

Lorem Ipsum

31. März 2021

Teil I Examples

1 Type Inference

Types can be inferred by a proof-like system with the following core rules:

$$\frac{}{\dots, x : \text{Int}, \dots \vdash x :: \text{Int}} \text{Var} \quad \frac{\Gamma, x : \sigma \vdash t :: \tau}{\Gamma \vdash (\lambda x. t) :: \sigma \rightarrow \tau} \text{Abs} \quad \frac{\Gamma \vdash t_1 :: \sigma \rightarrow \tau \quad \Gamma \vdash t_2 :: \sigma}{\Gamma \vdash (t_1 \ t_2) :: \tau} \text{App}$$

1. Start with $\vdash t :: \tau_0$
2. Build derivation tree bottom up, collect constraints
3. Solve constraints.

2 Types

- **Base Types:** Double, ...
- **Compound Types:** Lists, Tuples, ...
- **Type synonyms:**
 type Complex = (Double, Double)
- **Type Classes**, have Instances, offer restricted form of polymorphism. Similar to Interfaces. E.g the type class Eq represents a set of Types.
- **Algebraic Types**, similar to structs.
 - Enumeration Types
 - Product Types

In data Tree = Leaf Int | Node Tree Tree
 Tree is a algebraic Type, Leaf is a constructor.

3 Lazy Evaluation

In Haskell, substitution occurs without argument evaluation. Furthermore terms are only evaluated as far as needed, e.g for pattern matching. This allows **data-driven computation**: Simoulta-neous generation and processing of data on demand (similar to coroutines). E.g consider:

```

primes = sieve [2 ..]
sieve xs = head xs : sieve (dropMults (head xs)
                                   (tail xs))
  
```