Diplomarbeit

Entwurf eines TI-Coaching Systems

von

Christof Kuhn

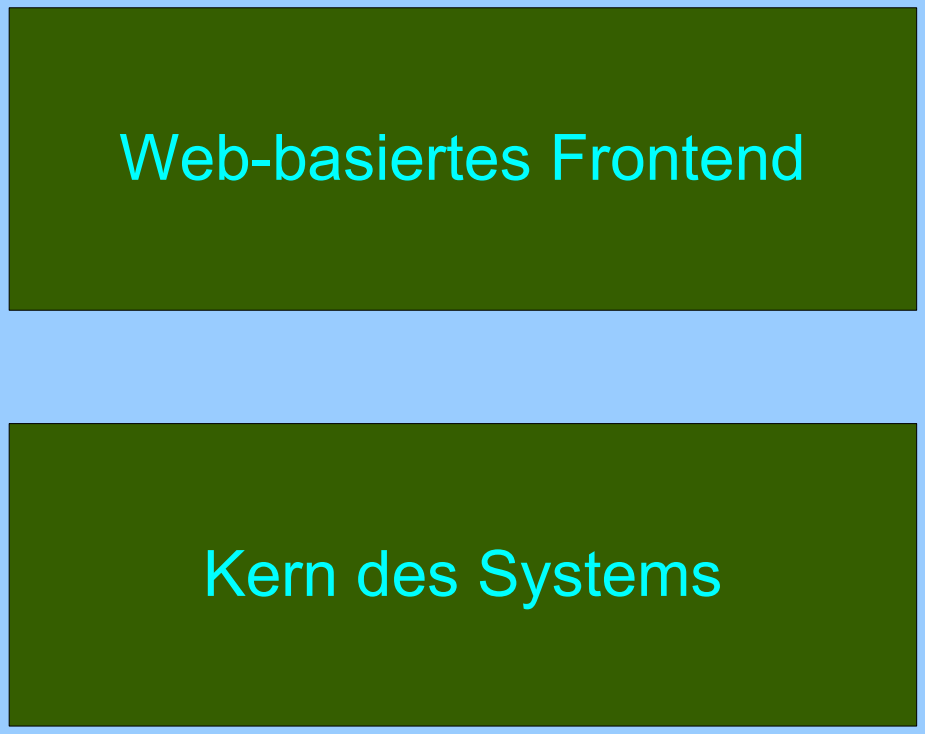
Warum TI-Coaching System?

- Theoretische Informatik fällt vielen schwer
- Sie ist aber sehr wichtig für die Informatik
- Sie ist nicht wirklich kompliziert
- Sie zu vermitteln ist nicht immer leicht
- Individuelle Betreuung kostet viel Zeit
- Ein automatisches System hat viel Zeit

Was soll das System können?

- Einem Studenten Aufgaben präsentieren
- Lösungen des Studenten verstehen
- Die Aufgaben selbst lösen
- Fehler in der Lösung des Studenten erkennen
- Die Lösung bewerten

Aufbau des Systems



Das Frontend

- Web-basiert für bequemen Zugriff von überall
- Web-basiert für ständige Aktualität
- Abstraktion vom eigentlichen Kern
- Unabhängigkeit in der Präsentation

Systemkern

- Verwaltung verschiedener Übungsaufgaben
- Verarbeitung eigener Lösungen
- Analyse studentischer Lösungen
- Speicherung verarbeiteter Lösungen

Zweiteilung und Kommunikation

- Implementierung lässt sich leichter aufteilen
- Verteilung der Last im Betrieb
- Isolierung bei Programmfehlern
- Kern sendet Ausgaben an das Frontend
- Kern steuert in Teilen das Frontend
- Frontend gibt relevante Eingaben an Kern weiter

Ansätze zum Entwurf des Kerns

- Auswahl geeigneter Aufgabentypen
- Analyse gewählter Aufgabenstellungen
- Automatische Verarbeitung von Aufgaben
- Vergleich von Lösungen und Fehlererkennung
- Bewertung von Lösungen
- Grundlagen der Kommunikation mit Frontend
- Verbindung der einzelnen Komponenten

Geeignete Aufgabenstellungen

- Typische Bereiche der Theoretischen Informatik
  - Automatentheorie und Formale Sprachen
  - Berechenbarkeitstheorie
  - Komplexitätstheorie
- AFS hat in der Regel den größten Umfang
- Aufgaben der AFS sind sehr mechanisch
- Der Teilbereich der Regulären Sprachen bietet schon einfache aber vielfältige Aufgaben

Aufgabenanalyse

- Verschiedene Aufgabentypen sind:
  - Synthese (Erzeugen Sie...)
  - Transformation (Wandeln Sie...)
  - Qualifikation (Zeigen Sie, dass...)
- Aufgaben lassen sich vielfältig kombinieren
- Transformationen sind leicht automatisierbar

Automatisierung

- Es gibt viele Ansätze:
  - Aufgaben als Planungsprobleme
  - Lösungen als PROLOG-Programme
  - Aufgabenstellung als Beweis
- Beliebige andere Implementierungen

Automatisierung – Beweiser

Eine sehr einfache Aufgabenstellung, ist die Umwandlung einer regulären Grammatik in einen nichtdeterministischen endlichen Automaten.

Diese soll nun als Beispiel für eine logische Beschreibung einer Umwandlung dienen.

Automatisierung - Beweiser

$vars_G(G, V)$: V ist die Menge der Variablen der Grammatik G

$term_G(G, \Sigma)$: Σ ist die Menge der Terminale der Grammatik G

$pro_G^1(G, x, y, z)$: $x \to yz$ ist eine Produktion für G

$pro_G^0(G, x, y)$: $x \to y$ ist eine Produktion für G

$start_G(G, S)$: S ist das Startsymbol von G

Automatisierung - Beweiser

$stat_N(M, Z)$: Z ist die Menge der Zustände des NFA M

$alph_N(M, \Sigma)$: Σ ist das Eingabealphabet von M

$trans_N(M, x, y, z)$: $\delta(x, y) \ni z$ ist ein Übergang für M

$start_N(M, S)$: S ist die Menge der Startzustände von M

$end_N(M, E)$: E ist die Menge der Endzustände von M

Automatisierung - Beweiser

$$vars_G(G, V) \rightarrow stat_N(M(G), V \cup \{E(G)\})$$

$$term_G(G, \Sigma) \rightarrow alph_N(M(G), \Sigma)$$

$$pro^1_G(G, x, y, z) \rightarrow trans_N(M(G), x, y, z)$$

$$pro^0_G(G, x, y) \rightarrow trans_N(M(G), x, y, E(G))$$

$$start_G(G, S) \rightarrow start_N(M(G), \{S\})$$

$$end_N(M(G), \{E(G)\})$$

Fehleranalyse und Erkennung

- Es gibt Allgemeine Fehler, die immer auftreten
- Spezifische Fehler für bestimmte Aufgaben
- Fehler sind einmalig oder systematisch
- Sie lassen sich nicht immer eindeutig erkennen
- Ein Ansatz ist Vergleich mit korrekter Lösung
- Lösungen können sich stark unterscheiden
- Nicht jeder Fehler lässt sich erkennen

Fehleranalyse - Transformationen

Aufgabentyp	***Fehlerkategorien***
Reg. Grammatik in NFA	Falsche Umwandlung
DFA (NFA) in Grammatik	Falsche Umwandlung, Endzustände (nur NFA), Fehlende Fälle (NFA)
NFA in DFA	Falsche Umwandlung, Endzustände, Fehlerzustand
Reg. Ausdruck in NFA	Rekursion, Sequenzautomat, Schleifenautomat
NFA in Reg. Ausdruck	Rekursion, Falsche Umwandlung, Falscher Einstieg
Minimierung DFA	Fehlende Wiederholung, Falsche Folgepaare, Identische Folgepaare
Gemeinsame Kategorien	Nachlässigkeit, Unvollständigkeit, Verwechslung

Aufgabenabwicklung/Bewertung

- Betrachtung einer Lösung als Ganzes
- Unterscheiden korrekter und falscher Lösungen
- Bewertung von aufgetretenen Fehlern
- Hinweise für bessere Lösungen
- Spielt entscheidende Rolle für Lerneffekt

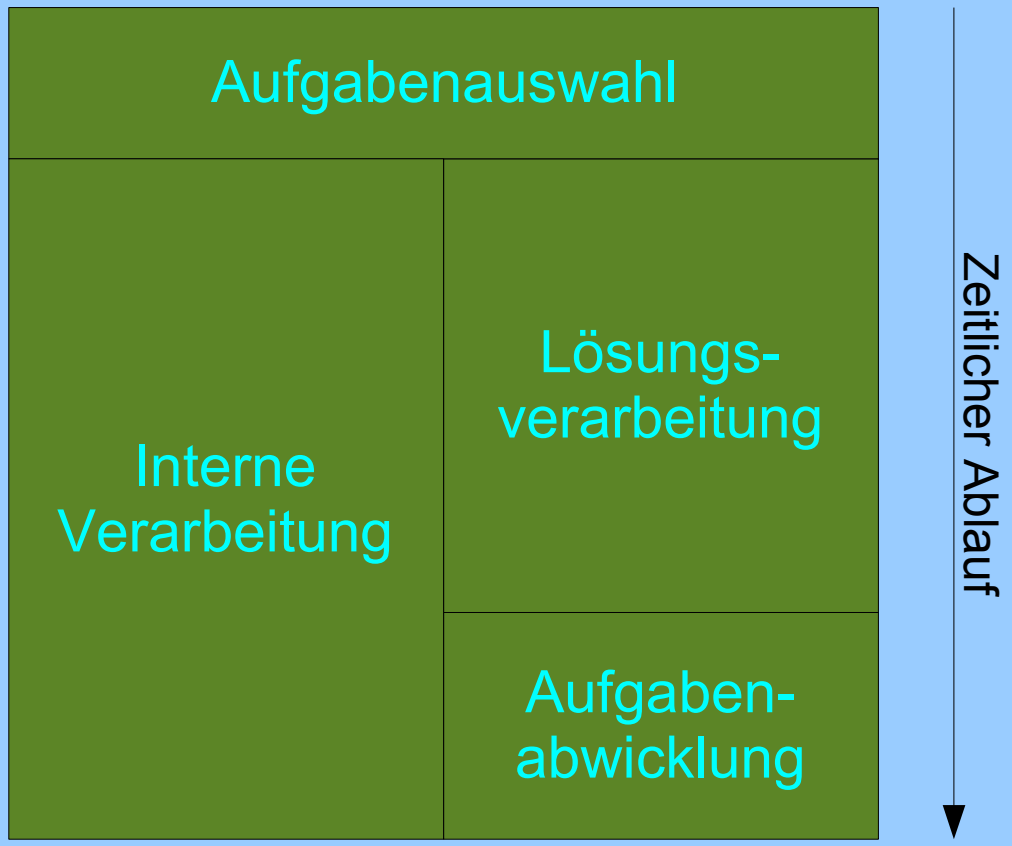
Kommunikation

- Kommunikation zwischen Student und System
- Kommunikation zwischen Frontend und Kern
- System sollte wissen ***was*** ein Student macht
- System sollte wissen ***wozu*** er es macht
- Es sollte wissen ***wie*** er es macht
- Kern muss nicht jede Eingabe kennen
- Frontend dient als Filter

Kommunikation - LBS

- Kerneingabe des Studenten ist eine Lösung
- Lösungsbeschreibungssprache ist sinnvoll
- Sprache beschreibt das ***Was***, ***Wozu*** und ***Wie***
- Aufgabenstellungen sind primitive Lösungen
- LBS funktioniert somit in beide Richtungen

Verarbeitungsstruktur



Struktur – Aufgabenauswahl

- Auswahl nach verschiedenen Kriterien
  - Aufgabentyp ist ein einfaches Kriterium
  - Aufgabenlänge ist ein weiteres
  - Schwierigkeitsgrad ist stark subjektiv
- Bestimmte Aufgaben zufällig generierbar
- Berücksichtigung von Studentenwünschen
- Unterschiedliche Aufgaben für gleichen Student

Struktur – Interne Verarbeitung

- Wichtiger Aspekt für "Intelligenz" des Systems
- Möglichst flexibel und selbstständig
- Effektiver Abgleich mit studentischer Lösung

Struktur - Lösungsverarbeitung

- Analyse der Eingaben des Studenten
- Erzeugen interner Repräsentation
- Allgemeine Fehlerüberprüfung
- Ständiger Abgleich mit interner Lösung
- Unterschiedlicher Grad der Interaktion
- Rückgriff auf interne Lösung bei Anfragen

Struktur - Aufgabenabwicklung

- Vergleich des Ergebnisses mit interner Lösung
- Feststellung der Korrektheit, falls möglich
- Bewertung nach geeigneten Kriterien
- Zusammenfassung aufgetretener Fehler
- Bei Bedarf Vorstellung eigener Lösung
- Hinweise für bessere Lösungen

Weitere Aspekte

- Aktive oder passive Fehlerbehandlung
- Konfiguration/Studentenprofile
- Speicherung von Informationen über Studenten
- Lernmotivation/Spielerisches