Automatisiertes Schließen mit formalen Begriffen

Implementierung

Karl Fritsche[1] und Pascal Hitzler[2*]

[1]Martin-Andersen-Nexö-Gymnasium Dresden
[2]AIFB, Universität Karlsruhe

Im Rahmen der in Klasse 11/12 zu absolvierenden *Besonderen Lernleistung* können Schüler des Dresdner Martin-Andersen-Nexö-Gymnasiums Dresden extern an der Universität betreut und damit in Forschungsprojekte eingebunden werden. Dieses Dokument beschreibt die Implementierung eines Systems zum automatisierten Schließen über Begriffshierarchien, die aus einer solchen Betreuung in den Jahren 2003 und 2004 hervorgegangen ist.

Die mathematischen Grundlagen des Projekts sind in [6] beschrieben. Die Implementierung erfolgte in Prolog und Java™ und wird im Folgenden vorgestellt. Die Software mit Quellcode ist im Internet unter http://www.aifb.uni-karlsruhe.de/WBS/phi/mint11/ erhältlich.

Wir bedanken uns bei Herrn Geißler und Frau Pabst vom Martin-Andersen-Nexö-Gymnasium Dresden, die das Projekt von schulischer Seite betreut haben. Paul Fritsche und Wieland Morgenstern waren bei Fragen rund um Java hilfreich.

Funktionalität

Das implementierte System findet min-Antwortmodelle im Sinne von [6].

An das System wird ein erweitertes oder normales Programm übergeben und außerdem ein formaler Kontext, über welchem geschlossen wird. Die Eingabe erfolgt über eine in Java programmierte GUI. Eingabedaten können in Standardformaten abgespeichert und damit wiederverwendet werden.

*Während des ersten Teils des Projektes war der Autor am Institut für Künstliche Intelligenz, Fakultät Informatik, Technische Universität Dresden.

Installation

Das System ist auf dem Betriebssystem Windows© XP programmiert wurden. Es sollte auf allen gängigen Windows© Betriebssystem laufen. Es läuft leider nicht auf anderen Betriebssystemen, wie Linux oder MacOS, da die verwendeten JPL Schnittstellenbibliotheken nur für Windows© bereitstehen. Sollten diese Bibliotheken auch auf andere Betriebssysteme portiert werden, kann das System diese sofort laden und arbeitet dann auch dort. Einzige Einschränkung ist, dass es für dieses Betriebssystem eine Java2™ und eine SWI-Prolog-Version geben muss.

Auf http://www.aifb.uni-karlsruhe.de/WBS/phi/mint11/ sind die Dateien verfügbar, die benötigt werden, um das System auf Windows© zu starten. Es benötigt zusätzlich nur Java2™ in mindestens der Version 1.4.0_01 und eine SWI-Prolog Version ab 5.4.5.

Das System wird mit der Batch-Datei Projekt.bat gestartet.

Auf der angegebenen Webseite befinden sich auch die Quelltexte und einige Beispielkontexte.

Bedienung

Bie Benutzeroberfläche besitzt drei Register (Programm, Kontext, Ergebnis), mit denen man zu drei unterschiedlichen Tabellen kommt, in denen Daten eingetragen und ausgelesen werden können. Wir werden erläutern, was in jeden Register ausgeführt werden kann.

Register Programm

Im Register *Programm* erstellt, lädt oder speichert man das Programm. Entsprechende Buttons dienen zum Laden und Speichern des Programmes im Textformat. Drei Eingabefelder in der Benutzeroberfläche dienen der manuellen Eingabe von Regeln des Programms. In das erste Feld kommt die Schlußfolgerung, in das zweite die *wenn*-Bedingung und das dritte Feld ist für die erweiterte Regel die *wenn nicht notwendig*-Bedingung. In die ersten beiden Felder kann man die entsprechenden Klauseln hineinschreiben. Um in das letzte Feld schreiben zu können, muss das Kontrollkästchen vor dem Eingabefeld aktiviert sein. In den Klauseln müssen alle Gegenstände und Merkmale mit Kommas getrennt werden. Mit dem Button *Hinzufügen* wird diese Regel dem Programm übergeben und steht in der darunterstehenden Tabelle. Wenn man in der Tabelle eine bestimmte Regel anklickt, kann man diese mit dem Button *Ausgewähltes Löschen* löschen. Durch einen Doppelklick auf eine bestimmte Klausel kann diese im Nachhinein wieder editiert werden. Der Button *Alles löschen* löscht die komplette Tabelle.

Register Kontext

Im Register *Kontext* erstellt, lädt oder speichert man den Kontext. Entsprechende Buttons dienen zum Laden und Speichern des Programmes im csv-Format. In der rechten Hälfte der Benutzeroberfläche sieht man die Kontexttabelle, am Anfang ist sie natürlich leer. Mit den Buttons *Gegenstand* und *Merkmal hinzufügen* kann man neue Merkmale und Gegenstände eingeben. Wenn ein Gegenstand ein bestimmtes Merkmal hat, geht man einfach in die Zelle, welche in der Zeile des Gegenstands und in der Spalte des Merkmales liegt. Dadurch erscheint ein $\times$ in der Zelle. Durch einen erneuten Klick in die Zelle verschwindet das $\times$ wieder. Mit *Alles Löschen* löscht man die komplette Tabelle und mit *Zeile Löschen* nur die Zeile, die angeklickt ist. Dasselbe gilt für *Spalte löschen*, dort wird die Spalte, in der die Markierung ist, gelöscht.

Register Ergebnis

Im Register *Ergebnis* berechnet man das min-Antwortmodell. Dies kann man hier auch speichern und laden, falls man große Kontexte hat, bei denen die Berechnung lange dauert. Beim Bestätigen des Buttons *Berechnen* berechnet das System das min-Antwortmodell und gibt es aus. Es werden die Intenten, Extenten und die Gegenstände und Merkmale, die diese Intenten/Extenten erzeugen, ausgegeben.

Ein Beispiel

Bei den Dateien findet man einen Beispielkontext zum Thema *Pizzen*. Die Merkmale des Kontextes sind die Beläge der Pizzen. Der Kontext umfasst ca. 30 Gegenstände (Pizzen) und mehr als 20 Merkmale (Beläge).

Betrachten wir als erstes das Programm bestehend aus der einzigen Regel $[salami] \leftarrow [\bot]$. Der Extent des min-Antwortmodells zeigt dann wie erwartet alle Pizzen an, die als Belag Salami haben.

Möchte man Pizzen mit einer bestimmten Menge von Belägen haben, gibt man entsprechend mehrere Regeln ein, z.B. $[salami] \leftarrow [\bot]$ und $[schinken] \leftarrow [\bot]$. Man bekommt dann als Extenten die entsprechenden Pizzen und als Intenten die Zutaten zu den jeweiligen Pizzen geliefert. Für das Beispiel liefert das System die Pizzen deutschland, titaniccalzone und venezia. Diese ersten beiden Pizzen enthalten champignons.

Was aber wenn man keine Champignons mag? Die Lösung ist einfach, man erweitert eine der beiden Regel oder auch beide wie folgt: Das Programm mit erweiterter Regel lautet z.B.

$$[salami] \leftarrow [\bot]. \qquad [schinken] \leftarrow [\bot], \sim [champignons].$$

Als Antwortmodell kommt nur die Pizza venezia, da es die einzige Pizza mit Salami und Schinken, aber ohne Champignons ist.

Funktionsweise

Das Sytem besteht aus einem in SWI-Prolog implementierten Kern und einer in Java™ realisierten GUI. Das Prologprogramm ist für die komplette Berechnung des min-Antwortmodells verantwortlich. Der Benutzer kommt damit nicht in Berührung, da er nur mit der GUI interagiert. Die verwendete Schnittstelle zwischen Prolog und Java™ heißt JPL. Die GUI selbst wurde mit dem GUI Maker von IntelliJ IDEA erstellt.

Das Javaprogramm besteht aus 3 Klassen. In der ersten Klasse Main.class wird das Programm aufgerufen und die Oberfläche initialisiert. Die zweite Klasse Gui.class beinhaltet den Aufbau der Oberfläche und alle ActionListener. Die dritte Klasse Calc.class führt die Schnittstelle aus, gibt die Anweisung zum Berechnen und gibt das Ergebnis zurück.

Beim Arbeiten mit dem System laufen folgende Vorgänge ab. Wenn alle benötigten Daten eingegeben oder geladen sind kann das entsprechende min-Antwortmodell berechnet werden. Dazu wird die Java-Prolog Schnittstelle initialisiert. Dann lädt das System den in Prolog geschriebenen Kern. Zur Übergabe von Kontext und Eingabeprogramm werden nun zur Laufzeit temporär eine Kontextdatei und eine Programmdatei aus den vom Benutzer eingegebenen Tabellen erstellt, und diese ins Prologprogramm eingelesen. Anschließend werden die temporären Dateien wieder gelöscht.

Das Javaprogramm stellt nun die Anfrage, dass das min-Antwortmodell erstellt werden soll. Auf diese Berechnung des min-Antwortmodells gehen wir nun näher ein. Als erstes wird der AOC des Kontextes erstellt. Danach werden alle Modelle der Regeln des eingegebenen Programms bestimmt, indem die Regeln nacheinander abgearbeitet werden. Dazu wird zunächst die Regel aus dem Programm ausgelesen und danach wird geprüft, ob es sich um eine erweiterte oder eine normale Regel handelt. Wenn es sich um eine erweiterte Regel handelt, wird die Komplementärmenge aller Modelle des letzten Teils, also dem *wenn nicht*-Teil der Regel, gebildet. Danach wird geprüft, welche Modelle im AOC vorkommen, also welches keine Modelle von der Klausel der Regel sind. Dies ist einfach möglich, da im ganzen System nur mit Intenten gerechnet wird, also Modelle und AOC in ein- und derselben Form vorliegen. Danach wird das erweiterte Programm als normales Progamm zurückgegeben und in ihre Klauseln geteilt. Von beiden Klauseln werden die Modelle gebildet. Nun wird geprüft, ob die Modelle der zweiten auch in der ersten Klausel auftreten. Alle Modelle, die in der ersten und zweiten Klausel vorkommen, werden dann in eine gesonderte Liste geschrieben. Von dieser gesonderten Liste werden nun eigentlich die Maxima, da aber der AOC invers ist, die Minima bestimmt. Diese Minima werden danach geprüft, ob sie ebenfalls im AOC vorkommen, wenn ja, ist das min-Antwortmodell gefunden. Die-

se Ergebnisse werden dann wieder an das Javaprogramm zurückgegeben und können dargestellt und ggf. abgespeichert werden.

Schlußbemerkung

In diesem Projekt wurde ein neuer Ansatz zum automatisierten Schließen mit formalen Begriffen erstmalig für Evaluationszwecke implementiert. Die durchgeführten Experimente mit dem System bestätigen wie erwartet, dass der Ansatz grundsätzlich ein intuitiv einleuchtendes Schlussverfahren liefert.

Literatur

[1] *IntelliJ IDEA 4.5 build #2233.*

[2] *Java(™) 2 Runtime Environment, Standard Edition (build 1.4.0_01-b03).* http://java.sun.com.

[3] Baumann, Rüdeger: *Informatik für die Sekundarstufe.* Ernst Klett Schulbuchverlag, 1996.

[4] Blackburn, Patrick, Johan Bos und Kristina Striegnitz: *Learn Prolog Now!* http://www.coli.uni-sb.de/~kris/learn-prolog-now, Oktober 2003.

[5] Ganter, Bernhard und Rudolf Wille: *Formal Concept Analysis — Mathematical Foundations.* Springer, Berlin, 1999.

[6] Hitzler, Pascal: *Automatisiertes Schließen mit formalen Begriffen: Mathematische Grundlagen.* In: *MINT Band 11.* 2005. In diesem Band.

[7] Schmitt, Dietmar: *Zur Entstehung der Programmiersprache Prolog — Herkünfte und Auswirkungen.* http://www.dietmarschmitt.de/essays/SGI/Prolog/, Mai 2005.

[8] Singleton, Paul, Fred Dushin und Jan Wielemaker: *JPL Java/Prolog interface 3.0.3.* http://www.swi-prolog.org.

[9] Steinmüller, Johannes: *Expertensysteme.* Vorlesung an der TU Chemnitz Sommersemester 2003, http://www.tu-chemnitz.de/~stj/stj.html, Januar 2005.

[10] Wielemaker, Jan: *SWI-Prolog version 5.4.5.* http://www.swi-prolog.org.

[11] Wielemaker, Jan: *SWI-Prolog 5.2 Reference Manual.* University of Amsterdam, 2003, http://www.swi-prolog.org, Oktober 2003.