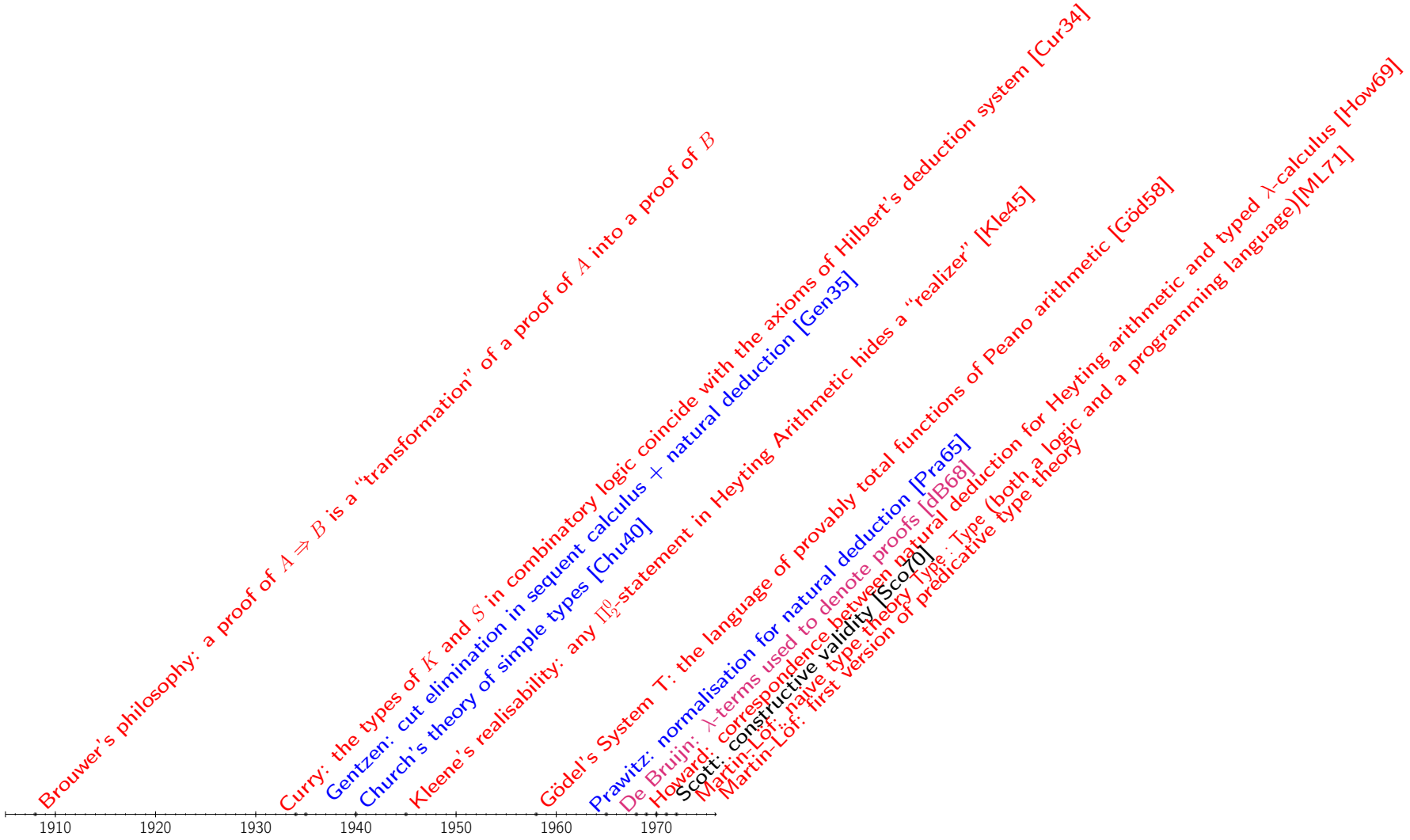
Introduction to (Homotopy) Type Theory

Intensive class - LMFI 2021

Hugo Herbelin

April 13, 2021

Historical context: the emergence of the awareness that proofs and programs are two sides of a common language

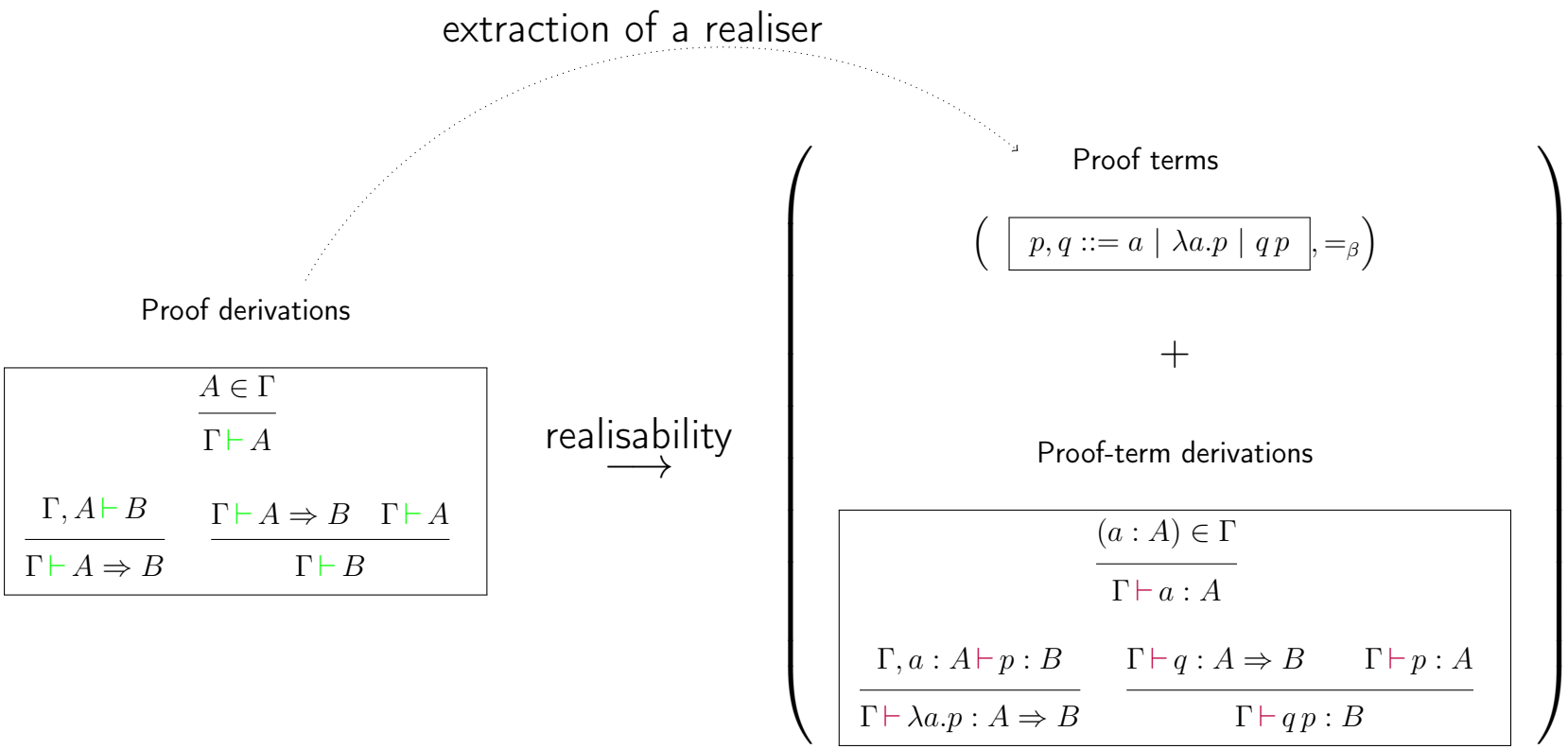


direct contribution to eventually being aware of the identity of structure between proofs and programs

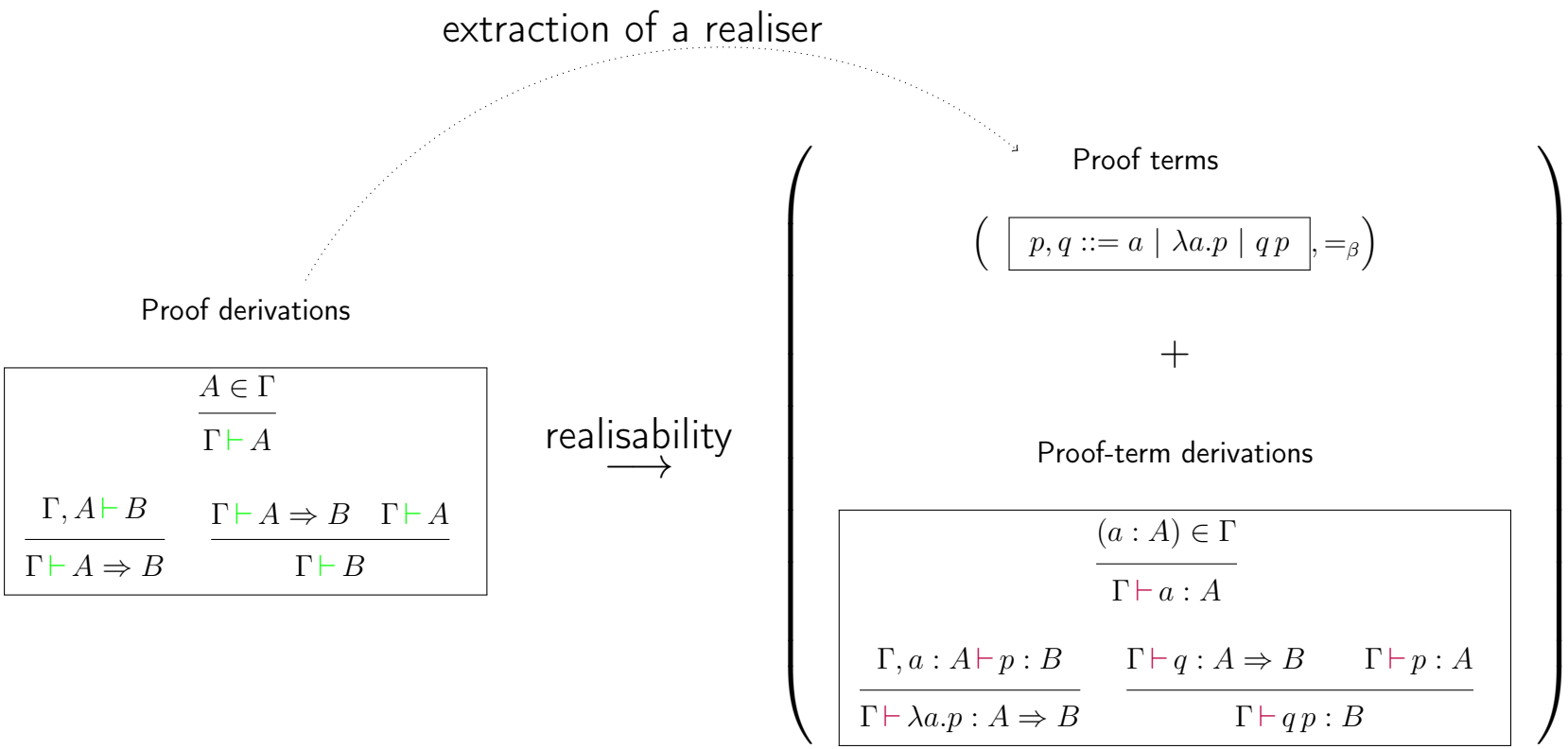
independent observations that proofs do "compute" (in some sense)

From realisability to the Curry-Howard correspondence

Kleene-style realisability: extraction of a realiser



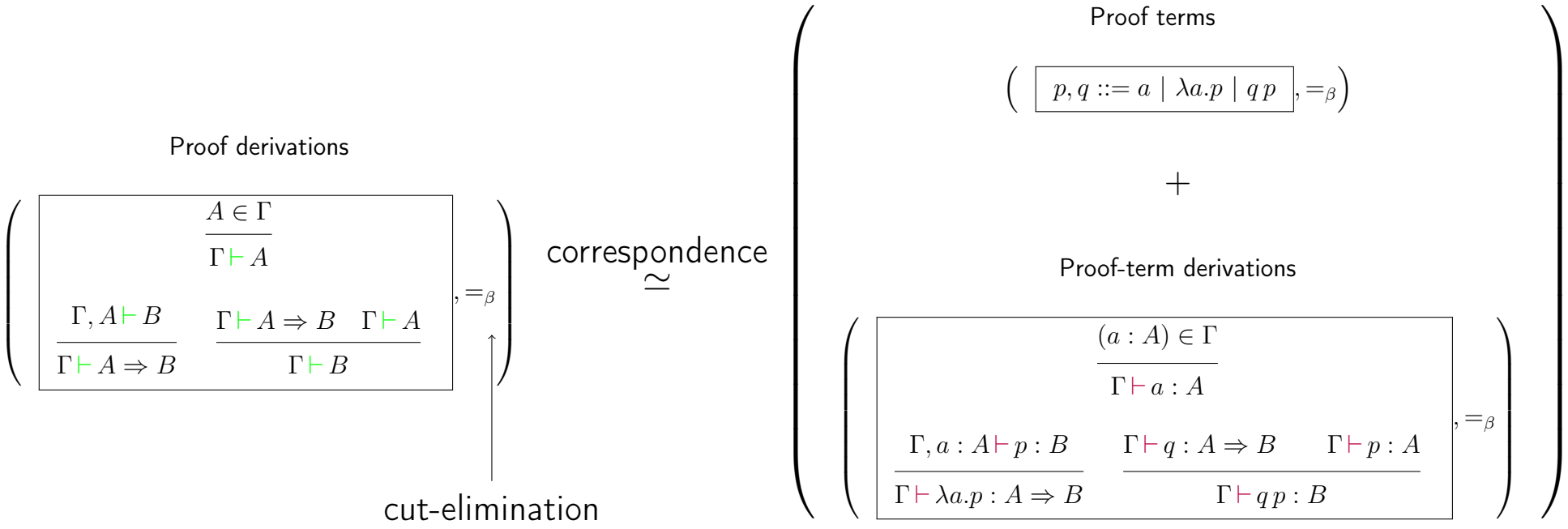
Kleene-style realisability: extraction of a realiser



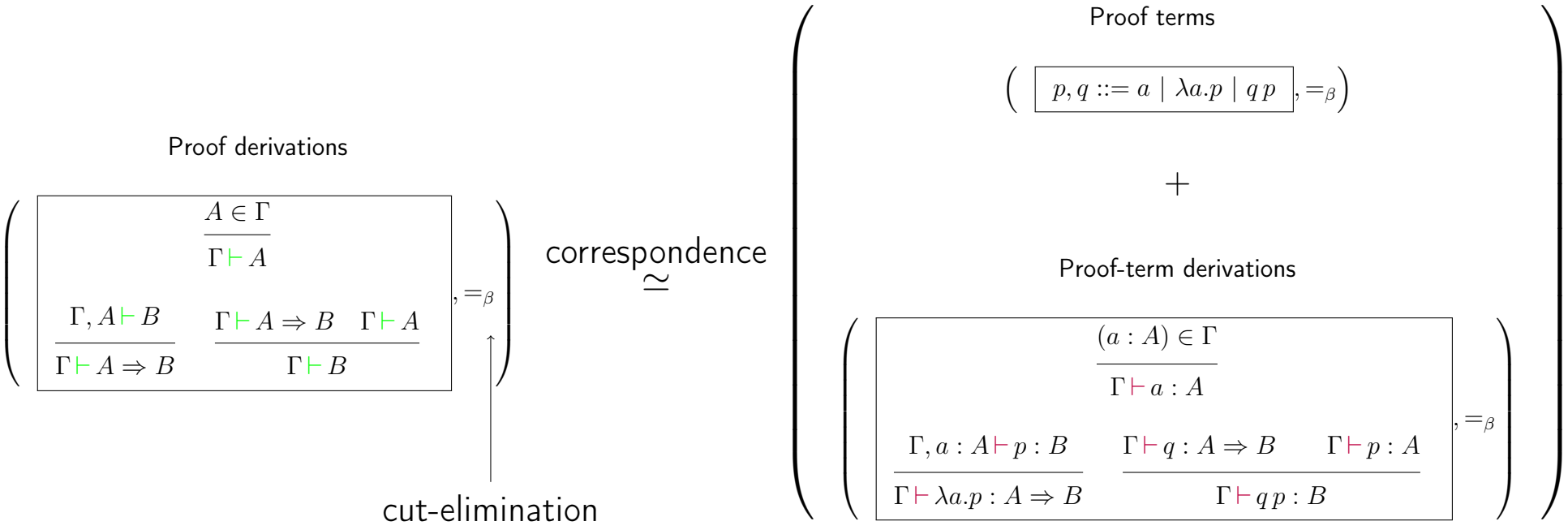
up to the fact that:

- Kleene used codes of partial recursion functions instead of untyped λ-terms
- it covered full Heyting arithmetic and used the symbol "r", meaning "realise", instead of ":" (they connect as follows: $q\ \mathsf{r}\ A \Rightarrow B \triangleq \forall a\,(a : A) \Rightarrow (q\,a : B)$ and $p : A \triangleq \forall q, q \to^*_\beta p \Rightarrow q\ \mathsf{r}\ A$)

Curry-Howard-style proofs-as-programs correspondence



Curry-Howard-style proofs-as-programs correspondence



and many variants, for instance,

- Howard actually used type-annotated proof-terms $p^A ::= a^A \mid (\lambda a^A.p^B)^{A \Rightarrow B} \mid (q^{A \Rightarrow B}\, p^A)^B$
- it covered full Heyting arithmetic and did not exactly show the simulation of β-reduction
- see Bernardy, Lasson [BL11, Las14] for the general picture

Curry-Howard-style proofs-as-programs correspondence with quantification

Proof derivations

$$\left(\boxed{\dfrac{A \in \Gamma}{\Gamma \vdash A} \qquad \dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \qquad \dfrac{\Gamma \vdash B(x) \quad x \notin FV(\Gamma)}{\Gamma \vdash \forall x\, B(x)} \qquad \dfrac{\Gamma \vdash \forall x\, B(x)}{\Gamma \vdash B(t)}}, =_\beta \right)$$

correspondence $\simeq$

Proof terms

$$\left(\boxed{a \mid \lambda a.p \mid q\,p \mid \lambda x.p \mid q\,t}, =_\beta \right)$$

$+$

Proof-term derivations

$$\left(\boxed{\dfrac{(a : A) \in \Gamma}{\Gamma \vdash a : A} \qquad \dfrac{\Gamma, a : A \vdash p : B}{\Gamma \vdash \lambda a.p : A \Rightarrow B} \qquad \dfrac{\Gamma \vdash q : A \Rightarrow B \qquad \Gamma \vdash p : A}{\Gamma \vdash q\,p : B} \qquad \dfrac{\Gamma \vdash p : B(x) \quad x \notin FV(\Gamma)}{\Gamma \vdash \lambda x.p : \forall x\, B(x)} \qquad \dfrac{\Gamma \vdash q : \forall x\, B(x)}{\Gamma \vdash q\,t : B(t)}}, =_\beta \right)$$

From predicate calculus to type theory

Remark: an explicit way to ensure freshness of quantified variables in predicate logic

Add declaration of variables in contexts and replace freshness by inference rules:

Well-formedness of propositions

$$A ::= P(t_1, ..., t_{\mathsf{ar}(P)}) \mid A \Rightarrow B \mid \forall x.\, A(x)$$

$$\frac{\ldots\ \Gamma \vdash t_i\ \ \mathsf{ok}\ \ldots}{\Gamma \vdash P(t_1, ..., t_{\mathsf{ar}(P)})\ \mathsf{prop}} \qquad \frac{\Gamma \vdash A\ \mathsf{prop} \quad \Gamma \vdash B\ \mathsf{prop}}{\Gamma \vdash A \Rightarrow B\ \mathsf{prop}}$$

$$\frac{\Gamma, x \vdash A(x)\ \mathsf{prop}}{\Gamma \vdash \forall x.\, A(x)\ \mathsf{prop}}$$

Well-formedness of terms

$$t ::= x \mid f(t_1, ..., t_{\mathsf{ar}(f)})$$

$$\frac{x \in \Gamma}{\Gamma \vdash x\ \ \mathsf{ok}}$$

$$\frac{\ldots\ \Gamma \vdash t_i\ \ \mathsf{ok}\ \ldots}{\Gamma \vdash f(t_1, ..., t_{\mathsf{ar}(f)})\ \ \mathsf{ok}}$$

Proof derivations

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

$$\frac{\Gamma \vdash A\ \mathsf{prop} \quad \Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

$$\frac{\Gamma, x \vdash B(x)}{\Gamma \vdash \forall x\, B(x)} \qquad \frac{\Gamma \vdash \forall x\, B(x) \quad \Gamma \vdash t\ \ \mathsf{ok}}{\Gamma \vdash B(t)}$$

Realisability associates canonical λ-calculi to logics

Let $\vdash$ be a logic, what are the recursive functions provably total in this logic[1]?

- Predicate calculus: simply-typed λ-calculus
- Heyting arithmetic: system T (Gödel [Göd58])
- Second-order Heyting arithmetic: system F (Girard [Gir71])
- Higher-order Heyting arithmetic: system F_ω (Girard [Gir71])

Therefore, there is a natural temptation to canonically extend a logic with quantification over all its provably total programs

- Predicate calculus: extend the logic with quantification over simply-typed λ-calculus (see LF, $\lambda\Pi$)
- Heyting arithmetic (HA): quantify over system T (Gödel [Göd58]) and get HA^ω (= arithmetic in finite types)
- Second-order Heyting arithmetic (HA_2): quantify over system F (Girard [Gir71]) (see Lasson [Las14])
- Higher-order Heyting arithmetic (HA_ω): quantify over system F_ω (Girard [Gir71]) (see the Calculus of Constructions [CH85] [Las14])

[1] assuming the logic to include an axiomatisation of recursive functions

Example: predicate logic over simply-typed λ-calculus

A function symbol f is now prescribed a type $\mathsf{arg}_1(f) \times \ldots \mathsf{arg}_{\mathsf{ar}(f)}(f) \to \mathsf{concl}(f)$ and similarity for predicate symbols.

Well-formedness of propositions

$$A ::= P(t_1, ..., t_{\mathsf{ar}(P)}) \mid A \Rightarrow B \mid \forall x.\, A(x)$$

$$\frac{\ldots\ \Gamma \vdash t_i : \mathsf{arg}_i(P)\ \ldots}{\Gamma \vdash P(t_1, ..., t_{\mathsf{ar}(P)})\ \mathsf{prop}} \qquad \frac{\Gamma \vdash A\ \mathsf{prop} \quad \Gamma \vdash B\ \mathsf{prop}}{\Gamma \vdash A \Rightarrow B\ \mathsf{prop}}$$

$$\frac{\Gamma, x \vdash A(x)\ \mathsf{prop}}{\Gamma \vdash \forall x.\, A(x)\ \mathsf{prop}}$$

Well-typing of terms

$$t, u ::= x \mid f(t_1, ..., t_{\mathsf{ar}(f)}) \mid \lambda x.t \mid u\, t$$

$$\frac{(x : T) \in \Gamma}{\Gamma \vdash x : T} \qquad \frac{\ldots\ \Gamma \vdash t_i\ : \mathsf{arg}_i(f)\ \ldots}{\Gamma \vdash f(t_1, ..., t_{\mathsf{ar}(f)}) : \mathsf{concl}(f)}$$

$$\frac{\Gamma, x : T \vdash t : U}{\Gamma \vdash \lambda x.t : T \to U} \qquad \frac{\Gamma \vdash u : T \to U \quad \Gamma \vdash t : T}{\Gamma \vdash u\, t : U}$$

Equality of propositions

$A \equiv_\beta B$
is the refl-sym-trans congruent closure on types of β-reduction on terms

Proof derivations

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

$$\frac{\Gamma \vdash A\ \mathsf{prop} \quad \Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

$$\frac{\Gamma, x : T \vdash B(x)}{\Gamma \vdash \forall x^T.\, B(x)} \qquad \frac{\Gamma \vdash \forall x^T.\, B(x) \quad \Gamma \vdash t : T}{\Gamma \vdash B(t)}$$

$$\frac{\Gamma \vdash A \quad A \equiv_\beta B}{\Gamma \vdash B}$$

Where are we?

- If we want to have more complex domains of quantification, the complexity of the logic increases
- Proofs have the same structure as programs: implication is an arrow type and universal quantification is a dependent function type
- Can we reduce the number of rules while still preserving a rich structure of types and connectives?

First step: move to the right-hand side of the Curry-Howard correspondence

Well-formedness of propositions

$$A ::= P(t_1, ..., t_{\mathsf{ar}(P)}) \mid A \Rightarrow B \mid \forall x.\, A(x)$$

$$\frac{\ldots\ \Gamma \vdash t_i : \mathsf{arg}_i(P)\ \ldots}{\Gamma \vdash P(t_1, ..., t_{\mathsf{ar}(P)})\ \mathsf{prop}} \qquad \frac{\Gamma \vdash A\ \mathsf{prop} \quad \Gamma \vdash B\ \mathsf{prop}}{\Gamma \vdash A \Rightarrow B\ \mathsf{prop}}$$

$$\frac{\Gamma, x \vdash A(x)\ \mathsf{prop}}{\Gamma \vdash \forall x.\, A(x)\ \mathsf{prop}}$$

Well-typing of terms

$$t, u ::= x \mid f(t_1, ..., t_{\mathsf{ar}(f)}) \mid \lambda x.t \mid u\, t$$

$$\frac{(x : T) \in \Gamma}{\Gamma \vdash x : T} \qquad \frac{\ldots\ \Gamma \vdash t_i\ : \mathsf{arg}_i(f)\ \ldots}{\Gamma \vdash f(t_1, ..., t_{\mathsf{ar}(f)}) : \mathsf{concl}(f)}$$

$$\frac{\Gamma, x : T \vdash t : U}{\Gamma \vdash \lambda x.t : T \to U} \qquad \frac{\Gamma \vdash u : T \to U \quad \Gamma \vdash t : T}{\Gamma \vdash u\, t : U}$$

Equality of propositions

$A \equiv_\beta B$
is the refl-sym-trans congruent closure on types of β-reduction on terms

Proof derivations

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

$$\frac{\Gamma \vdash A\ \mathsf{prop} \quad \Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

$$\frac{\Gamma, x : T \vdash B(x)}{\Gamma \vdash \forall x^T.\, B(x)} \qquad \frac{\Gamma \vdash \forall x^T.\, B(x) \quad \Gamma \vdash t : T}{\Gamma \vdash B(t)}$$

$$\frac{\Gamma \vdash A \quad A \equiv_\beta B}{\Gamma \vdash B}$$

First step: move to the right-hand side of the Curry-Howard correspondence

Well-formedness of propositions

$$A ::= P(t_1, ..., t_{\mathsf{ar}(P)}) \mid A \Rightarrow B \mid \forall x.\, A(x)$$

$$\frac{\ldots\ \Gamma \vdash t_i : \mathsf{arg}_i(P)\ \ldots}{\Gamma \vdash P(t_1, ..., t_{\mathsf{ar}(P)})\ \mathsf{prop}} \qquad \frac{\Gamma \vdash A\ \mathsf{prop} \quad \Gamma \vdash B\ \mathsf{prop}}{\Gamma \vdash A \Rightarrow B\ \mathsf{prop}}$$

$$\frac{\Gamma, x \vdash A(x)\ \mathsf{prop}}{\Gamma \vdash \forall x.\, A(x)\ \mathsf{prop}}$$

Well-typing of terms

$$t, u ::= x \mid f(t_1, ..., t_{\mathsf{ar}(f)}) \mid \lambda x.t \mid u\, t$$

$$\frac{(x : T) \in \Gamma}{\Gamma \vdash x : T} \qquad \frac{\ldots\ \Gamma \vdash t_i\ : \mathsf{arg}_i(f)\ \ldots}{\Gamma \vdash f(t_1, ..., t_{\mathsf{ar}(f)}) : \mathsf{concl}(f)}$$

$$\frac{\Gamma, x : T \vdash t : U}{\Gamma \vdash \lambda x.t : T \to U} \qquad \frac{\Gamma \vdash u : T \to U \quad \Gamma \vdash t : T}{\Gamma \vdash u\, t : U}$$

Equality of propositions

$A \equiv_\beta B$
is the refl-sym-trans congruent closure on types of β-reduction on terms

Proof-term derivations

$$\frac{(a : A) \in \Gamma}{\Gamma \vdash a : A}$$

$$\frac{\Gamma \vdash A\ \mathsf{prop} \quad \Gamma, a : A \vdash p : B}{\Gamma \vdash \lambda a.p : A \Rightarrow B} \qquad \frac{\Gamma \vdash q : A \Rightarrow B \qquad \Gamma \vdash p : A}{\Gamma \vdash q\, p : B}$$

$$\frac{\Gamma, x : T \vdash p : B(x)}{\Gamma \vdash \lambda x.p : \forall x^T.\, B(x)} \qquad \frac{\Gamma \vdash q : \forall x\, B(x) \quad \Gamma \vdash t : T}{\Gamma \vdash q\, t : B(t)}$$

$$\frac{\Gamma \vdash p : A \quad A \equiv_\beta B}{\Gamma \vdash p : B}\ \text{conversion rule}$$

Second step: use the unifying syntax of Pure Type Systems

See propositions as types, unify all instances of λ, unify $\Rightarrow$, $\forall$, $\rightarrow$ into a dependent function type Π:

when A and B are intended to be prop. and $a \notin FV(B)$
when T, and U are arbitrary types and $a \notin FV(U)$

$$
\begin{array}{lcl}
A \Rightarrow B & \triangleq & \Pi a^A.\, B \\
\forall x^T.\, A & \triangleq & \Pi x^T.\, A \\
T \rightarrow U & \triangleq & \Pi x^T.\, U
\end{array}
$$

Well-formedness of types

$$A ::= \Pi a^A.\, B$$

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, a : A \vdash B \text{ type}}{\Gamma \vdash \Pi a^A.\, B \text{ type}}$$

Proof-term derivations

$$p, q ::= a \mid \lambda a.p \mid q\, p$$

$$\frac{(a : A) \in \Gamma}{\Gamma \vdash a : A}$$

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, a : A \vdash p : B}{\Gamma \vdash \lambda a.p : \Pi a^A.\, B} \qquad \frac{\Gamma \vdash q : \Pi a^A.\, B \qquad \Gamma \vdash p : A}{\Gamma \vdash q\, p : B[p/a]}$$

$$\frac{\Gamma \vdash p : A \quad A \equiv_\beta B}{\Gamma \vdash p : B}$$

Equality of propositions

$A \equiv_\beta B$
is the refl-sym-trans congruent closure on types of β-reduction on terms

Where are we again?

- We have a unified syntax
- We are missing types such as unit type (= true connective), empty type (= false connective), sum types (= disjunction), product types (= conjunction), dependent product-types (= Σ-types), boolean values, natural numbers, sets, a way to quantify over types, ... all can be added later to give Martin-Löf's type theory
- The unified syntax gave us a new possibility: to treat proofs as objects and to quantify over them! This will be important later.

Type theory under the eyes of the Curry-Howard correspondence

Reminder: The basis of Curry-Howard correspondence

Proof derivations

$$\left(\boxed{\dfrac{A \in \Gamma}{\Gamma \vdash A} \qquad \dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}}, =_\beta \right)$$

correspondence $\simeq$

Proof terms

$$\left(\boxed{p, q ::= a \mid \lambda a.p \mid q\,p}, =_\beta \right)$$

$+$

Proof-term derivations

$$\left(\boxed{\dfrac{(a : A) \in \Gamma}{\Gamma \vdash a : A} \qquad \dfrac{\Gamma, a : A \vdash p : B}{\Gamma \vdash \lambda a.p : A \Rightarrow B} \qquad \dfrac{\Gamma \vdash q : A \Rightarrow B \quad \Gamma \vdash p : A}{\Gamma \vdash q\,p : B}}, =_\beta \right)$$

Curry-Howard correspondence for the unified syntax (the dreamt view)

Intrinsically well-formed types

$$\left(\boxed{\dfrac{\Gamma \vdash \text{ type} \quad \Gamma, A \vdash \text{ type}}{\Gamma \vdash \text{ type}}\ \Pi(A,B)}\ , \equiv_\beta \right)$$

Intrinsically well-typed terms

$$\left(\boxed{\begin{array}{c} \dfrac{A \in \Gamma}{\Gamma \vdash A}\ \mathsf{Var} \\[2ex] \dfrac{\Gamma, A \vdash B}{\Gamma \vdash \Pi A.\, B}\ \mathsf{Lam}(p) \qquad \dfrac{\Gamma \vdash \Pi A.\, B \quad \Gamma \vdash A}{\Gamma \vdash B[p]}\ \mathsf{App(q,p)} \\[2ex] \dfrac{\Gamma \vdash A \quad A \equiv_\beta B}{\Gamma \vdash B}\ \mathsf{Conv}(p,e) \end{array}}\ , =_\beta \right)$$

intrinsic syntax

ideal correspondence $\simeq$

Types $\left(\boxed{A, B ::= \Pi a^A.\, B}\ , \equiv_\beta \right)$

Terms $\left(\boxed{p, q ::= a \mid \lambda a.p \mid q\, p}\ , =_\beta \right)$

$+$

extrinsic syntax

Well-formed types

$$\left(\boxed{\dfrac{\Gamma \vdash A \text{ type} \quad \Gamma, a : A \vdash B \text{ type}}{\Gamma \vdash \Pi a^A.\, B \text{ type}}}\ , \equiv_\beta \right)$$

Well-typed terms

$$\left(\boxed{\begin{array}{c} \dfrac{(a : A) \in \Gamma}{\Gamma \vdash a : A} \\[2ex] \dfrac{\Gamma \vdash A \text{ type} \quad \Gamma, a : A \vdash p : B}{\Gamma \vdash \lambda a.p : \Pi a^A.\, B} \qquad \dfrac{\Gamma \vdash q : \Pi a^A.\, B \qquad \Gamma \vdash p : A}{\Gamma \vdash q\, p : B[p/a]} \\[2ex] \dfrac{\Gamma \vdash p : A \quad A \equiv_\beta B}{\Gamma \vdash p : B} \end{array}}\ , =_\beta \right)$$

actually an open problem, requiring a quotient-inductive-inductive-recursive construction, where the definitions of $=_\beta$, $\equiv_\beta$, $=_\beta$, $\equiv_\beta$, and the definitions of contexts have yet to be given precisely

Where are we in this dream?
(intrinsic side)

There are "model" syntaxes of type theory (with explicit substitution) which approximate the dreamt intrinsic syntax:

- Dybjer's categories with families (CwF) [Dyb95, CCD19] whose initial syntax can be seen as an intrinsic syntax with explicit substitutions
- Altenkirch-Kaposi's quotient-inductive-inductive presentations of CwF (on the indexed of basic Grothendieck's construction $\{A : \mathsf{Type} \& A \to B\} \simeq (B \to \mathsf{Type})$), with work by Boulier [Bou18] and Kaposi to prove the intrinsic/extrinsic correspondence in this case
- Cartmell's categories with attributes (on the fibered side of Grothendieck's construction)
- Many other variants: contextual categories, C-systems, ...
- See also the correspondence with topos

In particular, they define $=_\beta$, $\equiv_\beta$ to be the ambient equality of the meta-language

Where are we in this dream?
(extrinsic side)

In practice, implementations of type theories rely on the extrinsic view; themselves exist in many variants:

- With judgemental equality: $A \equiv_\beta B$ is replaced by a typed version $\Gamma \vdash A \equiv_\beta B$, as in original papers by Martin-Löf, common in theoretical papers, see e.g. Agda
- With untyped equality: in the Pure Type Systems tradition (Calculus of Constructions, Barendregt's cube, ...), see e.g. Coq
- With Typed-Parallel-One-Step-Reduction (TPOSR): $A \equiv_\beta B$ is replaced by a typed parallel one-step reduction $\Gamma \vdash A \triangleright_\beta B$ [Ada06, SH12] suitable to prove the basic metatheoretic properties
- With some amount of η-rules
- With some amount of extensional decidable equality, as in the Calculus of Algebraic Constructions or Strub's CoqMT
- With fully extensional (undecidable) equality, as in Martin-Löf's Extensional Type Theory (ETT), or the proof assistants NuPrl and Ljubljana's Andromeda
- With the $\equiv_\beta$ quotient made fully explicit, as in Winterhalter's Weak Type Theory [Win20]
- Generally with enough information in the proof-term so as to decidably reconstruct a whole typing derivation when one exists (decidability of type-checking, see Miller's ProofCert project)

Possible next steps

- More on equality: Martin-Löf's identity type, strict equality, weak equality, axiom of univalence, h-levels
- More on the syntax of type theory: contexts, typing rules, universes à la Tarski, universes à la Russell
- More on natural numbers, System T and Heyting Arithmetic inside Martin-Löf's type theory
- More on Aczel's sets and set theory inside Martin-Löf's type theory

References

[Ada06] Robin Adams. Pure type systems with judgemental equality. *J. Funct. Program.*, 16(2):219–246, 2006.

[BL11] Jean-Philippe Bernardy and Marc Lasson. Realizability and parametricity in pure type systems. In *Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, volume 6604 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2011.

[Bou18] Simon Boulier. *Extending Type Theory with Syntactical Models*. Phd thesis, University of Nantes, September 2018.

[CCD19] Simon Castellan, Pierre Clairambault, and Peter Dybjer. Categories with families: Unityped, simply typed, and dependently typed. *CoRR*, abs/1904.00827, 2019.

[CH85] Thierry Coquand and Gérard Huet. Constructions : A Higher Order Proof System for Mechanizing Mathematics. In *EUROCAL'85*, volume 203 of *Lecture Notes in Computer Science*, Linz, 1985. Springer-Verlag.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5(2):56–68, 1940.

[Cur34] Haskell B. Curry. Functionality in combinatory logic. *Proceedings of the National Academy of Sciences of the United States of America*, 20:584–590, 1934.

[dB68] Nicolaas de Bruijn. Automath, a language for mathematics. Technical Report 66-WSK-05, Technological University Eindhoven, November 1968.

[Dyb95] Peter Dybjer. Internal type theory. In Stefano Berardi and Mario Coppo, editors, *Types for Proofs and Programs, International Workshop TYPES'95, Torino, Italy, June 5-8, 1995, Selected Papers*, volume 1158 of *Lecture Notes in Computer Science*, pages 120–134. Springer, 1995.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210,405–431, 1935. English Translation in [Sza69], "Investigations into logical deduction", pages 68-131.

[Gir71] Jean-Yves Girard. Une extension de l'interpretation de gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In J.E. Fenstad, editor, *Second Scandinavian Logic Symposium*, number 63 in Studies in Logic and the Foundations of Mathematics, pages 63–92. North Holland, 1971.

[Göd58] Kurt Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12(3):280–287, December 1958.

[How69] William A. Howard. The formulae-as-types notion of constructions. In *to H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1969. Published only in 1980.

[Kle45] Stephen C. Kleene. On the interpretation of intuitionistic number theory. *The Journal of Symbolic Logic*, 10(4):109–124, 1945.

[Las14] Marc Lasson. *Réalisabilité et Paramétricité dans les Systèmes de Types Purs*. Thèse de doctorat, Université de Lyon, November 2014.

[ML71] Per Martin-Löf. A theory of types. Technical Report 71-3, University of Stockholm, 1971.

[Pra65] Dag Prawitz. *Natural Deduction, a Proof-Theoretical Study*. Almqvist and Wiksell, Stockholm, 1965.

[Sco70] Dana Scott. Constructive validity. In Michel Laudet, Daniel Lacombe, Louis Nolin, and Marcel-Paul Schützenberger, editors, *Symposium on Automatic Demonstration*, pages 237–275, Berlin, Heidelberg, 1970. Springer Berlin Heidelberg.

[SH12] Vincent Siles and Hugo Herbelin. Pure type systems conversion is always typable. *Journal of Functional Programming*, 22(2):153–180, Mar 2012.

[Sza69] Manfred E. Szabo, editor. *The Collected Works of Gerhard Gentzen*. North Holland, Amsterdam, 1969.

[Win20] Théo Winterhalter. *Formalisation and Meta-Theory of Type Theory*. Phd thesis, University of Nantes, September 2020.