Homotopy type theory

- http://homotopytypetheory.org/book/
- https://vimeo.com/68761218
- **Dependent type theory** + homotopy theory = HoTT
- DTT by Per Martin-Löf
- *"It was really thrilling," said co-organizer Steve Awodey of Carnegie Mellon University in Pittsburgh, PA. "You felt like you were part of the Manhattan Project of computer science. Everybody had the feeling they were part of something important, and I think they were. I think we really made some amazing progress there, the consequences of which are going to take time now to play out."*
- Univalent Foundations of Mathematics : Identity is equivalent to equivalence.
- Ale zároveň vhodné pro automatické dokozazování (Coq).

Jak to souvisí s typovaným GP?

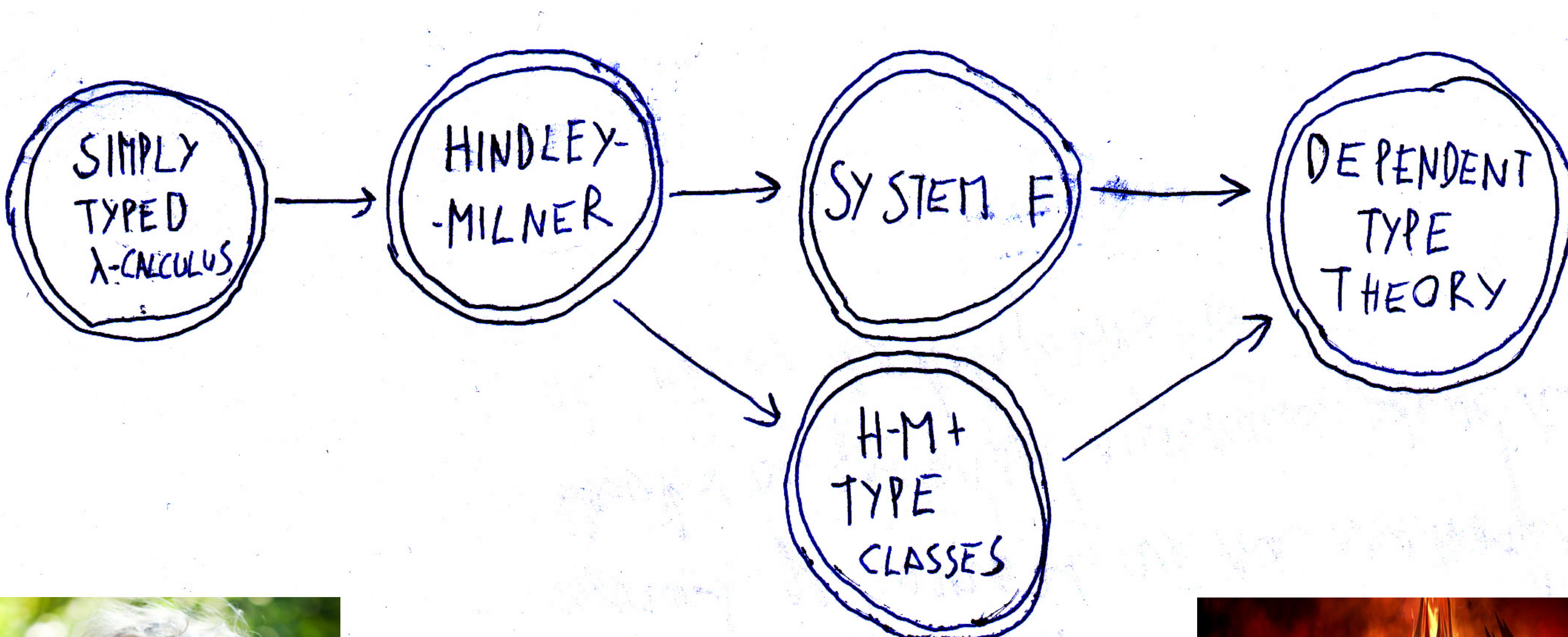
Aneb není to s kanónem na komára?

- DTT jako vršek hierarchie (moci) typových systémů
- Poskytuje jednotící optiku pro chápání konstruktů v různých typových systémech
- DTT a HoTT jsou navrženy tak, aby byly použitelné v automatických dokazovačích.
  - Induktivní vs. deduktivní systémy

Induktivní a deduktivní

Styl	*Induktivní*	*Induktivní i deduktivní*	*Deduktivní*
Výpočetní systém	**Klasické GP**	**Hybrid GP/Dokazovač**	**Automatické dokazování vět**
Obtížnost vygenerovat k typu program	Triviální.	Středně těžké. Na tom by pracoval Dokazovač.	Obtížné. Z Curry–Howardova izomorfizmu plyne ekvivalence s obtížností dokazování vět.
Co dál s vygenerovaným programem?	Vygenerované programy jsou hloupé. Veškerá kreativita se schovává ve snaze postupným kombinováním hloupých programů získávat chytřejší.	Snad středně těžké. Na tom by pracovalo GP.	Čím mocnější máme typový systém, tím spíš se nám povede, že můžeme typ formulovat tak, že nalezení libovolného programu daného typu je řešením problému.
Škála složitosti typových systémů	Netypovaný	Středně mocný typový systém	Hodně mocný typový systém

Hierarchie moci



Typ: Tvrzení nebo Množina? Oboje!

	Čeho je to kolekce?	$a : A$
MNOŽINA	Prvků	$a \in A$
TVRZENÍ	Důkazů	Věta: A Dk: a

$A \rightarrow B$

- Množinový pohled
  - $(A \rightarrow B)$ je množina funkcí z A do B
  - Funkce je mn. dvojic, že..
    - „velká tabulka“
- Logický pohled
  - $(A \rightarrow B)$ je tvrzení „A implikuje B“
  - Funkce je důkaz implikace.
    - Důkazy tvaru: „Nechť platí A, potom <důkaz B>.“

A×B

- Množinový pohled
  - (Binární) kartézský součin
    - (A×B) je množina všech dvojic (a,b), kde $a \in A$, $b \in B$
      - (Malá tabulka)
- Logický pohled
  - (A×B) je tvrzení (A&B)
  - Konkrétní (a,b) je dvojice důkazů pro tvrzení A a B.
- Programátorský pohled
  - „(binární) struct z C“

A+B

- Množinový pohled
  - (Binární) disjunktní sjednocení
- Logický pohled
  - OR
- Programátorský podled
  - „(binární) union z C“
  - | z Haskellu

$$\prod_{x:A} B(x)$$

- Zobecnění → (ale také ×)

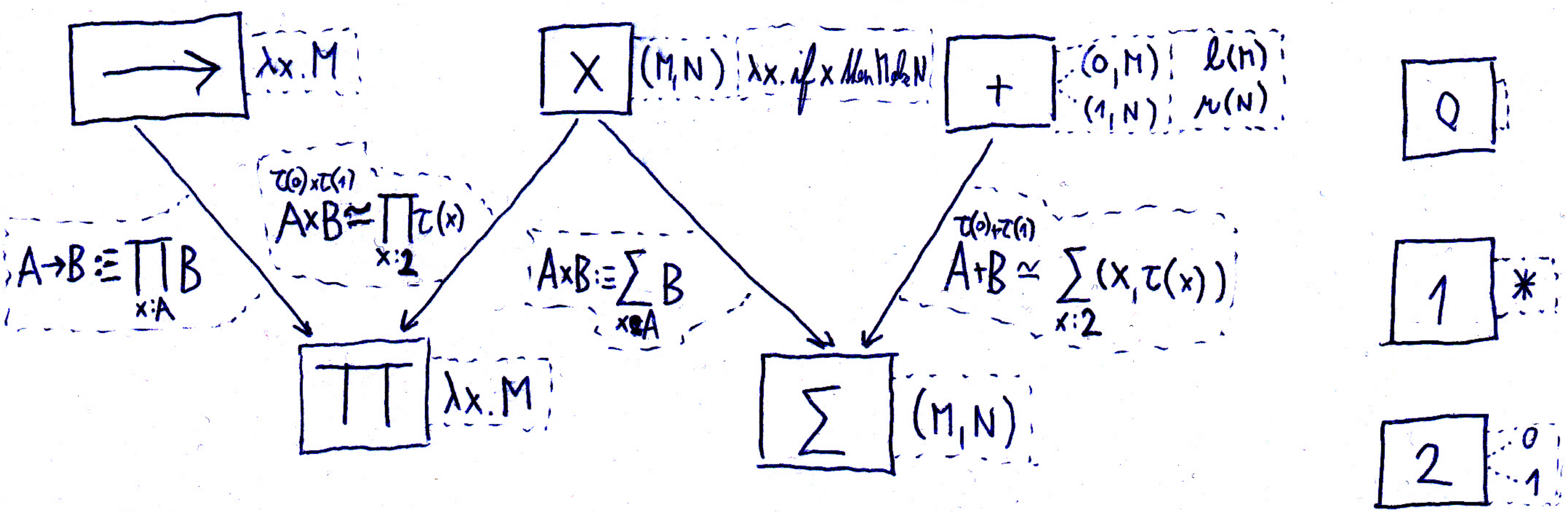
- Množinový pohled
  - (Velký) kartezský součin přes indexovou množinu
  - Prvky jsou „A-tice“ (velká tabulka)
- Logický pohled
  - ∀

$$\sum_{x:A} B(x)$$

- Zobecnění × (ale také +)

- Množinový pohled
  - (Velké) disjunktní sjednocení přes indexovou množinu
  - Prvky jsou dvojice (index, prvek)
- Logický pohled
  - $\exists$

Vztahy

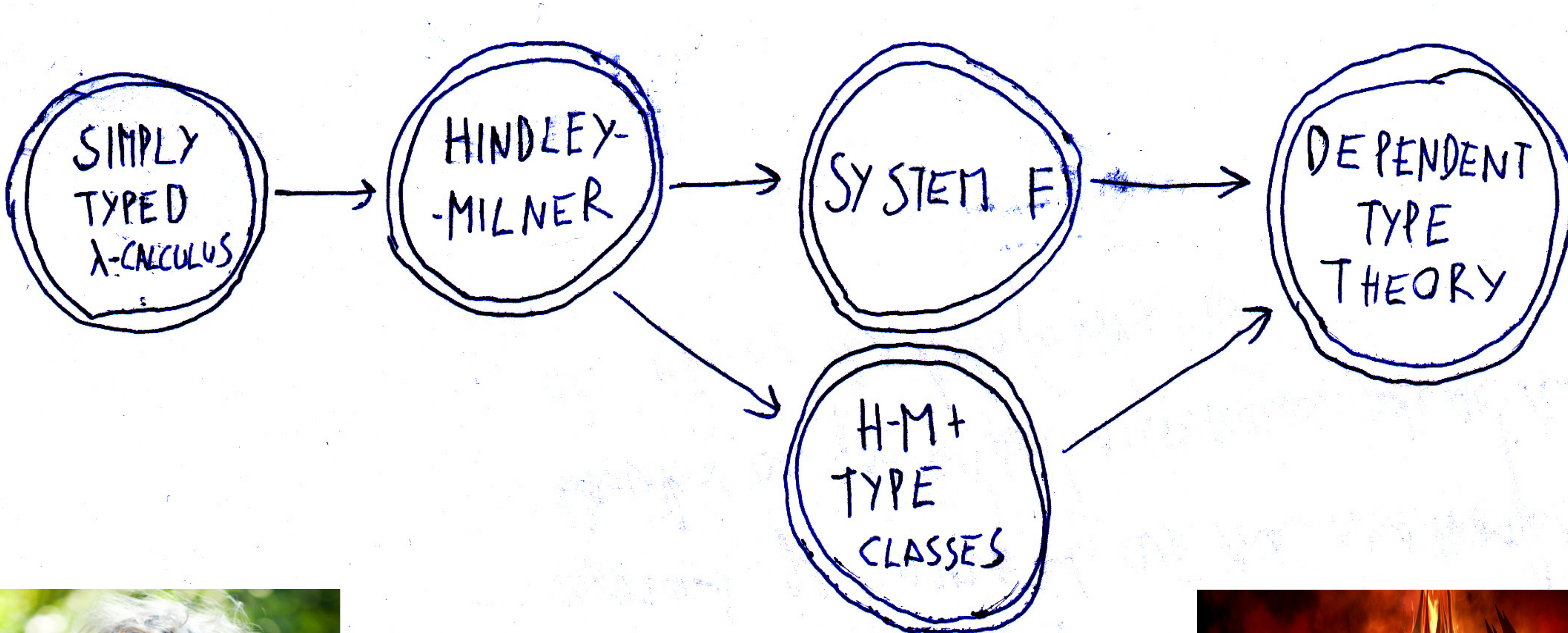


λx.M
(M,N)
λx. if x then M else N
(0,M)
(1,N)
l(M)
r(N)
τ(0)×τ(1)
A×B ≃ ∏ τ(x), x:2
A→B :≡ ∏ B, x:A
A×B :≡ ∑ B, x:A
τ(0)+τ(1)
A+B ≃ ∑ (x, τ(x)), x:2
λx.M
(M,N)
0
1
*
2
0
1

=

- Podle zásady „typy odpovídají tvrzením“, by měl existovat typ pro tvrzení „x=y“ kde x,y:A.
- Na důkaz rovnosti se můžeme koukat jako na (1D) cestu
- Na důkaz rovnosti dvou důkazů rovností se můžeme koukat jako na homotopii (2D cestu)
- Tyto triky inspirované homotopy teorií (odvětví topologie) jsou nástrojem pro vyslovení univalence axiomu pro efektivní práci s izomorfními strukturami.

Hierarchie moci



Simply Typed Lambda Calculus

- Jen atomické typy a →
  - Atomické typy nejsou dále strukturovány
    - (List Int) nedělitelný symbol

Jednoduchý generující algoritmus

Hindley–Milner type system

- Navíc typové proměné (a zjednodušený ∀) a parametrické typy (jakoby funkční symboly)
  - Díky parametrickým typům chápeme typy „strukturovaněji“ než v ST lambda kalkulu.
    - (List Int) už chápeme jako dva symboly
  - Díky typovým proměným máme polymorfizmus
- Generující algoritmus rozšíříme o práci s unifikacemi

HM + Type Classes

- „K Hindley–Milnerovi se přidají predikáty.“
- Typová třída je konstrukt používaný v Haskellu
  - „Mocné interfacy z Javy“
- Třída, instance a funkce s „predikátovým předznamenáním“
  - Např:
    - Eq, Functor, Category, Arrow
    - Num, IsA
- Při generování navíc běh logického programu

System F

- HM lze chápat jako zjednodušení Systému F, aby v něm šlo rychle a hezky otypovat neanotovaný lambda term.
  - V termech se už musí objevovat typy
- Kvantifikátory kdekoliv
  - Díky tomu mohu být typy ještě strukturovanější než v HM, např List nyní chápeme jako zkratku za
    - $\forall a,b : (a \to b \to b) \to b \to b$