Lorem Ipsum

2. Juni 2021

Teil I
Examples

1 Math

Consider the formula

$$p = (1 - \lambda) \cdot \lambda$$

but ignore it.

2 Type Inference

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

$$\frac{}{\ldots, x : \overline{\tau}, \ldots \vdash x :: \overline{\tau}}\ Var \qquad \frac{\Gamma, x : \sigma \vdash t :: \tau}{\Gamma \vdash (\lambda x.\, t) :: \sigma \to \tau}\ Abs \qquad \frac{\Gamma \vdash t_1 :: \sigma \to \tau \qquad \Gamma \vdash t_2 :: \sigma}{\Gamma \vdash (t_1\ t_2) :: \tau}\ App$$

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

3 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.

4 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**: Simoultaneous 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))
```