TECHNISCHE UNIVERSITÄT DRESDEN

# PROGRAMMIERUNG

## Übung 4: Haskell – Typpolymorphie & Unifikation

**Eric Kunze**
eric.kunze@mailbox.tu-dresden.de

TU Dresden, 1. Mai 2019

# Unifikation

**Verlegung an Himmelfahrt:**
von Do, 30.05., 1. DS auf **Mittwoch, 29.05., 1.DS, APB E001**

**Motivation:** Typüberprüfung

```
f :: (t, Char) -> (t, [Char])
f (...) = ...

g :: (Int, [u]) -> Int
g (...) = ...

h = g . f
```

Beide Typausdrücke können in Übereinstimmung gebracht werden, wenn die Typterme $trans($`(t,[Char])`$)$ und $trans($`((Int, [u]))`$)$ unifizierbar sind

→ `t = Int` und `u = Char`

# Unifikationsalgorithmus

- **Ziel.** Am Ende sollen nur paarweise verschiedene Variablen in der oberen Zeile stehen.
- **beliebter Fehler.** Verwechslung von Elimination von Variablen $(x_i, x_i)$ und Dekomposition von nullären Symbolen $(\alpha, \alpha)$.

# Unifikationsalgorithmus – Regeln

- **Dekomposition.** Sei $\delta \in \Sigma$ ein $k$-stelliger Konstruktor, $s_1, \ldots, s_k, t_1, \ldots, t_k$ Terme über Konstruktoren und Variablen.

$$\begin{pmatrix} \delta(s_1, \ldots, s_k) \\ \delta(t_1, \ldots, t_k) \end{pmatrix} \quad \rightsquigarrow \quad \begin{pmatrix} s_1 \\ t_1 \end{pmatrix}, \ldots, \begin{pmatrix} s_k \\ t_k \end{pmatrix}$$

- **Elimination.**Sei $x$ eine Variable !

$$\begin{pmatrix} x \\ x \end{pmatrix} \quad \rightsquigarrow \quad \emptyset$$

- **Vertauschung.** Sei $t$ keine Variable.

$$\begin{pmatrix} t \\ x \end{pmatrix} \quad \rightsquigarrow \quad \begin{pmatrix} x \\ t \end{pmatrix}$$

- **Substitution.** Sei $x$ eine Variable, $t$ keine Variable und $x$ kommt nicht in $t$ vor (occur check). Dann ersetze in jedem anderen Term die Variable $x$ durch $t$.