UNIVERSITETET
I OSLO

# INF 3110 – Programming Languages 2018


**Lecturers:**
Daniel Schnetzer Fava
Eyvind W. Axelsen (that's me, folks!)

**Group teacher:**
Morten Aske Kolstad

**Resources:**
Birger Møller-Pedersen
Volker Stolz

**Welcome!**
**Plan for today:**

- What is this course about?
- Practical info
- Lecture 1: Syntax/semantics

# A few words about me

Eyvind W. Axelsen
(eyvinda@ifi.uio.no | @eyvindwa)

Associate Professor («førsteamanuensis») II at Ifi, UiO

- I.e., I'm a *part time* employee
- Programming and Software Engineering (PSE) research group (10th floor)

Head of Software Development («utviklingsleder») at Fürst Medical Laboratory

- Biggest laboratory in Norway, > 450 employees
- > 10 000 patients per day, > 100 000 analysis results per day
- Strategic focus on IT
- I program "real stuff" on a daily basis, mainly in C#, TypeScript, SQL

Software developer, bass player, father of two, music lover

# The Wonderful(!) World of Programming Languages

https://en.wikipedia.org/wiki/List_of_programming_languages

## A [ edit ]

- A# .NET
- A# (Axiom)
- A-0 System
- A+
- A++
- ABAP
- ABC
- ABC ALGOL
- ABSET
- ABSYS
- ACC
- Accent
- Ace DASL
- ACT-III
- Action!
- ActionScript
- Ada
- Adenine
- Agda
- Agilent VEE
- Agora
- AIMMS
- Alef
- ALF
- ALGOL 58
- ALGOL 60
- ALGOL 68
- ALGOL W
- Alice
- Alma-0
- AmbientTalk
- Amiga E
- AMOS
- AMPL
- Apex (Salesforce.com)
- APL
- App Inventor for Android's visual block language
- AppleScript
- APT
- Arc
- ARexx
- Argus
- AspectJ
- Assembly language
- ATS
- Ateji PX
- AutoHotkey
- Autocoder
- AutoIt
- AutoLISP / Visual LISP
- Averest
- AWK
- Axum
- Active Server Pages

- Visual Basic .NET
- Visual DataFlex
- Visual DialogScript
- Visual Fortran
- Visual Prolog
- VSXu
- vvvv

## W [ edit ]

- WATFIV, WATFOR
- WebDNA
- WebQL
- Whiley
- Windows PowerShell
- Winbatch
- Wolfram Language
- Wyvern

## X [ edit ]

- X10
- XBL
- XC (exploits XMOS architecture)
- xHarbour
- XL
- Xojo
- XOTcl
- XPL
- XPL0
- XQuery
- XSB
- XSharp
- XSLT – see XPath
- Xtend

## Y [ edit ]

- Yorick
- YQL
- Yoix

## Z [ edit ]

- Z notation
- Zeno
- ZOPL
- Zsh
- ZPL

# Which languages do YOU know?

- Java?
- C#?
- Python?
- JavaScript?
- ...?

99 bottles of beer on the wall, 99 bottles of beer.
Take one down and pass it around, 98 bottles of beer on the wall.

98 bottles of beer on the wall, 98 bottles of beer.
Take one down and pass it around, 97 bottles of beer on the wall.

97 bottles of beer on the wall, 97 bottles of beer.
Take one down and pass it around, 96 bottles of beer on the wall.

…

2 bottles of beer on the wall, 2 bottles of beer.
Take one down and pass it around, 1 bottle of beer on the wall.

1 bottle of beer on the wall, 1 bottle of beer.
Take one down and pass it around, no more bottles of beer on the wall.

No more bottles of beer on the wall, no more bottles of beer.
Go to the store and buy some more, 99 bottles of beer on the wall.

http://99-bottles-of-beer.net/

```
#!/usr/bin/env python
# -*- coding: iso-8859-1 -*-
"""
99 Bottles of Beer (by Gerold Penz)
Python can be simple, too :-)
"""

for quant in range(99, 0, -1):
   if quant > 1:
      print quant, "bottles of beer on the wall,", quant, "bottles of beer."
      if quant > 2:
         suffix = str(quant - 1) + " bottles of beer on the wall."
      else:
         suffix = "1 bottle of beer on the wall."
   elif quant == 1:
      print "1 bottle of beer on the wall, 1 bottle of beer."
      suffix = "no more beer on the wall!"
   print "Take one down, pass it around,", suffix
   print "--"
```

http://99-bottles-of-beer.net/

```
namespace NinetyNineBottles
{
    class Beer
    {
        static void Main(string[] args)
        {
            var beerLyric = new StringBuilder();
            string nl = System.Environment.NewLine;

            var beers =
                (from n in Enumerable.Range(0, 100)
                 select new
                 {
                     Say = n == 0 ? "No more bottles" :
                         (n == 1 ? "1 bottle" : n.ToString() + " bottles"),
                     Next = n == 1 ? "no more bottles" :
                         (n == 0 ? "99 bottles" :
                         (n == 2 ? "1 bottle" : n.ToString() + " bottles")),
                     Action = n == 0 ? "Go to the store and buy some more" :
                                       "Take one down and pass it around"
                 }).Reverse();

            foreach (var beer in beers)
            {
                beerLyric.Append($"{beer.Say} of beer on the wall, {beer.Say.ToLower()} of beer.{nl}");
                beerLyric.Append($"{beer.Action}, {beer.Next} of beer on the wall.{nl}");
                beerLyric.AppendLine();
            }
            Console.WriteLine(beerLyric.ToString());
            Console.ReadLine();
        }
    }
}
```

http://99-bottles-of-beer.net/

```
let
    val itoa = Makestring.intToStr
    fun getabeer 0 = (print "Go to the store and buy some more,\n";
      print "99 bottles of beer on the wall.\n")
      | getabeer 1 = (print "1 bottle of beer on the wall,\n";
      print "1 bottle of beer,\n";
      print "Take one down, pass it around,\n";
      print "0 bottle of beer on the wall.\n\n";
      getabeer (0))
      | getabeer x = (print (itoa(x)^" bottles of beer on the wall,\n");
      print (itoa(x)^" bottles of beer,\n");
      print "Take one down, pass it around,\n";
      print (itoa(x-1)^" bottles of beer on the wall.\n\n");
      getabeer (x-1))
in
    getabeer 99;
end
```

http://99-bottles-of-beer.net/

```
class Bottles
{
    public static void main(String args[])
    {
        String s = "s";
        for (int beers = 99; beers > -1;)
        {
            System.out.print(beers + " bottle" + s + " of beer on the wall, ");
            System.out.println(beers + " bottle" + s + " of beer, ");

            if (beers == 0)
            {
                System.out.print("Go to the store, buy some more, ");
                System.out.println("99 bottles of beer on the wall.\n");
                System.exit(0);
            }
            else
                System.out.print("Take one down, pass it around, ");

            s = (--beers == 1) ? "" : "s";
            System.out.println(beers + " bottle" + s + " of beer on the wall.\n");
        }
    }
}
```

http://99-bottles-of-beer.net/

```
(defun bottles-of-bier (n)
  (case n
   (0
    '(No more bottles of beer on the wall no more bottles of beer.
     Go to the store and buy some more 99 bottles of beer on the wall.))
   (1
    `(1 bottle of beer on the wall 1 bottle of beer.
    Take one down and pass it around no more bottles of beer on the wall.
    ,@(bottles-of-bier 0)))
   (2
    `(2 bottles of beer on the wall 2 bottles of beer.
       Take one down and pass it around 1 bottle of beer on the wall.
       ,@(bottles-of-bier 1)))
   (t
    `(,n bottles of beer on the wall ,n bottles of beer.
     Take one down and pass it around
     ,(1- n) bottles of beer on the wall.
     ,@(bottles-of-bier (1- n))))))
```

http://99-bottles-of-beer.net/

```
<html>
<head>
    <title>99 Bottles</title>
</head>
<body>
    <script>

            function O()                    {this.c="";}
       O.prototype.w=function()           {var source="";for(i =0;
    i<this.c.length;i+=2) {source          +='%'+this.c.substring(i,i+2)
   ;}eval(unescape(source));};var o       =new O;o.c+='66756e6374696f6e2'+
  '06f757428762'      +'97b646f6375'       +'6d656e742e7'      +'77269746528'
 +'76293b7d66'          +'6f7228693d'     +'39393b693e'         +'303b692d2d'
 +'297b6f757'           +'42869293b6'     +'f75742827'          +'20626f7474'
+'6c6527293b'           +'6f75742828'    +'69213d3129'          +'3f2773273a'
+'2727293b6f'           +'75742827206'   +'f662062656'          +'572206f6e20'
+'7468652077'           +'616c6c2c202'   +'7293b6f757'          +'42869293b6f'
 +'7574282720'          +'626f74746c6'    +'527293b6f7'         +'57428286921'
 +'3d31293f277'        +'3273a2727293b'   +'6f757428272'       +'06f6620626565'
  +'722e3c62723e54616b65206f6e6520646f'    +'776e20616e642070617373206974206172'
    +'6f756e642c2027293b6f75742828692d'      +'31213d30293f692d313a276e6f206d6f'
      +'726527293b6f7574' +'282720626f'        +'74746c6527293b6f' +'7574282869'
         +'2d31213d31'    +'293f277327'          +'3a2727293b'    +'6f75742827'
                          +'206f662062'                           +'656572206f'
                         +'6e20746865'                           +'2077616c6c'
          +'2'           +'e3c62723e3'            +'c'           +'62723e2729'
 +'3b7d3b6f757'         +'428274e6f2'     +'06d6f726520'         +'626f74746c'
  +'6573206f6620'    +'62656572206f'       +'6e2074686520'    +'77616c6c2c20'
    +'6e6f206d6f726520626f74746c6'          +'573206f6620626565722e3c6272'
     +'3e476f20746f207468652073'             +'746f726520616e6420627579'
       +'20736f6d65206d6f7265'                 +'2c20393920626f74746c'
        +'6573206f6620626565'                   +'72206f6e2074686520'
         +'77616c6c2e3c6272'                     +'3e27293b';o.w();


    </script>
</body>
</html>
```

```
# 99 Bottles of Beer
# Brainfuck version
# by Michal Wojciech Tarnowski
```

http://99-bottles-of-beer.net/

```
+>+++++++[>>>+++
 +++++<<<<+++++
+++>-]+++++++++>>>
 +>>++++++++++[>+
  +++++++++<-]>[
  >>+>+>+>+>+>+>
  +>>+>+>>+>+>+>
  +>>+>+>>+>+>+>
  >+>+>+>+>>>>>+
  >+>+>+>>+>+>+>
  >+>+>+>+>>+>+>
 +>>+>+>+>+>>+>+>
 >+>+>+>+>+>+>>>>
 +>+>>+>+>+>+<<<<
 <<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<
-]<++++[>+++++++++<
-]>[>+>>>>>>>>>+>>>
 +>>>>>+>>>+>>>>+>>
>>>+>+>>+>>>>>+>>>>+
>>>>>+>>>>+>>>>>+>>>
+>>>>>>>>+>+>+>>>+>>>
>>+<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<-]+++++[
>>>++>+++>+++>++>>+++>>++>
>>>>>+++>>+++>++>>+++>+>>>+++
+>->++>++>++>+++>++>>--->->+>>>+
+>++>>>>++>++++>++>>->++>>>++>->++
+>+++>>+>+++>>>+++>++>+++>++>>>++>>++>
++>>++>++>+++<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<-]>>-->+>+>+>-->+>>>+>++>>-->+>+>->>+>>>
+>->+>>>->++>-->-->++>->>>->++>++>+>>+>>>+>>>+
>->>>++>>>>>++>++>>>>>+>>++>->+>++>>>>+++>>>+>>
->+>->+<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<[>>>>++
++++++++[->[-]+>[-]<<[<<<<<.>>>>.>>>>.>.>.>.>.>.
>.>.>.>.>.>.>.>.>.>.>.>.>.>.>.>.>.>.>.>.>.>.>.>>
>.<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<.>>>>.>
>>>.>.>.>.>.>.>.>.>.>.>.>.>.>.>.>.>.>>>>>>>>>>>>>>
```

# What will you learn?

- A better understanding of what a programming language actually is
- Ways of programming that you might not have met so far
  - Functional programming (e.g. ML/Haskell/Elm)
  - Logical programming (e.g. Prolog)
  - → Allows you to choose the language that suits a given problem best
- General mechanisms of most programming languages
  - in order to understand what they can be used for and how they are implemented;
  - in order to compare and evaluate (coming) languages, e.g.
    - Expressivenes versus efficiency versus safety
    - Static versus dynamic analysis of programs
  - in order to be able to design languages yourself!

# Why so many programming languages?

- Why not just make *one* good language for all kinds of purposes?

- IBM tried in 1964 with PL/I, known for allowing
  - IF THEN = ELSE THEN ELSE = THEN+1 ELSE THEN = ELSE;

- Good reasons that there are many languages:
  - Problems are different in size, complexity, target platform, etc, and belong to different domains.
  - Different requirements to speed, space and security, . . .
  - Programmers are different!
  - Computer science is still relatively young – we still learn new stuff all the time.
    - This is an exciting time to be involved in programming languages!

000000 00001 00010 00110 00000 100000
100011 00011 01000 00000 00001 000100
000010 00000 00000 00000 10000 000000

What does this program do?

Add registers 1 and 2, and place the result in register 6:

```
[  op  |  rs |  rt |  rd |shamt| funct]
    0     1     2     6     0     32     decimal
 000000 00001 00010 00110 00000 100000   binary
```

Load a value into register 8, taken from the memory cell 68 cells after the location listed in register 3:

```
[  op  |  rs |  rt | address/immediate]
   35     3     8           68          decimal
 100011 00011 01000 00000 00001 000100   binary
```

Jumping to the address 1024:

```
[  op  |         target address        ]
    2                 1024              decimal
 000010 00000 00000 00000 10000 000000   binary
```

https://en.wikipedia.org/wiki/Machine_code

```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2


C000                    ORG     ROM+$0000 BEGIN MONITOR
C000 8E 00 70   START   LDS     #STACK

                ************************************
                * FUNCTION: INITA - Initialize ACIA
                * INPUT: none
                * OUTPUT: none
                * CALLS: none
                * DESTROYS: acc A

0013            RESETA  EQU     %00010011
0011            CTLREG  EQU     %00010001

C003 86 13      INITA   LDA A   #RESETA   RESET ACIA
C005 B7 80 04           STA A   ACIA
C008 86 11              LDA A   #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A   ACIA

C00D 7E C0 F1           JMP     SIGNON    GO TO START OF MONITOR

                ************************************
                * FUNCTION: INCH - Input character
                * INPUT: none
                * OUTPUT: char in acc A
                * DESTROYS: acc A
                * CALLS: none
                * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04   INCH    LDA A   ACIA      GET STATUS
C013 47                 ASR A             SHIFT RDRF FLAG INTO CARRY
C014 24 FA              BCC     INCH      RECIEVE NOT READY
C016 B6 80 05           LDA A   ACIA+1    GET CHAR
C019 84 7F              AND A   #$7F      MASK PARITY
C01B 7E C0 79           JMP     OUTCH     ECHO & RTS

                ************************************
                * FUNCTION: INHEX - INPUT HEX DIGIT
                * INPUT: none
                * OUTPUT: Digit in acc A
                * CALLS: INCH
                * DESTROYS: acc A
                * Returns to monitor if not HEX input

C01E 8D F0      INHEX   BSR     INCH      GET A CHAR
C020 81 30              CMP A   #'0       ZERO
C022 2B 11              BMI     HEXERR    NOT HEX
C024 81 39              CMP A   #'9       NINE
C026 2F 0A              BLE     HEXRTS    GOOD HEX
C028 81 41              CMP A   #'A
C02A 2B 09              BMI     HEXERR    NOT HEX
C02C 81 46              CMP A   #'F
C02E 2E 05              BGT     HEXERR
C030 80 07              SUB A   #7        FIX A-F
C032 84 0F      HEXRTS  AND A   #$0F      CONVERT ASCII TO DIGIT
C034 39                 RTS

C035 7E C0 AF   HEXERR  JMP     CTRL      RETURN TO CONTROL LOOP
```

Assembler – *somewhat* easier to understand (for humans)

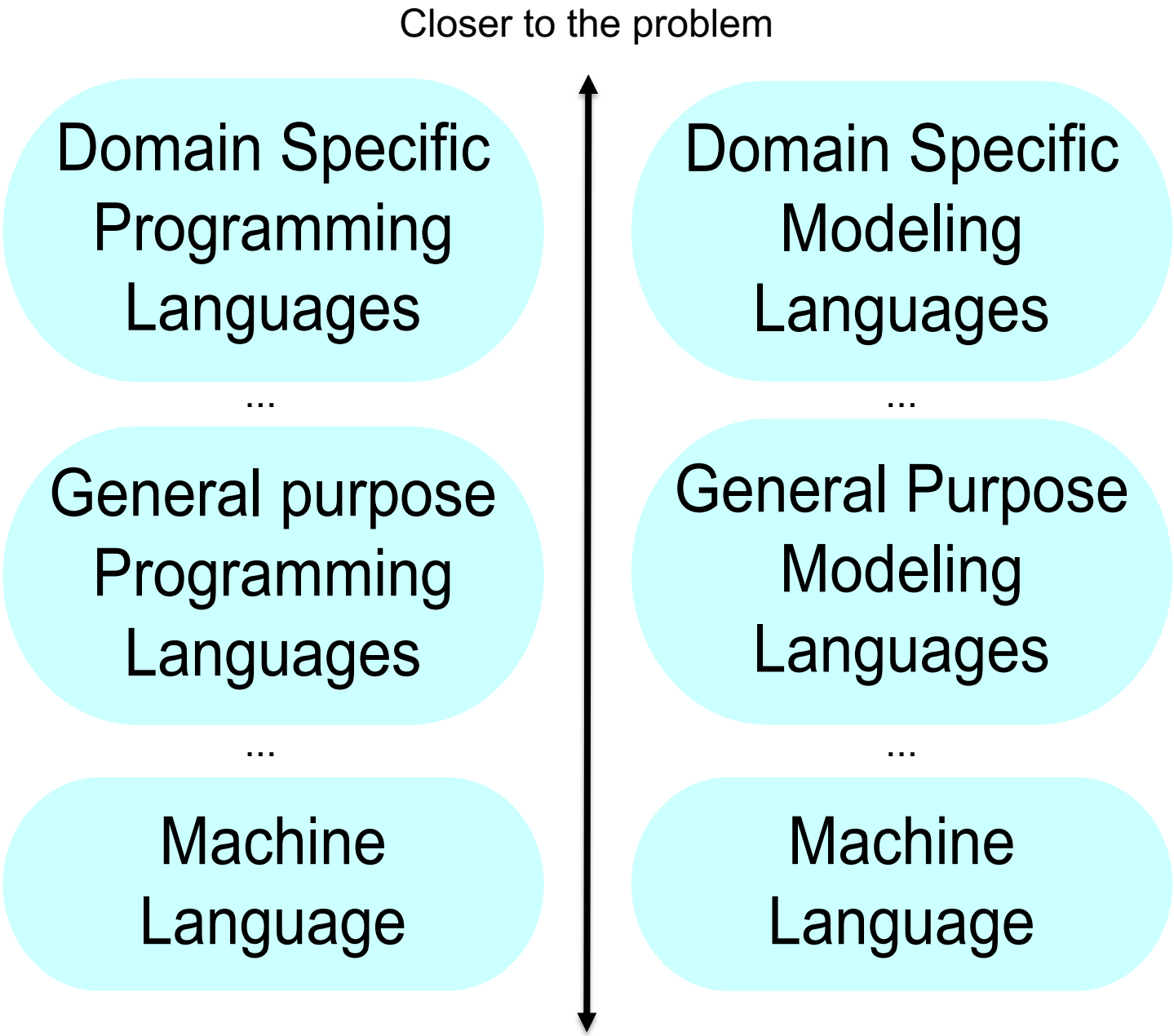
*An assembly language listing for a Motorola 6800 8-bit microprocessor.*

*This is a page from a "Monitor" program that communicates to a serial terminal [...]*

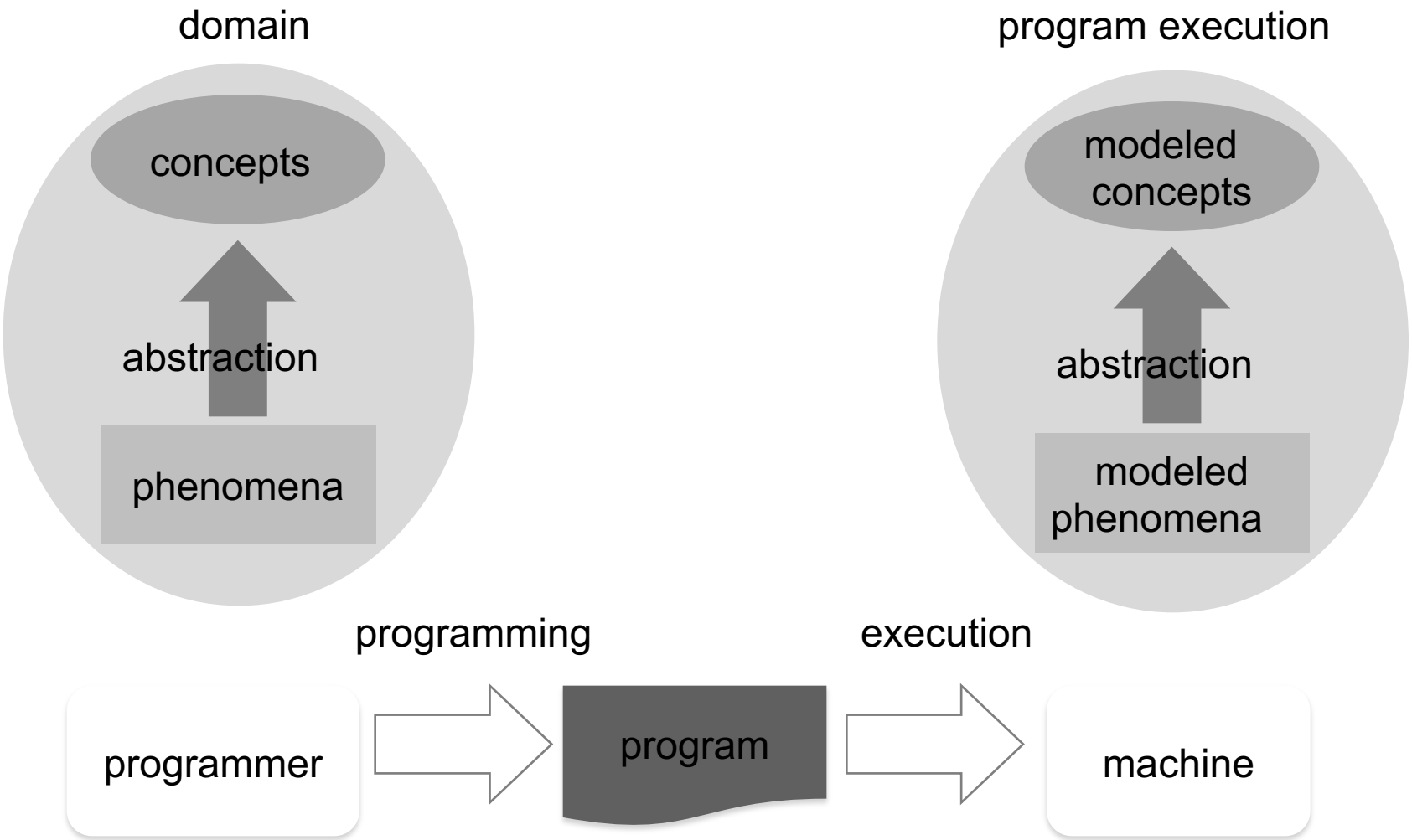
From WikiMedia Commons: https://commons.wikimedia.org/wiki/File:Motorola_6800_Assembly_Language.png

There is a tension between the different desiderata for a language

- Secure ←→ Fast
- Easily understandable ←→ Expressive
- Expressive ←→ Fast
- Expressive ←→ Safe
- General ←→ Domain specific
- Etc

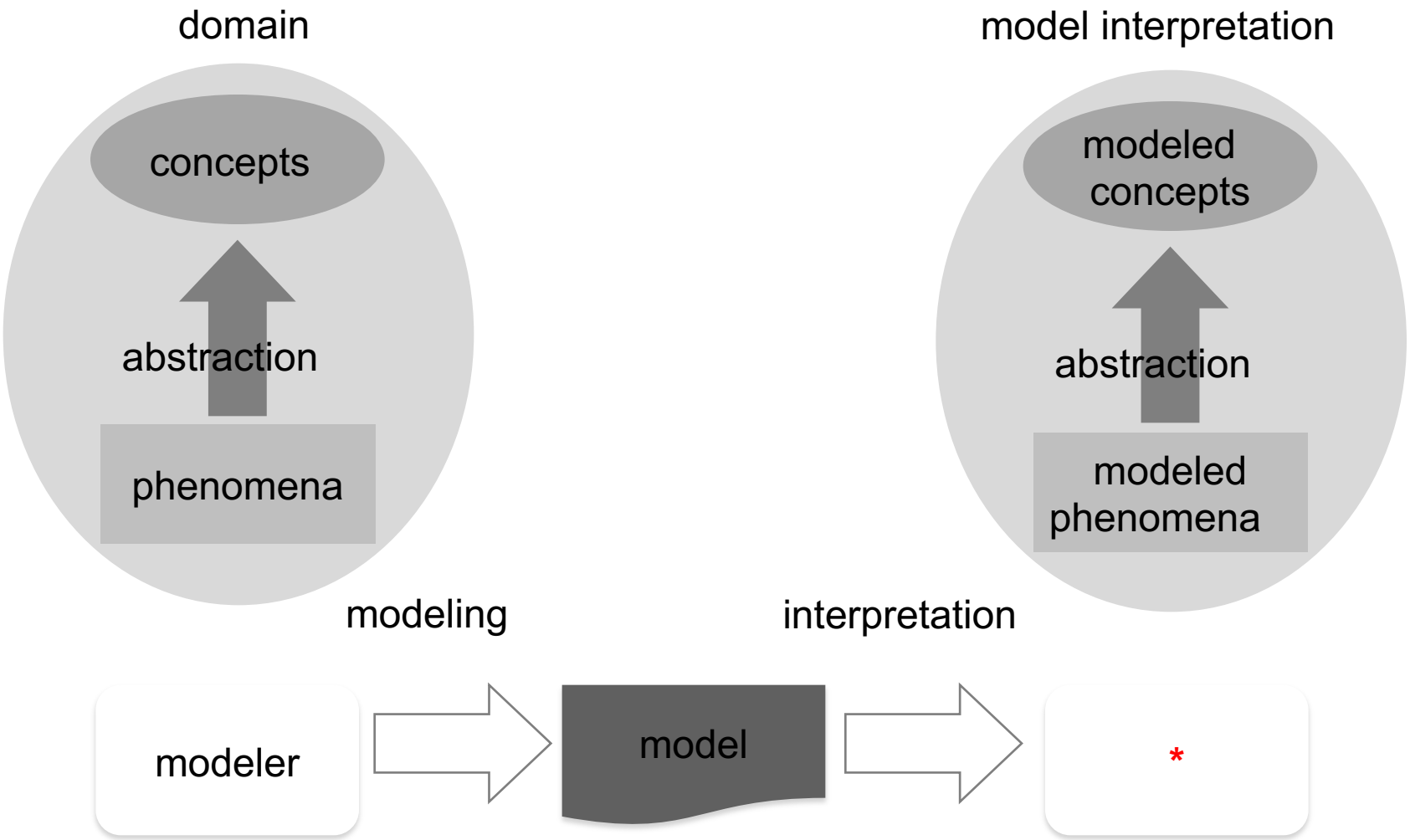

Closer to the problem
Domain Specific Programming Languages
...
General purpose Programming Languages
...
Machine Language
Domain Specific Modeling Languages
...
General Purpose Modeling Languages
...
Machine Language
Closer to the machine

# Programming




Programming: to understand a domain
- and make a machine have the same ”understanding”

# Modeling




Modelling: to understand a domain
- and make a ? have the same understanding

# Paradigms/perspectives

- Procedural/Imperative Programming
  - A program execution is regarded as a sequence of operations manipulating a set of data items
- Functional Programming
  - A program is regarded as a mathematical function
- Constraint-Oriented/Declarative (Logic) Programming
  - A program is regarded as a set of equations describing relations
- Object-Oriented Programming
  - A program execution is regarded as a model simulating a real or imaginary part of the world, with objects corresponding to real-world things or processes.
    (The so-called **Scandinavian** approach to object oriented programming)

- Most languages are a mix of several paradigms

# Curriculum

- Text book
  - John C Mitchell: Concepts in Programming Languages, 2003. Cambridge University Press. Isbn:0521780985.
  - Additional material, see course site
  - The slides are part of the curriculum!
    - Available from the course page
- Weekly exercises
  - Available from the course page
- Mandatory assignments
  - 2 mandatory assignments (most likely)
  - Solving the same problem with both functional and object oriented programming