Foundations of Proof Systems
PRFSYS

Benjamin Werner — Dominik Kirst
LIX, Ecole polytechnique
Team Partout, Inria Saclay

Two faces of Computer Science

Today's context:

A computer is a machine able to execute *algorithms*

- It does it *precisely*
- It does it *fast*, even for *large* algorithms and *big* data

$2^{136,279,841} - 1$ is prime

What are Mathematics ?

All humans are mortal, Socrates is a human, *thus* Socrates is mortal

It is the syntax of the premisses which allows to do the deduction

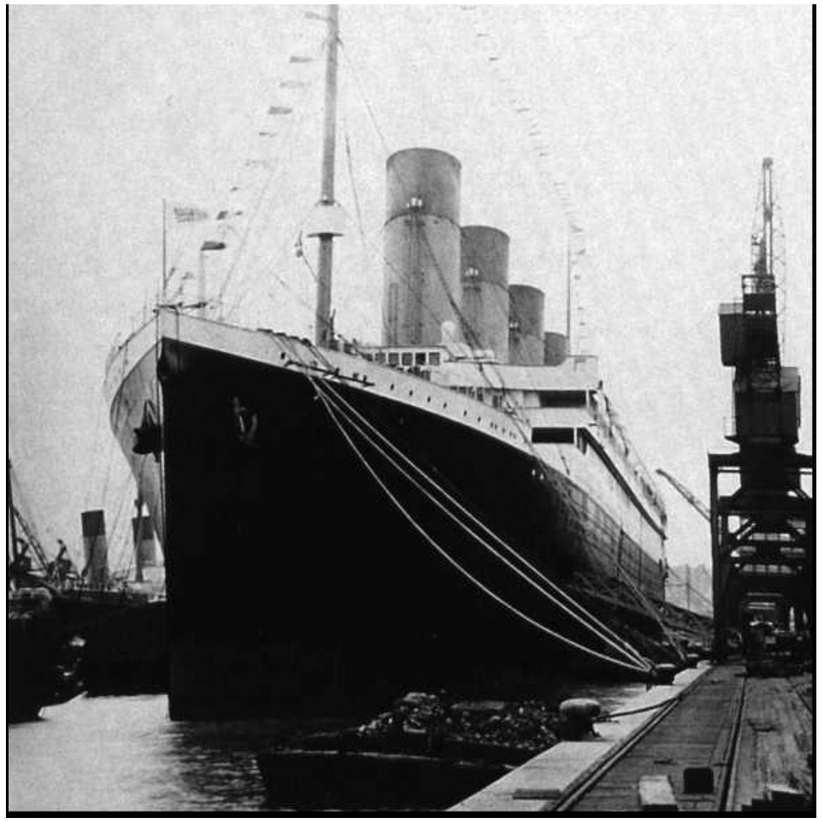
The logical point of view : a mathematical proofs can be detailed up to a point where it only consists of the application of such logical rules.

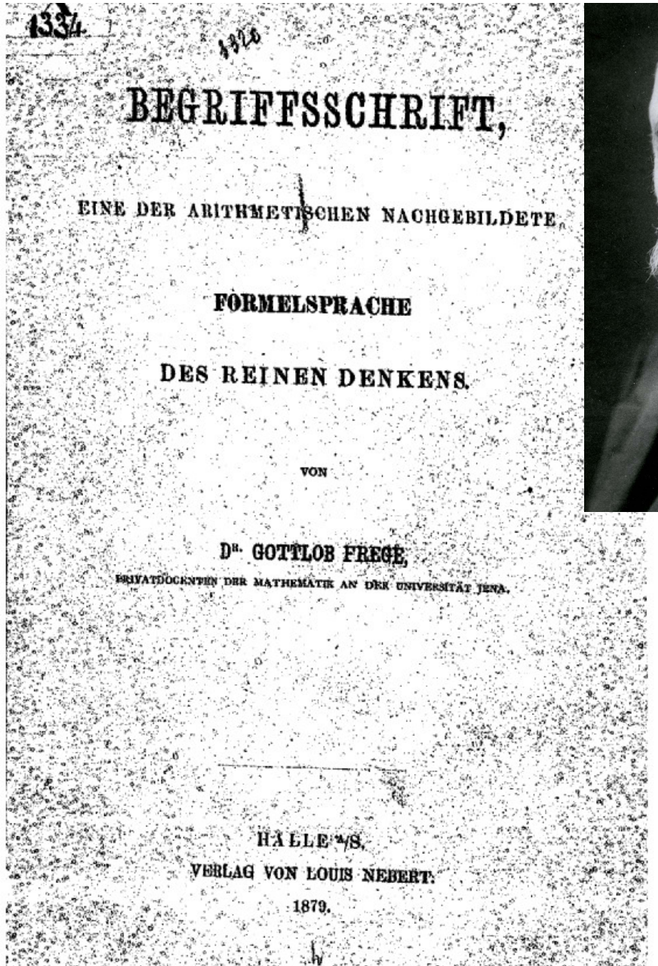
These rules are the *bricks* to construct mathematical proofs.

Verifying such a proof is a purely syntactical task.

Birth of Modern Mathematical Logic

Since second half of XIXth century: first attempts towards formal foundations for all of mathematics.
Cantor, Peano, Frege, Russell & Whitehead , Zermelo & Fraenkel, Church…




1334.

BEGRIFFSSCHRIFT,

EINE DER ARITHMETISCHEN NACHGEBILDETE

FORMELSPRACHE

DES REINEN DENKENS.

VON

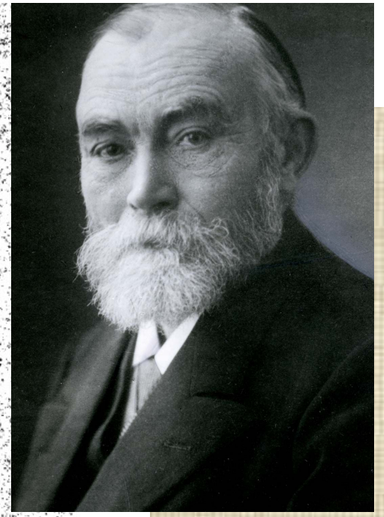
Dr. GOTTLOB FREGE,

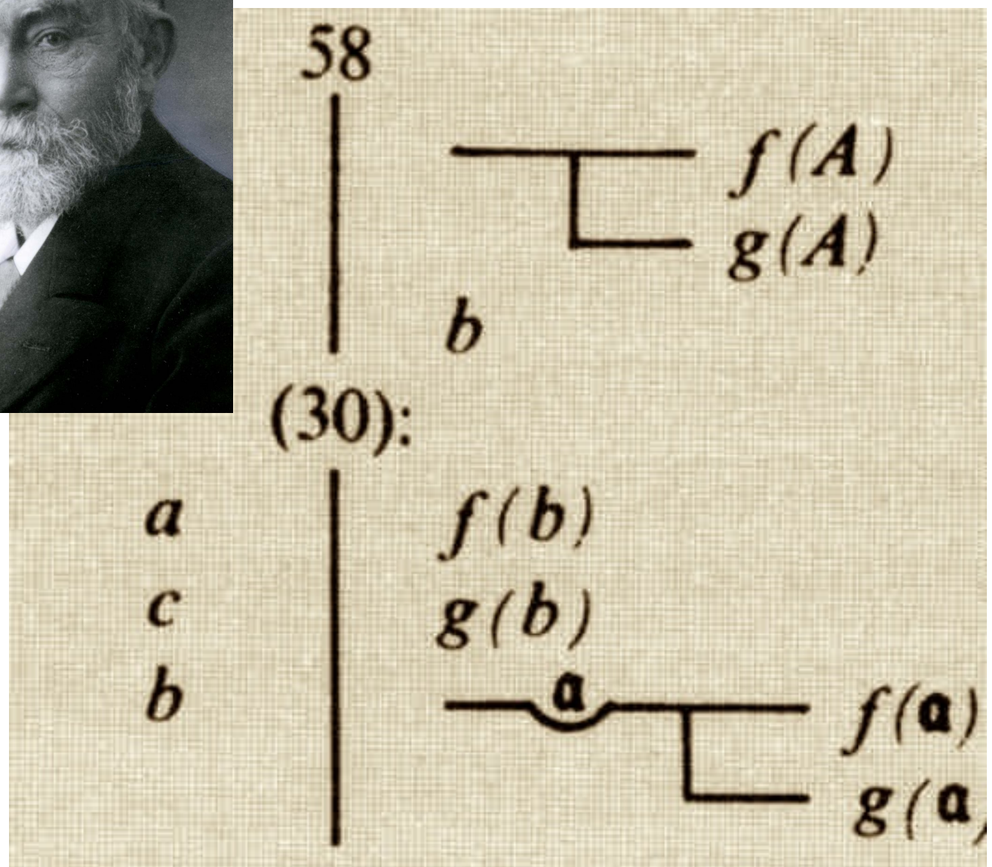
PRIVATDOCENTEN DER MATHEMATIK AN DER UNIVERSITÄT JENA.

HALLE a/S.

VERLAG VON LOUIS NEBERT.

1879.




58

$f(A)$
$g(A)$
b

(30):

a
c
b

$f(b)$
$g(b)$
a
$f(a)$
$g(a$

In the old world

- Logic provides a way to define precisely mathematical correctness
- A theorem is correct if a exists proof exists
- A mathematical text is an informal argument to convince the reader that the formal proof exists
- But one almost never writes a full formal proofs

- Mathematicians are happy to know formal foundations exist, but they rarely care about their details
- Logic is a quite exotic branch of mathematics, with little *applications*

Enters the computer

1918-2012

N.G. de Bruijn 1967

Automath

First *proof system*

Formal objects and formal proofs are actually constructed.

They exist and are verified in the computer.

Modern proof-systems : Rocq (Coq), Lean

The question of the formalism

Formal proofs are *actually* built and verified in the computer.

The choice of the formalism becomes more important
A proof system implements a formalism like a compiler implements a programming language

- The formalism needs to be not only expressive, but also convenient
- Sticking to the mathematical intuition
- Compact proofs
- Efficient checking
- Clear specification (to avoid errors in the implementation of the proof checker)

Context today: loads of new AI generated proofs boost the need for formal checking

Some Objectives of This Course

- Present and study some formalisms meant for proof systems
- Study: how to show ***coherence***, decidability of checking…
- Focus on *Type Theories*
- Proofs that make use of *computations*
- Interest in Curry-Howard (*proofs as objects,* but also *constructivity)*
- Some limits of type theories: paradoxes (links between impredicativity, constructivity…)

About this course

- Done with Dominik Kirst (Inria-Saclay, LIX)
- Lectures generally followed by exercise sessions (generally on paper)
- With respect to previous years: We want to go further - thus somewhat faster at the beginning
- Course notes largely rewritten
- Grading : final exam

Today: starting with basics

- First-Order Logic
- Simply Typed λ-calculi (including System T)
- Cut free proofs, constructivity
- Cut elimination
- A first (somewhat) computational formalism: deduction modulo

Reminders : λ-calculus

$t ::= x \mid \lambda x.t \mid (t\ t)$

$\lambda x.t$ is $x \mapsto t$ for instance $\lambda x.(x + 2)$

$(t\ u\ v\ w)$ stands for $((t\ u)\ v)\ w)$

$(\lambda x.t\ u) \triangleright_\beta t[x \setminus u]$

$(\lambda x.(x\ x)\ \ \lambda x.(x\ x)) \triangleright_\beta (x\ x)[x \setminus \lambda x.(x\ x)] = (\lambda x.(x\ x)\ \ \lambda x.(x\ x))$

no termination !

but also:

$(\lambda x.\ f\ (x\ x)\ \ \lambda x.f\ (x\ x)) \triangleright_\beta = f\ (\lambda x.f\ (x\ x)\ \ \lambda x.f\ (x\ x))$

$Y_f \triangleright_\beta (f\ Y_f)$ Fixpoint ! λ–calculus is Turing-complete

Some Historical Remarks

$Y_f \quad \triangleright_\beta \quad (f \;\; Y_f)$ Fixpoint ! λ–calculus is Turing-complete

Church Numerals

0 = λ f. λ x. x

1 = λ f. λ x. (f x)

2 = λ f. λ x. (f (f x))

3 = λ f. λ x. (f (f (f x))) …

- Pure (untyped) λ-calculus invented by Church as a base for a logical formalism
- First attempt turned out paradoxical because of non-termination
- Turing joined Church as a post-doc precisely at this time !
- Fix for the paradox: typed λ-calculus

We will see the paradox later. Today we will use simply typed calculus as a tool for something else

Simply typed λ-calculus

$$T ::= At \mid T \to T \qquad t ::= x^T \mid \lambda x^T.t \mid (t\ t)$$

$$\frac{}{\Gamma \vdash x^A : A}\ \text{Var} \qquad \frac{\vdash t : B}{\Gamma \vdash \lambda x^A.t : A \to B}\ \text{Lam}$$

$$\frac{\vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash (t\ u) : B}\ \text{App}$$

Simple remarks about the definition

$$\frac{}{\vdash x^A : A}\ \text{Var} \qquad \frac{\vdash t : B}{\vdash \lambda x^A.t : A \to B}\ \text{Lam}$$

$$\frac{\vdash t : A \to B \qquad \vdash u : A}{\vdash (t\ u) : B}\ \text{App}$$

- This is an inductive definition
- A derivation of $\vdash t : A$ has the same structure as t
- $x^A : B$ iff $A = B$
- $(t\ u) : B$ iff $\exists A . \vdash t : A \to B \wedge \vdash u : A$
- $\lambda x^A.t : C$ iff $\exists B . \vdash t : B \wedge C = A \to B$

} Inversion lemmas

Proving Normalization for simple types

Blackboard

Adding sum and product types to $\lambda\!\rightarrow$

$$T ::= \alpha \mid T \rightarrow T \mid T \times T \mid T + T$$

$$t ::= x \mid \lambda x.t \mid t\ t \mid (t, t) \mid \pi_1(t) \mid \pi_2(t) \mid i(t) \mid j(t) \mid \delta(t, x.t, x.t)$$

$$\lambda x.t\ u \quad \triangleright_\beta \quad t[x \setminus u]$$

$$\pi_1(t, u) \quad \triangleright_\beta \quad t$$

$$\pi_2(t, u) \quad \triangleright_\beta \quad u$$

$$\frac{\vdash t : A \qquad \vdash u : B}{\vdash (t,u) : A \times B}$$

$$\frac{\vdash t : A \times B}{\vdash \pi_1(t) : A} \qquad \frac{\vdash t : A \times B}{\vdash \pi_2(t) : B}$$

Adding sum and product types to $\lambda\!\rightarrow$ (2)

$$T ::= \alpha \mid T \rightarrow T \mid T \times T \mid T + T$$

$$t ::= x \mid \lambda\, x.t \mid t\ t \mid (t, t) \mid \pi_1(t) \mid \pi_2(t) \mid i(t) \mid j(t) \mid \delta(t, x.t, x.t)$$

$$\lambda\, x.t\ u \quad \triangleright_\beta \quad t[x \setminus u]$$

$$\pi_1(t, u) \quad \triangleright_\beta \quad t$$

$$\pi_2(t, u) \quad \triangleright_\beta \quad u$$

$$\frac{\vdash t : A}{\vdash i(t) : A + B} \qquad \frac{\vdash u : B}{\vdash j(u) : A + B}$$

$$\delta(i(t), x^A.u, y^B.v) \quad \triangleright_\beta \quad u[x^A \setminus t]$$

$$\delta(j(t), x^A.u, y^B.v) \quad \triangleright_\beta \quad v[y^B \setminus t]$$

$$\frac{\vdash t : A + B \quad \vdash u : C \quad \vdash v : C}{\vdash \delta(t, x^A.u, y^B.v) : C}$$

Normalization of $\lambda{\to}{\times}{+}$

$$|\, A \times B \,| \equiv \{\, t \in SN \;\mid\; t \rhd^* (u, v) \;\Rightarrow\; u \in |A| \wedge v \in |B| \,\}$$

$$|\, A + B \,| \equiv \{\, t \in SN \;\mid\; (t \rhd^* i(u) \Rightarrow u \in |A|) \wedge (t \rhd^* j(v) \Rightarrow v \in |B|) \,\}$$

$\mathcal{N}$ = terms which are not of the form: $\lambda\, x.t$, (u,v), $i(u)$ or $j(v)$

In general, the metatheory follows the same schemes: inversion lemmas, type inference, subject reduction, **strong normalization**…

Logic

Natural Deduction

First-order language: Function symbols and predicate symbols with an *arity*

Objects: $t ::= x \mid f(\underbrace{t, \ldots, t}_{\text{arity of f}})$

Propositions: $A ::= P(\underbrace{t, \ldots, t}_{\text{arity of P}}) \mid A \wedge A \mid A \vee A \mid A \Rightarrow A \mid \forall\, x.A \mid \exists\, x.A \mid \bot$

Set of *Axioms*

Proof derivations

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \rightarrow\text{-I}$$

$$\frac{\Gamma \vdash A \rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \rightarrow\text{-E}$$

modus ponens

Natural Deduction (2)

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-I}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-E (g)} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-E (d)}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee\text{-I (g)} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee\text{-I (d)}$$

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee\text{-E}$$

Natural Deduction (3)

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x.\, A}\ \forall\text{-I}$$

condition : x is not free if Γ
(« x fresh »)

$$\frac{\Gamma \vdash \forall x.\, A}{\Gamma \vdash A[x \setminus t]}\ \forall\text{-E}$$

$$\frac{\Gamma \vdash A[t/x]}{\Gamma \vdash \exists x.\, A}\ \exists\text{-I}$$

$$\frac{\Gamma \vdash \exists x.\, A \qquad \Gamma, A \vdash B}{\Gamma \vdash B}\ \exists\text{-E}$$

condition : x is not free if Γ or B

Natural Deduction (4)

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot\text{-E}$$

Intuitionistic logic

$$\frac{}{\Gamma \vdash A \vee \neg A}\ \text{EM}$$

classical logic

First-Order Theories : Arithmetic

Function Symbols: 0, S, +,×

Predicate Symbols: =

Axioms

Giuseppe Peano
1889

- $\forall x, 0 + x = x$
- $\forall x \forall y, S(x) + y = S(x + y)$
- $\forall x, 0 \neq S(x)$
- $\forall x \forall y, S(x) = S(y) \Rightarrow x = y$
- $\forall x, 0 \times x = 0$
- $\forall x \forall y, S(x) \times y = y + (x \times y)$
- $\forall x, x = x$
- $\forall x \forall y, P(x) \wedge x = y \Rightarrow P(y)$
- $P(0) \wedge (\forall y, P(y) \Rightarrow P(S(y))) \Rightarrow \forall x, P(x)$

First-Order Theories : Set Theory

Function Symbols: none

Predicate Symbols: $\in$, =

Cantor, Zermelo, Fraenkel…

Axioms

- Extensionality : $\forall x\ y, (\forall z, z \in x \Leftrightarrow z \in y) \Rightarrow x=y$
- Pair {a; b} exists
- Comprehension $\{x \in A \mid P(x)\}$ exists
- A set exists (thus $\varnothing$ exists)
- Powerset : $\mathcal{P}(x)$ exists
- Union : $\bigcup X$ exists
- infinity
- Foundation : $\in$ is well-ordered
- ZF : replacement scheme, ZFC: choice

- We will not study set theory in this course
- We will use it informally as a framework
- One important point : it is very *extensional*

Logical Cuts

What is a cut ?

1. Prove $\forall$ a. $\forall$ b.$(a + b)^2 = a^2 + b^2 + 2ab$ (ends with $\forall$-intro)
2. Deduce $\forall$ b.$(3 + b)^2 = 9 + b^2 + 6b$ (use $\forall$-elim)

We could have proved (2) directly (following the same scheme as (1)

A Cut:
an introduction rule followed by the corresponding elimination rule

Logical Cuts (2)

$$\frac{\sigma}{\Gamma\,;\,A \vdash B} \qquad \frac{\theta}{\Gamma \vdash A} \qquad \Rightarrow \qquad \frac{\sigma[A \setminus \theta]}{\Gamma \vdash B}$$

We replace the axiom rule for A in σ by a copy of θ

A cut for implication:

$$\dfrac{\dfrac{\dfrac{\sigma}{\Gamma\,;\,A \vdash B}}{\Gamma \vdash A \Rightarrow B} \qquad \dfrac{\theta}{\Gamma \vdash A}}{\Gamma \vdash B}$$

« simplifies » to :

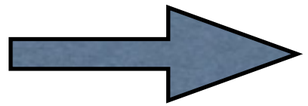
$$\Rightarrow \qquad \frac{\sigma[A \setminus \theta]}{\Gamma \vdash B}$$

Logical Cuts (3)

A cut for conjunction:

$$\dfrac{\dfrac{\dfrac{\sigma_1}{\Gamma \vdash A} \quad \dfrac{\sigma_2}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}}{\Gamma \vdash A}$$

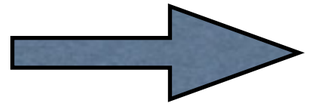
simplifies to :



$$\dfrac{\sigma_1}{\Gamma \vdash A}$$

$$\dfrac{\dfrac{\sigma}{\Gamma \vdash A}}{\dfrac{\Gamma \vdash A \vee B \quad \dfrac{\theta_1}{\Gamma; A \vdash C} \quad \dfrac{\theta_2}{\Gamma; B \vdash C}}{\Gamma \vdash C}}$$

« simplifies » to :

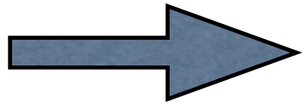


$$\dfrac{\theta_1[A \setminus \sigma]}{\Gamma \vdash C}$$

Logical Cuts (4)

$$\dfrac{\dfrac{\dfrac{\sigma}{\Gamma \vdash A}}{\Gamma \vdash \forall\, x\,.\,A}}{\Gamma \vdash A[x \setminus t]}$$

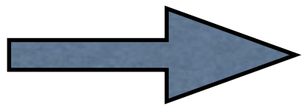
« simplifies » to :



$$\dfrac{\sigma[x \setminus t]}{\Gamma \vdash A[x \setminus t]}$$

$$\dfrac{\dfrac{\dfrac{\sigma}{\Gamma \vdash A[x \setminus t]}}{\Gamma \vdash \exists\, x\,.\,A} \qquad \dfrac{\theta}{\Gamma; A \vdash B}}{\Gamma \vdash B}$$

« simplifies » to :



$$\dfrac{\theta[A \setminus \sigma]}{\Gamma \vdash B}$$

Cut Free Proofs

1. A cut free proof of $[] \vdash A$ ends with a introduction rule
2. The cut-elimination steps described before terminate

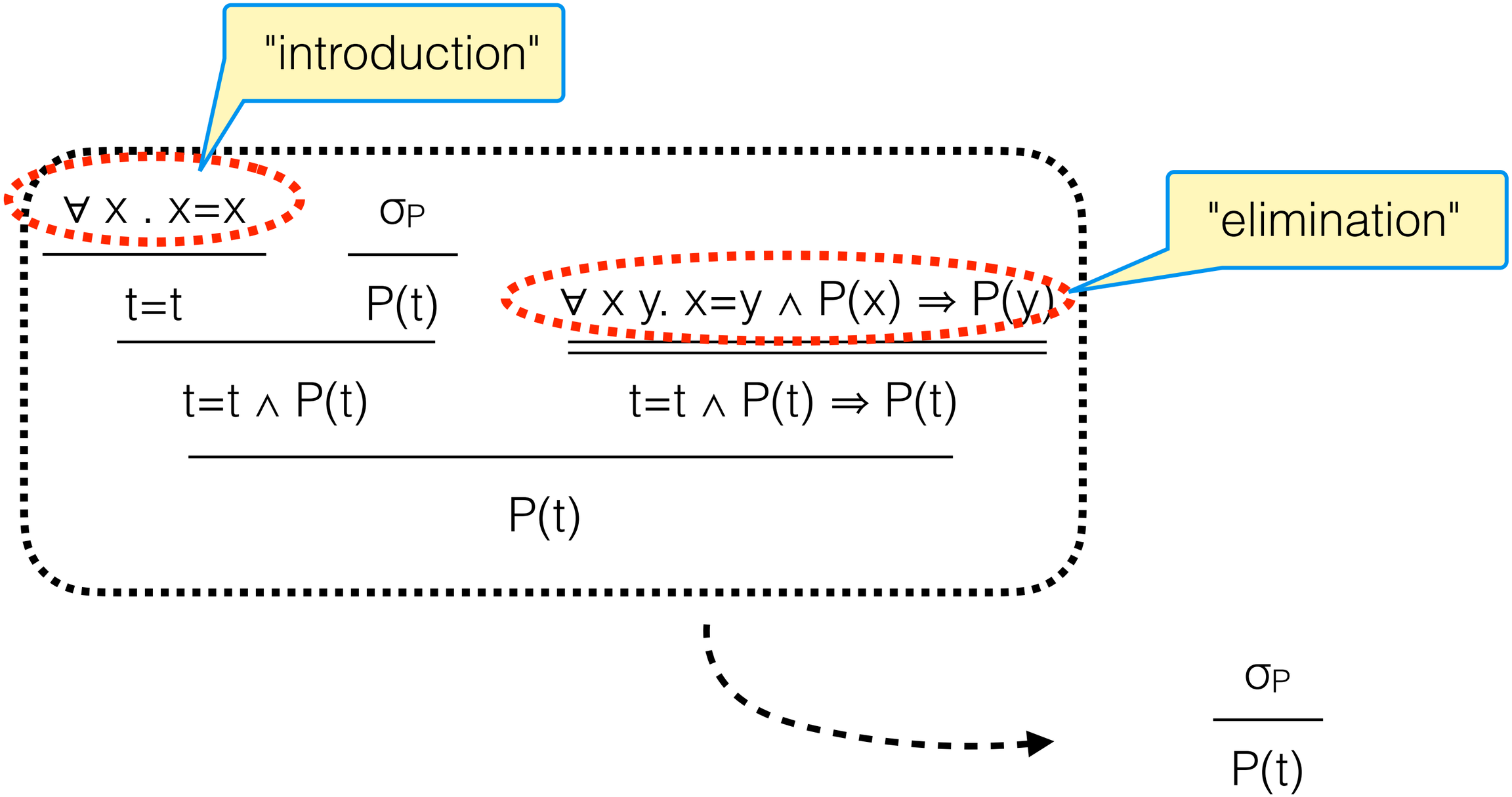
Heyting's semantics

To make the point of *constructivity*

- a proof of $n=n$ is 0 (some trivial object)
- a proof of $A \wedge B$ is (can be reduced to) (a,b) with a:A and b:B
- a canonical proof of $A \vee B$ is (ε,c) with $\varepsilon=0$ and c:A or $\varepsilon=1$ and c:B
- a proof of $A \Rightarrow B$ is a computational function f, s.t. if a:A, then $f(a)$: B
- a canonical proof of $\exists$ x.A is a pair (t,a) s.t. a: A[x \ t]
- a proof of $\forall$ x.A is a computational function f, s.t. for all n, $f(n)$: A[x \ n]

Axiomatic Cuts

Equality Cut

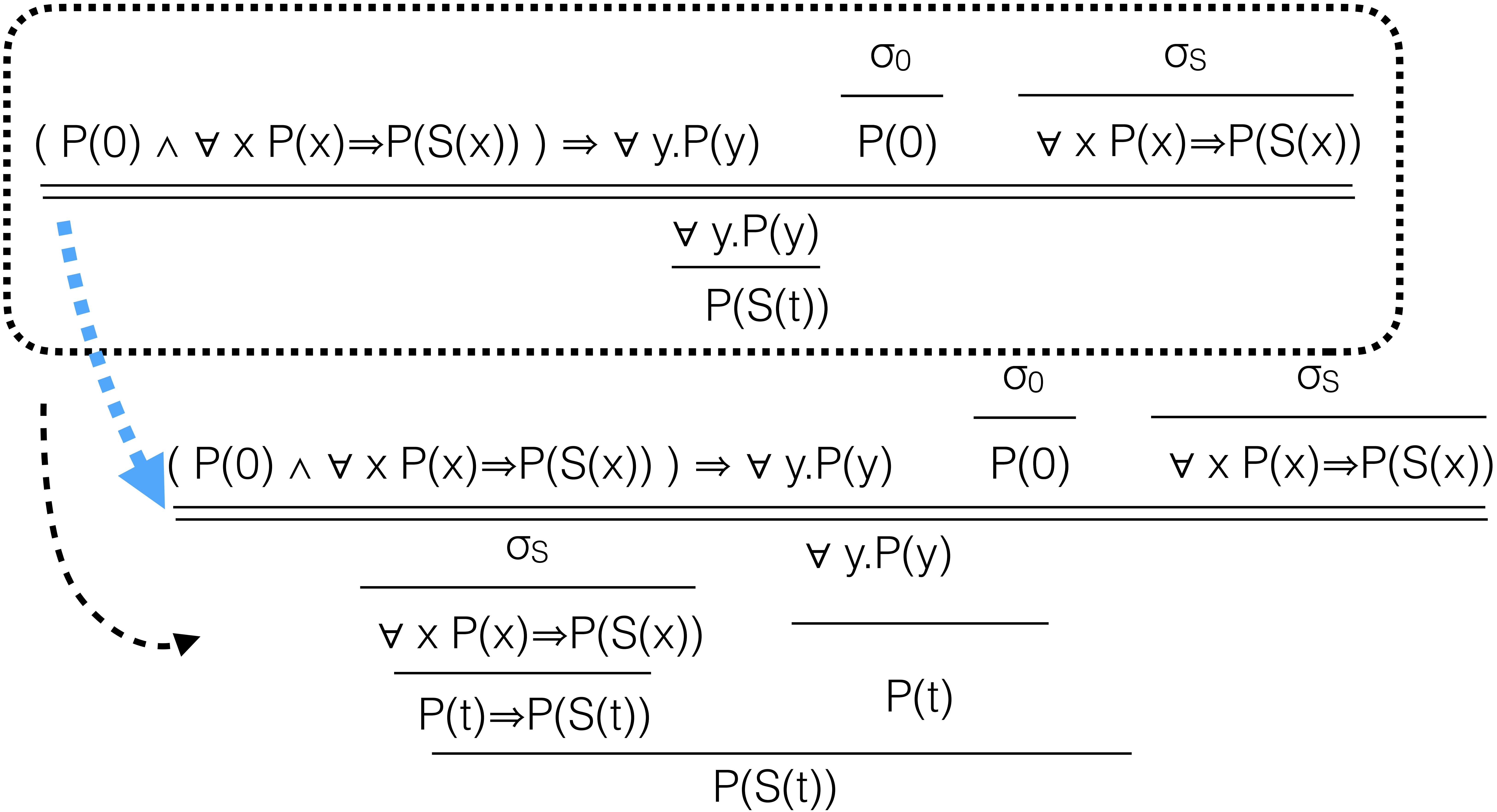


Induction Cut (1)

$$\dfrac{\dfrac{}{(\ P(0) \wedge \forall\ x\ P(x) \Rightarrow P(S(x))\ ) \Rightarrow \forall\ y.P(y)} \qquad \dfrac{\sigma_0}{P(0)} \qquad \dfrac{\sigma_S}{\forall\ x\ P(x) \Rightarrow P(S(x))}}{\dfrac{\forall\ y.P(y)}{P(0)}}$$

$$\dfrac{\sigma_0}{P(0)}$$

Induction cut (2)



Cut Free Proofs

Properties

easy:

If t is a term without free variables, then $t \rhd^* S^n(0)$

Cut free proofs:

Take A without free variables. Any cut-free proof of A in HA either :

- ends with an introduction
- is refl or t=t (from refl)
- is Leibniz or partial application of L : $\forall\, y.\ t=y \wedge P(t) \Rightarrow P(y)$, $u=t \wedge P(t) \Rightarrow P(u)$
- Is Induction or a partial application of it: $\forall\, y.\ P(y)$

by induction over the structure of the proof (somewhat tedious)

A without free variables. A cut-free proof of A in HA is either :

- ends with an introduction
- is refl or t=t (from refl)
- is Leibniz or partial application of L : $\forall$ y. t=y$\wedge$ P(t) $\Rightarrow$ P(y), u=y$\wedge$ P(t) $\Rightarrow$ P(u)
- Is Induction of proof partial application: $\forall$ y. P(y)

Constructivity :

- If $\vdash_{HA}$ A$\vee$ B, then either $\vdash_{HA}$ A or $\vdash_{HA}$ B
- if $\vdash_{HA}$ $\exists$ x. A(x) then we can extract n and a proof of $\vdash_{HA}$ A(n)

Consider : $\forall$ x. $\exists$ y. x=y+y $\vee$ x = S(y+y)