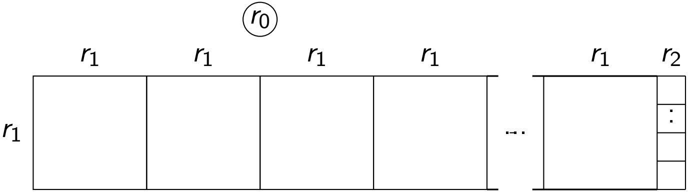


Figure 1: Anthyphairesis

Ratio of 12 to 5 $= [2,2,2] = 2 + \cfrac{1}{2 + \cfrac{1}{2}}$

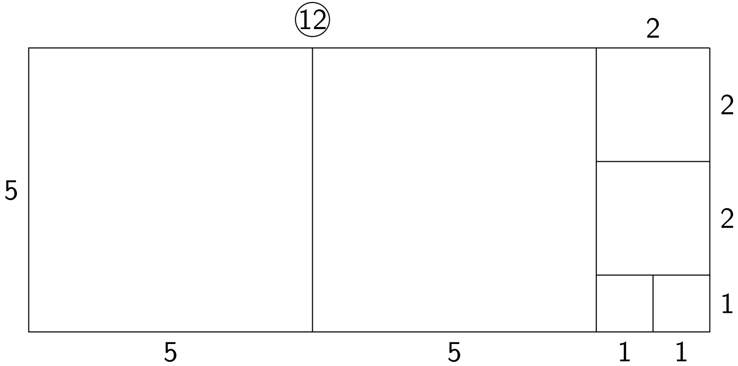


Figure 2: Ratio of 12 to 5

Ratio of 22 to 6 $= [3,1,2] = 3 + \cfrac{1}{1 + \cfrac{1}{2}}$

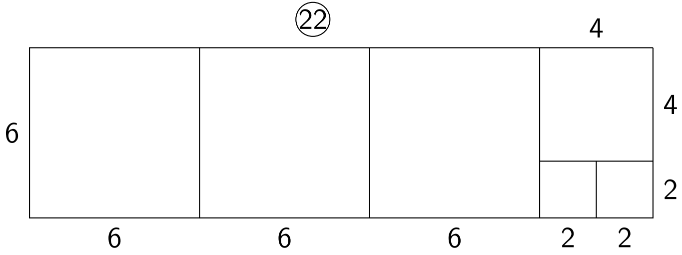
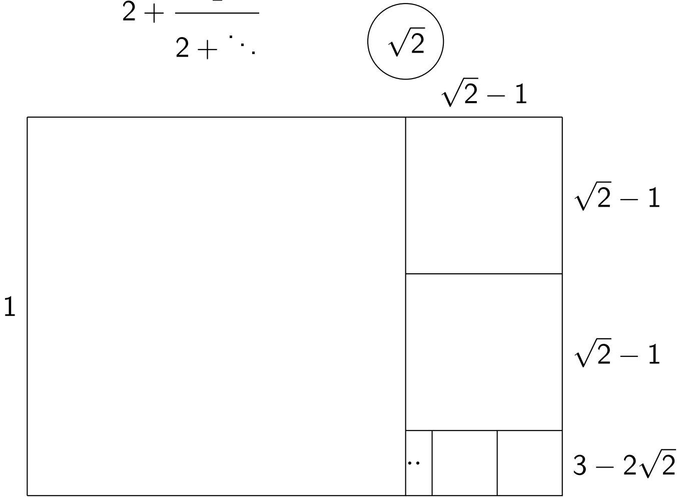


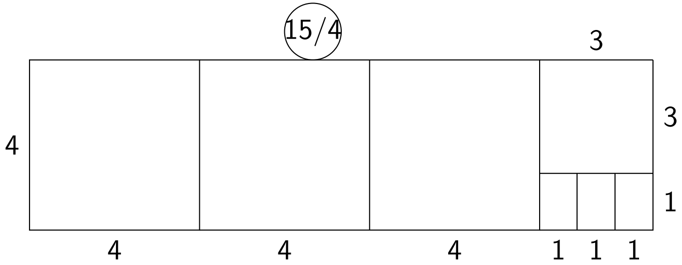
Figure 3: Ratio of 22 to 6

Ratio of $\sqrt{2}$ to 1 is $[1, 2, 2, 2, \ldots]$

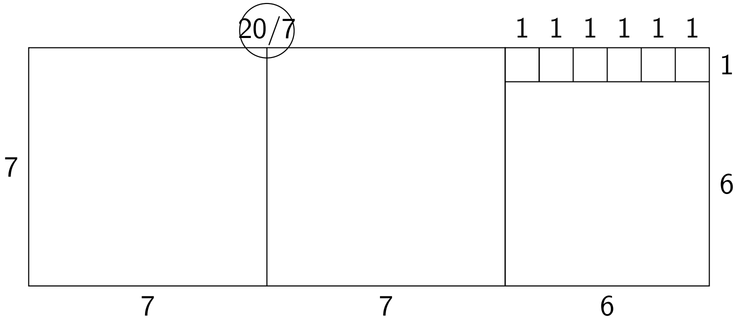
$$\sqrt{2} = 1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \ddots}}}} \quad \text{etc.}$$



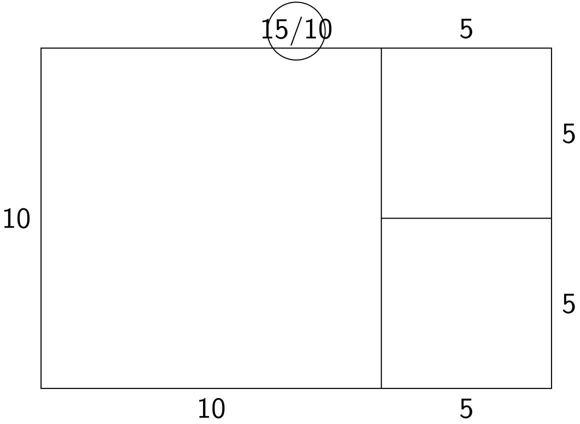
The ratio of 15 to 4 is $[3,1,3] = 3 + \cfrac{1}{1 + \cfrac{1}{3}}$



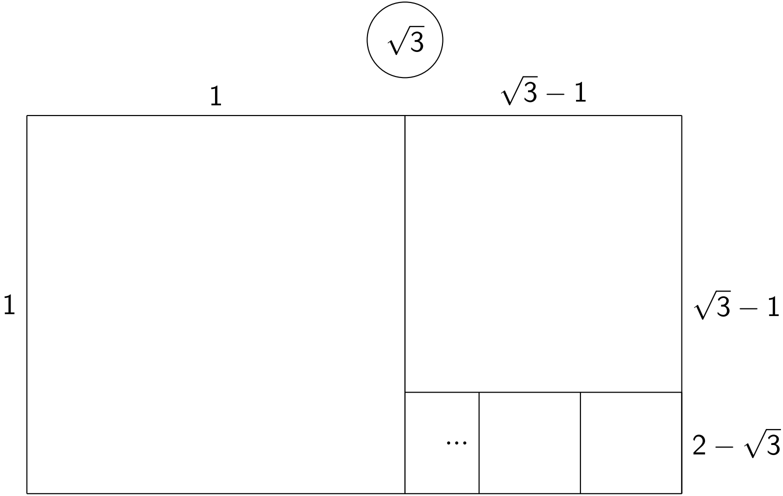
The ratio of 20 to 7 is $[2, 1, 6] = 2 + \cfrac{1}{1 + \cfrac{1}{6}}$



The ratio of 15 to 10 is $[1, 2] = 1 + \frac{1}{2}$



The ratio of $\sqrt{3}$ to 1 is characterised by $[1, 1, 2, 1, 2, 1, 2, ...] = [1, \overline{1, 2}]$



Real numbers need to be constructed (using approximations like Dedekind cuts, Cauchy sequences, etc.)

- The idea of using fractions to approximate continuous magnitudes developed first in the Arab world during the middle ages, and only came to Europe in the 16th and 17th centuries.
- This idea would have been assumed by both Newton and Leibniz.
- *Although the Greeks did not construct magnitudes (real numbers), they still studied them after discovery of incommensurability.*

In 18th and 19th century, irrational numbers were divided into two categories: *Algebraic* and *transcendental*

- A number is algebraic if it is the root of a non-zero polynomial with rational coefficients. For example, $\sqrt{2}$ is algebraic since it is the solution to $x^2 - 2 = 0$.
- An irrational number that is not algebraic is called *transcendental* (i.e. cannot be made of algebraic equations). For example, π is transcendental.
- *Transcendental was coined by Leibniz in 17th cenetury who showed that $sin(x)$ is not an algebraic function of x.*
- Until the discovery of irrationals like $\sqrt{2}$, the pythagorean expected all numbers to be rational.

The discovery of transcendental numbers

- Until the 17th century it was expected that numbers *should fit the algebraic mould.*
- In the *18th century* it was shown that π is irrational and *conjectured* that π is transcendental.
- In the 19th century, *proofs were given* of the existence of transcendtal numbers and that π *is transcendental.*
- Once π was shown transcendtal meant that the old problem of *squaring the circle became impossible.*

Researchers in the 19th century continued to go deeper into numbers

- 1821: Many controversies in analysis were solved by *Cauchy.* E.g., he gave a precise definition of convergence in his *Cours d'Analyse*.
- 1872: Due to the more *exact definition of real numbers given by Dedekind*, the rules for reasoning with real numbers became even more precise.
- 1895-1897: *Cantor began formalizing set theory* and made contributions to number theory.
- 1889: *Peano formalized arithmetic*, but did not seriously treat logic or quantification.
- *Cantor's diagnolisation argument and the size of the natural numbers versus the size of the real numbers* will impact the size of what can be *computable* versus what cannot.

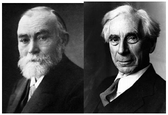
- Cantor proved that *algebraic numbers are countable*.
- Hence there are only *as many algebraic numbers as there are natural numbers.*
- Cantor proved that *the transcendental numbers are uncountable*.
- Cantor proved that the size of the algebraic numbers is infinite, but is the *smallest infinite that exists*.
- The size of the transcendental numbers is a much much larger infinite.

Depressing implication for computation

Later on it was shown that:

- The size of the computable functions is the size of the algebraic numbers, the smallest infinite $\aleph_0$.
- The size of the non-computable functions is the size of the transcendental numbers (the monster infinite), which according to Cantor's *Continuum hypothesis* is the infinite $\aleph_1$ which is the next one up after $\aleph_0$.
- This means that there are a lot more functions that are impossible to compute than there are computable functions.

From Cantor to Frege, Russell and Type Theory



- *General definition of function 1879* [8] is key to Frege's *formalisation of logic*.
- 1892-1903 Frege's *Grundgesetze der Arithmetik*, could handle elementary arithmetic, set theory, logic, and quantification.
- *Self-application of functions* was at the heart of *Russell's paradox 1902* [29].
- To *avoid paradox* Russell controled function application via *type theory*.
- Russell [30] *1903* gives the first type theory: the *Ramified Type Theory* (RTT).
- RTT is used in Russell and Whitehead's Principia Mathematica [32] 1910–1912.

Hilbert

- *Simple theory of types* (STT): Ramsey [25] *1926*, Hilbert and Ackermann [12] *1928*.
- Following from Leibniz and Frege, researchers started calling for *logical methods that could decisively answer questions at hand.*
- Hilbert believed that every mathematical problem should *either have a solution* or we should definitely know that *no such solution exists.*

We must Know. We will know.

- Back in 1928, Hilbert posed the *the Entscheidung/Decision Problem*.
- This problem dates back to *Leibniz* and asks for an *algorithm* that takes as input a statement of first order logic and returns as output one of two possible answers: *yes* when the statement is always valid, or *no* otherwise.

Hilbert advocated the idea (which became known as *Hilbert's program*) that there should be an axiomatization of all of mathematics that is

- *a complete* (i.e., every true formula can be derived) and
- *consistent* (i.e., does not contain a contradiction)

such that *every mathematical problem should either have a solution or we should definitely know that no such solution exist.*

- The results of the 1930s would establish the limitations of computers even before computers were built.
- No matter how fast and advanced computers get (and they are advancing at an amzing speed, considering that they did not exist in 1930s).
- Before we knew what computers could do, we had results telling us what computers could never do.
- These results of the limitations of the computer, will never change.
- They are set in stone just like the impossibility of squaring a circle.

Can we solve/compute everything?

- Turing answered the question via a *machine for running/computing programs*.
  a function f is computable iff f can be computed on a Turing machine.
- Church invented the λ-calculus, a *language for describing programs*.
  a function f is computable iff f can be described in the λ-calculus.
- Note that Church's λ-calculus was initially intended as a language of programs and logic, but it turned out to be inconsistent (Kleene and Rosser) and Church restricted the λ-calculus to programs.
- Goedel's result meant that *no absolute guarantee can be given that many significant branches of mathematics are entirely free of contradictions.*
- This means: we can compute a very small (∞ly countable, size of $\mathbb{N}$) amount compared to what we will never be able to compute (uncountable, size of $\mathbb{R}$).

Church, Turing, Goedel

- Church's *simply typed λ-calculus* $\lambda\rightarrow$ [4] 1940 $=$ λ-calculus $+$ STT.
- The hierarchies of types/orders in RTT and STT are *unsatisfactory*.
- Hence, birth of *different systems of functions and types*, each with *different functional power*.

- Simply typed λ-calculus was adopted in theorem provers like HOL and was used to make sense of other programming languages (e.g., pascal).
- Then, simple types were independently extended to *polymorphic* logic and programming languages.
- *Dependent* types (necessary for reasoning about proofs inside the system) were also introduced in Automath by de Bruijn.
- *Types* continue to play an influential role in the design and implementation of programming languages and theorem provers.

Syntax of λ-calculus

Type Free

- $A ::= x \mid AB \mid \lambda x.B$
- $(\lambda x.B)C \to_\beta B[x := C]$.

With simple types:

- $\sigma ::= T \mid \sigma \to \tau$
- $A ::= x \mid AB \mid \lambda x{:}\sigma.B$
- $(\lambda x : \sigma.B)C \to_\beta B[x := C]$.

With dependent types:

- $A ::= x \mid * \mid \Box \mid AB \mid \lambda x{:}A.B \mid \Pi x{:}A.B$
- $(\lambda x : A.B)C \to_\beta B[x := C]$.
- Sometimes also with $(\Pi x : A.B)C \to_\Pi B[x := C]$.

Church's Simply Typed λ-calculus in modern notation

- Terms $A ::= x \mid AB \mid \lambda x{:}\sigma.B$
- Types $::= T \mid \mid \sigma \to \tau$
- Γ is an environment (set of declaration).
- Rules:

(start)	$\dfrac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}$
(λ)	$\dfrac{\Gamma, x{:}\sigma \vdash A : \tau}{\Gamma \vdash \lambda_{x:\sigma}.A : \sigma \to \tau}$
(app_{Π})	$\dfrac{\Gamma \vdash A : \sigma \to \tau \qquad \Gamma \vdash B : \sigma}{\Gamma \vdash AB : \tau}$

Common features of modern types and functions

- We can *construct* a type by abstraction. (Write $A : *$ for *A is a type*)
  - $\lambda_{y:A}.y$, the identity over A *has type* $A \to A$
  - $\lambda_{A:*}.\lambda_{y:A}.y$, the polymorphic identity *has type* $\Pi_{A:*}.A \to A$
- We can *instantiate* types. E.g., if $A = \mathbb{N}$, then the identity over $\mathbb{N}$
  - $(\lambda_{y:A}.y)[A := \mathbb{N}]$ *has type* $(A \to A)[A := \mathbb{N}]$ or $\mathbb{N} \to \mathbb{N}$.
  - $(\lambda_{A:*}.\lambda_{y:A}.y)\mathbb{N}$ *has type* $(\Pi_{A:*}.A \to A)\mathbb{N} = (A \to A)[A := \mathbb{N}]$ or $\mathbb{N} \to \mathbb{N}$.
- $(\lambda x{:}\alpha.A)B \to_\beta A[x := B]$ $\qquad$ $(\Pi x{:}\alpha.A)B \to_\Pi A[x := B]$
- Write $A \to A$ as $\Pi_{y:A}.A$ when y not free in A.

- Syntax: $s ::= * \mid \Box_i$ *where* $i \geq 1$.
  We assume $\mathcal{S}$ set of sorts s.
  $A ::= x^s \mid s \mid AB \mid \lambda x^s{:}A.B \mid \Pi x^s{:}A.B$
- $\mathcal{A}_{\vec{F_\omega}} = \mathcal{A}_{\vec{CC}} = \{(*, \Box_1), (\Box_i, \Box_{i+1})\}$
  $\mathcal{A}_Z = \{(*, \Box_1), (\Box_1, \Box_2), (\Box_2, \Box_3)\}$
- $\boldsymbol{R}_{\vec{CC}} = \{(*, *, *), (\Box_i, *, *), (*, \Box_i, \Box_j), (\Box_i, \Box_j, \Box_{\max\{i,j\}})\}$
  $\boldsymbol{R}_{\vec{F_\omega}} = \{(*, *, *), (\Box_i, *, *), (\Box_i, \Box_j, \Box_{\max\{i,j\}})\}$
  $\boldsymbol{R}_Z = \{(*, *, *)\} \cup \{(\Box_i, *, *) | 1 \leq i \leq 3\} \cup \{(\Box_i, \Box_j, \Box_{\max\{i,j\}}) | 1 \leq i, j \leq 2\}$
- $\mathcal{A}_Z \subset \mathcal{A}_{\vec{F_\omega}} = \mathcal{A}_{\vec{CC}}$ and $\boldsymbol{R}_Z \subset \boldsymbol{R}_{\vec{F_\omega}} \subset \boldsymbol{R}_{\vec{CC}}$.
- Formation rule:

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x^s{:}A \vdash B : s_2}{\Gamma \vdash \Pi x^s{:}A.B : s_3} \qquad \textit{if } (s_1, s_2, s_3) \in \boldsymbol{R}$$

The PTS rules

(axiom)	$\langle\rangle \vdash s_1 : s_2 \ \mathit{if}\ (s_1, s_2) \in \mathcal{A}$
(start)	$\dfrac{\Gamma \vdash A : s \qquad x^s \notin \mathrm{DOM}(\Gamma)}{\Gamma, x^s{:}A \vdash x^s : A}$
(weak)	$\dfrac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s \quad x^s \notin \mathrm{DOM}(\Gamma)}{\Gamma, x^s{:}C \vdash A : B}$
(Π)	$\dfrac{\Gamma \vdash A : s_1 \qquad \Gamma, x^s{:}A \vdash B : s_2 \quad (s_1, s_2, s_3) \in \boldsymbol{R}}{\Gamma \vdash \Pi_{x^s:A}.B : s_2}$
(λ)	$\dfrac{\Gamma, x^s{:}A \vdash b : B \qquad \Gamma \vdash \Pi_{x^s:A}.B : s'}{\Gamma \vdash \lambda_{x^s:A}.b : \Pi_{x^s:A}.B}$
(conv_β)	$\dfrac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s \qquad B =_\beta B'}{\Gamma \vdash A : B'}$
(app_Π)	$\dfrac{\Gamma \vdash F : \Pi_{x^s:A}.B \qquad \Gamma \vdash a : A}{\Gamma \vdash Fa : B[x^s{:=}a]}$

Desirable properties of a type system with reduction r

- *r is Church Rosser (CR)*
  If $A \twoheadrightarrow_r A'$ and $A \twoheadrightarrow_r A''$ then there is a B such that $A' \twoheadrightarrow_r B$ and $A'' \twoheadrightarrow_r B$.
- *Typing is preserved under reduction*
  If $\Gamma \vdash A : B$ and ($A \twoheadrightarrow_r A'$ or $B \twoheadrightarrow_r B'$ or $\Gamma \twoheadrightarrow_r \Gamma'$) then $\Gamma' \vdash A' : B'$.
- *Strong Normalisation (SN)*
  If $\Gamma \vdash A : B$ then $\mathrm{SN}_{\rightarrow_r}(A)$ and $\mathrm{SN}_{\rightarrow_r}(B)$.
  SN properties for CC and F_ω have the same proof-theoretic strength as higher-order arithmetic (HA_ω).
  CC and F_ω can be proven consistent within Heyting arithmetic.

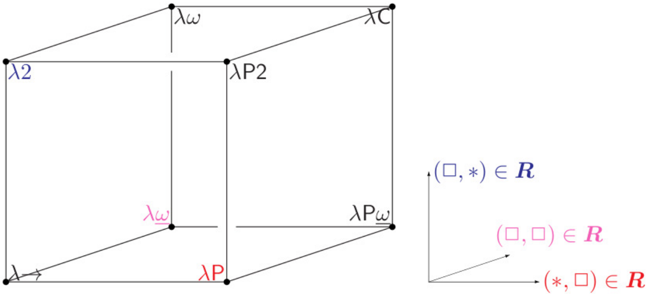
What about???

- *Do we always need CR to hold* For example, in Krivine's λ_c (the language of realisers in classical realisability which is λ-calculus plus the control operator cc), CR does not hold. Instead we have determinism.
- *Subject Expansion*
If $\Gamma \vdash A : B$ and $A' \twoheadrightarrow_r A$ then $\Gamma \vdash A' : B$.
- *The Type System Characterizes Strong Normalisation*
If M is a type free term such that $\mathrm{SN}_{\rightarrow_r}(M)$ then there are Γ, A, B, such that $\Gamma \vdash A : B$ and $TE(A) = M$.
- *Decidability of Type checking and typability*
Given A, B, and Γ do we have $\Gamma \vdash A : B$??.
Given A are there Γ, B such that $\Gamma \vdash A : B$??.
Given A and Γ is there B such that $\Gamma \vdash A : B$??.

Assume only $*, \Box$. Let $\mathcal{A} = \{(*, \Box)\}$. Write (s_1, s_2, s_2) as (s_1, s_2) and let $\boldsymbol{R} \subseteq \{(*, *), (*, \Box), (\Box, *), (\Box, \Box)\}$.

	Simple	Poly-morphic	Depend-ent	Constr-uctors	Related system	Refs.
$\lambda{\rightarrow}$	$(*, *)$				λ^τ	[4, 2, 13]
$\lambda 2$	$(*, *)$	$(\Box, *)$			F	[10, 28]
λP	$(*, *)$		$(*, \Box)$		AUT-QE, LF	[6, 11]
$\lambda\underline{\omega}$	$(*, *)$			$(\Box, \Box)$	POLYREC	[27]
$\lambda P2$	$(*, *)$	$(\Box, *)$	$(*, \Box)$			[22]
$\lambda\omega$	$(*, *)$	$(\Box, *)$		$(\Box, \Box)$	Fω	[10]
$\lambda P\underline{\omega}$	$(*, *)$		$(*, \Box)$	$(\Box, \Box)$		
λC	$(*, *)$	$(\Box, *)$	$(*, \Box)$	$(\Box, \Box)$	CC	[5]

The Barendregt Cube



Typing Polymorphic identity needs $(\Box, *)$

- $$\frac{y:*\vdash y:* \qquad y:*,x{:}y\vdash y:*}{y:*\vdash \Pi x{:}y.y:*} \text{ by } (\Pi)\ (*,*)$$
- $$\frac{y:*,x:y\vdash x:y \qquad y:*\vdash \Pi x{:}y.y:*}{y:*\vdash \lambda x:y.x:\Pi x{:}y.y} \text{ by } (\lambda)$$
- $$\frac{\vdash *:\Box \qquad y:*\vdash \Pi x{:}y.y:*}{\vdash \Pi y:*.\Pi x{:}y.y:*} \text{ by } (\Pi) \text{ by } (\Box,*)$$
- $$\frac{y:*\vdash \lambda x:y.x:\Pi x{:}y.y \qquad \vdash \Pi y:*.\Pi x{:}y.y:*}{\vdash \lambda y:*.\lambda x:y.x:\Pi y:*.\Pi x{:}y.y} \text{ by } (\lambda)$$

The Cube with parametric constants

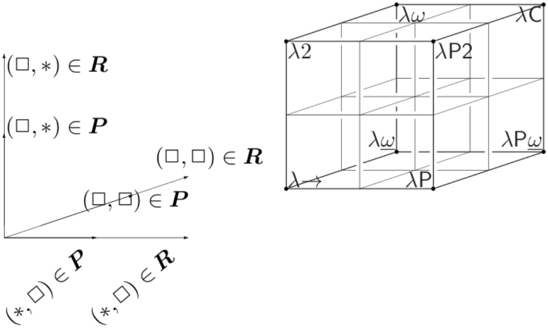
- Let $(*,*) \subseteq \mathbf{R}, \boldsymbol{P} \subseteq \{(*,*),(*,\Box),(\Box,*),(\Box,\Box)\}$.
- $\lambda\mathbf{R}\boldsymbol{P} = \lambda\mathbf{R}$ and the two rules $(\vec{\mathbf{C}}$**-weak)** and $(\vec{\mathbf{C}}$**-app)**:

$$\frac{\Gamma \vdash b : B \qquad \Gamma, \Delta_i \vdash B_i : s_i \qquad \Gamma, \Delta \vdash A : s}{\Gamma, c(\Delta) : A \vdash b : B} \quad (s_i, s) \in \boldsymbol{P}, c \text{ is } \Gamma\text{-fresl}$$

$$\frac{\begin{array}{lcll} \Gamma_1, c(\Delta){:}A, \Gamma_2 & \vdash & b_i{:}B_i[x_j{:=}b_j]_{j=1}^{i-1} & (i = 1, \ldots, n) \\ \Gamma_1, c(\Delta){:}A, \Gamma_2 & \vdash & A : s & (\text{if } n = 0) \end{array}}{\Gamma_1, c(\Delta){:}A, \Gamma_2 \vdash c(b_1, \ldots, b_n) : A[x_j{:=}b_j]_{j=1}^{n}}$$

$\Delta \equiv x_1{:}B_1, \ldots, x_n{:}B_n$.
$\Delta_i \equiv x_1{:}B_1, \ldots, x_{i-1}{:}B_{i-1}$

The refined Barendregt Cube



The π-cube: $R_\pi = R_\beta \setminus (\text{conv}_\beta) \cup (\text{conv}_{\beta\Pi}), \to_{\beta\Pi}$

- $(\lambda x{:}\alpha.A)B \to_\beta A[x := B]$
- $(\Pi x{:}\alpha.A)B \to_\Pi A[x := B]$

(axiom) (start) (weak) (Π) (λ) (app_Π)

$$(\text{conv}_{\beta\Pi}) \quad \frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s \qquad B =_{\beta\Pi} B'}{\Gamma \vdash A : B'}$$

Lemma: $\Gamma \vdash_\beta A : B$ iff $\Gamma \vdash_\pi A : B$

Lemma: The β-cube and the π-cube satisfy the desirable for type systems.

The π_i-cube: $R_{\pi_i} = R_\pi \setminus (\text{app}_\Pi) \cup (\text{i-app}_\Pi)$, $\to_{\beta\Pi}$

(app_Π)	$\dfrac{\Gamma \vdash F : \Pi_{x:A}.B \qquad \Gamma \vdash a : A}{\Gamma \vdash Fa : B[x{:=}a]}$
(axiom)	(start) (weak) (Π) (λ)
$(\text{conv}_{\beta\Pi})$	$\dfrac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s \qquad B =_{\beta\Pi} B'}{\Gamma \vdash A : B'}$
(i-app_Π)	$\dfrac{\Gamma \vdash F : \Pi_{x:A}.B \qquad \Gamma \vdash a : A}{\Gamma \vdash Fa : (\Pi_{x:A}.B)a}$

Lemma:

- If $\Gamma \vdash_\beta A : B$ then $\Gamma \vdash_{\pi_i} A : B$.
- If $\Gamma \vdash_{\pi_i} A : B$ then $\Gamma \vdash_\beta A : [B]_\Pi$ where $[B]_\Pi$ is the Π-normal form of B.

The π_i-cube

- The π_i-cube loses TC and SR properties
  Let $\Gamma = z : *, x : z$. We have that $\Gamma \vdash_{\pi_i} (\lambda_{y:z}.y)x : (\Pi_{y:z}.z)x$.
  - *We do not have TC* $(\Pi_{y:z}.z)x \not\equiv \Box$ and $\Gamma \not\vdash_{\pi_i} (\Pi_{y:z}.z)x : s$.
  - *We do not have SR* $(\lambda_{y:z}.y)x \rightarrow_{\beta\Pi} x$ but $\Gamma \not\vdash_{\pi_i} x : (\Pi_{y:z}.z)x$.
- But we have:
  - *We have STT*
  - *We have PT*
  - *We have SN*
  - *We have a weak form of TC* If $\Gamma \vdash_{\pi_i} A : B$ and *B does not have a Π-redex* then either $B \equiv \Box$ or $\Gamma \vdash_{\pi_i} B : s$.
  - *We have a weak form of SR* If $\Gamma \vdash_{\pi_i} A : B$, *$B$ is not a Π-redex* and $A \twoheadrightarrow_{\beta\Pi} A'$ then $\Gamma \vdash_{\pi_i} A' : B$.

The problem can be solved by re-incorporating Frege and Russell's notions of low level functions (which was lost in Church's notion of function)

(start-a)	$\dfrac{\Gamma \vdash A : s \qquad \Gamma \vdash B : A}{\Gamma, x = B{:}A \vdash x : A}$	$x \notin \mathrm{DOM}(\Gamma)$
(weak-a)	$\dfrac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s \qquad \Gamma \vdash D : C}{\Gamma, x = D{:}C \vdash A : B}$	$x \notin \mathrm{DOM}(\Gamma)$

Figure 4: Basic abbreviation rules BA

$(\mathrm{let}_\backslash)$	$\dfrac{\Gamma, x = B{:}A \vdash C : D}{\Gamma \vdash (\backslash_{x:A}.C)B : D[x := B]}$

Figure 5: $(\mathrm{let}_\backslash)$ where $\backslash = \lambda$ or $\backslash = \Pi$

The β_a-cube: $R_{\beta_a} = R_\beta$ + BA + let$_\beta$, $\to_\beta$

(axiom) (start) (weak) (Π) (λ) (app$_\Pi$) (conv$_\beta$)

(start-a) $$\frac{\Gamma \vdash A : s \quad \Gamma \vdash B : A}{\Gamma, x = B{:}A \vdash x : A} \qquad x \notin \mathrm{DOM}(\Gamma)$$

(weak-a) $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s \quad \Gamma \vdash D : C}{\Gamma, x = D{:}C \vdash A : B} \qquad x \notin \mathrm{DOM}(\Gamma)$$

(let$_\beta$) $$\frac{\Gamma, x = B{:}A \vdash C : D}{\Gamma \vdash (\lambda_{x:A}.C)B : D[x := B]}$$

Lemma: The β_a-cube satisfies the desirable properties except for typability of subterms.
If A is $\vdash$-legal and B is a subterm of A such that every bachelor $\lambda_{x:D}$ in B is also bachelor in A, then B is $\vdash$-legal.

The π_a-cube: $R_{\pi_a} = R_\pi + \mathrm{BA} + \mathrm{let}_\beta + \mathrm{let}_\Pi$, $\to_{\beta\Pi}$

(axiom) (start) (weak) (Π) (λ) (app_Π) ($\mathrm{conv}_{\beta\Pi}$)

(start-a) $$\frac{\Gamma \vdash A : s \qquad \Gamma \vdash B : A}{\Gamma, x = B{:}A \vdash x : A} \qquad x \notin \mathrm{DOM}(\Gamma)$$

(weak-a) $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s \quad \Gamma \vdash D : C}{\Gamma, x = D{:}C \vdash A : B} \qquad x \notin \mathrm{DOM}(\Gamma)$$

(let_β) $$\frac{\Gamma, x = B{:}A \vdash C : D}{\Gamma \vdash (\lambda_{x:A}.C)B : D[x := B]}$$

(let_Π) $$\frac{\Gamma, x = B{:}A \vdash C : D}{\Gamma \vdash (\Pi_{x:A}.C)B : D[x := B]}$$

Lemma: The π_a-cube satisfies the same properties as the β_a.

Let $\Gamma = z : *, x : z$. We have that $\Gamma \vdash_{\pi_{ai}} (\lambda_{y:z}.y)x : (\Pi_{y:z}.z)x$.

- *We NOW have TC* although $\Gamma \not\vdash_{\pi_i} (\Pi_{y:z}.z)x : s$, we have $\Gamma \vdash_{\pi_{ai}} (\Pi_{y:z}.z)x : s$
  By (weak-a) $z : *, x : z, y = x : z \vdash_{\pi_{ai}} z : *$.
  Hence by (let_Π) $z : *, x : z \vdash_{\pi_{ai}} (\Pi_{y:z}.z)x : *[y := x] \equiv *$.
- *We NOW have SR* $(\lambda_{y:z}.y)x \to_{\beta\Pi} x$.
  Although $\Gamma \not\vdash_{\pi_i} x : (\Pi_{y:z}.z)x$, we have $\Gamma \vdash_{\pi_{ai}} x : (\Pi_{y:z}.z)x$
  Since $z : *, x : z \vdash_{\pi_{ai}} x : z$, and $z : *, x : z \vdash_{\pi_{ai}} (\Pi_{y:z}.z)x : *$ and $z : *, x : z \Vdash z =_{\beta\Pi} (\Pi_{y:z}.z)x$, we use ($\text{conv}_{\beta\Pi}$) to get:
  $z : *, x : z \vdash_{\pi_{ai}} x : (\Pi_{y:z}.z)x$.

Degrees of terms

- We define $\sharp : \Lambda \to \{0, 1, 2, 3\}$ by $\sharp(\Box) = 3$, $\sharp(*) = 2$, $\sharp(x^{\varsigma}) = \sharp(\varsigma) - 2$, $\sharp(\pi\delta.A) = \sharp(AB) = \sharp(A)$.
- For $A \in \Lambda$, $\sharp(A)$ is called the degree of A.
- $A : B$ is OK iff $\sharp(A) = \sharp(B) - 1$.
- If $\Gamma \vdash A : B$ then for any $C : D$ in either Γ or A or B we have $C : D$ is OK.
- A is *kind* iff $\sharp(A) = 2$,
  A is *constructor* or A is *type* iff $\sharp(A) = 1$,
  A is *object* iff $\sharp(A) = 0$.
  $\sharp(A) = 3$ iff $A = \Box$.

Guy Steele's discussion of most popular programming language in computer science

Computer Science Metanotation (CSM)

- Data Types:
  - Built-in: numbers, arrays, lists, etc.
  - User-defined: Records, Abstract Data Types or Symbolic Expressions (written in BNF).
- Code: Inference rules (written in Gentzen notation)
- Conditionals: rule dispatch via nondeterministic pattern-matching
- Repetition: overlines and/or ellipsis notations, and sometimes iterators
- Primitive expressions: logic and mathematics
- Special operation: capture-free substitution within a symbolic expression

Guy Steele, Its Time for a New Old Language

According to Steele

Early contributors include

- Gentzen
- Bakus
- Naur
- Church

Steel focuses around difficulties of use of BNF notation:

- Substitution
- Overline and Ellipsis
- Formalisation and Mechanisation of CSM.

Gentzen

- Gerhard Gentzen with his rule metanotation for natural deduction:

  "3.1. Eine Schluβfigur läβt sich in der Form Screiben:

  $$\frac{\mathfrak{A}_1 \ldots \mathfrak{A}_\nu}{\mathfrak{B}} \qquad (\nu \geq 1),$$

  wobei $\mathfrak{A}_1, \ldots, \mathfrak{A}_\nu$ Formeln sind. $\mathfrak{A}_1, \ldots, \mathfrak{A}_\nu$ heiβen dann die Oberformlen, $\mathfrak{B}$ heiβt die Unterformel der Scluβfigur."

  (Gerhard Gentzen, 1934 [9])

Backus/Naur

- John Backus influenced by Emil Post's productions gives a syntax to write production rules *with multiple alternatives* for a context-free grammar for the International Algorithmic Language:

  $$\text{“}< digit >:\equiv 0 \;\overline{or}\; 1 \;\overline{or}\; 2 \;\overline{or}\; 3 \;\overline{or}\; 4 \;\overline{or}\; 5 \;\overline{or}\; 6 \;\overline{or}\; 7 \;\overline{or}\; 8 \;\overline{or}\; 9$$
  $$< integer >:\equiv< digit > \overline{or} < integer >< digit >\text{”}$$

  (John Backus, 1959)

- Peter Naur uses Backus notation where
  - $:\equiv \Longrightarrow ::=$ and
  - $\overline{or} \Longrightarrow |$ and *gave nonterminals the same names used in the text.*

  $$\text{“} < unsigned\ integer >::=< digit > | < unsigned\ integer >< digit >$$
  $$< integer >::=< unsigned\ integer > |+ < unsigned\ integer > |- < unsigned\ integer > \text{”}$$

  (Naur, report on Algol 60, CACM)

Data Declaration à la Alonzo Church

Type Free

- $A ::= x \mid AB \mid \lambda x.B$
- $(\lambda x.B)C \to_\beta B[x := C]$.

With simple types:

- $\sigma ::= T \mid \sigma \to \tau$
- $A ::= x \mid AB \mid \lambda x{:}\sigma.B$
- $(\lambda x : \sigma.B)C \to_\beta B[x := C]$.

With dependent types:

- $A ::= x \mid * \mid \Box \mid AB \mid \lambda x{:}A.B \mid \Pi x{:}A.B$
- $(\lambda x : A.B)C \to_\beta B[x := C]$.
- Sometimes also with $(\Pi x : A.B)C \to_\Pi B[x := C]$.

Code à la Alonzo Church

With dependent/polymorphic types

$$(\lambda) \qquad \frac{\Gamma, x{:}A \vdash b : B \qquad \Gamma \vdash \Pi_{x:A}.B : s}{\Gamma \vdash \lambda_{x:A}.b : \Pi_{x:A}.B}$$

$$(\mathrm{app}_\Pi) \qquad \frac{\Gamma \vdash F : \Pi_{x:A}.B \qquad \Gamma \vdash a : A}{\Gamma \vdash Fa : B[x{:=}a]}$$

$$(\text{N-app}_\Pi) \qquad \frac{\Gamma \vdash F : \Pi_{x:A}.B \qquad \Gamma \vdash a : A}{\Gamma \vdash Fa : (\Pi_{x:A}.B)a}$$

With simple types:

$$(\lambda_\rightarrow) \qquad \frac{\Gamma, x{:}\sigma \vdash b : \tau}{\Gamma \vdash \lambda_{x:A}.b : \sigma \rightarrow \tau} \qquad (\Gamma \vdash \sigma \rightarrow \tau : *)$$

$$(\mathrm{app}) \qquad \frac{\Gamma \vdash F : \tau \rightarrow \sigma \qquad \Gamma \vdash a : \tau}{\Gamma \vdash Fa : \sigma}$$

In this talk we concentrate on the development of calculi rather than the development of the notation

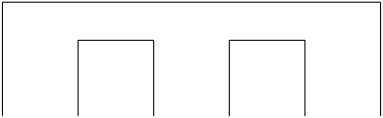
- The challenge is to develop expressive calculi that have clear syntax, semantics, and the desirable properties (Church-Rosser, correctness, termination).
- Nonetheless, notation is important.
- For example, simply changing the order of functions and arguments, and restructuring parenthesis, enable us to:
  - Express things that would be hard to do in the old notation.
  - Reduce proofs of strong normalisation to proofs of weak normalisation.
  - Make computations more efficient.
  - Avoid unnecessary/redundant computations and allow for free lazy, local, or global reductions.

Lambda Calculus à la de Bruijn

- $A := x \mid AB \mid \lambda x.B$ $\qquad\qquad$ $A := x \mid\ < B > A \mid [x]B$
- $\mathcal{I}(x) = x, \qquad \mathcal{I}(\lambda x.B) = [x]\mathcal{I}(B), \qquad \mathcal{I}(AB) = \langle\mathcal{I}(B)\rangle\mathcal{I}(A)$
- $(\lambda x.\lambda y.xy)z$ translates to $\langle z\rangle[x][y]\langle y\rangle x$.
- The *applicator wagon* $\langle z\rangle$ and *abstractor wagon* $[x]$ occur NEXT to each other.
- $(\lambda x.A)B \rightarrow_\beta A[x := B]$ becomes $\langle B\rangle[x]\ A \rightarrow_\beta [x := B]A$
- The "bracketing structure" of $((\ \lambda_x.(\lambda_y.\lambda_z.–)c)ba)$ is '$[_1\ \ [_2\ \ [_3\ \ ]_2\ \ ]_1\ \ ]_3$', where '$[_i$' and '$]_i$' match.
- The bracketing structure of $\langle a\rangle\langle b\rangle[x]\langle c\rangle[y][z]\ \langle d\rangle$ is simpler: $[\ [\ ][\ ]]$.
- $\langle b\rangle[x]$ and $\langle c\rangle[y]$ are AT-pairs whereas $\langle a\rangle[z]$ is an AT-couple.

Redexes in de Bruijn's notation

Classical Notation	de Bruijn's Notation
$(\underline{(\lambda_x.(\lambda_y.\lambda_z.zd)c)b})a$	$\langle a\rangle\underline{\langle b\rangle[x]}\langle c\rangle[y][z]\langle d\rangle z$
$\downarrow_\beta$	$\downarrow_\beta$
$(\underline{(\lambda_y.\lambda_z.zd)c})a$	$\langle a\rangle\underline{\langle c\rangle[y]}[z]\langle d\rangle z$
$\downarrow_\beta$	$\downarrow_\beta$
$\underline{(\lambda_z.zd)a}$	$\underline{\langle a\rangle[z]}\langle d\rangle z$
$\downarrow_\beta$	$\downarrow_\beta$
ad	$\langle d\rangle a$



$\langle a\rangle \quad \langle b\rangle \quad [x] \quad \langle c\rangle \quad [y] \quad [z] \quad \langle d\rangle \quad z$

- This maks it easy to study local/global/mini reductions into the λ-calculus, Kamareddine etal [16, 17]

Some notions of reduction studied in the literature

Name	*In Classical Notation*	*In de Bruijn's notation*
(θ)	$((\lambda_x.N)P)Q \downarrow (\lambda_x.NQ)P$	$\langle Q\rangle\langle P\rangle[x]N \downarrow \langle P\rangle[x]\langle Q\rangle N$
(γ)	$(\lambda_x.\lambda_y.N)P \downarrow \lambda_y.(\lambda_x.N)P$	$\langle P\rangle[x][y]N \downarrow [y]\langle P\rangle[x]N$
(γ_C)	$((\lambda_x.\lambda_y.N)P)Q \downarrow (\lambda_y.(\lambda_x.N)P)Q$	$\langle Q\rangle\langle P\rangle[x][y]N \downarrow \langle Q\rangle[y]\langle P\rangle[x]N$
(g)	$((\lambda_x.\lambda_y.N)P)Q \downarrow (\lambda_x.N[y := Q])P$	$\langle Q\rangle\langle P\rangle[x][y]N \downarrow \langle P\rangle[x][y := Q]N$
(β_e)	? $\downarrow$?	$\langle Q\rangle\overline{s}[y]N \downarrow \overline{s}[y := Q]N$

A Few Uses of these reductions/term reshuffling

- Regnier [26] uses θ and γ in analyzing perpetual reduction strategies.
- Term reshuffling is used by Kfoury, Tiuryn, Urzyczyn, Wells in [21, 19] in analyzing typability problems.
- Nederpelt [23], de Groote [7], Kfoury+ Wells [20], and Kamareddine [15] use generalised reduction and/or term reshuffling in relating SN to WN.
- Ariola etal [1] uses a form of term-reshuffling in obtaining a calculus that corresponds to lazy functional evaluation.
- Kamareddine etal [16, 14, 18, 3] show that they could reduce space/time needs in computation.

Even more: de Bruijn's generalised reduction has better properties

$$
\begin{array}{lll}
(\beta) & (\lambda_x.M)N \to M[x := N] & \\
(\beta_I) & (\lambda_x.M)N \to M[x := N] & \text{if } x \in FV(M) \\
(\beta_K) & (\lambda_x.M)N \to M & \text{if } x \notin FV(M) \\
(\theta) & (\lambda_x.N)PQ \to (\lambda_x.NQ)P & \\
(\beta_e) & (M)\overline{s}[x]N \to \overline{s}\{N[x := M] & \text{for } \overline{s} \text{ well-balanced.}
\end{array}
$$

- Kamareddine [15] shows that β_e satisfies *Church Rosser*, *PSN*, postponment of K-contraction and conservation (latter 2 properties fail for β-reduction).
- *Conservation of β_e*: If A is $\beta_e I$-normalisable then A is β_e-strongly normalisable.
- Postponment of K-contraction : Hence, discard arguments of K-redexes after I-reduction. This gives flexibility in implementation: *unnecessary work can be delayed, or even completely avoided.*

- Attempts have been made at establishing some reduction relations for which postponement of K-contractions and conservation hold.
- The picture is as follows (-N stands for normalising and $r \in \{\beta_I, \theta_K\}$).

(β_K-postponement for r)	If $M \to_{\beta_K} N \to_r O$ then $\exists P$ such that $M \twoheadrightarrow^+_{\beta_I \theta_K} P \twoheadrightarrow_{\beta_K} O$
(Conservation for β_I)	If M is β_I-N then M is β_I-SN Barendregt's book
(Conservation for $\beta + \theta$)	If M is $\beta_I \theta_K$-N then M is β-SN [7]

- De Groote does not produce these results for a single reduction relation, but for $\beta + \theta$ (this is more restrictive than β_e).
- β_e is the first single relation to satisfy β_K-postponement and conservation.
- Kamareddine [15] shows that:

(β_{eK}-postponement for β_e)	If $M \to_{\beta_{eK}} N \to_{\beta_{eI}} O$ then $\exists P$ such that $M \to_{\beta_{eI}} P \twoheadrightarrow^+_{\beta_{eK}} O$
(Conservation for β_e)	If M is β_{eI}-N then M is β_e-SN

Canonical typing

There are reasons why separating the questions "what is the type of a term" (via τ) and "is the term typable" (via $\vdash$), is advantageous:

- The canonical type of A is easy to calculate.
- $\tau(A)$ plays the role of a preference type for A. If $A \equiv \lambda_{x:*}.(\lambda_{y:*}.y)x$ then $\tau(<>, A) \equiv \Pi_{x:*}.(\Pi_{y:*}.*)x \twoheadrightarrow_{\beta\Pi} \Pi_{y:*}.*$, the type of A.
- The conversion rule is no longer needed as a separate rule in the definition of $\vdash$. It is accommodated in our application rule:

$$\frac{\Gamma \vdash A \qquad\qquad \Gamma \vdash B}{\Gamma \vdash AB} \text{ if } \tau(\Gamma, A) =_{\beta\Pi} \Pi_{x:C}.D \text{ and } \tau(\Gamma, B) =_{\beta\Pi} C$$

- Higher degrees:If we use λ^1 for Π and λ^2 for λ then we can aim for a possible generalization. In fact, we can extend our system by incorporating more different λ's. For example, with an infinity of λ's, viz. λ^0, λ^1, λ^2, λ^3 ..., we replace $\tau(\Gamma, \lambda_{x:A}.B) \equiv \Pi_{x:A}.\tau(\Gamma.\lambda_{x:A}, B)$ and $\tau(\Gamma, \Pi_{x:A}.B) \equiv \tau(\Gamma.\lambda_{x:A}, B)$ by the following:

  $$\tau(\Gamma, \lambda^{i+1}_{x:A}.B) \equiv \lambda^{i}_{x:A}.\tau(\Gamma.\lambda_{x:A}, B), \text{for } i = 0, 1, 2, \ldots \text{ where } \lambda^0_{x:A}.B \equiv B$$

  There may be circumstances in which one desires to have more "layers" of λ's (see de Bruijn 1974).

- This notion enables one to separate the judgement $\Gamma \vdash A : B$ in two: $\Gamma \vdash A$ and $\tau(\Gamma, A) = B$.

$$
\begin{array}{lcll}
\tau(\Gamma, *) & \equiv & \Box & \\
\tau(\Gamma, x) & \equiv & A \text{ if } (A\lambda_x) \in \Gamma & \\
\tau(\Gamma, (a\delta)F) & \equiv & (a\delta)\tau(\Gamma, F) & \\
\tau(\Gamma, (A\lambda_x)B) & \equiv & (A\Pi_x)\tau(\Gamma(A\lambda_x), B) & \text{if } x \notin dom(\Gamma) \\
\tau(\Gamma, (A\Pi_x)B) & \equiv & \tau(\Gamma(A\lambda_x), B) & \text{if } x \notin dom(\Gamma)
\end{array}
$$

- In usual type theory:
  - the type of $(*\lambda_x)(x\lambda_y)y$ is $(*\Pi_x)(x\Pi_y)x$ and
  - the type of $(*\Pi_x)(x\Pi_y)x$ is $*$.
- With our τ, we get the same result:
  - $\tau(<>,(*\lambda_x)(x\lambda_y)y) \equiv (*\Pi_x)\tau((*\lambda_x),(x\lambda_y)y) \equiv (*\Pi_x)(x\Pi_y)\tau((*\lambda_x)(x\lambda_y),y) \equiv (*\Pi_x)(x\Pi_y)x$ and
  - $\tau(<>,(*\Pi_x)(x\Pi_y)x) \equiv \tau((*\lambda_x),(x\Pi_y)x) \equiv \tau((*\lambda_x)(x\lambda_y),x) \equiv *$

Let $\Gamma_0 \equiv <>$, $\Gamma_1 \equiv (*\lambda_z)$, $\Gamma_2 \equiv (*\lambda_z)(*\lambda_y)$, $\Gamma_3 \equiv \Gamma_2(*\lambda_x)$. We want to find the canonical type of $(*\Pi_z)(B\delta)(*\lambda_y)(y\delta)(*\lambda_x)x$ in Γ_0.

$$
\begin{array}{cccccccccccc}
(\Gamma_0\tau) & (*\Pi_z) & & (B\delta) & & (*\lambda_y) & & (y\delta) & & (*\lambda_x) & & x \\
 & & (\Gamma_1\tau) & (B\delta) & & (*\lambda_y) & & (y\delta) & & (*\lambda_x) & & x \\
 & & & (B\delta) & (\Gamma_1\tau) & (*\lambda_y) & & (y\delta) & & (*\lambda_x) & & x \\
 & & & (B\delta) & & (*\Pi_y) & (\Gamma_2\tau) & (y\delta) & & (*\lambda_x) & & x \\
 & & & (B\delta) & & (*\Pi_y) & & (y\delta) & (\Gamma_2\tau) & (*\lambda_x) & & x \\
 & & & (B\delta) & & (*\Pi_y) & & (y\delta) & & (*\Pi_x) & (\Gamma_3\tau) & x \\
 & & & (B\delta) & & (*\Pi_y) & & (y\delta) & & (*\Pi_x) & & *
\end{array}
$$

New Typability

($\vdash$-axiom)	$<>\vdash *$
($\vdash$-start rule)	$\dfrac{\Gamma \vdash A}{\Gamma(A\lambda_x) \vdash x}$ if vc
($\vdash$-weakening rule)	$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash D}{\Gamma(A\lambda_x) \vdash D}$ if vc
($\vdash$-application rule)	$\dfrac{\Gamma \vdash F \qquad \Gamma \vdash a}{\Gamma \vdash (a\delta)F}$ if ap
($\vdash$-abstraction rule)	$\dfrac{\Gamma(A\lambda_x) \vdash b \qquad \Gamma \vdash (A\Pi_x)B}{\Gamma \vdash (A\lambda_x)b}$ if ab
($\vdash$-formation)	$\dfrac{\Gamma \vdash A \qquad \Gamma(A\lambda_x) \vdash B}{\Gamma \vdash (A\Pi_x)B}$ if fc

- vc (variable condition): $x \notin \Gamma$ and $\tau(\Gamma, A) \twoheadrightarrow_{\beta\Pi} S$ for some S
- ap (application condition): $\tau(\Gamma, F) =_{\beta\Pi} (A\Pi_x)B$ and $\tau(\Gamma, a) =_{\beta\Pi} A$ for some A, B.
- ab (abstraction condition): $\tau(\Gamma(A\lambda_x), b) =_{\beta\Pi} B$ and $\tau(\Gamma, (A\Pi_x)B) \twoheadrightarrow_{\beta\Pi} S$ for some S.
- fc (formation condition): $\tau(\Gamma, A) \twoheadrightarrow_{\beta\Pi} S_1$ and $\tau(\Gamma(A\lambda_x), B) \twoheadrightarrow_{\beta\Pi} S_2$ for some rule (S_1, S_2).

Properties of $\vdash$

Define $\overline{A}$ to be the $\beta\Pi$-normal form of A.

Lemma

If $\Gamma \vdash A$ *then* $\downarrow \overline{\tau(\Gamma, A)}$ *and* $\Gamma \vdash_\beta A : \overline{\tau(\Gamma, A)}$

Lemma

(Subject Reduction for $\vdash$ *and* τ*)*
$\Gamma \vdash A \wedge A \twoheadrightarrow_{\beta\Pi} A' \Rightarrow [\Gamma \vdash A' \wedge \tau(\Gamma, A) =_{\beta\Pi} \tau(\Gamma, A')]$

Theorem

(Strong Normalisation for $\vdash$*)*
If A *is* $\Gamma^{\vdash}$*-legal, then* $SN_{\twoheadrightarrow_\beta}(A)$.

Lemma

$\Gamma \vdash_\beta A : B \Longleftrightarrow \Gamma \vdash A$ *and* $\tau(\Gamma, A) =_{\beta\Pi} B$ *and* B *is* $\vdash_\beta$*-legal type.*

From Frege's low level functions to PTSs that capture strong normalisation

- Kamareddine and Wells 2017, has incorporated Frege's low level of functions to create PTSs with intersection types which contain all the ordinary PTSs (including the β-cube given above and its extensions with parameters/Frege's functions.
- The f-cube is the β-cube extended with finite set declarations in the form of ordinary mathematical notion of function.
- **Theorem:** If $\Gamma \vdash_f A : B$ then A and B are strongly normalising.
- **Theorem:** If a type free term of the λ-calculus M is strongly Normalising then M is typable in the f-cube.
- Urzyczyn proved $U = (\lambda r.\, h(r(\lambda f \lambda s.\, f\, s))(r(\lambda q.\lambda g.\, g\, q)))(\lambda o.\, o\, o\, o)$ is untypable in Fω. Hence U is untypable in any system of the cube.
- But U is strongly normalising.
- Kamareddine and Wells 2017 prove that U is typable in the f-cube: There are Γ, A such that $\Gamma \vdash_f U : A$.

Conclusion

- A hierarchy of systems that classify important properties of CR, SN, SR.
- Not only types are used to derive important properties and avoid paradoxes and non termination, but also types classify non termination.

[1] Zena M. Ariola, Matthias Felleisen, John Maraist, Martin Odersky, and Philip Wadler. The call-by-need lambda calculus. pages 233–246.

[2] H[endrik] P[ieter] Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, revised edition, 1984.

[3] R. Bloo, F. Kamareddine, and R. Nederpelt. The Barendregt Cube with Definitions and Generalised Reduction. *Information and Computation*, 126(2):123–143, 1996.

[4] Alonzo Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5:56–68, 1940.

[5] Thierry Coquand and Gérard Huet. The Calculus of Constructions. *Inform. & Comput.*, 76:95–120, 1988.

[6] N.G. de Bruijn. The mathematical language Automath – its usage and some of its extensions. In L. Laudet, D. Lacombe, and M. Schuetzenberger, editors, *Symposium on Automatic Demonstration*, volume 125 of *Lecture Notes in Mathematics*, pages 29–61, Heidelberg, 1970. Springer-Verlag. Reprinted in [24, A.2].

[7] Philippe de Groote. The conservation theorem revisited. pages 163–178.

[8] Gottlob Frege. *Begriffsschrift: eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Nebert, Halle, 1879. Can be found on pp. 1–82 in [31].

[9] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1934.

[10] J[ean]-Y[ves] Girard. *Interprétation Fonctionnelle et Elimination des Coupures de l'Arithmétique d'Ordre Supérieur*. Thèse d'Etat, Université de Paris VII, 1972.

[11] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. In *Proceedings Second Symposium on Logic in Computer Science*, pages 194–204, Washington D.C., 1987. IEEE.

[12] D. Hilbert and W. Ackermann. *Grundzüge der Theoretischen Logik*. Die Grundlehren der Mathematischen Wissenschaften in Einzeldarstellungen, Band XXVII. Springer Verlag, Berlin, first edition, 1928.

[13] J. Roger Hindley and Jonathan P. Seldin. *Introduction to Combinators and λ-calculus*, volume 1 of *London Mathematical Society Student Texts*. Cambridge University Press, 1986.

[14] F. Kamareddine, R. Bloo, and R. Nederpelt. On Π-conversion in the λ-cube and the combination with abbreviations. *Ann. Pure Appl. Logic*, 97(1–3):27–45, 1999.

[15] Fairouz Kamareddine. Postponement, conservation and preservation of strong normalisation for generalised reduction. *J. Logic Comput.*, 10(5):721–738, 2000.

[16] Fairouz Kamareddine and Rob Nederpelt. Refining reduction in the λ-calculus. *J. Funct. Programming*, 5(4):637–651, October 1995.

[17] Fairouz Kamareddine and Rob Nederpelt. A useful λ-notation. *Theoret. Comput. Sci.*, 155(1):85–109, 1996.

[18] Fairouz Kamareddine, Alejandro Ríos, and J. B. Wells. Calculi of generalised β-reduction and explicit substitutions: The type free and simply typed versions. *J. Funct. Logic Programming*, 1998(5), June 1998.

[19] A. J. Kfoury and J. B. Wells. A direct algorithm for type inference in the rank-2 fragment of the second-order λ-calculus. pages 196–207.

[20] A. J. Kfoury and J. B. Wells. New notions of reduction and non-semantic proofs of β-strong normalization in typed λ-calculi. pages 311–321.

[21] Assaf J. Kfoury, Jerzy Tiuryn, and Paweł Urzyczyn. An analysis of ML typability. *J. ACM*, 41(2):368–398, March 1994.

[22] G. Longo and E. Moggi. Constructive natural deduction and its modest interpretation. Technical Report CMU-CS-88-131, Carnegie Mellon University, Pittsburgh, USA, 1988.

[23] Rob Nederpelt. *Strong Normalization in a Typed Lambda Calculus With Lambda Structured Types*. PhD thesis, Technical University of Eindhoven, 1973.

[24] Rob Nederpelt, J. H. Geuvers, and Roel C. de Vrijer. *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1994.

[25] F.P. Ramsey. The foundations of mathematics. *Proceedings of the London Mathematical Society,* 2nd series, 25:338–384, 1926.

[26] Laurent Regnier. *Lambda calcul et réseaux*. PhD thesis, University Paris 7, 1992.

[27] G.R. Renardel de Lavalette. Strictness analysis via abstract interpretation for recursively defined types. *Information and Computation*, 99:154–177, 1991.

[28] J. C. Reynolds. Towards a theory of type structure. In *Colloque sur la Programmation*, volume 19 of *LNCS*, pages 408–425. Springer-Verlag, 1974.

[29] B. Russell. Letter to Frege. English translation in [31], pages 124–125, 1902.

[30] B. Russell. *The Principles of Mathematics*. Allen & Unwin, London, 1903.

[31] J. van Heijenoort. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Harvard University Press, 1967.

[32] Alfred North Whitehead and Bertrand Russel. *Principia Mathematica*. Cambridge University Press, 1910–1913. In three volumes published from 1910 through 1913. Second edition published from 1925 through 1927. Abridged edition published in 1962.