Das `autotool` als Service-Provider

Christian Wanka und Johannes Waldmann, HTWK Leipzig

Weiterentwicklung

Semantik-Dienste (Aufgaben-Erzeugung und Korrektur) als Webservice (XML-RPC):

- kann von anderen E-Learn-Portalen benutzt werden (Prototypen: Lips, Elate)
- Service-Schnittstelle kann durch andere Semantik-Provider implementiert werden

Seminatik-Service soll eine *zustandslose Komponent* sein.

Bestandteile des `autotool`

- Semantik-Bibliothek
  (Automaten, Grammatiken, Graphen, . . .)
- Generator-Programme
- Korrektur-Programme

- Datenbank (2002)
  Konfiguration der Generatoren, Aufgaben
  erreichte Punkte
- Web-Schnittstelle
  für Studenten (2003), für Tutoren (2005)

Service-Schnittstelle (Anforderungen)

Kernforderungen

- Generieren von Aufgaben-Instanzen
- Bewerten von Lösungen zu Instanzen

Ergänzungen

- Information über verfügbare Aufgabentypen
- Verifzieren der Generator-Konfiguration

Datenaustausch unabhängig von Programmiersprachen, Betriebssystem

Service-Schnittstelle (Ansatz)

```
ist_types :: List<Task>

-- | liefert Beispiel-Konfiguration:
et_config :: Task -> Config

erify_config :: Task -> Config -> Boolean

-- | liefert persönliche Instanz und Beispiel-
et_instance :: Task -> Config -> Seed -> Pair

rade :: Task -> Instance -> Solution -> Boole
```

Service-Schnittstelle (genauer)

```
class Documented<E> { E contents, String docum
class Signed<E> { E contents, String signature
```

Ausgaben dokumentieren und Eingaben nur akzeptieren, wenn sie vorher verifiziert (und danach signiert) wurden.

```
get_config :: Task -> Documented<Config>
verify_config :: Task -> Config
    -> Signed<Config>
get_instance :: Task -> Signed<Config> -> Seed
   -> Pair<Doc<Signed<Instance>,Doc<Solution>>
grade :: Task -> Signed<Instance> -> Solution
   -> Documented<Boolean>
```

Implementierungen

lerzeit Prototypen

- autotool ist komplett als Server verfügbar (in Haskell)
- Client in Python/Zope für Lips (Christian Wanka, HTWK)
- Client in Java für Elate (Steffen Dienst, Uni Leipzi

Erweiterungen (I)

- Bewertung der Einsendung nicht nur qualitativ (boolean), sondern zusätzlich quantitativ (für Highscore)
- Dokumentation der Ausgabe nicht als String, sondern strukturiert (z. B. HTML)
- Eingabe nicht String, sondern strukturiert (XML): Benutzung schemagesteuerter Editoren.

Erweiterungen (II)

Eingabe und Bewertung nicht als Ganzes, sondern schrittweise. Viele Aufgaben gestatten den Begriff *partiell korrekte Lösung* (= kann zu total korrekter fortgesetzt werden). Bsp:

- COL: konfliktfreie Färbung für Teilgraphen
- PCP: Lösungswort, das Präfixbedingung erfüllt.

Vorteile:

- Eingabe einfacher Schritte: point/click (statt Text)
- Feedback nach jedem Lösungsschritt
- bessere Behandlung mehrerer Quantorenwechsel (z. B. Pumping-Lemma) (Student: $\exists$, autotool: $\forall$)

Standards?

ür

- Verwaltung von Aufgaben (Generatoren und Instanzen)
- Anzeige von Aufgaben
- (schrittweise) Eingabe von Lösungen
- Verwaltung von bewerteten Lösungen

Ziel: gemeinsame Schnittstelle für autotool, multiple choice-Software usw.