Programming Languages: Lecture 5
Functors and Compiling

Rishabh Dhiman

13 January 2022

1 Functors

Definition 1 (Functor)**.** A *functor* is a structure that takes other structures as parameters and yields a new structure,

1. A functor can be applied to argument structures to yield a new structure.
2. A functor can be applied only to structures that match certain signature constraints.
3. Functors may be used to test existing structures or to create new structures.
4. Functors may be used to express generic algorithms.

2 Compiling

In general, a compiler/interpreter for a source language $\mathcal{S}$ written in some language $\mathcal{C}$ translates code written in $\mathcal{S}$ to a target $\mathcal{T}$.

2.1 The Compiling Process

Besides $\mathcal{S}, \mathcal{C}, \mathcal{T}$, there could be several other intermediate lanauages, $\mathcal{I}_1, \mathcal{I}_2, \ldots$ called *intermediate representations*, int which the source program could be translated in the process of compiling or interpreting the source programs written in $\mathcal{S}$.

Some of these intermediate representations could just be datatypes of OOPs languages.

2.2 Phases of a Compiler

The phases may be different from the various passes in compilation.

↓ stream of characters
Scanner (Lexical Analyzer)
↓ stream of tokens
Parser (Syntactic Analyzer)
↓ parse tree (syntax tree)
Semantic Analyzer – Till here was the frontend of the compiler
↓ abstract syntax tree
I.R. Code Generator
↓ intermediate representation
Optimizer
↓ optimized intermediate representation
Code Generator
↓ target code

Symbol table manager on right, and Error-handler on left.