TYPE THEORY *

Zoltán Csörnyei
Eötvös University
Department of General Computer Science
Budapest, Hungary
csz@inf.elte.hu

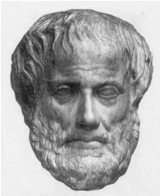
* Supported by OTKA T037742

1

Preliminaries I.

LOGIC

- Aristotle (384 BC – 322 BC)



2

Aristotle

Aristotle proposed the now famous Aristotelian syllogistic, form of argument consisting of two premises and a conclusion. His example is:

(i) *Every Greek is a person.*

(ii) *Every person is mortal.*

(iii) *Every Greek is mortal.*

3

- William of Ockham (1288 – 1348)

4

W. of Ockham

Ockham's Razor:

- *Frustra fit per plura, quod fieri potest per pauciora.*
  It is vain to do with more what can be done with less.
  or
  Essentia non sunt multiplicanda praeter necessitatem.
  Entities should not be multiplied unnecessarily.

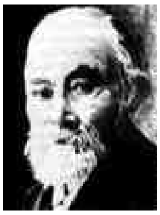
5

W. of Ockham

- He considered a *three valued logic* where propositions can take one of three truth values. This became important for mathematics in the 20th Century but it is remarkable that it was first studied by Ockham 600 years earlier.

6

- Gottlob Frege (1848 – 1925)



7

G. Frege

- one of the founders of modern symbolic logic

He was the first to fully develop the main thesis of logicism, that mathematics is reducible to logic.

(The Russell paradox gave contradiction in Frege's system of axioms.)

8

- David Hilbert (1862 – 1943)

9

D. Hilbert

- Axioms

1. |- $A \to A$
2. |- $A \to (B \to A)$
3. |- $(A \to (B\to C)) \to ((A \to B) \to A \to C))$

4. $\dfrac{\text{|- } A \to B \quad \text{|- } A}{\text{|- } B}$

10

- Gerhard Gentzen (1909 – 1945)

11

G. Gentzen

- He introduced the notion of 'logical consequence'

  $B_1, B_2, \ldots, B_n$ |- A
- Natural deduction
- Sequent calculus
- Derivation tree

12

- Luitzen E. Jan Brouwer (1881 – 1966)

13

L.E.J. Brouwer

- The foundations of intuitionism.
- Judgements about statements are based on the existence of a *proof* or *construction* of that statements.
- There are two irrational numbers *x* and *y*, such that x^y is rational.

14

Preliminaries II.

- The untyped combinatory logics
  Schönfinkel, 1924
- The untyped lambda calculus
  Church, 1934
- Turing machines
  Turing, 1936

15

- Moses Schönfinkel (1889 – 1942(?))

16

Combinatory logic I.

- *expr* ::= *var*
  | K | S
  | (*expr expr*)
- *reductions*
  K *x y* → *x*
  S *x y z* → *x z* (*y z*)
- *normal form*

17

Combinatory logic II.

- *True* ≡ K
  false ≡ K I   I ≡ S K K
- *if E F G* ≡ *E F G*
  and E F ≡ *E F false*
  or E F ≡ *E true F*
  not E ≡ *E false true*

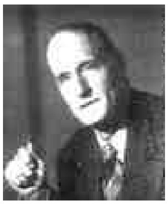
18

Combinatory logic III.

- *natural numbers*
  $n \equiv (\, S\, B\,)^n (\, K\, I\,)$ $\quad B\, E\, F\, G \equiv E\, (F\, G)$
- $succ \equiv S\, B$ $\quad C\, E\, F\, G \equiv E\, G\, F$
  $zero \equiv C\, (CI\, (\, true\ false))true$
- $add\ E\, F \equiv S'\, B'\, E\, F$ $\quad S'CEFG \equiv C(EG)(FG)$
  $mul\ E\, F \equiv B'\, E\, F$ $\quad B'CEFG \equiv CE(FG)$
  $exp\ E\, F \equiv F\, E$

19

- Stephen C. Kleene (1904 – 1994)



20

Combinatory logic IV.

- *Definable functions:*
  partial recursive numerical function
  (Kleene, 1936.)

21

- Alonzo Church (1903 – 1995)

22

Lambda calculus I.

- *expr ::= var*
  | *λ var . expr*
  | *(expr expr)*
- *reductions*
  $(\lambda x\, .\, E_1)\, E_2 \rightarrow_\beta E_1 [\, x := E_2\,]$
  $\lambda x\, .\, E \rightarrow_\alpha \lambda y\, .\, E\, [\, x := y\,]$
  $\lambda x\, .\, E\, x \rightarrow_\eta E$

23

Lambda calculus II.

- *true* ≡ *λ xy.x*
  false ≡ *λ xy.y*
- *if* ≡ *λ xyz.xyz*
  and ≡ *λ xy.xy false*
  or ≡ *λ xy.x true y*
  not ≡ *λ x.x false true*

24

Lambda calculus III.

- *natural numbers*
  $n \equiv \lambda fx \,.\, f^{n}(x)$
- $succ \equiv \lambda\, nfx \,.\, f\,(\, nfx\,)$
  $zero \equiv \lambda\, x \,.\, x\,(\, true\ false\,)\ true$

- $add \equiv \lambda\, xypq \,.\, xp\,(\, ypq\,)$
  $mul \equiv \lambda\, xyp \,.\, x\,(\, yp\,)$
  $exp \equiv \lambda\, xy \,.\, yx$

25

Lambda calculus IV.

- *Definable functions:*
  partial recursive numerical function

(*undefined* ≡ *unsolvable*)
solvable: $\exists F,\ (\lambda x.E)\, F = \mathrm{I}$
(Kleene, 1936., Wadsworth, 1971.)

26

- Alan M. Turing (1912 – 1954)

27

Turing machine

- Turing Theorem (1936.)

combinatory logics ≡
lambda calculus ≡
Turing machine

28

Formal Type System

- Formal Type System: (S, J, R)
  S *syntax*, J *judgements*, R *rules*
- Type environment Γ
- Rule

$$\frac{\Gamma \vdash I_1 \ \dots\ \Gamma \vdash I_n}{\Gamma \vdash I}$$

29

Type System F_1

- *type ::= basic_type*
  | *type* → *type*
- *expr ::= var*
  | λ *var : type . expr*
  | *(expr expr)*

„type checking”
(Pascal, C++, …)

30

Rules (F_1)

- $\Gamma, x{:}A, \Gamma' \text{ |- } wf$
  ------------------------- [Val x]
  $\Gamma, x{:}A, \Gamma' \text{ |- } x{:}A$
- $\Gamma, x{:}A \text{ |- } E : B$
  ----------------------------- [Val Fun]
  $\Gamma \text{ |- } \lambda{:}A . E : A \to B$
- $\Gamma \text{ |- } E : A \to B \quad \Gamma \text{ |- } F : A$
  ------------------------------------- [Val Appl]
  $\Gamma \text{ |- } E\,F : B$

Theorems (F_1)

- Type preservation
  If E : A and E ->> F then F : A .
- Type unumbigouos
  If Γ |- E : A and Γ |- E : B then $A \equiv B$.
- Finite reductions
  there is no infinite reduction sequence.

The power (F_1)

- (Schwichtenberg, 1976.)
  The lambda definable functions are exactly the extended polinomials.
- *constant functions, projections, signum function, addition, multiplication.*

(Type System F_1 / Curry)

- *type ::= type_var*
  | *type → type*
- *expr ::= var*
  | *λ var . expr*
  | *(expr expr)*

„type inference”
(ML, Haskell, …)

Type System F_2

- *Id-Nat ≡ λ x : Nat . x*
  Id-Bool ≡ λ x : Bool . x
  Id-Int ≡ λ x : Int. x
- *Id-α ≡ λ x : α. x*
  Id ≡ Λ α. λ x : α. x polimorphic function
- *Id : ∀ α. α → α* type of polimorphic function

Type System F_2 II.

- *type ::= type_var*
  | *type → type*
  | *∀ type_var . type*
- *expr ::= var*
  | *λ var : type . expr*
  | *(expr expr)*
  | *λ type_var . expr*
  | (*expr*) [*type*]

Type System F_2 III.

- *Id* [*Nat*] ≡ (Λ*α* . λ *x* : *α* . *x*)[*Nat*] $\rightarrow_\beta$
  λ *x* : *Nat* . *x* : *Nat* → *Nat*

- *Type of Id* [*Nat*] ?
  Λ*ω* . *ω* → *ω*
- *ω* type of type-expression ≡ *kind*

37

Type System F_3

- * ≡ type of types of expressions
- *kind* ::= *
  | (* → *kind*)
- *type* ::= *type_var*
  | *type* → *type*
  | ∀ *type_var* : *kind* . *type*
  | Λ *type_var* : *kind* . *type*
  | (*type type*)

38

Type System F_4, F_5,...

- F_4 *kind* ::= *
  | (* → *kind*)
  | (*kind* → *)
- F_5 *kind* ::= *
  | (* → *kind*)
  | (*kind* → *)
  | ((* → *)→ *kind*)

39

Type System F^ω

- $F^\omega = \cup F_i$
- *kind* ::= *
  | (*kind* → *kind*)

40

Extension of Type System, $F_\leq^\omega$ I.

- $F_\leq^\omega$ (F-omega-sub)
  subtyping
- *Top*
- *type* ≤ *Top*
- Λ *type_var* ≤ *type* . e*xpr*

41

Extension of Type System, $F_\leq^\omega$ II.

- *Subtyping*

$$\frac{\Gamma \vdash E : A \quad \Gamma \vdash A \leq B}{\Gamma \vdash E : B} \text{ [Subsumption]}$$

$$\frac{\Gamma \vdash A \leq B \quad \Gamma \vdash C \leq D}{\Gamma \vdash B \rightarrow C \leq A \rightarrow D} \text{ [Sub Arrow]}$$

42

Extension of Type System, $F_{\leq}^{\omega}$ III.

- *Existential type*
- $\exists\alpha.A$
  pack, unpack (open)
- *Recursive type*
- $\mu\alpha.A$
  fold, unfold

43

The power of $F_{\leq}^{\omega}$

- class of functions definiable in $F_{\leq}^{\omega}$ is much larger than the *primitive recursive* functions.
- Ackermann's function is definiable.

44

Isomorphism (Curry, Feys 1956.)

- Isomorphism between

combinators	Hilbert's axioms
1. $\lambda x.x$	$\vdash A \to A$
2. $\lambda xy.x$	$\vdash A \to (B \to A)$
3. $\lambda xyz.xz(yz)$	$\vdash (A\to(B\to C))\to((A\to B)\to(A\to C))$
4. function application	$\vdash A \to B \quad \vdash A$ / -------------------- / $\vdash B$

45

The Curry-Howard isomorphism (Howard, 1959.)

- isomorphism between

lambda calculus	intuitive prop.logic
term variable	assumption
term	proof
type	formula
type constructor	connective
reduction	normalization
…	…

46