Teoreetilisest arvutiteadusest ja funktsionaalprogrammeerimisest, isiklikult pinnalt

Tarmo Uustalu

TÜ, 13.12.2023

Kes ma olen?

- matemaatik (?)
- teoreetiline arvutiteadlane (?)
- loogik (?)
- TTÜ juhtivteadur loogikas ja semantikas, `https://cs.ioc.ee/lsg/`
- Reykjaviki Ülikooli professor programmeerimiskeelte teoorias

Millest täna räägin?

- Natuke endast, jagamaks oma kogemust akadeemilisest või teadusmaailmast
- Funktsionaalprogrammeerimisest kui “praktilisest” tehnoloogiast, millele minu teadus on aluseks

Akadeemiline elulugu

- s 1969
- 1976–1987 Tln 44. Kk (tänaseks TMG)
- 1976–1984 Nõmme LMK (tänaseks Nõmme MK)
- 1982–2016 Kübl (likvideeriti)
- 1984–1988 Otsa Kool (tänaseks MUBA)
- 1987–1992, 2002– TPI/TTÜ (tänaseks ümber nimetatud)
- 1991 NTH (tänaseks NTNU)
- 1992–1999 KTH
- 2000–2002 U do Minho
- 2017– Reykjavik U

Mõjutajad matemaatikas / teoreetilises arvutiteaduses

- minu isa
- Maali Pärt
- Peeter Lorents
- Grigori Mints
- Enn Tõugu
- Varmo Vene
- Jaan Penjam
- mitmed teised vanemad ja uuemal ajal ka eriti nooremad kolleegid

Teaduslikud huvid

- matemaatiline loogika, iseäranis tõestusteooria
  (aga filosoofiline loogika ka serva peal)
- sh eriti tüübiteooria
- tõestusassistendid,
  matemaatika/programmeerimisteooria formaliseerimine
- kategooriateooria
- programmeerimiskeelte teooria:
  semantika, programmianalüüsid ja -teisendused,
  tüübisüsteemid ja programmiloogikad,
  formaalsete keelte ja automaatide teooria,
  konkurentsuse teooria

Matemaatika

- “Ma olen matemaatik”
- Matemaatik on professioon, mitte tiitel
- Ala on mittemateriaalne, abstraktne, fundamentaalne
- aitab elu mõtestada
- Tegemise protsessis intuitiivne, mitteformaalne, isegi verbaliseerimatu
- Lõppproduktina täpselt vastupidise iseloomuga, kuiv, range
- (Vrd muusika)
- Ei reosta
- Ei ole ideoloogiline
- Ei tee kurja (?) Aga (otse või läbi IT): (info)relvad, AI, . . .

Teoreetiline arvutiteadus

- Matemaatilisem osa arvutiteadusest, mõnikord kutsutud ka klassikaliseks arvutiteaduseks
- Enamvähem see, mis arvutiteadus oli enne, kui poliitikud leiutasid “infotehnoloogia” ja “arvutiteadused”
- Püüab mõista arvutamist ja programmeerimist fundamentaalsel tasemel, sh nende piire
- Hea valdkonnasisene kliima, üldiselt ühesugused väärtused ja head kollegiaalsed suhted, solidaarsus
- Arvutiteaduse instituudid annavad tööd väga paljudele puhastele matemaatikutele

Akadeemiline “karjäär”

- Ei ole “normaalsetele” inimestele
- Ei ole karjäär tavamõistes, pigem selle farss
- Mõte on teha, kui on sisemine sund, enesedistsipliin (*drive*)
- Palk ei kompenseeri stressi vastutusest ega jaburust süsteemis
- Tegelik hüvitus on rõõm teha asja, millest ise oled huvitatud ja oma elu olulisel määral ise juhtida
- samuti eluaegsed sõprused sarnastelt mõtlevate inimestega kogu maailmast

Akadeemiline maailm 1980.-test 2020.-teni

- 1980.-tel võis akadeemiline maailm olla mõnevõrra tolmune elevandiluutorn, mida juhtisid mitte alati väga targalt akadeemikud ise
- 40 a hiljem puudub tolm täiesti; ülikooli on totaalselt üle võtnud administraatorid auditiühiskonna ja poliitkorrektsuse vaimus
- Valed motiivid (*publish and perish* jm), uued ideoloogilised pseudoteadused, teadlaste moraali lagundamine
- sh mitte harva “teaduseetika” sildi all

Doktorantuur välismaal

- Soovitan, kui on eraeluliselt võimalik
- sest on teaduslikult väga arendav, kui õnnestub valida õige koht
- Ka avardab elamine teises kultuuris maailmapilti
- Kindlasti tuleks ette määratleda enda temaatiline huvi
- luua otsekontakt potentsiaalse juhendajaga
- Alles vastastikuse huvi alusel esitada ametlik avaldus administraatoritele

Postdokikohad ehk mobiilsus

- “Mobiilsuse” fetišeerimine on viigileht akadeemilise maailma struktuursete probleemide varjamiseks
- Doktoreid, kes tahaks jätkata akadeemias, toodetakse globaalselt suurusjärk ülearu võrreldes kohtade arvuga järgmistel karjääriastmetel
- Enamuse jaoks on postdokikoht/kohad enesepettus
- Lotovõitjale on postdokk siiski akadeemiliselt üsna mõttekas iseseisva teadusprogrammi ja võrgustiku kasvatamise jaoks
- kui sobitub eraeluga
- “Kahe keha probleem” on akadeemilises maailmas väga massiline

Teoreetiline arvutiteadus Eestis

- minu rühm Tallinnas
- Paweł Sobociński rühm Tallinnas
- Vesal Vojdani rühm TÜs
- end Dominique Unruh', Vitaly Skacheki, Helger Lipmaa rühm TÜs
- Dirk Theisi rühm TÜs
- Meelis Kulli rühm TÜs
- Peeter Laud jt Cyberneticas

Ühistegemised

- ülemaailmse loogikapäeva Eesti töötoad (2021–2024, `https://cs.ioc.ee/lsg/wld24/`
- Eesti ja Eesti-Läti arvutiteaduse teooriapäevad (2002–2019, 2022, 2024), `https://cs.ioc.ee/lsg/tdays/`
- Eesti arvutiteaduse talvekoolid (1996–2020, 2024), `https://cs.ioc.ee/ewscs/`
- rida rahvusvahelisi konverentse on toimunud Eestis al. 1990.-test (ja varemgi), 2024. a-l tulevad siia LICS+ICALP+FCSD

The functional programming revolution

Tarmo Uustalu
Reykjavik University

UTmessan, 8 February 2019

The mankind's oldest (?) algorithm on infinite data

- Eratosthenes (3rd c BC) proposed:
  - Write down *all* numbers starting from 2
  - Put the first number aside, remove its multiples from the rest
  - Repeat, *ad infinitum*
  - See what you've got
- If you could do so, this would happen:

2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
2	3		5		7		9		11		13		15	...
2	3		5		7				11		13			...
2	3		5		7				11		13			...
								...						
2	3		5		7				11		13			...

- But *can* you?

Direct coding in a modern programming language

- In Haskell, we write:

```
from n = n : from (n + 1)

sieve (n : ns) = n : sieve (pruneMult n ns)
  where
    mult       k n  = n `mod` k == 0
    pruneMult k ns = filter (not . mult k) ns

primes = sieve (from 2)
```

- Running `primes`, we get

```
[2,3,5,7,11,13,...
```

- Should this not be impossible?

This talk

- What is functional programming (FP)?
- What are some functional languages?
- What makes FP appealing?
- Applicability

Functional programming?

- The word could mean "it works"...
- ...but means "you (almost) only program *functions*"
- ie transformations of arguments to return values

Purely functional programming?

- Even what *seems not to be* a function . . .
  - eg a computation that can raise an exception instead of returning a value
  - or an interactive input-output computation instead of an argument-to-return-value computation
- . . . *is still* a function, if you look at it in the *right way*
- . . . so you can program it as such

The basis of FP: lambda calculus

- We have expressions involving *variables*

$$\ldots x \ldots y \ldots$$

- built of λ-*abstractions* (that make functions)

$$\lambda x.\, t$$

- and *applications* (that use functions)

$$t\, u$$

- Computation is reduction:

$$(\lambda x.\ t)\, u \longrightarrow t[u/x]$$

Alonzo Church

Functional and typed?

- Everything you program is *function*,
- moreover it is *typed* (the function has a domain and a range)
- either by you or the language processor

- Type *checking* or *inference ensures*
  the program cannot go wrong
- Programmer-provided types also serve as documentation;
  machine-inferred types serve as confirmation

Typed lambda calculus

- Expressions are typed

$$t : A$$

  with base types and function types

$$A \to B$$

- Typing of λ-abstractions:

$$\frac{x : A \vdash t : B}{\lambda x.\, t : A \to B}$$

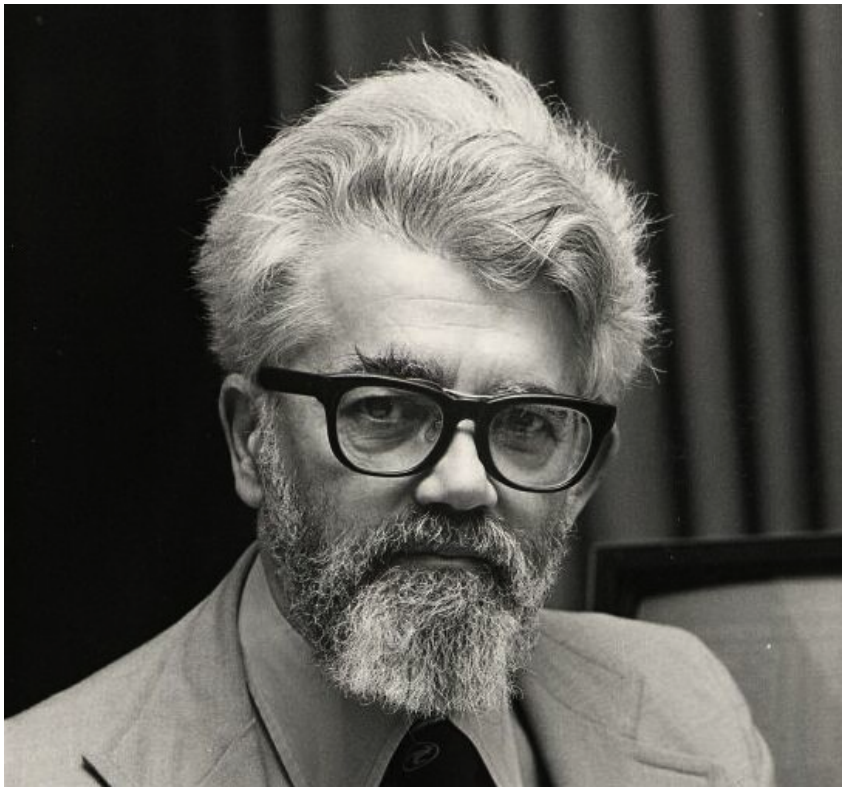
- Typing of applications:

$$\frac{t : A \to B \quad u : A}{t\, u : B}$$

- Reduction preserves an expression's type

Lisp, the first FP language

- LISP was developed at MIT in 1958, for AI applications
- It is the 2nd oldest high-level programming language still in use today (following Fortran)
- Lisp is impure and untyped (and uses fully parenthesized prefix syntax)



John McCarthy

The zoo of FP languages

- Lisp family:
  Common Lisp, Scheme, Clojure (untyped, eager)
- ML family:
  Standard ML, OCaml, F# (typed, eager)
- Clean and Haskell (typed, lazy)
- Scala (functional + OO)
- Erlang (concurrency)
- PureScript (scripting)
- Agda, Idris (dependently typed, type-checking proves your program correct)

The spirit of FP: abstraction

- When something recurs in your code, *abstract* it
- This leads to more modular code that is easier to understand
- (or not, if you go too far)

- You want to choose a language supporting abstracting those kinds of recurring things you care about

Higher-order and polymorphic functions

- Functions may take *function arguments*
- Functions can be defined *for every type*

- `filter :: (a -> Bool) -> [a] -> [a]`
- `(.) :: (b -> c) -> (a -> b) -> a -> c`

Type classes

- Collect together types (or type constructors) that share *similar functionality*

- ```
  class (Real a, Enum a) => Integral a where
    quotRem :: a -> a -> (a, a)
    toInteger :: a -> Integer
  ```
- ```
  class Show a where
    showsPrec :: Int -> a -> ShowS
    show :: a -> String
  ```

- (Cf: OO classes collect objects with similar functionality)

Monad – the most (in)famous type class

- Collects type constructors that model impure notions of computation
- Eg exceptions, nondeterminism, probability, mutable state, interactive I/O

More abstractions

- User-defined *recursive data and record types*
- *Generic* programming
- *Type-level* programming

- *Dependent* types – types indexed by values of other types, allow for very precise specification of functions

Laziness (in languages like Haskell)

- The argument in an application is evaluated only when the evaluated function *first needs* it
- Of a data structure, only as much is produced as is consumed
- This enables programming with infinite data structures for free

- `from n = n : from (n+1)`
- `take 5 (from 3)`
  `===> [3,4,5,6,7]`

Strengths of FP

- Programs *high-level/declarative*, easy to develop and understand
- *Safety* from strong typing
- Abstraction helps tame *complexity*;
  programs *modular*, easy to maintain
- Programs amenable to automatic or semi-automatic *analysis*, *reasoning* (correctness proofs)

- Well-suited for *mathematical domains*
- Well-suited for manipulation of *linguistic data*
  (metaprogramming, eg DSL development)
  and data conforming to *complex schemes* in general

Weaknesses

Perceived weaknesses

- Poor *performance* –
  only applies to domains where direct low-level programmer control of resources is desirable
- Purity and strong typing limit *expressiveness* –
  just wrong

Real issues

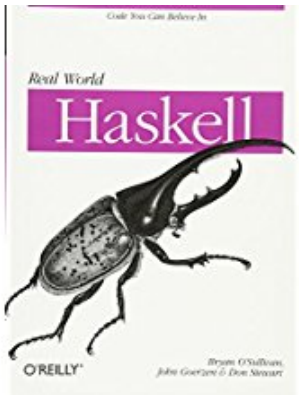
- *Education* not geared for FP, so not enough programmers
- Weak *support* (libraries, developer tools, documentation), FP languages are developed by small communities

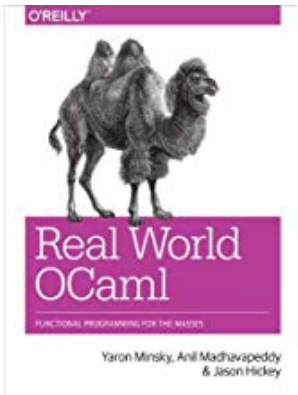
FP applications in the real world

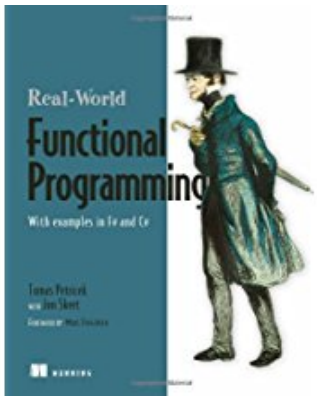
- Numerical computing with complex algorithms, eg in fintech
- Big and rich data analytics, machine learning
- Web, concurrent and parallel programming
- Correctness and security critical software

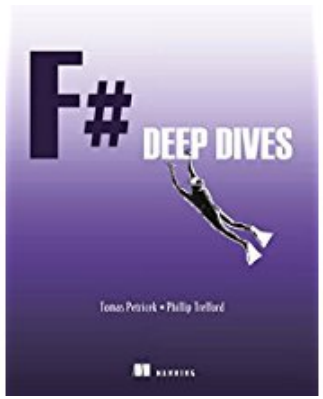
Books on real-world applications

- Sullivan, Stewart, Goerzen, Real World Haskell
- Minsky, Real World OCaml
- Petricek, Skeet, Real-World Functional Programming: With Examples in F# and C#
- Petricek, Trelfort, F# Deep Dives









Functional Art and Music, FARM

- Workshop on applications of FP to art and music at ICFP
- FP applications to music, eg harmonization, composition
- Functional art, modelling, design

- Cellular automata scarves (Fabien Serrière)