

# Semantische Bewertung und personalisierte Erzeugung von Übungsaufgaben zu Mathematik, Logik, Informatik

Johannes Waldmann (HTWK Leipzig)

September 2, 2014

# Beispiel (Sicht des Studenten)

Gesucht ist ein Hamiltonkreis in

```
Graph { knoten = mkSet
        [ 1 , 2 , 3 , 4 , 5 , 6 , 7
          , 8 , 9 , 10 ]
      , kanten = mkSet
        [ kante 1 2 , kante 1 3
          , kante 1 4 , kante 1 7, ...
```

**Lösungsversuch:**

```
[ 1 , 10 , 2 , 9 , 3 , 8 , 4 , 7, 5 , 6 ]
```

**Bewertung:**

... Knotenfolge bildet Kreis?

Graph enthält nicht: kante 2 10

# Beispiel (Sicht des Tutors)

- ▶ **Server:**

```
http://kernkraft.imn.htwk-leipzig.de/  
cgi-bin/autotool-latest.cgi
```

- ▶ **Task Type: Hamilton-Quiz**

- ▶ **Highscore: Kein**

- ▶ **Status: Mandatory**

- ▶ **Konfiguration:**

```
Config { nodes = 10, edges = 30 }
```

# Was der Student sieht

## Web-Oberfläche

- ▶ zur Auswahl und Bearbeitung von Übungsaufgaben
- ▶ ausführliche Fehlermeldungen sofort
- ▶ Anzeige von Punkten, Highscores

## Formate:

- ▶ Eingabe ist immer textuell  
(Terme, JSON-ähnliche Notation)  
(kein click click, kein WYSIWIG)
- ▶ Ausgabe meist textuell,  
machmal zusätzlich grafisch

# Was der Tutor sieht

## Web-Oberfläche

- ▶ für Auswahl von Aufgaben-Typen
- ▶ Konfiguration von Parametern der jeweiligen Aufgabeninstanz
- ▶ (... bzw. des Instanzen-Generators)
- ▶ Arbeitsfläche wie für Student  
(zum Ausprobieren konfigurierter Aufgaben)
- ▶ Anzeigen von Punkten, Übungsgruppen usw.

# Semantik

Die Semantik eines Aufgabentyps ist gegeben durch

- ▶ eine Funktion zur Bewertung:

`Instanz x Einsendung -> Bool`

- ▶ eine Funktion zur Berechnung einer (syntaktisch korrekten, aber semantisch falschen) Beispiel-Einsendung:

`Instanz -> Einsendung`

- ▶ eine Funktion zum Erzeugen von Instanzen:

`Konfiguration x Matrikelnummer  
-> Instanz`

# Was ist besonders?

- ▶ es wird geprüft, ob die Lösung die *Spezifikation* erfüllt.
- ▶ es wird *nicht* geprüft, ob die Lösung mit einer Musterlösung übereinstimmt.
- ▶ es wird *nicht* geprüft, auf welchem Weg eine Lösung erhalten wurde.
- ▶ es gibt aber Aufgaben, bei denen die Lösung aus einer Folge von Schritten besteht (z.B.: eine Resolutionsableitung der leeren Klausel)
- ▶ für nicht personalisierte Aufgaben:  
Highscore (z.B. kleinste, früheste Lösung)

# Anwendungen

in Lehrveranstaltungen an Uni Leipzig, Uni Halle, Uni Karlsruhe, FH Zwickau, HTWK Leipzig

- ▶ Automaten und Formale Sprachen
- ▶ Berechenbarkeit und Komplexität
- ▶ Prinzipien von Programmiersprachen
- ▶ Diskrete Mathematik und Logik
- ▶ Aussagen- und Prädikatenlogik



# Rechnen mit Mengen

Gesucht ist ein Ausdruck (Term) mit Wert:

$\{1, \{2\}\}$

Der Ausdruck soll höchstens die Größe 12 haben.

Sie dürfen diese Symbole benutzen

zweistellige : [ + , - , & ]

einstellige : [ pow ] , nullstellige : [ ]

und diese vordefinierten Konstanten:

$A = \{1, 3\}$  ,  $B = \{2, 3\}$

**Beispiel: gemeint ist  $2^{A \setminus B} \cup (A \cap B)$**

pow (A - B) + (A & B)

Der Wert Ihres Terms ist  $\{3, \{\}, \{1\}\}$

stimmen die Werte überein?

Nein, die symmetrische Differenzmenge ist

$\{1, 3, \{\}, \{1\}, \{2\}\}$

# Rechnen mit Relationen

(Bsp.) gesucht ist Ausdruck mit Wert

$\{(1, 4), (2, 1), (3, 1), (4, 3), (4, 5), (5, 1)\}$

- ▶ Operationen (zweistellig): Vereinigung  $+$ , Differenz  $-$ , Durchschnitt  $\&$ , Produkt  $\cdot$
- ▶ Operationen (einstellig)  
Inverse, transitive Hülle, reflexive Hülle
- ▶ Konstanten, Bsp:

$R = \{(1, 4), (2, 1), (3, 1), (4, 3), (4, 5)\}$

# Geschichte

- ▶ Entwicklung begann ca. 2000 an Ifl Uni Leipzig  
(ursprünglich mit Mail-Oberfläche)
- ▶ seit 2003 an HTWK Leipzig  
(Web-Schnittstelle)
- ▶ 2009/10: Projekt autOlat  
(Abtrennung Semantik-Server,  
Olat-Schnittstelle)
- ▶ ab 2013: Virtualisierung,  
Shibboleth-Authentifizierung,  
Frontend-Refactoring

# Technische Realisierung

- ▶ die Semantik-Funktionen werden durch einen (zustandslosen) XML-RPC-Server bereitgestellt (Backend)
- ▶ dieser kann durch verschiedene Frontends benutzt werden, die Datenhaltung und Präsentation realisieren:
  - ▶ “klassisches autotool”
  - ▶ autOlat (eine Olat-Erweiterung)

# Neue Aufgabentypen

die gewünschte (Syntax und) Semantik muß als (Haskell-)Quelltextmodul realisiert werden, dann der Server neu kompiliert.

- ▶ abstrakte Syntax durch `data`-Deklarationen
- ▶ konkrete Syntax (mit Parser) wird automatisch abgeleitet
- ▶ Implementierung der Bewertung (richtig/falsch) ist oft trivial,
- ▶ sinnvolle Fehlermeldungen sind etwas umständlicher,
- ▶ am aufwendigsten ist oft der Generator.

# Was fehlt (Sicht des Dozenten)?

- ▶ aktuelle OPAL-Anbindung  
(autOlat funktioniert, aber OPAL ist nicht Olat)
- ▶ leichter nachnutzbare  
Aufgaben-Konfigurationen  
z.B. derzeit kein (mechanischer) Austausch  
zw. Uni Leipzig und HTWK Leipzig möglich