The Book of Magne

INF222 Crashcourse - v2022

Sander Wiig

Institute for Informatics
University of Bergen

19 Mai 2022

UNIVERSITY OF BERGEN
Faculty of Mathematics and Natural Scien

Script and Presentation

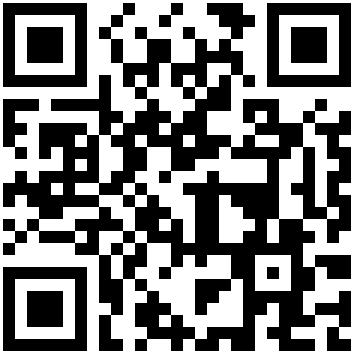


Figure: https://tinyurl.com/book-of-magne

What is a programming language?

A programming language

- is an artificial language(i.e made by us humans on purpose)

What is a programming language?

A programming language

- is an artificial language(i.e made by us humans on purpose)
- is used to tell machines what to do

Grouping Languages

- Many different ways of grouping

Grouping Languages

- Many different ways of grouping
- Ex. purpose, typing, paradigm, generality vs. specificity

Grouping Languages

- Many different ways of grouping
- Ex. purpose, typing, paradigm, generality vs. specificity
- Look at; generality vs. specificity, and paradigm

Generality vs. Specificity

Languages are usually grouped into two categories when based on their specificity

- DSLs, small, targeted at specific problems. Internal/embedded vs external

Generality vs. Specificity

Languages are usually grouped into two categories when based on their specificity

- DSLs, small, targeted at specific problems. Internal/embedded vs external
- GPLs, large, many uses.

Generality vs. Specificity

Characteristic	DSL	GPL
Domain	Small, Well-defined	Large, Not specialized
Size	Small	Large
Lifespan	Weeks, Months	Years, Decades

Paradigms

- Imperative Languages

Paradigms

- Imperative Languages
- Functional Languages

Paradigms

- Imperative Languages
- Functional Languages
- OOP languages

Paradigms

- Imperative Languages
- Functional Languages
- OOP languages
- Logic languages

Syntax and Semantics

All languages consist of two parts

- Syntax

Syntax and Semantics

All languages consist of two parts

- Syntax
- Semantics

The Interpreter Process

Interpreters differ from Compilers since they execute code instead of translating it.

- Lexical Analyzer

The Interpreter Process

Interpreters differ from Compilers since they execute code instead of translating it.

- Lexical Analyzer
- Syntax Analyzer

The Interpreter Process

Interpreters differ from Compilers since they execute code instead of translating it.

- Lexical Analyzer
- Syntax Analyzer
- Semantic Analyzer

The Interpreter Process

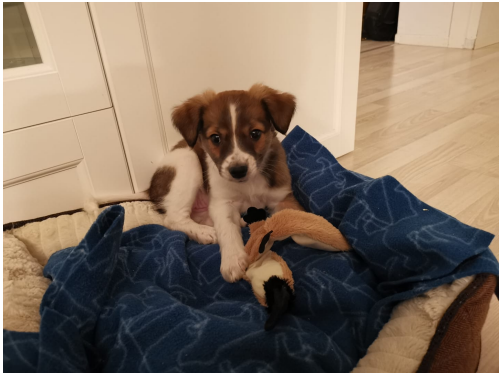
Interpreters differ from Compilers since they execute code instead of translating it.

- Lexical Analyzer
- Syntax Analyzer
- Semantic Analyzer
- Evaluator

Meta Programming

A metaprogram is a program that works on *other* programs.
When you see a data structure like BSL or BIPL in Haskell it represents those programs.

Questions?



Abstract Syntax Trees

ASTs represent the abstract syntax of a program. Abstract means that unnecessary tokens are removed, ex. parenthesis, curly braces.
It's produced by the Syntax Analyzer.
Often used as an intermediary representation by the interpreter.

Sum of Products

```
| B Bool
| C
```

Sum of Products

$$\underbrace{(\text{Bool} \times \text{Bool} \times \text{Bool})}_{A} + \underbrace{\text{Bool}}_{B} + \underbrace{1}_{C}$$

The Bool type can take on 2 different values (False and True), so the A constructor can construct $2 * 2 * 2 = 8$ different values.
the total numbers of values of type SomeType is $8 + 2 + 1 = 11$, as the data type is the sum of the three products we've just described.

Type Checking

Typing falls into two categories

- Static

Type Checking

Typing falls into two categories

- Static
- Dynamic

Type Checking

Example!

Wellformedness

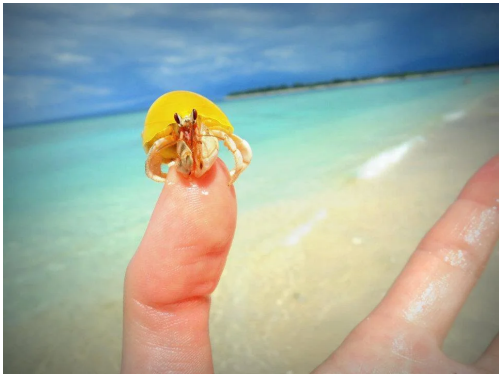
For a program to be Wellformedit needs to satisfy all the constraints(kinda like rules) on it. This means that the program follows all the rules for it like;

- The program conforms to the AST
- Typed correctly
- all procs/declarations are wellformed
- other things

Type Inference

Some languages don't specify the type of expressions and therefore need to be inferred. Example!

Questions?



Store

A store lets us "store"[1] things.
Usually array of bytes or values(depends on abstraction level)
Items are stored at specific index.
Usually grows upwards.

[1] duh

Variables and Enviroment

Variables associate names with storage locations.
Environments are the map of variables.
Kinda like a phonebook.

Scoping

A scope is an overview of what exists when
Usually talk about the scope of something like a var.
Describes when it exists.
Two types

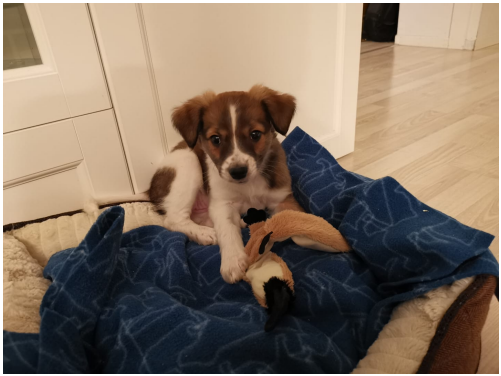
- Static/Lexical - Determined at compiletime

Scoping

A scope is an overview of what exists when
Usually talk about the scope of something like a var.
Describes when it exists.
Two types

- Static/Lexical - Determined at compiletime
- Dynamic - Determined at runtime

Questions?



Lykke til på eksamen!

Takk for oss :)