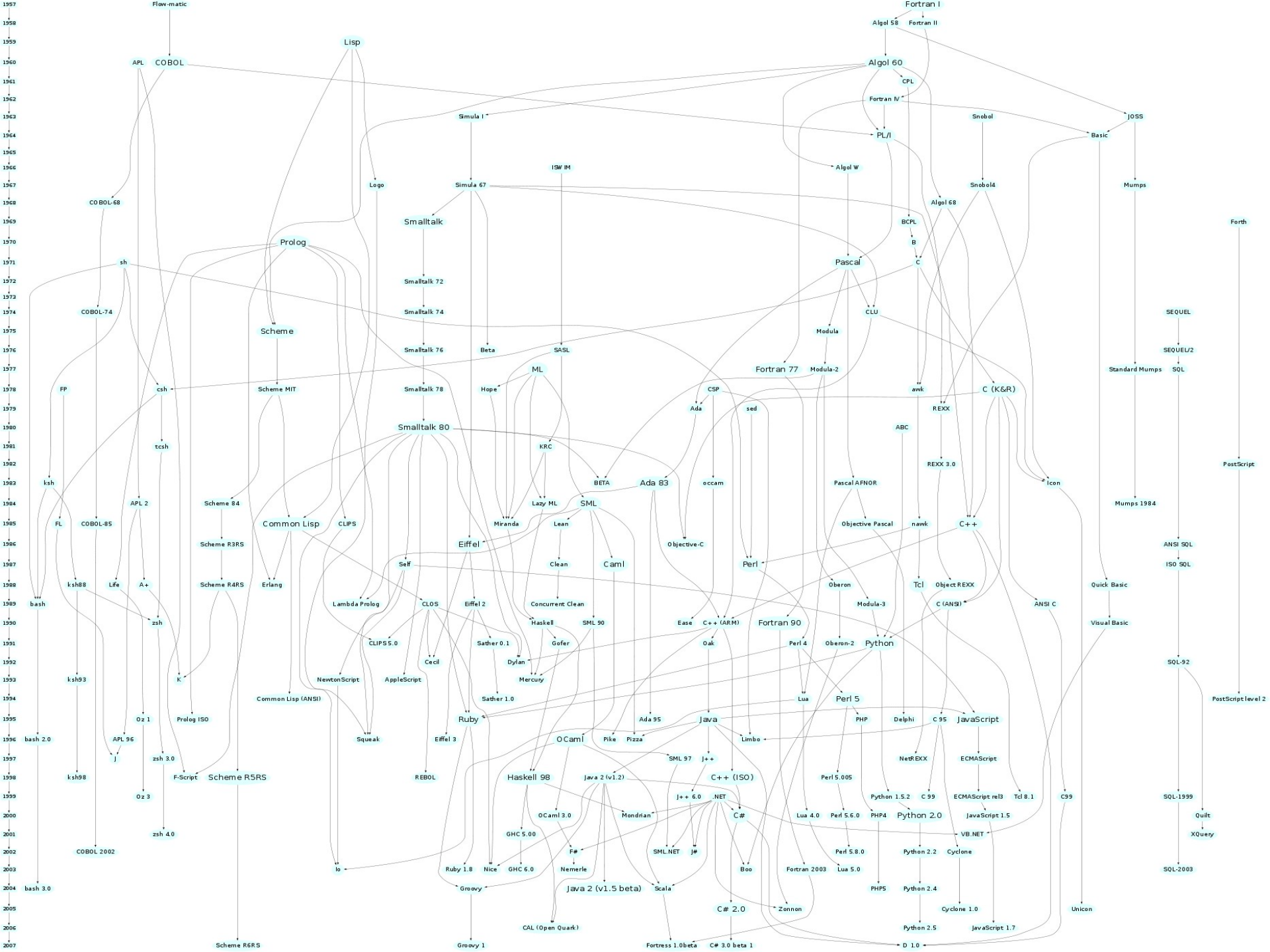
Teorija programskih jezikov

I.Savnik, FAMNIT, 2012/13

“Computer Science is no more about computers than astronomy is about telescopes.”

- E.W.Dijkstra


1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
Flow-matic
Fortran I
Algol 58
Fortran II
Lisp
APL
COBOL
Algol 60
CPL
Fortran IV
Simula I
Snobol
JOSS
PL/I
Basic
Algol W
ISWIM
Logo
Simula 67
Snobol4
Mumps
Algol 68
COBOL-68
Smalltalk
BCPL
B
C
Forth
Prolog
Pascal
sh
Smalltalk 72
COBOL-74
Smalltalk 74
CLU
SEQUEL
Scheme
Modula
Smalltalk 76
Beta
SASL
SEQUEL/2
ML
Fortran 77
Modula-2
Standard Mumps
SQL
FP
csh
Scheme MIT
Hope
CSP
awk
C (K&R)
Smalltalk 78
Ada
sed
REXX
Smalltalk 80
ABC
tcsh
KRC
REXX 3.0
PostScript
ksh
BETA
Ada 83
occam
Pascal APNOR
Icon
APL 2
Scheme 84
Lazy ML
SML
Mumps 1984
FL
COBOL-85
Common Lisp
CLIPS
Miranda
Lean
Objective Pascal
nawk
C++
Eiffel
Objective-C
ANSI SQL
Self
Clean
Caml
Perl
ISO SQL
bash
Life
A+
ksh88
Scheme R3RS
Erlang
Oberon
Tcl
Object REXX
Quick Basic
Lambda Prolog
CLOS
Eiffel 2
Concurrent Clean
Modula-3
C (ANSI)
ANSI C
zsh
Haskell
SML 90
Ease
C++ (ARM)
Fortran 90
Visual Basic
CLIPS 5.0
Gofer
Oak
Perl 4
Oberon-2
Python
Scheme R4RS
ksh93
K
Cecil
Sather 0.1
Dylan
SQL-92
NewtonScript
AppleScript
Mercury
Common Lisp (ANSI)
Sather 1.0
Lua
Perl 5
PostScript level 2
Oz 1
Prolog ISO
Ruby
Ada 95
Java
PHP
Delphi
C 95
JavaScript
bash 2.0
APL 96
Squeak
Eiffel 3
OCaml
Pike
Pizza
Limbo
J
SML 97
J++
NetREXX
ECMAScript
ksh98
zsh 3.0
F-Script
Scheme R5RS
REBOL
Haskell 98
Java 2 (v1.2)
C++ (ISO)
Perl 5.005
Oz 3
J++ 6.0
.NET
Python 1.5.2
C 99
ECMAScript rel3
Tcl 8.1
SQL-1999
Quilt
OCaml 3.0
Mondrian
C#
Lua 4.0
Perl 5.6.0
PHP4
Python 2.0
JavaScript 1.5
zsh 4.0
GHC 5.00
VB.NET
XQuery
COBOL 2002
F#
SML.NET
J#
Perl 5.8.0
Python 2.2
Cyclone
lo
Ruby 1.8
Nice
GHC 6.0
Nemerle
Boo
Fortran 2003
Lua 5.0
SQL-2003
bash 3.0
Groovy
Java 2 (v1.5 beta)
Scala
PHP5
Python 2.4
C# 2.0
Zonnon
Cyclone 1.0
Unicon
CAL (Open Quark)
Python 2.5
JavaScript 1.7
Scheme R6RS
Groovy 1
Fortress 1.0beta
C# 3.0 beta 1
D 1.0

Potek

- Osnovni podatki
- Zgodovina
- Semantika jezikov
- Uporaba tipov
- Praktična uporaba
- Mejniki v razvoju prog.jezikov

Osnovni podatki

Naslov:	Teorija programskih jezikov
Predavatelj:	doc.dr. Iztok Savnik
Vaje:	domače naloge
Točke:	9 KT
Komunikacija:	e-učilnica, forumi, e-posta, govorilne ure
Govorilne ure:	po predavanju
E-učilnica:	e.famnit.upr.si

Ocenjevanje

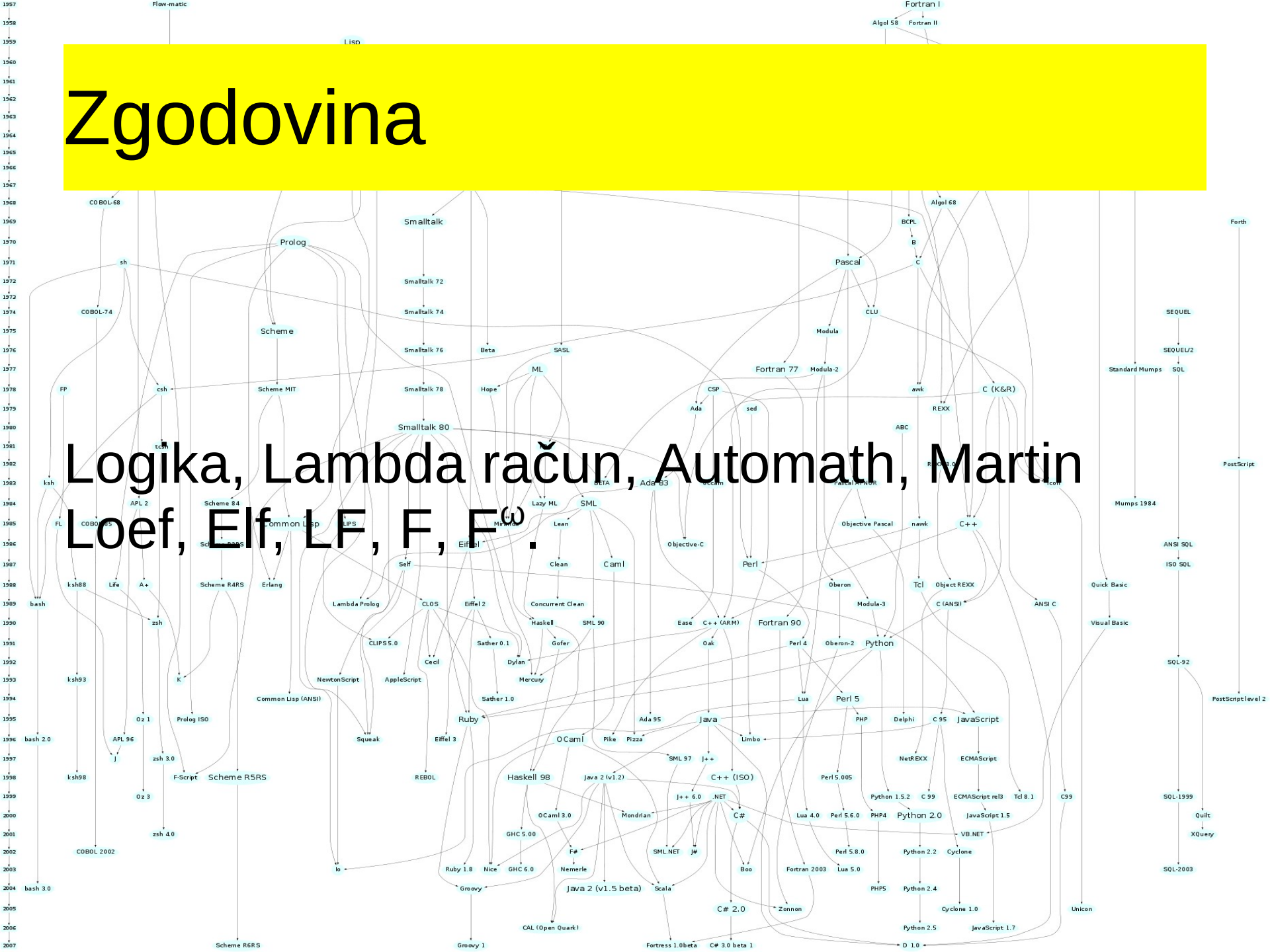
- Sestava ocene = 6KT
  - Ocena pisnega izpita – 50%
  - Ocena ustnega izpita – 50%
- Domače naloge = 3KT
  - 3-4 domače naloge

- Vsaka od delnih ocen mora biti pozitivna!

Vsebina

1) Pregled konceptov programskih jezikov
2) Sklepanje
3) Sintaksa
4) Operacijska semantika
5) Denotacijska semantika
6) Λ-račun
7) Tipi
8) Rekurzija
9) Strukture
10) Rekurzivni tipi
11) Λ-račun 2. reda
12) LF in Twelf
13) Jezik Min-ML

Zgodovina



Semantika programskih jezikov

Teoretične študije lastnosti programskih jezikov.

Teorija tipov formalno obravnava tipe.

Nadaljevanje veje logike (Russel,Frege,Church).

Teorija domen preučuje matematične lastnosti izvajanja programov.

Logika

- Gottfried Wilhelm Leibniz, 1646
- Calculus ratioconator
  - Osnova simbolične logike
  - Osnova za Lambda račun in Turingov stroj

Leibniz je imel naslednji ideal:

1) Kreiraj univerzalni jezik, ki lahko izrazi vse probleme.

2) Poišči metodo, ki reši vse probleme opisane v univerzalnem jeziku.

Oblika teorije množic izražene v predikatnem računu (1).

[Entscheidungsproblem] Ali lahko rešimo vse probleme izražene v univerzalnem jeziku (2)

Entscheidungsproblem

- David Hilbert, 1928
- Problem: Je mogoče iz opisa formalnega jezika in stavka v tem jeziku napisati algoritem, ki bo odgovoril true v primeru, da je stavek pravilen in false sicer

Logika

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.

- Gottlob Frege, 1848
  - Aksiomatska predikatna logika
  - Kvantificirane spremenljivke
- Logika je formalna osnova programskih jezikov in raznovrstne modele !

Lambda račun

Zgodovina:

Izviri λ-računa so v delu Leibniza v 17. stoletju (Calculus Ratiocinator)

λ-račun je bil uporabljen v slavnem članku l.1936 o obstoju neodločljivega problema

Turing je v tistem času naredil svoj stroj.

λ: Alonzo Church, Stephen Cole Kleene, 1930

Študija osnov matematike

Idealiziran in minimalističen prog. jezik

Izrazno enakovreden Turingovem stroju

Prog.jeziki: Erlang, Haskell, Lisp, ML, Scheme.

Teorija tipov

- Teorija tipov je področje, ki se ukvarja z načrtovanjem, analizo in študijem tipov.

  Matematika: Teorija tipov nadomesti naivno teorijo množic.

- Bertram Russel, 1901

  Teorija tipov je nastala kot odgovor na paradoks Fregejeve logike

  Russel predlaga svojo teorijo tipov

- Churcheva teorija tipov

  Lambda račun s tipi

Teorija tipov

Definicija formalnega sistema,

ki obravnava tipe.

> Objekti, ki jih preučujemo z uporabo matematičnega aparata so raznovrste strukture: množice, n-terice, zapisi, ...

Opazujemo lastnosti tipov, urejenosti, ...

Primerjamo z drugimi formalizmi, še posebaj z logiko.

Automath - de Bruijn

- Formalen jezik, Nicolaas Govert de Bruijn, začetek, 1967
- Opisovanje matematičnih teorij
- Vsebuje orodja za avtomatsko dokazovanje izrekov
- Veliko idej iz Automath so kasneje prenesli na lambda račun.

Martin-Loef:teorija tipov

Martin-Löf, 1972.

Delal je na filozofskih osnovah logike.

re-konstrukciji logike

Osnovana na analogiji med izjavami (logika) in tipi (Curry-Howard isomorfizm).

Izjavni račun ⇔ lambda račun s tipi

Predikatni račun ⇔ Odvisni tipi

Π-tipi, Σ-tipi, Induktivni tipi

Formalizacija teorije tipov

System F, F^{ω}

- Polimorfični lambda račun
  - Jean-Yves Girard
  - John C. Reynolds
  - λ-račun 2. reda
- Formalizacija polimorfizma v prog. Jezikih.
- Rekurzivne konstrukcije tipov podobno kot teorija M.P.Loef.
- Preverjanje tipov ni odločljivo.
- Induktivno definirana hierarhija.

Logika za izračunljive funkcije

- LCF (R.Milner)
  - Logic for computable functions (LCF)
  - Stanford 1970-71, Edinburgh 1972-1995
- Meta-jezik za LCF sistem
  - Projekt za automatizacijo logike
  - Dokazovanje
  - Sistem tipov
  - Notacija za programe
  - Funkcije višjega reda
  - Programi, ki poiščejo dokaz

Edinburgh Logical Framework

Elf, LF: Orodje za predstavitev logike.

Lambda račun z odvisnimi tipi = λΠ-račun

Odvisni tipi prvega reda – izjave so tipi.

Za opis logičnega okolja definiramo:

opis razredov objektov; primeren metajezik; definicijo mehanizmov s katerimi je logika predstavljena.

Preverjanje tipov je odločljivo.

Sklepanje s tipi ni odločljivo.

LF implementacija v Twelf (CMU)

Semantika jezikov

Denotacijska semantika programskih jezikov

Operacijska semantika programskih jezikov

Aksiomatska semantika programskih jezikov

Vrste formalnih jezikov

Cela števila

Lambda račun

Tipi

Rekurzija

Strukture

Rekurzivni tipi

Lambda račun 2. reda

Lambda račun višjih redov

Predstavitev formalnih jezikov

- Vsak jezik bo formalno definiran
  - Statična in dinamična semantika
- Predstavljene bodo lastnosti vsakega nivoja

  determinizem, normalizacija

  napredek + ohranitev = varnost

  primerjava s hierarhijo Chomskega
- Predstavljeni bodo primeri programov

Različne semantike jezikov

Običajno bomo uporabili operacijsko semantiko za predstavitev jezika

Uporabljali bomo pravila

Večkrat bomo definirali tudi denotacijsko semantiko jezika

While, PCF, F

Abstrakten model jezika

Formalna semantika predstavlja model računskega pomena programa.

> Model običajno predstavi samo del celotnega pomena programa.

Predstavljen del je definiran zadosti formalno, da lahko analiziramo lastnosti jezika ter opazujemo lastnosti pomembnejših obdelav programa kot so npr. preverjanje tipov, evaluacija, itd.

Abstrakten model jezika

Računski pomen je preveč kompleksen

Opazujemo željen nivo in aspekt jezika.

Abstrahiramo značilnosti, ki nas ne zanimajo.

Prerez jezika, ki ga želimo opazovati.

Na primer, lahko pozabimo na shranjevanje podatkov, računski čas, itd.

Lastnosti jezikov

Definiramo striktno preverjanje tipov.

Definiramo evaluacijsko metodo.

Preučujemo implementacijo jezika.

Natančna specifikacija pomena gradnikov jezika.

...

Lastnosti programov

Velikokrat bi želeli podrobneje pogledati nekatere lastnosti programov.

Dokaz, da se program napisan v danem jeziku vedno zaključi.

Preverjanje pravilnosti programa.

Preverjanjo varnosti tipov danega jezika.

Dokaz, da so programi danega jezika deterministični.

...

Koncepti programskih jezikov

Na formalen način si bomo ogledali vrsto konceptov programskih jezikov

- Tipe
- Funkcije
- Podtipe
- Polimorfizem
- Objekte
- Razrede
- Dedovanje
- Spremenljivke tipov
- Preverjanje tipov
- Rekurzijo
- Rekurzivne tipe

Študij jezikov

Formalen zapis pomena izrazov danega programskega jezika ⇒

Pomen izraza je interpretacija (model) izraza, formalen postopek za evaluacijo izrazov, ...

Študij gradnikov programskih jezikov.

Striktna definicija gradnikov.

Osnova za implementacijo gradnikov...

Uporaba tipov

Odkrivanje napak

Abstrakcija

Dokumentacija

Varnost

Učinkovitost

Tipi - odkrivanje napak (1)

Preverjanje tipov omogoča odkrivanje nekaterih napak v času prevajanja oz. v času izvajanja.

Statično preverjanje tipov lahko odkrije presenetljivo veliko napak.

Programi preprosto ``delajo"... :)

Tipi - odkrivanje napak (2)

Moč tega učinka je odvisna od izrazne moči sistema za delo s tipi.

Razlog za ta učinek je predvsem v formi, ki jo daje struktura tipov programom.

Programska koda (posebaj v OO jezikih) je vezana na strukture, ki jih tvorijo tipi.

Zato je enostavneje obvladljiva.

Pomembno je dobro poznati lastnosti tipov!

Abstrakcije

Tipi podpirajo proces programiranja tako, da zahtevajo disciplinirano programiranje.

Tipi so osnovno okostje sistema okoli katerega je implementirana koda.

Sistem tipov tvori osnovo za module--tip modula dejansko ustreza vmesniku modula.

Strukturiranje večjih sistemov v module omogoča bolj abstraktno zasnovo programskih sistemov.

Dokumentacija

Strukture tipov služijo kot osnova za dokumentacijo sistema.

OO jeziki: enostavno generiranje dokumentacije iz parcialnih opisov, ki so vezani na hierarhijo tipov.

Dokumentacija je del samega sistema in jo je tako veliko lažje vzdrževati.

Komentarje običajno lahko enostavno dodajamo na same sintaktične konstrukte programskega jezika

Varnost

Pierce neformalno definira varen

jezik kot jezik, ki ti onemogoči, da se ustreliš v lastno nogo medtem, ko programiraš.

Povedano na drug način: jezik je varen, če varuje lastne abstrakcije.

Polja:

programer ne more narediti nič zelo čudnega s polji. kot na primer spreminjati vsebino polja tako, da piše preko meja neke druge podatkovne strukture.

Učinkovitost

Prvo preverjanje tipov se je začelo pri Fortranu, ko so želeli ločiti med numeričnim računanju s celimi števili in računanju z realnimi števili.

Statično preverjanje tipov omogoča bolj učinkovito delo z vrednostmi.

Odpade mariskatero preverjanje v času izvajanja.

Učinkovitost

Marsikatera odločitev pri generiranju kode pride iz tipov izrazov.

Tip kazalcev vpliva na izbiro inštrukcij za delo s kazalci.

Tehnike za izboljšanje algoritmov za čiščenje pomnilniškega prostora.

...

Praktična uporaba

Načrtovanje jezikov

Meta-programski jeziki

Aplikacije

Načrtovanje jezikov

Programski jeziki morajo biti načrtovani skupaj s sistemom za preverjanje tipov.

Sicer je preverjanje tipov zelo težko.

Dobro načrtovan jezik redko potrebuje podatke o tipih izrazov. Večino lahko izpelje sistem tipe sam.

Meta-programski jeziki omogočajo enostavno načrtovanje jezika kot tudi študij lastnosti načrtovanega programskega jezika.

Meta-programski jeziki

Novejša programska orodja kot je na primer Twelf nam omogočajo prestavitev formalne semantike jezika v meta-programskem jeziku.

Meta jezik nam omogoča implementacijo analiz programskega jezika kot na primer:

preverjanje tipov izraza, programsko dokazovanje lastnosti jezikov, interpretacijo izrazov jezika in podobno.

Aplikacije

Zasnova in implementacija enostavnih programskih jezikov je pogosto potrebna v aplikacijah.

Nov jezik tipično uporabljamo pri zasnovi: uporabniških vmesnikov, aplikacijskih protokolih v porazdeljenih okoljih, ukaznih jezikih aplikacijskega stržnika ali za definicijo formata za shranjevanje podatkov.

Pregled programskih jezikov

Mejniki v razvoju programskih jezikov

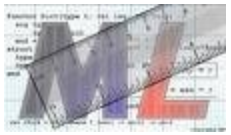
Algol

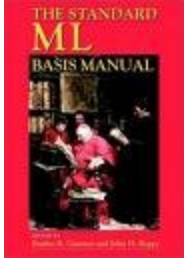
- 1950, Družina programskih jezikov
- Funkcije so lahko del programa
  - Funcija ima lahko parametre
  - Funkcija lahko vrne rezultat
  - Funkcije si bomo natančno ogledali!
- Jezik ima bloke
  - **`begin … end`**
- Algol, Pascal, C, … Modula

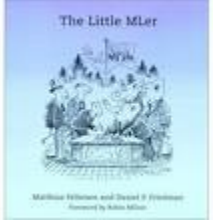
Fortran

- John W. Backus, 1953, IBM
- The IBM Mathematical **For**mula **Tran**slating System
  - IBM 360
  - Luknjane kartice
- Gradniki Fortran
  - DO, GOTO, IF, SUBRUTINE, CALL
- Zelo popularen jezik za numerično procesiranje
- Še vedno zelo živ jezik!
- Implicitni paralelizem
- Fortran program lahko lahko učinkovito paraleliziramo
- Edini takšen jezik!

ML



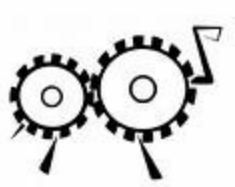




- Meta Language
- Eden od bolj popularnih funkcijskih jezikov
  - Strogi tipi, stdlib, čist jezik
- Edinburgh, 1974, skupina Robin Milner
- Obstaja množica dialektov
  - Standarden ML: SML New Yersey, Moscow SML, MLKit
  - Ocaml je odlična implementacija ML in množice razširitev

Ocaml

- Objektni Caml, Inria, Francija, >1985
- Zelo dobro zasnovan in implementiran programski jezik
  - Funkcijski, imperatitivni in objektni
- Množica orodij, vmesnikov, knjižnic, ... vse kar je običajno in potrebno.
- Zelo hiter jezik:
  - konkurira C,C++
  - Interpreter, prevajalniki za procesorje
  - Zelo aktiven “community”



Objektno usmerjeni jeziki

- Prvi objektni jezik je bil **`Simula`**.
  - Simulacija dejanskega dogajanja.
- **`Smalltalk`** je eden od najslavnejših jezikov
  - Adele Goldberg, Xerox, Palo Alto
  - Vse na tem svetu so samo objekti!
- Vsak objekt je član razreda
  - Prototipni objekt
  - Razred je objekt!
  - Kompleksne hierarhije dedovanja
  - C++ + Java + vsi novi jeziki !
- Java
  - J.Gosling, B.Joy, G.Steele, G.Bracha
    The Java Language Specification
    http://java.sun.com/docs/books/jls/download/langspec-3.0.pdf

Prolog

- Programming in Logic, Robert Kowalski
- Predikatni račun
  - Hornovi stavki
  - Unifikacija, resolucija
- Zelo močen, enostaven in abstrakten jezik
- Uporaba vračanja (back-tracking)
  - Deklarativen in proceduralen pomen Prologa
  - Operator CUT (!)
- Jeziki za delo s podatkovnimi bazami
  - Datalog
- Sicstus, SB Prolog, SWI Prolog,…

Literatura

- Benjamin Pierce

  Types and Programming languages, MIT Press, 200
- Robert Harper

  Practical Foundations for Programming Languages, Draft, 2010.
- Glynn Winskel

  The Formal Semantics of Programming Languages: An Introduction, MIT Press, 1993 (1-8).
- Iztok Savnik

  Teorija programskih jezikov, Zapiski, 2012.