aus

Oliver J. Bott, Peter Fricke, Uta Priss, Michael Striewe (Hrsg.)

Automatisierte Bewertung in der Programmierausbildung

Digitale Medien in der Hochschullehre Band 6
2017, 420 Seiten, br., 42,90 €, ISBN 978-3-8309-3606-0

Waxmann Verlag GmbH
www.waxmann.com info@waxmann.com

19 Integration automatisierter Programmbewertung in LON-CAPA

Frauke Sprengel und Oliver Rod

Zusammenfassung
Das Lernmanagementsystem LON-CAPA stellt schon von sich aus eine Vielzahl möglicher Aufgabentypen, sowie vielfältige Möglichkeiten zur Kombinierbarkeit und Programmierbarkeit von Aufgaben bereit. Daneben gibt es eine einfache Schnittstelle zur externen Bewertung, die natürlich auch zur Programmbewertung benutzt werden kann und wird (z. B. an der Ostfalia und der Hochschule Hannover). Diese soll im Folgenden vorgestellt werden.

19.1 LON-CAPA

Das Lernmanagementsystem LON-CAPA hat seinen Ursprung an der Michigan State University in CAPA (a Computer-Assisted Personalized Approach). Dort wurde es eingesetzt, um automatisch Hausaufgaben zu randomisieren und anschließend zu bewerten. Der Fokus lag damals auf computergenerierten Aufgabenzetteln für Studierende. Im Jahre 2000 haben sich zwei weitere Gruppen der Michigan State University, welche sich ebenso mit dem elektronischen Bereitstellen von Unterlagen mit CAPA beschäftigt haben, zusammengetan und LON-CAPA gegründet. LON-CAPA ist quelloffen und nach der Gnu GPL lizenziert. Bereits 2007 hatten 120 Institutionen Inhalte in LON-CAPA bereitgestellt (vgl. [Kor+08]). Der Austausch wurde besonders für Einführungsveranstaltungen genutzt, da die Erstellung neuer Inhalte sehr zeitaufwendig war bzw. ist.

Teile dieses Kapitels entstanden im Rahmen des Projekts eCULT, Teilvorhaben eAssessment, gefördert durch das Bundesministerium für Bildung und Forschung unter dem Förderkennzeichen 01PL16066H. Die Verantwortung für den Inhalt dieses Kapitels liegt bei der Autorin und dem Autor.

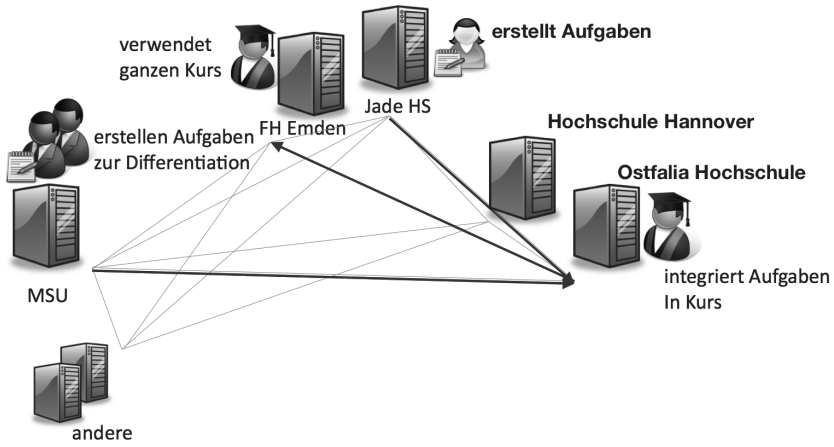


Abbildung 19.1: Veranschaulichung des LON-CAPA-Netzwerkes und der Austausch von Aufgaben.

19.1.1 Aufgabenaustausch

Derzeit (Stand 2016) haben sich über 60 Institutionen[1] weltweit dem LON-CAPA-Netzwerk angeschlossen. In diesem Netzwerk sind insgesamt über 200 000 Aufgaben verfügbar – in LON-CAPA Probleme genannt. Die meisten dieser Aufgaben sind in den technischen Fakultäten angesiedelt und im Rahmen von Forschungsprojekten entstanden. Die Aufgaben sind allen zugänglich. Jede Einrichtung innerhalb des LON-CAPA-Netzwerkes kann sie nutzen oder eigene erstellen. LON-CAPA bietet somit ein Repository mit Versionierung für die Aufgaben an und ermöglicht darüber hinaus deren Austausch. Es ist aber auch möglich, ganze Kurse innerhalb des Netzwerkes zu klonen.
Die Abbildung 19.1[2] soll vereinfacht die Vernetzung der einzelnen Hochschulen im LON-CAPA-Netzwerk verdeutlichen. Es wird dabei sichtbar, dass die verwendeten Aufgaben nicht nur aus der eigenen Hochschule (Domäne) kommen müssen. Dies ermöglicht Lehrenden bereits erstellte Aufgaben anderer Mitglieder im Netzwerk zu nutzen und die gewonnene Zeit in ihre Lehre fließen zu lassen.

1 http://www.lon-capa.org/institutions.html
2 http://www.lon-capa.org/presentations/futureparc.pdf

19.1.2 Aufgabentypen

Bis heute ist LON-CAPA eines der meist eingesetzten Lernmanagementsysteme im Bereich der Hochschullehre. Neben dem Aufgabenaustausch liegt der Schwerpunkt des Systems im Erstellen und Bereitstellen verschiedener Inhalte und Aufgabentypen. So unterstützt LON-CAPA:

- Algebraische Aufgaben (Formel mit Computer-Algebra-System),
- Aufgaben mit freier Gestaltung (Funktionsplot-Antwort mit Hintergrund-Plot),
- Auswahlaufgaben (1-aus-n, Rangordnung, Zuordnung mit Optionen),
- Chemische Aufgaben (Chemische Reaktion, Organisches Material),
- Eingabeabhängige Aufgaben,
- Manuell bewertete Aufgaben (Essay, Dropbox),
- Numerische Aufgaben,
- Verschiedenes (Klick-ins-Bild),
- Externe Bewertung (externalresponse).

Die verschiedenen Aufgabentypen in LON-CAPA werden „problems“ genannt und sind in einer XML-ähnlichen Struktur aufgebaut. Alle problem-Dateien können auch Gebrauch von der Skriptsprache Perl machen, in welcher LON-CAPA zu großen Teilen geschrieben ist. Darüber hinaus verwenden einzelne Aufgabentypen das Computer-Algebra-System Maxima oder die Statistiksprache R. Für Aufgaben im Gebiet der Chemie gibt es zudem die Möglichkeit, den JSME Molecular Editor zu nutzen. Für die Eingabe oder Darstellung von Formeln kann LaTeX verwendet werden (vgl. [Gro15]). Prinzipiell sind in LON-CAPA Gruppenabgaben und Datei-Uploads als Abgabe möglich, allerdings beides nur für den Aufgabentyp *essay*.

Außerdem bietet LON-CAPA eine Schnittstelle für Aufgaben, die außerhalb des LMS bewertet werden sollen. Diese Aufgaben nutzen den Aufgabentyp *externalresponse*. Hier können Texteingaben von einzelnen Studierenden externen Programmen zur Bewertung übergeben werden.

Der Fokus in diesem Kapitel liegt in der Erstellung und Nutzung von External-Response-Aufgaben. Im Folgenden werden zunächst der Aufbau, die Parameter und anschließend die Anzeige vorgestellt. Danach wird am Beispiel von JFLAP eine Einbindung veranschaulicht.

19.2 External Response

Am Beispiel des in Abbildung 19.2 dargestellten XML-Codes werden die wichtigsten Bestandteile einer External-Response-Aufgabe erläutert.

19.2.1 Aufbau einer External-Response-Aufgabe

Die vereinfacht dargestellte Aufgabe soll den prinzipiellen Aufbau einer External-Response-Aufgabe erläutern[3]. Die Zahlen am linken Rand der Abbildung korrespondieren mit Erläuterungen der nachfolgenden Liste.

1. Die komplette Aufgabe befindet sich im Problem-Element.
2. Das Import-Statement erlaubt externe Bibliotheken mit dieser Problem-Datei zu verbinden. Hier bietet es sich an, eine Bibliothek zu laden. Es können somit komplexe Skripte vor dem Kurskoordinator verborgen oder die

```
1> <problem>
2>   <import id="11">/Pfad/zur/header-library</import>
3>   <script type="loncapa/perl">
       # URL des externen Service festlegen
       $externalurl = '$url'
       # Formular-Daten des HTTP-POST-Request festlegen
       %args = 'key => value'</script>
4>   <startouttext/> Aufgabentext <endouttext/>
5>   <instructorcomment>
         $error
     </instructorcomment>
6>   <startouttext/> $ausgabe
       <div id="codemirror-textfield">
     <endouttext/>
7>   <externalresponse answerdisplay="" answer=""
       url="$externalurl" form="%args" id="1">
       <textfield> </textfield>
     </externalresponse>
6>   <startouttext/>
       </div>
     <endouttext/>
8>   <import id="92">/Pfad/zur/footer-library</import>
   </problem>
```

Abbildung 19.2: Minimalbeispiel einer External-Response-Aufgabe (https://media.elan-ev.de/proforma/ProFormA/LON-CAPA/JFLAP-Minimal.problem).

3 Ein ausführlicheres „Hello World“-Beispiel findet sich im LON-CAPA-Aufgabenkatalog: /res/fhwf/ecult/Java/Hello_World.problem

Darstellung des Textfeldes optimiert werden. Die Verwendung von CodeMirror[4] ermöglicht Syntax-Highlighting, automatisches Einrücken und andere Features, welches aus dem Textfeld einen Editor zum Programmieren machen. Es wird aber immer eine Footer-Library benötigt, da erst diese auf das komplette Document Object Model zugreifen kann.

3. In diesem Abschnitt werden die Formulardaten und die Zieladresse des externen Dienstes für die Bewertung festgelegt.

4. Aufgabentext für die jeweilige Aufgabe.

5. Im Instructor-Comment kann man Ausgaben, wie zum Beispiel Fehlerausgaben, vor den Nutzern verbergen bzw. Hinweise nur dem Kurskoordinator anzeigen.

6. Die Variable `$ausgabe` ist ein Platzhalter für aktuelle bzw. alte Ausgaben. Das DIV „codemirror-textfield“ soll die eindeutige Zuordnung für das Eingabetextfeld ermöglichen.

7. In diesem Tag befinden sich alle wichtigen Parameter einer External-Response-Aufgabe. Die drei Hauptbestandteile[5] sind die externe *URL*, die Musterlösung *answer* und Formulardaten *form*. Die *URL* zeigt auf eine Ressource im Internet, welche dieses Problem bewerten soll.

8. In der Footer-Library sollten Skripte geladen werden, die das Textfeld oder die Rückmeldung des externen Dienstes manipulieren.

In den Formulardaten des POST-Request werden von LON-CAPA, ohne dass der Autor weitere Key-Value-Paare hinzufügt, die folgenden drei Felder übermittelt:

1. LONCAPA_correct_answer => answer LON-CAPA nimmt an, dass immer eine Musterlösung für die Bewertung der studentischen Einreichung notwendig ist. Dies kann aber auch vernachlässigt werden.

2. LONCAPA_student_response => textfield Dies entspricht dem kompletten studentischen Text, welcher im Textfeld eingegeben wurde.

3. LONCAPA_language => Dieses Feld wird von LON-CAPA automatisch mit der spezifizierten Sprache der Ressource gefüllt.

4 CodeMirror ist ein in JavaScript programmierter vielseitiger Text-Editor (https://codemirror.net).

5 https://s3.lite.msu.edu/adm/help/Authoring_ExternalResponse.hlp

19.2.2 Antwortformat

Der externe Service hinter der *URL* aus Abschnitt 19.2.1 würde die Einreichung automatisiert testen und das Ergebnis an LON-CAPA zurücksenden. Die Antwort muss folgender XML-Struktur folgen:

```
<loncapagrade>
        <awarddetail></awarddetail>
        <message></message>
        <awarded></awarded>
</loncapagrade>
```

Alle möglichen Antworten finden sich in der LON-CAPA-Hilfe[6]. Es soll hier nur auf Beispiele eingegangen werden, die häufig in der Praxis verwendet werden:

EXACT_ANS, APPROX_ANS: Die Einreichung ist korrekt.

INCORRECT: Die Einreichung ist falsch.

ASSIGNED_SCORE: Die Einreichung ist nur teilweise richtig. Hier wird aber noch ein Gleitkommazahl zwischen 0 und 1 im XML-Element *awarded* erwartet.

ERROR: Bei der Einreichung ist ein Fehler aufgetreten. Der Versuchszähler des Studenten wird nicht erhöht.

In der zurückzugebenden Nachricht (Message) können weitere XML-Elemente enthalten sein. Falls keine zusätzliche Formatierung durch den Aufgabenautoren (oder Benutzer einer Bibliothek) erfolgt, wird der enthaltene Text unter der studentischen Eingabe einfach als gelb unterlegter Text angezeigt.

19.3 Beispiel: JFLAP für die theoretische Informatik

Das Programm JFLAP ist ein Werkzeug, das z. B. im Grundstudium an der Hochschule Hannover in der theoretischen Informatik eingesetzt wird. Es hat eine graphische Oberfläche und kann sowohl Bilder als auch ein internes XML-Format exportieren. Das Programm wurde entwickelt unter der Leitung von Susan Rodgers an der Duke University. Genauere Informationen liefern die JFLAP-Webseite [Rod] und das Buch [RF06].

6 http://source.lon-capa.org/cgi-bin/cvsweb.cgi/doc/homework/datastorage?rev=1.26

Die Studierenden können in der graphischen Oberfläche Automaten und Maschinen konstruieren und testen (z. B. endliche Automaten und Maschinen, Kellerautomaten, Turing-Maschinen). Automaten können in passende Grammatiken (oder reguläre Ausdrücke) umgewandelt werden und umgekehrt. Automaten und Grammatiken können auf ihren Typ getestet werden, sowie auf die Akzeptanz bzw. die Erzeugung von einzugebenden Wörtern. Nicht-Determinismus in den Automaten kann man sich hervorheben lassen. Daneben existieren Veranschaulichungen von beliebten Algorithmen: NFA[7] zu DFA[8], Minimierung von DFA, vollständige Automaten und einige mehr.
Die Benutzung ist einfach und mehr oder minder selbsterklärend. Ein Tutorial findet sich auf der JFLAP-Webseite [Rod], wo man sich auch das Programm selbst herunterladen kann.

Die aktuell stabile und für Lehrveranstaltungen empfohlene Version ist JFLAP7. An der Hochschule Hannover ist JFLAP7 unter der Leitung von Carsten Kleiner internationalisiert worden. Diese Version kann auf Nachfrage bereitgestellt werden.

Es existiert zudem eine Weiterentwicklung (JFLAP8-Beta), die einige Ungenauigkeiten und Fehler in JFLAP7 behebt. Leider sind die Formate nicht in allen Fällen abwärtskompatibel. Auch stehen nicht alle Funktionen von JFLAP7 zur Verfügung, dafür einige andere. Die oben genannten Funktionen gibt es aber in beiden Versionen.

An der Hochschule Hannover ist unter der Leitung von Frauke Sprengel ein Wrapper für JFLAP7 (mittlerweile auch JFLAP8) entwickelt worden ([Spr16], [Tos13], [Hel16]), der die Fähigkeiten des Programms für eine Grading Engine

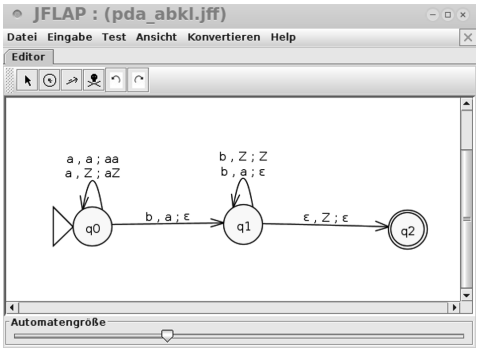


Abbildung 19.3: JFLAP7 mit deutscher Oberfläche (Kellerautomat).

7 NFA: Nondeterministic Finite Automaton, (nichtdeterministischer endlicher Automat)
8 DFA: Deterministic Finite Automaton, (deterministischer endlicher Automat)

benutzt, die innerhalb eines LMS wie z. B. LON-CAPA verwendet werden kann. Die Studierenden geben dazu einen String ein oder kopieren eine aus JFLAP exportierte XML-Datei in ein Textfeld. Die Bewertung von Automaten gleicht hierbei im Wesentlichen einer Programmbewertung, da sowohl „Syntax“ als auch Funktionalität des Automaten überprüft werden müssen.

Aktuell können z. B. folgende Arten von Fragen gestellt werden:

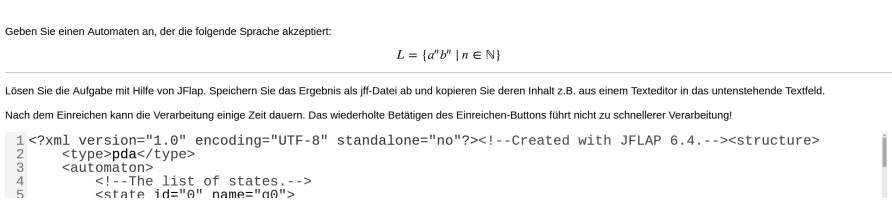
- Zu einer gegebenen Sprache (definiert z. B. durch eine Grammatik, einen regulären Ausdruck, eine Menge, eine Beschreibung als Text) soll ein akzeptierender Automat konstruiert werden (endlicher Keller-/Turing-Automat).

- Zu einer gegebenen Sprache (definiert z. B. durch einen Automaten, einen regulären Ausdruck, eine Menge, eine Beschreibung als Text, . . .) soll eine erzeugende Grammatik (eines bestimmten Typs) angegeben werden.

- Zu einer gegebenen Sprache (definiert z. B. durch eine Grammatik, einen Automaten, einen regulären Ausdruck, eine Menge, eine Beschreibung als Text, . . .) soll eine Anzahl von Wörtern angegeben werden.

- Zu einem vorgegebenen Typ von Automat oder Grammatik soll ein Beispiel gegeben werden.

Die im Fragentext gegebene Sprache wird an das Programm durch Angabe eines regulären Ausdrucks, einer Grammatik oder zwei Mengen von Wörtern (Wörter aus der Sprache bzw. nicht aus der Sprache) übergeben.


Geben Sie einen Automaten an, der die folgende Sprache akzeptiert:

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

Lösen Sie die Aufgabe mit Hilfe von JFlap. Speichern Sie das Ergebnis als jff-Datei ab und kopieren Sie deren Inhalt z.B. aus einem Texteditor in das untenstehende Textfeld.

Nach dem Einreichen kann die Verarbeitung einige Zeit dauern. Das wiederholte Betätigen des Einreichen-Buttons führt nicht zu schnellerer Verarbeitung!

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?><!--Created with JFLAP 6.4.--><structure>
2     <type>pda</type>
3     <automaton>
4         <!--The list of states.-->
5         <state id="0" name="q0">
```

Abbildung 19.4: LON-CAPA-Aufgabe (Ausschnitt): Hier ist der Kellerautomat aus Abbildung 19.3 gesucht.

19.4 Anbindung von JFLAP an LON-CAPA

19.4.1 Eine Beispielaufgabe

Die Sprache aus dem Beispiel der Abbildungen 19.3 und 19.4 wird erzeugt durch die kontextfreie Grammatik $S \to aSb \mid ab$. In einer LON-CAPA-Aufgabe, die den Automaten dazu verlangt, könnte man das Alphabet noch randomisieren und als Autor eine Aufgabe[9] verfassen wie in Abbildung 19.5.

In der ersten eingebundenen Bibliothek verbergen sich Funktionen für die Zusammenstellung aller nötigen Informationen für den JFLAP-Aufruf (`jflapUrl-Input`), die Darstellung der studentischen Eingaben (`listOfSubmissions`) und der Rückgabe von JFLAP. Die dazu benötigten Angaben werden in den Abschnitten 19.4.4 und 19.4.5 beschrieben. In den beiden anderen Bibliotheken werden nur das Syntax-Highlighting im Eingabefeld und einige Ausgaben (wiederkehrender Text, Fehlerausgaben) erledigt.
Der externe Aufruf als solcher verwendet hier nur die Felder `answer` und `url`. Auf die zusätzliche Angabe von Key-Value-Paaren wurde hier verzichtet.

Als Ergebnis kann man dann in LON-CAPA die Ausgabe wie in Abbildung 19.6 erhalten.

19.4.2 JFLAP als externer Service

Die Anbindung von JFLAP erfolgte in starker Anlehnung an das Skript zur Beispielaufgabe für externe Aufrufe, die auf jedem LON-CAPA-Server[10] liegt.

Es wird ein Perl-Skript auf einem Webserver aufgerufen, das die nötigen Angaben aus dem <externalresponse>-Tag verarbeitet und eine XML-Struktur wie in Abschnitt 19.2.2 beschrieben zurückliefert:

```
 ... Vorbereitung
POST Variablen werden in %FORM geladen
...
my $call_jflap= "java -jar JFLAP.jar"
   . " \'$FORM{'LONCAPA_correct_answer'}\' "
   . " \'$FORM{'LONCAPA_student_response'}\'";
... Fehler und Timeout-Behandlung ...
```

9 In LON-CAPA zu finden unter:
/res/fh-hannover/sprengel/Informatik/TheoretischeInformatik/PushDownAutomata/pda_anbn_jflap.problem.

10 z. B. https://loncapa.hs-hannover.de/res/adm/includes/templates/sampleexternal.pl

```
$result = `$call_jflap`;
print (<<ENDOUT);
$result
ENDOUT
exit;
```

```
<problem>
  <import Vorbereitung />
  <script type="loncapa/perl">
    @letter = ('a' .. 'g');
    $choice = random(0,$#letter-2,2);
    $a = $letter[$choice];
    $b = $letter[$choice+1];
    $given = 'S->'.$a.'S'.$b.'|'.$a.$b;
    $mode = 'agtw';
    $type = "pda";
    $numberOfWords = 20;
    $maxlength = 20;
    $tasktitle = "Automat für $a^n $b^n";
    ($externalurl,$input,$error) =
     jflapUrlInput($mode,$given,$type,
     $tasktitle,$numberOfWords,$maxlength);
    $listSubmission =  listOfSubmissions(JFC,JFlapCall);
  </script>
  <startouttext />
    Geben Sie einen Automaten an, der die folgende
    Sprache akzeptiert:
    <m eval="on">
     \[ L = \{ $a ^n $b ^n  \,|\, n \in
      \mathbb{N} \}
    </m>  <p />
  <endouttext />
  <import AutomatonHead>
  <part id="JFC">
     <externalresponse answer="$input" id="JFlapCall"
           url="$externalurl">
         <textfield spellcheck="none" />
     </externalresponse>
  </part>
  <import AutomatonFoot>
</problem>
```

Abbildung 19.5: LON-CAPA-Aufgabe: Codebeispiel für JFLAP-Aufgabe.

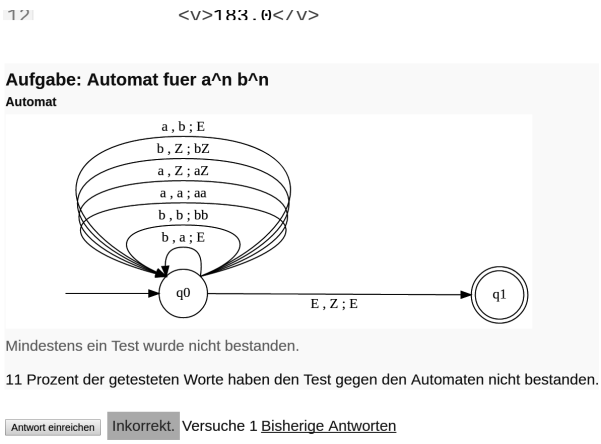


Abbildung 19.6: LON-CAPA-Ergebnis: Hier ist der Kellerautomat aus Abbildung 19.3 und 19.4 nicht sofort richtig eingegeben worden.

Wenn noch weitere Variablen in den POST-Variablen übermittelt werden, können diese auch verwendet werden. Die XML-Struktur wird hier direkt von JFLAP erzeugt. Für dieses Beispiel müssen auf dem Webserver neben Perl nur noch ein Java Runtume Environment (JRE) und die Graphhviz-Bibliothek zur Darstellung der Automaten als SVG installiert sein.

19.4.3 Einzubindende Bibliotheken

Alle im Folgenden beschriebenen Bibliotheken stehen als Sourcecode[11] zur Verfügung. Benutzt werden muss auf jeden Fall

JFlap_call_preparation.library: Hier erfolgt die gesamte Vorbereitung. Der Name des Servers ist dort im Code hinterlegt und muss geändert werden für einen eigenen Server.

Die restlichen Bibliotheken müssen nicht benutzt werden, beinhalten aber häufig verwendeten Code und sorgen für das Syntaxhighligthing und die Darstellung der Ergebnisse:

JFlap_call_automaton_head.library, JFlap_call_automaton_foot.library: Header und Footer für den Aufgabenteil mit dem externen Aufruf von JFLAP.

11 In LON-CAPA zu finden unter: /res/fh-hannover/sprengel/Informatik/TheoretischeInformatik/Libraries, dort findet man auch Beispielaufgaben.

Sorgt für die Ausgabe wiederkehrende Formulierungen und Syntax-Highlighting der JFLAP-Datei und die Darstellung der Ergebnisse.

jff_automaton_parts_head.library, jff_automaton_parts_foot.library: Header und Footer für einen Aufgabenteil, der die Eingabe der JFLAP-Datei liest und die Komponenten des Automaten abfragt (für endliche und Kellerautomaten).

JFlap_call_grammar.library: Aufgabenteil (sorgt für die Ausgabe wiederkehrender Formulierungen und den Aufruf von JFLAP für Grammatiken).

JFlap_call_words_head.library, JFlap_call_words_foot.library: Header und Footer für einen Aufgabenteil, in dem nach Beispielwörtern einer gegebene Sprache gefragt wird.

19.4.4 Funktion für die URL und die Parameter des JFLAP-Aufrufs

Im Perl-Skript der LON-CAPA-Aufgabe sollte sich solch ein Funktionsaufruf befinden:

```
($externalurl,$input,$error) =
     jflapUrlInput($mode,$given,$type,$tasktitle,
                   $numberOfWords,$maxlength);
```

Dabei ist es wichtig, die Ausgabevariablen `($externalurl, $input, $error)` genauso zu benennen, wie im obigen Aufruf in der External-Response-Aufgabe aus dem letzten Abschnitt. Sie enthalten dann

- `$externalurl`: Link zum aufzurufenden Perl-Skript,
- `$input`: im Wesentlichen eine Konkatenation der nötigen Parameter, einem Eingabeparameter für JFLAP,
- `$error`: falls der Aufruf fehlschlägt, kann dieser String in einem Instructor-Comment für den Aufgabenautoren oder Kurskoordinator angezeigt werden.

Für den Aufruf der Funktion `jflapUrlInput` sind die folgenden Parameter nötig:

- `$mode` (String): Einer der Modi wie im nächsten Abschnitt 19.4.5 beschrieben.

- `$given` (String): Regulärer Ausdruck oder Grammatik zur Beschreibung der Sprache aus dem Aufgabentext (falls nur ein Beispiel gefragt wird: leerer String).

Dazu kommen noch optionale Parameter, die von hinten nach vorn weggelassen werden können

- `$type` (String): Typ der Grammatik oder des Automaten (Abschnitt 19.4.5), Default: non.
- `$tasktitle` (String): Titel der Aufgabe, Default: JFlap-Aufgabe oder JFlap task, in Abhängigkeit von der Sprache
- `$numberOfWords` (Integer): Anzahl der zu generierenden Wörter, mit denen getestet werden soll, Default: 10.
- `$maxlength` (Integer): Maximale Länge der zu generierenden Wörter, mit denen getestet werden soll, Default: 10.

19.4.5 Modi und Typen

Die JFLAP Grading Engine kann bisher mit folgenden Modi benutzt werden:

- `ar[t][w]` (Automaton, Regex[, Type][, Words]): Gesucht ist ein endlicher Automat, interne Beschreibung der Sprache als regulärer Ausdruck.
- `ag[t][w]` (Automaton, Grammar[, Type][, Words]): Gesucht ist ein Automat, interne Beschreibung der Sprache als Grammatik.
- `gg[t][w]` (Grammar, Grammar[, Type][, Words]): Gesucht ist eine Grammatik, interne Beschreibung der Sprache als Grammatik.
- `egt` (Example, Grammar, Type): Gesucht ist ein Beispiel für eine Grammatik von bestimmtem Typ.
- `eat` (Example, Automaton, Type): Gesucht ist ein Beispiel für einen Automaten von bestimmtem Typ.

Bei der Angabe von `t` sollte ein bestimmter Typ verlangt werden. Bei der Angabe von `w` werden die Wörter auf LON-CAPA-Seite erzeugt.

Automatentypen sind (entweder mit `d`=deterministisch oder `n`=nichtdeterministisch oder ohne=egal) `[d/n]fa` für endliche Automaten, `[d/n]pda` für Kellerautomaten, `[d/n]ta` für Turing-Automaten oder `non` nicht angegeben.

Grammatiktypen sind wie erwartet `rl` rechtslinear, `rlcfg` rechtslinear oder kontextfrei, `cfg` kontextfrei, aber nicht rechtslinear, `ncfg` nicht kontextfrei oder `non` nicht angegeben.

19.4.6 Reguläre Ausdrücke und Grammatiken

Die regulären Ausdrücke werden entweder in Java (ohne `w`) oder in Perl (mit `w`) verarbeitet. Somit hat man viele Formulierungsmöglichkeiten. Als Terminale sind alle Kleinbuchstaben (`a ... z`) sowie `0,1` erlaubt.

Eine Grammatik wird für die Verarbeitung in JFLAP wie folgt als String geschrieben:

```
S -> aB, B -> bS | E | a, S -> aSa
```

Als Konvention wird hiernach verwendet: Kommata zur Trennung der Regeln, ein Pfeil (`->`) zwischen linker und rechter Seite der Regel, der senkrechte Strich `|` zur Trennung mehrerer rechter Seiten, ein großes `E` (statt ε) für das leere Wort, ein großes `S` als Startsymbol, alle anderen Großbuchstaben als Nichtterminale, alle Kleinbuchstaben (`a ... z`) und `0,1` als Terminale. Dazwischen sind beliebig viele Leerzeichen erlaubt.

Innerhalb der LON-CAPA-Aufgabe kann man wie gewohnt die Perl-Funktionen zur Randomisierung z. B. der Buchstaben des Alphabets der Grammatik oder des regulären Ausdrucks benutzen.

19.5 Fazit

In LON-CAPA ist die Anbindung externer Grader für Programmbewertung oder ähnliches durch die Problemart External-Response sehr einfach möglich. Dem aufgerufenen Grader können neben der studentischen Eingabe weitere Parameter als POST-Request übergeben werden. Studentische Programmeingaben sind nur einzeln und nur über das Textfeld und nicht als hochzuladende Dateien möglich. Rückgabeinformationen sind strukturiert in einer XML-Nachricht möglich und durch JavaScript formatierbar.

Literatur für dieses Kapitel

[Gro15] LON-CAPA Group. *Learning Online Network with CAPA – Author's Tutorial And Manual*. Michigan State University, Mai 2015. URL: https://loncapa.msu.edu/adm/help/author.manual.pdf.

[Hel16] Benjamin Held. *Erweiterung einer Bewertungssoftware für LON-CAPA unter Verwendung von JFLAP*. Masterarbeit. Hannover, 2016.

[Kor+08] Gerd Kortemeyer u. a. „Experiences using the open-source learning content management and assessment system LON-CAPA in introductory physics courses". en. In: *American Journal of Physics* 76.4 (2008).

[RF06] Susan H. Rodgers und Thomas W. Finley. *JFLAP: An Interactive Formal Languages and Automata Package*. Sudbury, MA: Jones & Bartlett Publishers, 2006.

[Rod] Susan H. Rodgers. *JFLAP web page*. URL: www.jflap.org.

[Spr16] Frauke Sprengel. *Electronic Exercises in Theoretical Computer Science*. (verfügbar in LON-CAPA /res/fh-hannover/sprengel). Hochschule Hannover, 2016.

[Tos13] Ufuk Tosun. *Automatische Bewertung von Studierendenabgaben in der theoretischen Informatik auf Basis von JFLAP*. Bachelorarbeit. Hannover, 2013.