

Interaktive Übungen zu Grundlagen von Programmiersprachen

Johannes Waldmann
HTWK Leipzig, Germany

September 22, 2015

Johannes Waldmann HTWK Leipzig, Germany Interaktive Übungen zu Grundlagen von Programmiersprachen September 22, 2015 1 / 7

Aufgabe zur (einfachen) Typisierung

Gesucht ist ein Ausdruck vom Typ Birne in der Signatur

```
Pflaume d;  
static Birne a (Apfel x, Apfel y, Pflaume z );  
static Apfel b (Birne x );  
static Apfel c (Pflaume x, Pflaume y, Pflaume z  
a ( c ( d, d, d ), c ( d, d, d ), d )
```

- ▶ textuelle Eingabe (nicht: GUI)
- ▶ sofortige semantische Kontrolle (nicht: syntaktischer Vergleich mit Musterlösung)

```
a ( b ( a ( c ( d, d, d ), c ( d, d, d ), d ))  
c ( d, d, d ), d )
```

Johannes Waldmann HTWK Leipzig, Germany Interaktive Übungen zu Grundlagen von Programmiersprachen September 22, 2015 2 / 7

Erzeugen von Aufgabeninstanzen

```
Conf { max_arity = 3  
      , types = [ Apfel, Birne, Pflaume ]  
      , min_symbols = 4, max_symbols = 4  
      , min_size = 6, max_size = 10 }
```

- ▶ deterministischer Pseudozufallsgenerator
- ▶ würfelt Signatur
- ▶ bestimme endlichen Baum-Automaten für Menge der korrekt getypten Terme
- ▶ bestimme kleinste Elemente

Johannes Waldmann HTWK Leipzig, Germany Interaktive Übungen zu Grundlagen von Programmiersprachen September 22, 2015 3 / 7

Aufgabe zu polymorphen Typen

Gesucht ist ein Ausdruck vom Typ

```
Fozzie<Kermit, Kermit>  
in der Signatur class S {  
  static <T2> Piggy<Piggy<Animal>>  
    statler ( Piggy<T2> x , Piggy<T2> y );  
  static <T2> Kermit waldorf ( Piggy<T2> x );  
  static Piggy<Fozzie<Animal, Animal>> bunsen ( );  
  static <T2, T1> T1  
    chef ( Piggy<Piggy<T2>> x , Piggy<Piggy<T1>>  
  static <T2> Fozzie<Kermit, T2>  
    rowlf ( T2 x, Animal y );  
}  
S.<Kermit>rowlf  
(S.<Fozzie<Animal, Animal>>waldorf  
(S.bunsen()), ...
```

Johannes Waldmann HTWK Leipzig, Germany Interaktive Übungen zu Grundlagen von Programmiersprachen September 22, 2015 4 / 7

Prinzipien von Programmiersprachen

Aufgaben zu den Bereichen

- ▶ (konkrete) Syntax (reg. Ausdrücke, CFG)
- ▶ Semantik
 - ▶ statisch (Typen)
 - ▶ dynamisch
 - ▶ operational (While-Programme, ...)
 - ▶ denotational (Gleichungssysteme, ...)

im autotool-Rahmen: Aufgabentyp = DSL

- ▶ konkrete Syntax (spezifisch oder generisch)
- ▶ abstrakte Syntax = algebraischer Datentyp
- ▶ Semantik = Interpreter (führt Probe durch)
- ▶ Instanzen-Generator

Johannes Waldmann HTWK Leipzig, Germany Interaktive Übungen zu Grundlagen von Programmiersprachen September 22, 2015 5 / 7

Bemerkungen zur Implementierung

entscheidend sind (auch hier) die Typen

- ▶ typisierte Texteingaben
- ▶ Kombinator-Parser (<http://hackage.haskell.org/package/parsec/>)
 - ▶ automatische Fehlermeldungen
 - ▶ ggf. durch compile-time reflection erzeugt
- ▶ automatisch mit URL zu API-Doc des abstrakten Syntaxtyps
- ▶ von dort Link to echten Quelltexten ...
- ▶ ... diese sind die wörtliche Realisierung des Vorlesungsskriptes (z.B. Semantik von While)

Johannes Waldmann HTWK Leipzig, Germany Interaktive Übungen zu Grundlagen von Programmiersprachen September 22, 2015 6 / 7

Thesen zum Design

- ▶ Nur die semantische Bewertung (objektive Probe) realisiert den wissenschaftlichen Anspruch des Studiums ...
- ▶ ... und ist eine Grundlage für die intelligente Erzeugung von personalisierten Aufgabeninstanzen.
- ▶ Eine ausschließlich textuelle Eingabe mit uniformer Syntax unterstreicht die Rolle der mathematischen Notation in der Informatik.

*Was sich überhaupt sagen läßt,
läßt sich klar sagen.* (Wittgenstein 1921)

Johannes Waldmann HTWK Leipzig, Germany Interaktive Übungen zu Grundlagen von Programmiersprachen September 22, 2015 7 / 7