Languages all the way down

Alexander Gryzlov,
IMDEA Software Institute

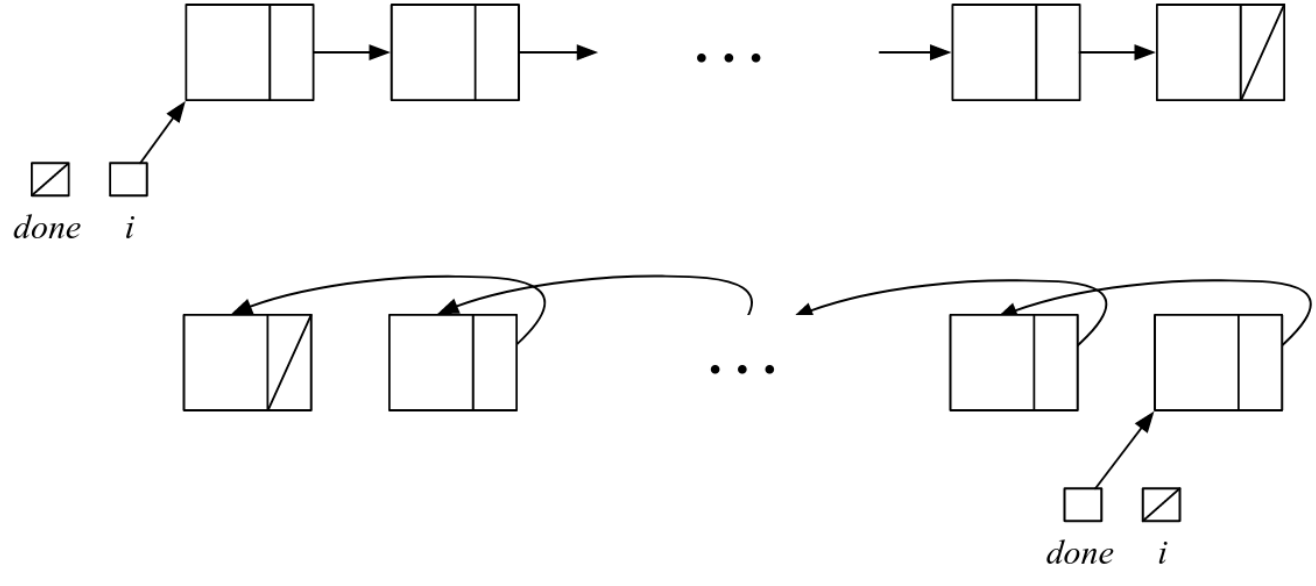
Celonis, Madrid, 19/10/2022

Intro

- Research software engineer
- Write proofs for imperative (concurrent) algorithms
- Programs that assume a global storage and manipulate pointers to it

List reversal

```
done ← null;
while (i ≠ null) do {
    k ← !(i + 1);
    i + 1 := done;
    done ← i;
    i ← k;
}
```



List reversal

- Written in some "midlevel" language
- You can compile it to machine code
- You can specify it with some abstraction

Compilation

$done \leftarrow$ null;
while $(i \neq$ null$)$ do $\{$
  $k \leftarrow !(i+1)$;
  $i+1 := done$;
  $done \leftarrow i$;
  $i \leftarrow k$;
$\}$

```
.LBB2_1:
        cmp     qword ptr [rbp - 32], 0
        je      .LBB2_3
        mov     rax, qword ptr [rbp - 32]
        mov     rax, qword ptr [rax + 8]
        mov     qword ptr [rbp - 16], rax
        mov     rcx, qword ptr [rbp - 24]
        mov     rax, qword ptr [rbp - 32]
        mov     qword ptr [rax + 8], rcx
        mov     rax, qword ptr [rbp - 32]
        mov     qword ptr [rbp - 24], rax
        mov     rax, qword ptr [rbp - 16]
        mov     qword ptr [rbp - 32], rax
        jmp     .LBB2_1
```

Specification

```
reverse :=
done ← null;
while (i ≠ null) do {
   k ← !(i + 1);
   i + 1 := done;
   done ← i;
   i ← k;
}
```

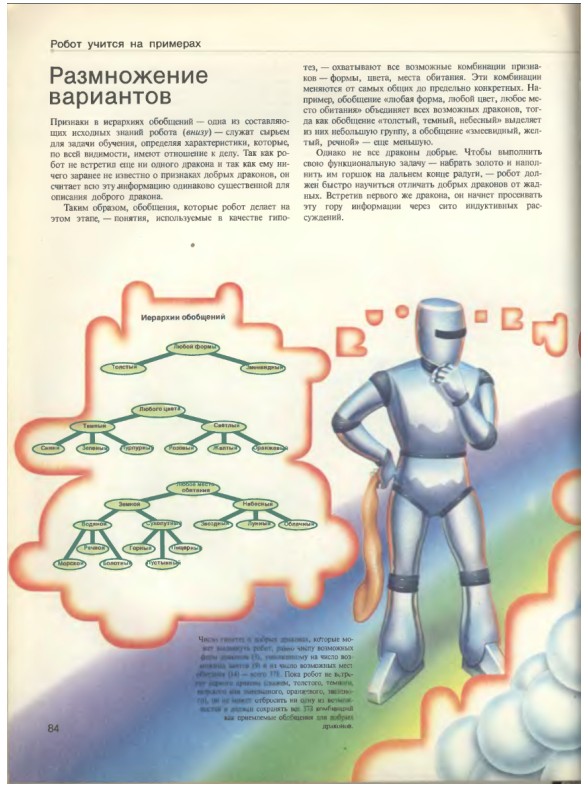
```
list A :=   nil
          | :: of A & list A

rev nil    = nil
rev x::xs = rev xs ++ x::nil

{is_list i l}
reverse
{is_list done (rev l)}
```

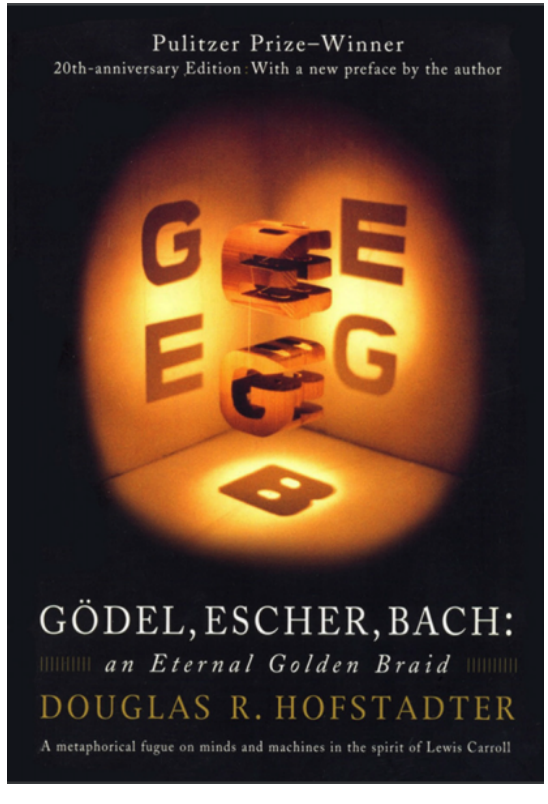
Language stack

- Interaction between machine and human
- The gap is large, so we have a series of languages
- Programming is about navigating this stack
- Can computers also help with this navigation?

Me and languages

- A linguist father
- Got a computer in early 90s for text processing
- "Computer obtains a mind", 1985-1990
- "Goedel, Escher, Bach", 1979





Me and languages

- Perceptrons, expert systems, semantics, learning
- Quines, fixpoints, self-modifying code
- Went to get a CS degree in mid 2000s

input to a program = string in a new language

the program itself = its interpreter

Language-Oriented Programming

Early work experience

- C/C++/Java (also some Smalltalk)
- Always about manipulating state ("mid-stack")
- Leaking abstractions: segfaults, null pointers, bizarre errors
- Spent a few years playing detective

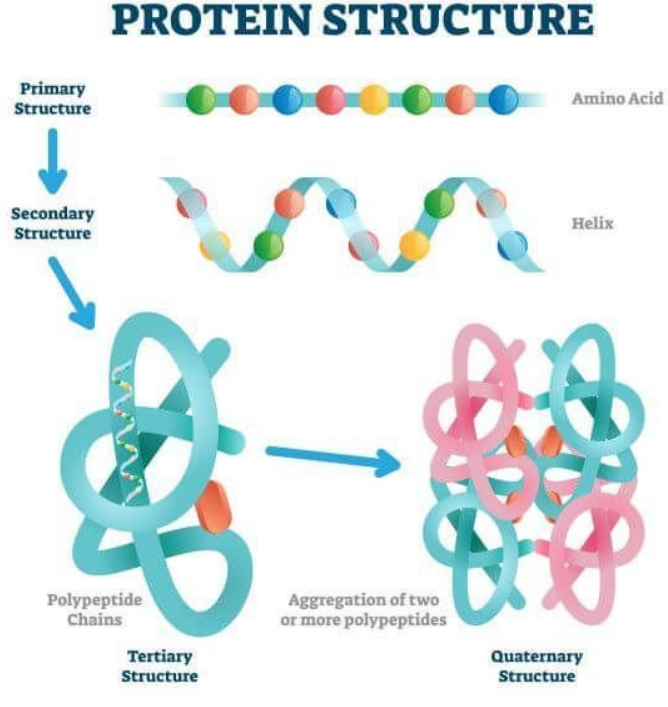
$$
\begin{array}{l}
done \leftarrow \mathsf{null}; \\
\mathsf{while}\ (i \neq \mathsf{null})\ \mathsf{do}\ \{ \\
\quad k \leftarrow\ !(i+1); \\
\quad i+1 := done; \\
\quad done \leftarrow i; \\
\quad i \leftarrow k; \\
\}
\end{array}
$$

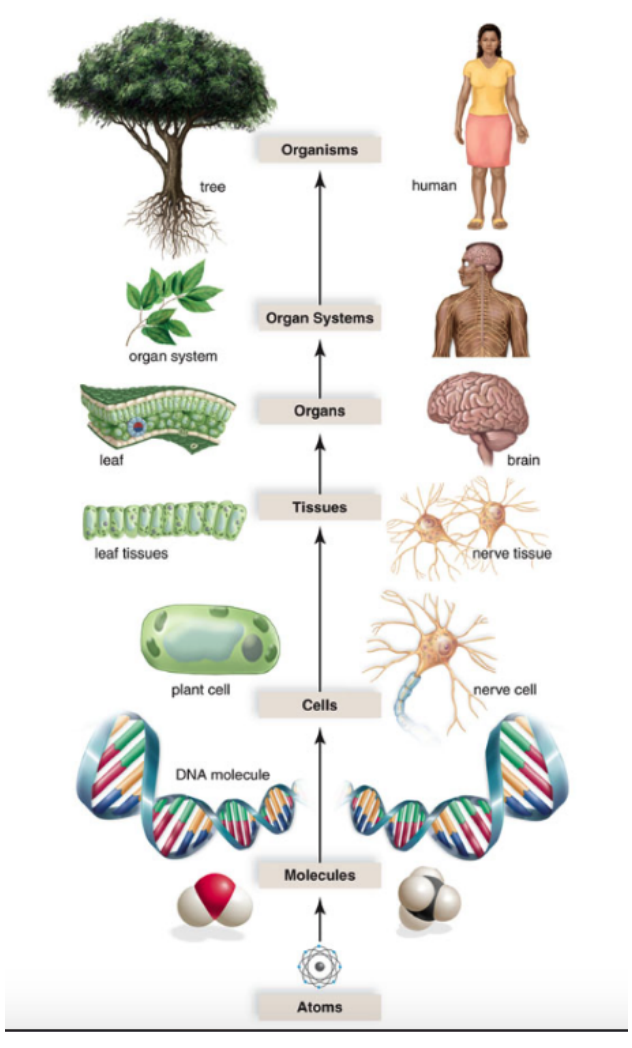
Early work experience

- Nice theory, rough practice
- The computer is not a very coherent dialogue partner
- "This is just bit shuffling"
- Need to look for languages somewhere else

Biology

- Can be seen as "language stacks" incarnate
- Biosemiotics, code, signalling





Systems biology

Lin et al., Journal of Biological Engineering [2018]

"Synthesis of control unit for future biocomputer"

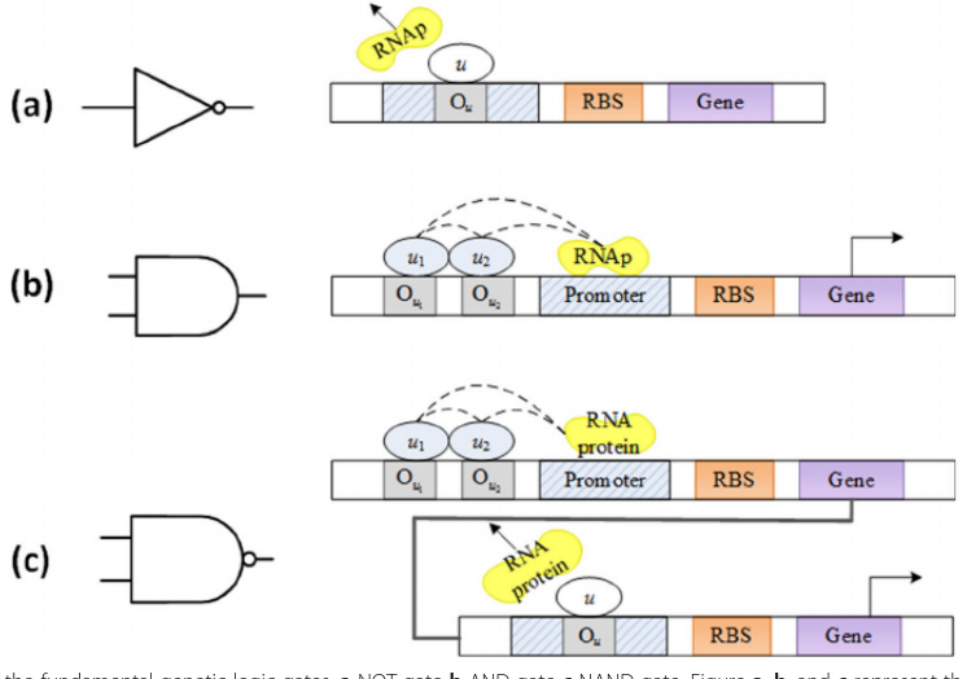


Fig. 2 Structure of the fundamental genetic logic gates. **a** NOT gate **b** AND gate **c** NAND gate. Figure **a**, **b**, and **c** represent the genetic sequences for expressing the logic functions, respectively

Functional programming

- Played with Lisp before
- Started getting interested in typed functional languages like OCaml and Scala (early 2010s)
- Began with parsers and data pipelines
- Bioperl, Biocaml, ...

Bioinformatics

- Applied to a PhD in Plant Systems Biology
- Studied for 2.5 years before dropping out
- More interested in quantitative effects than in big picture
- "Computational glue" pipelines (Perl, Fortran, MATLAB, etc), ad-hoc write-and-forget scripts
- Even less trustworthy!

Script troubles

Characterization of Leptazolines A–D, Polar Oxazolines from the Cyanobacterium *Leptolyngbya* sp., Reveals a Glitch with the "Willoughby–Hoye" Scripts for Calculating NMR Chemical Shifts

Jayanti Bhandari Neupane, Ram P. Neupane, Yuheng Luo, Wesley Y. Yoshida, Rui Sun, and Philip G. Williams*

Department of Chemistry, University of Hawai'i at Mānoa, 2545 McCarthy Mall, Honolulu, Hawaii 96822, United States

e.g., this paper from 2019 suggests 150+ papers could have wrong results:

"The error is the result of a simple file sorting problem. On operating systems without default file name sorting, the script fails to match the files containing a conformer's free energy with its chemical shift – leading to an overall wrong value."

Data engineering

- Went back to the industry, wishing for more rigor
- R&D in ad-tech
- Recommender engines, distributed data pipelines
- Started fully embracing FP (mid 2010s)
- "Stats and monads" department

Specs and types

```
list A :=   nil
          | :: of A & list A

rev nil   = nil
rev x::xs = rev xs ++ x::nil
```

FP is executable specs:

```
def rev[A] : List[A] => List[A] = {
  case Nil => Nil
  case (x::xs) => rev(xs) ++ List(x)
}
```

Monads & DSLs

- Embrace Language-Oriented Programming, use mini-languages
- Need for meta-language constructs
- Monads model sequential composition
- `cause => effect[result]`
- In CS computation is thought of as directed
- E.g. multiplying numbers vs factoring primes

Scalability

- Trustworthiness is really useful for a foundation
- Performance will suffer
- Scale horizontally => parallelism & concurrency
- Pandora's box: all about fine-grained communication
- No longer interacting with computer 1-1, you're outnumbered
- Sequentiality also starts leaking

Break up monads

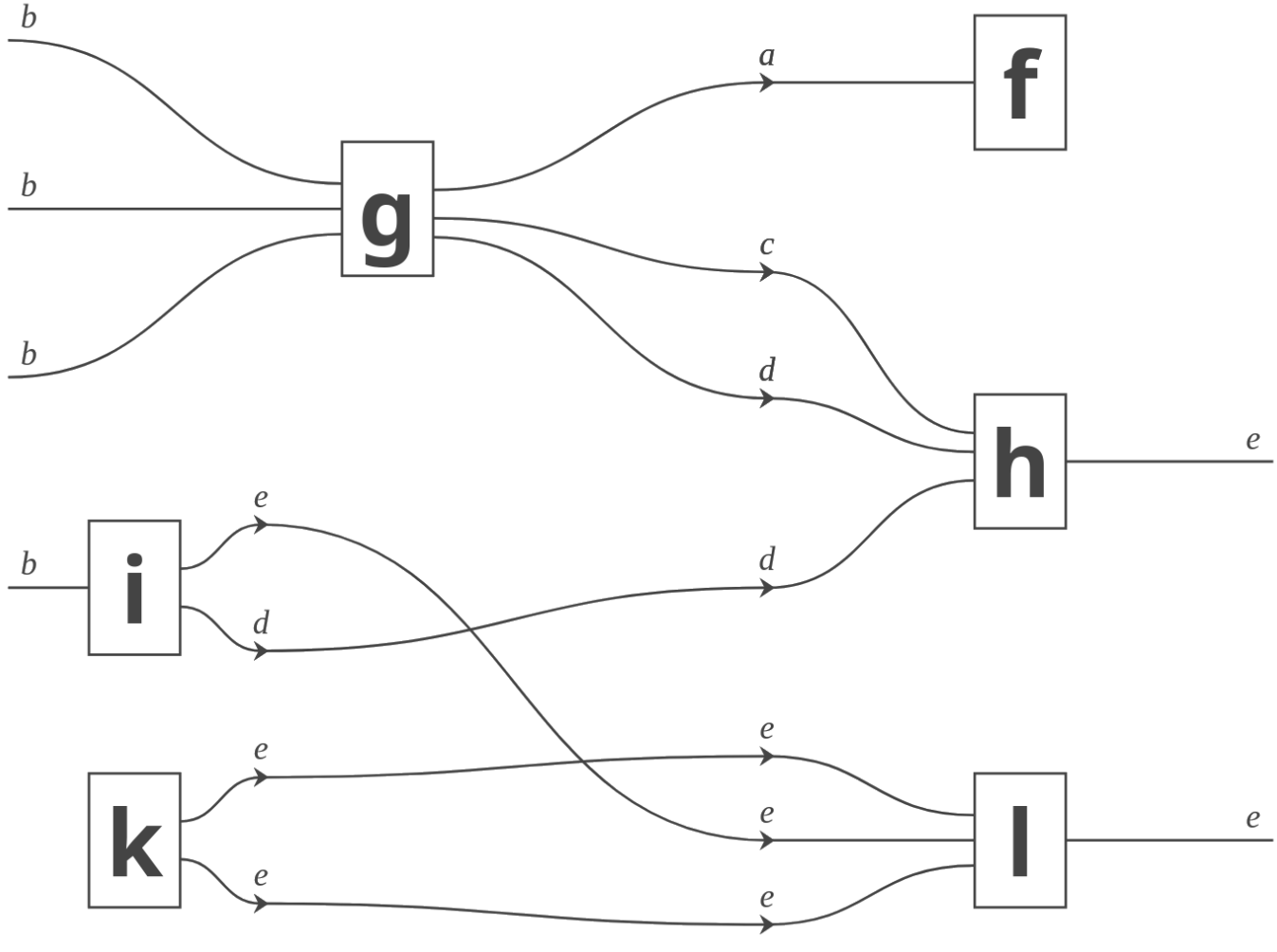
Applicatives

```
  def pure[A](a: A): F[A]
  def apply[A, B](f: F[A => B]): F[A] => F[B]
```

Arrows

```
Arrow[F[_, _]]

def lift[A,B](f: A => B): F[A,B]
def dimap[A,B,C,D](fab: F[A,B])
         (f: C=>A)(g: B=>D): F[C,D]
def second[A,B,C]
          (fa: F[A,B]): F[(C,A),(C,B)]
def split[A,B,C,D]
         (f: F[A,B], g: F[C,D]): F[(A,C),(B,D)]
```

Visual programming



Categorical cybernetics

Capucci, Gavranovic, Hedges, Rischel [2021]

"Towards foundations of categorical cybernetics"

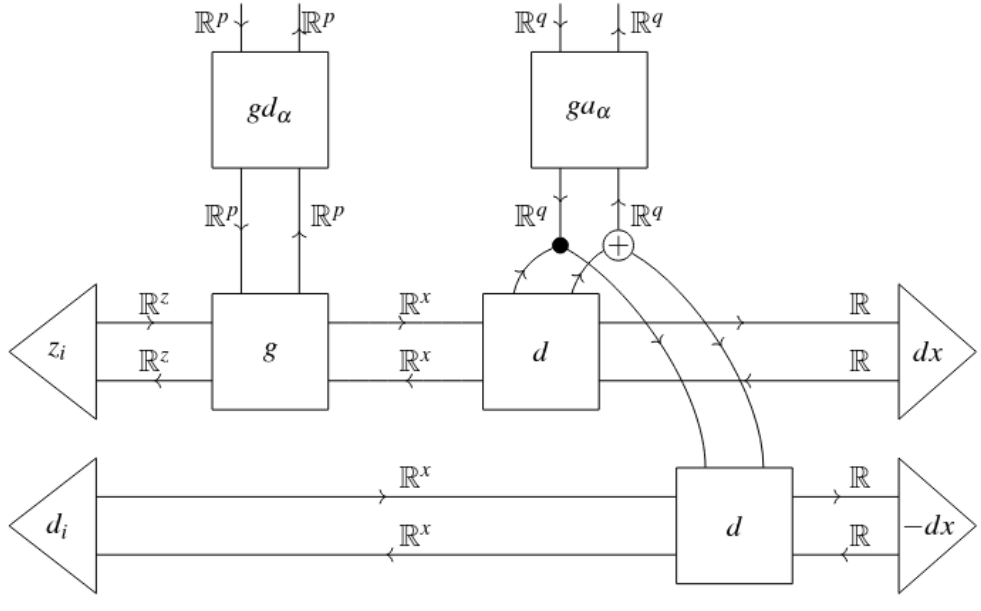


Figure 7: A generative adversarial network as a closed system.

Build on monads

- We wanted expressive declarative DSLs
- FP - higher-order functions + higher-order types
- Abstracts and hides some of the complexity
- The computer can have coherent dialogue but can be very excrutiating

Logic programming

- A second side of declarative is logic programming
- ***Solving*** as the default mode of computation
- Norvig's Corollary to Greenspun's Tenth Law of Programming: *"Any sufficiently complicated LISP program is going to contain a slow implementation of half of Prolog"*
- Filling the gaps: implicits

Proof engineering

- Dependent types are a powerful metalanguage
- Solving becomes undecidable, manual proofs
- Programming -> constructive math & logic
- Equality, ordering, choice, finiteness
- Countability ~ serialization

AI vs IA

- Thinking is scarce
- Moshe Vardi: Fast and Slow Thinking
- Expressive specs mean more powerful tools
- Typechecker feedback loop
- An interesting dialogue with machine

List reversal in Coq/HTT

Demo!

Contacts

- https://software.imdea.org/~aliaksandr.hryzlou/
- https://www.linkedin.com/in/alexgryzlov/
- https://github.com/clayrat/
- https://twitter.com/clayrat/