Genauer, schneller, geduldiger, preiswerter als studentische Hilfskräfte!

Automatisierte Übungen zu Prinzipien von Programmiersprachen

Johannes Waldmann, HTWK Leipzig

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 )
```


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 Genauer, schneller, geduldiger, preiswerter 23. Februar 2017 2 / 1

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

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()), ...
```


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 Genauer, schneller, geduldiger, preiswerter 23. Februar 2017 4 / 1

Aufgabe zu Frames

- UP-Aufruf erzeugt Aktivationsverbund (Frame)
- jeder Frame hat Vorgänger:
  - dynamisch: zu Frame des Aufrufenden (benutzt für Rückkehr)
  - statisch: zu Frame des textuell umgebenden UP (benutzt für Zugriff auf fremde lokale Namen)
- Aufgabe:
  - gegeben: stat. und dyn. Graph G
  - gesucht: Programmtext, bei dem G entsteht
- welche G sind überhaupt realisierbar?

Prinzipien von Programmiersprachen

Aufgaben zu den Bereichen

- (konkrete) Syntax (reg. Ausdrücke, CFG)

Prinzipien von Programmiersprachen

Aufgaben zu den Bereichen

- (konkrete) Syntax (reg. Ausdrücke, CFG)
- Semantik

Prinzipien von Programmiersprachen

Aufgaben zu den Bereichen

- (konkrete) Syntax (reg. Ausdrücke, CFG)
- Semantik
  - statische (Typen)

Prinzipien von Programmiersprachen

Aufgaben zu den Bereichen

- (konkrete) Syntax (reg. Ausdrücke, CFG)
- Semantik
  - statische (Typen)
  - dynamische

Prinzipien von Programmiersprachen

Aufgaben zu den Bereichen

- (konkrete) Syntax (reg. Ausdrücke, CFG)
- Semantik
  - statische (Typen)
  - dynamische
    - operational (Frames, ...)

Prinzipien von Programmiersprachen

Aufgaben zu den Bereichen

- (konkrete) Syntax (reg. Ausdrücke, CFG)
- Semantik
  - statische (Typen)
  - dynamische
    - operational (Frames, ...)
    - denotational (Spursprachen, ...)

Prinzipien von Programmiersprachen

Aufgaben zu den Bereichen

- (konkrete) Syntax (reg. Ausdrücke, CFG)
- Semantik
  - statische (Typen)
  - dynamische
    - operational (Frames, ...)
    - denotational (Spursprachen, ...)

Prinzipien von Programmiersprachen

Aufgaben zu den Bereichen

- (konkrete) Syntax (reg. Ausdrücke, CFG)
- Semantik
  - statische (Typen)
  - dynamische
    - operational (Frames, ...)
    - denotational (Spursprachen, ...)

im autotool-Rahmen: Aufgabentyp = DSL

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

Prinzipien von Programmiersprachen

Aufgaben zu den Bereichen

- (konkrete) Syntax (reg. Ausdrücke, CFG)
- Semantik
  - statische (Typen)
  - dynamische
    - operational (Frames, . . .)
    - denotational (Spursprachen, . . .)

im autotool-Rahmen: Aufgabentyp = DSL

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

Prinzipien von Programmiersprachen
Aufgaben zu den Bereichen
- (konkrete) Syntax (reg. Ausdrücke, CFG)
- Semantik
  - statische (Typen)
  - dynamische
    - operational (Frames, . . .)
    - denotational (Spursprachen, . . .)
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 Genauer, schneller, geduldiger, preiswerter 23. Februar 2017 6 / 1

Prinzipien von Programmiersprachen
Aufgaben zu den Bereichen
- (konkrete) Syntax (reg. Ausdrücke, CFG)
- Semantik
  - statische (Typen)
  - dynamische
    - operational (Frames, . . .)
    - denotational (Spursprachen, . . .)
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 Genauer, schneller, geduldiger, preiswerter 23. Februar 2017 6 / 1

Bemerkungen zur Implementierung

entscheidend sind (auch hier) die Typen

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

Bemerkungen zur Implementierung
entscheidend sind (auch hier) die Typen
- typisierte Texteingaben
- Kombinator-Parser (http://hackage.haskell.org/package/parsec/)
  - automatische Fehlermeldungen
  - Parser ggf. automatisch erzeugt (TH)
- automatisch mit URL zu API-Doc des abstraken Syntaxtyps
- von dort Link to echten Quelltexten . . .
- . . . diese sind die wörtliche Realisierung des Vorlesungsskriptes (z.B. Semantik von While)
Johannes Waldmann, HTWK Leipzig Genauer, schneller, geduldiger, preiswerter 23. Februar 2017 7 / 1

Bemerkungen zur Implementierung
entscheidend sind (auch hier) die Typen
- typisierte Texteingaben
- Kombinator-Parser (http://hackage.haskell.org/package/parsec/)
  - automatische Fehlermeldungen
  - Parser ggf. automatisch erzeugt (TH)
- automatisch mit URL zu API-Doc des abstraken Syntaxtyps
- von dort Link to echten Quelltexten . . .
- . . . diese sind die wörtliche Realisierung des Vorlesungsskriptes (z.B. Semantik von While)
Johannes Waldmann, HTWK Leipzig Genauer, schneller, geduldiger, preiswerter 23. Februar 2017 7 / 1

Bemerkungen zur Implementierung
entscheidend sind (auch hier) die Typen
- typisierte Texteingaben
- Kombinator-Parser (http://hackage.haskell.org/package/parsec/)
  - automatische Fehlermeldungen
  - Parser ggf. automatisch erzeugt (TH)
- automatisch mit URL zu API-Doc des abstraken Syntaxtyps
- von dort Link to echten Quelltexten . . .
- . . . diese sind die wörtliche Realisierung des Vorlesungsskriptes (z.B. Semantik von While)
Johannes Waldmann, HTWK Leipzig Genauer, schneller, geduldiger, preiswerter 23. Februar 2017 7 / 1

Bemerkungen zur Implementierung
entscheidend sind (auch hier) die Typen
- typisierte Texteingaben
- Kombinator-Parser (http://hackage.haskell.org/package/parsec/)
  - automatische Fehlermeldungen
  - Parser ggf. automatisch erzeugt (TH)
- automatisch mit URL zu API-Doc des abstraken Syntaxtyps
- von dort Link to echten Quelltexten . . .
- . . . diese sind die wörtliche Realisierung des Vorlesungsskriptes (z.B. Semantik von While)
Johannes Waldmann, HTWK Leipzig Genauer, schneller, geduldiger, preiswerter 23. Februar 2017 7 / 1

Bemerkungen zur Implementierung

entscheidend sind (auch hier) die Typen

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

Bemerkungen zur Implementierung

entscheidend sind (auch hier) die Typen

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

Das wollen Sie auch haben?

- meine autotool-Instanz benutzen
  - anonym, keine Speicherung: sofort möglich
  - pseudonym: in Vorbereitung (shibboleth, DFN-AAI)
- eine eigene Instanz betreiben
- Integration in andere E-Learning-Systeme? QTI-Spec. erweitern um *externe* Semantik-Server (bisher nur *interne* Auswertung für multiple choice)

Das wollen Sie auch haben?

- meine autotool-Instanz benutzen
  - anonym, keine Speicherung: sofort möglich
  - pseudonym: in Vorbereitung (shibboleth, DFN-AAI)
- eine eigene Instanz betreiben
- Integration in andere E-Learning-Systeme? QTI-Spec. erweitern um *externe* Semantik-Server (bisher nur *interne* Auswertung für multiple choice)

Das wollen Sie auch haben?

- meine autotool-Instanz benutzen
  - anonym, keine Speicherung: sofort möglich
  - pseudonym: in Vorbereitung (shibboleth, DFN-AAI)
- eine eigene Instanz betreiben
- Integration in andere E-Learning-Systeme? QTI-Spec. erweitern um *externe* Semantik-Server (bisher nur *interne* Auswertung für multiple choice)

Das wollen Sie auch haben?

- meine autotool-Instanz benutzen
  - anonym, keine Speicherung: sofort möglich
  - pseudonym: in Vorbereitung (shibboleth, DFN-AAI)
- eine eigene Instanz betreiben
  - stand-alone, front-end: klassisch (jetzt) oder yesod (in Vorbereitung)
  - als Teil von open-olat
- Integration in andere E-Learning-Systeme? QTI-Spec. erweitern um *externe* Semantik-Server (bisher nur *interne* Auswertung für multiple choice)

Das wollen Sie auch haben?

- meine autotool-Instanz benutzen
  - anonym, keine Speicherung: sofort möglich
  - pseudonym: in Vorbereitung (shibboleth, DFN-AAI)
- eine eigene Instanz betreiben
  - stand-alone, front-end: klassisch (jetzt) oder yesod (in Vorbereitung)
  - als Teil von open-olat
- Integration in andere E-Learning-Systeme? QTI-Spec. erweitern um *externe* Semantik-Server (bisher nur *interne* Auswertung für multiple choice)

Das wollen Sie auch haben?

- meine autotool-Instanz benutzen
  - anonym, keine Speicherung: sofort möglich
  - pseudonym: in Vorbereitung (shibboleth, DFN-AAI)
- eine eigene Instanz betreiben
  - stand-alone, front-end: klassisch (jetzt) oder yesod (in Vorbereitung)
  - als Teil von open-olat
- Integration in andere E-Learning-Systeme? QTI-Spec. erweitern um *externe* Semantik-Server (bisher nur *interne* Auswertung für multiple choice)

Das wollen Sie auch haben?

- meine autotool-Instanz benutzen
  - anonym, keine Speicherung: sofort möglich
  - pseudonym: in Vorbereitung (shibboleth, DFN-AAI)
- eine eigene Instanz betreiben
  - stand-alone, front-end: klassisch (jetzt) oder yesod (in Vorbereitung)
  - als Teil von open-olat
- Integration in andere E-Learning-Systeme? QTI-Spec. erweitern um *externe* Semantik-Server (bisher nur *interne* Auswertung für multiple choice)

Haskell in Leipzig HaL-10

`http://nfa.imn.htwk-leipzig.de/HAL2015/`

- **bis 2. November: Einreichung von Beiträgen**
- 5. November: Bekanntgabe des Programms
- bis 27. November: Anmeldung
- 4. und 5. Dezember: Workshop

. . . und *nächstes* Jahr:

≈ 12.–14. Sept. 2016: HaL + WFLP + WLP

Haskell in Leipzig HaL-10

`http://nfa.imn.htwk-leipzig.de/HAL2015/`

- **bis 2. November: Einreichung von Beiträgen**
- 5. November: Bekanntgabe des Programms
- bis 27. November: Anmeldung
- 4. und 5. Dezember: Workshop

. . . und *nächstes* Jahr:

≈ 12.–14. Sept. 2016: HaL + WFLP + WLP

Johannes Waldmann, HTWK Leipzig Genauer, schneller, geduldiger, preiswerter 23. Februar 2017 9 / 1