Automatisiertes Schließen mit formalen Begriffen

Mathematische Grundlagen

Pascal Hitzler*

AIFB, Universität Karlsruhe

1 Einführung

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. Im vorliegenden Fall sollte ein neuer, in [Hit04] vorgestellter Ansatz zum automatisierten Schließen über Begriffshierarchien erstmals implementiert werden, mit dem Ziel, die Plausibilität dieses Ansatzes zu zeigen. Dieses Dokument führt knapp in die mathematischen Grundlagen ein, die zum Verständnis der Problemstellung und zur Implementierung benötigt werden. Es werden keine mathematischen Vorkenntnisse benötigt, die über Schulwissen hinausgehen. Für manche der Aufgaben wird Grundwissen in der Programmiersprache Prolog vorausgesetzt. Sie führen auf eine Implementierung hin. Die Implementierung des Gesamtsystems ist in [FH05] beschrieben.

2 Formale Kontexte und Begriffe; Begriffsverbände

2.1 Formale Kontexte

Ausgangspunkt ist ein *formaler Kontext*. Am besten stellt man ihn sich vor als eine Tabelle, in der Gegenstände (je eine Zeile) und Merkmale (je eine Spalte) eingetragen sind (Beispiele gibt's in [GW99]). Die Einträge in den Tabellen geben lediglich an, ob ein Gegenstand das entsprechende Merkmal hat.

2.1 Definition
Ein *formaler Kontext* besteht aus einer Menge G von *Gegenständen*, einer Menge

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

M von *Merkmalen* sowie einer vollständigen Angabe, welcher Gegenstand welche Merkmale hat. Letzteres drücken wir aus durch eine Menge I von Paaren (g, m), wobei $(g, m) \in I$ genau dann, wenn der Gegenstand g das Merkmal m hat. (Wir schreiben $I \subseteq G \times M$ um auszudrücken, dass I eine Menge von solchen Paaren ist.)

Aufgabe 2.1
Stelle für irgendwelche Dinge aus Deinem persönlichen Umfeld Kontexttabellen auf. Gegenstände könnten z.B. sein: Kochrezepte (Merkmale: welche Gewürze werden verwendet); Einkaufszettel (Merkmale: die Artikel auf den Einkaufszetteln); Kinofilme (Merkmale: bekannte Darsteller); natürliche Zahlen bis 20 (Merkmale: prim, gerade, ungerade, etc.)

Aufgabe 2.2
Überlege Dir verschiedene Möglichkeiten, formale Kontexte in Prolog darzustellen, z.B. mit Hilfe von Listen.

2.2 Formale Begriffe

Jeder Menge von Gegenständen A ordnen wir nun die Menge aller Merkmale zu, die diese Gegenstände gemeinsam haben. Wir bezeichnen diese Merkmalsmenge mit A'. Ebenso ordnen wir jeder Menge von Merkmalen B die Menge B' aller Gegenstände zu, die alle Merkmale aus B haben.

2.2 Definition
Ist $A \subseteq G$, dann definieren wir den *Intent* A' von A durch

$$A' = \{m \in M \mid (g, m) \in I \text{ für alle } g \in A\}.$$

Ist $B \subseteq M$, dann definieren wir den *Extent* B' von B durch

$$B' = \{g \in G \mid (g, m) \in I \text{ für alle } m \in B\}.$$

Aufgabe 2.3
Betrachte Beispiele aus Aufgabe 2.1 oder aus [GW99]. Lege beliebige Mengen A von Gegenständen und Mengen B von Merkmalen des jeweiligen Beispiels fest, und bestimme A' und B'.

Aufgabe 2.4
Schreibe ein Prolog-Programm, das bei gegebenem formalem Kontext zu jeder Menge von Merkmalen den zugehörigen Extent liefert.

Wir definieren nun, was ein formaler Begriff ist. Dazu gehen wir von einer Menge A von Gegenständen aus. Dann bestimmen wir zunächst den Intent A' von A, d.h. die Menge aller Merkmale, die alle Gegenstände in A haben. Dann bestimmen wir wiederum den Extent A'' von A', d.h. die Menge aller Gegenstände, die alle Merkmale aus A' haben. Ein Paar (A'', A') nennen wir eine *formalen Begriff*.

2.3 Definition
Ist ein formaler Kontext mit Gegenständen G gegeben, dann ist die Menge $\mathcal{B}$ aller zugehörigen *formalen Begriffe* definiert als die Menge aller Paare (A'', A') für $A \subseteq G$.

Beachte, dass auch für die leere Menge $\emptyset$ immer ein zugehöriger Begriff existiert: Es ist $\emptyset' = M$, also ist der zugehöriger Begriff (M', M).

Es ist auch zu bemerken, dass für verschiedene Mengen A_1 und A_2 von Gegenständen auch $A_1' = A_2'$ möglich ist (Aufgabe: finde dazu ein Beispiel). In diesem Fall ist $(A_1'', A_1') = (A_2'', A_2')$, d.h. es handelt sich um ein und denselben formalen Begriff!

Aufgabe 2.5
Betrachte Beispiele für formale Kontexte wie in Aufgabe 2.3. Lege beliebige Mengen A von Gegenständen fest und bestimme die zugehörigen formalen Begriffe.

Aufgabe 2.6
Wähle einen kleinen *formalen Kontext (z.B.* 4 *Gegenstände und* 6 *Merkmale) und bestimme* alle *zugehörigen formalen Begriffe.*

Aufgabe 2.7
Schreibe ein Prolog-Programm, das zu gegebenem formalem Kontext alle *formalen Begriffe findet. Es soll dabei jeden Begriff nur einmal angeben.*

Der folgende Satz kann bewiesen werden.

2.4 Satz
Sei ein formaler Kontext mit Gegenständen G und Merkmalen M gegeben, und sei $\mathcal{B}$ die Menge aller zugehörigen formalen Begriffe. Dann gelten die folgenden Aussagen.

(i) $\mathcal{B}$ ist genau die Menge aller Paare (B', B''), für die $B \subseteq M$ ist.

(ii) $\mathcal{B}$ ist genau die Menge aller Paare (A, B) (mit $A \subseteq G$ und $B \subseteq M$), für die $A' = B$ und $B' = A$ gilt.

Aufgabe 2.8
Mache Dir die Aussagen von Satz 2.4 anhand von Beispielen klar.

In Teil 3 werden wir nicht alle Begriffe eines Kontexts betrachten, sondern nur Begriffe, die aus einem Gegenstand (oder einem Merkmal) entstehen.

2.5 Definition
Ist $g \in G$ ein Gegenstand, so nennen wir die Menge $\{g\}'$ aller Merkmale von g den *Gegenstandsintent* von g, und den Begriff $(\{g\}'', \{g\}')$ den *Gegenstandsbegriff* von g. Ist $m \in M$ ein Merkmal, so nennen wir die Menge $\{m\}'$ aller Gegenstände mit Merkmal m den *Merkmalsextent* von m, und den Begriff $(\{m\}', \{m\}'')$ den *Merkmalsbegriff* von m.

Aufgabe 2.9
Bestimme für Dein Beispiel aus Aufgabe 2.6 die Menge aller zugehörigen Gegenstands- und Merkmalsbegriffe.

Aufgabe 2.10
Modifiziere Dein Programm aus Aufgabe 2.7, so dass es nur die Gegenstands- und Merkmalsbegriffe ausgibt.

2.3 Begriffsverbände

Die formalen Begriffe eines Kontexts lassen sich auf intuitive Weise miteinander vergleichen, je nachdem, ob sie mehr- oder weniger Gegenstände umfassen.

2.6 Definition
Sind (A_1, B_1) und (A_2, B_2) Begriffe, dann schreiben wir $(A_1, B_1) \leq (A_2, B_2)$ falls $A_1 \subseteq A_2$ ist, d.h. falls es in A_2 mindestens genau so viele Gegenstände gibt, wie in A_1.

Man kann auf diese Weise alle Begriffe eines Kontexts ordnen und graphisch darstellen (siehe die Abbildungen in [GW99]). Diese Ordnungsstruktur nennt man dann den *Begriffsverband* des Kontextes. Um diese Abbildungen übersichtlich zu beschriften, beschriftet man nur die Gegenstands- und Merkmalsbegriffe, und zwar mit dem jeweiligen Gegenstand bzw. Merkmal.

Aufgabe 2.11
Stelle den Begriffsverband zu dem von Dir gewählten Kontext aus Aufgabe 2.6 graphisch dar und beschrifte ihn wie oben beschrieben. Mache Dir klar, warum diese Art der Beschriftung sinnvoll ist (Hinweis: Wie stehen die Begriffe über einem Gegenstandsbegriff in Beziehung zu diesem Gegenstand?).

Es gibt umsonst erhältliche Software, um Begriffsverbände graphisch darzustellen. Am besten ist wohl `ConExp`, erhältlich unter `http://www.mathematik.tu-darmstadt.de/ags/ag1/Software/ConExp/`.

Aufgabe 2.12
Mache Dich mit dem Programm `ConExp` vertraut, indem Du für Deine Beispiele von oben Begriffsverbände erstellst.

Man kann folgenden Satz beweisen.

2.7 Satz
Sei ein formaler Kontext gegeben. Dann gelten die folgenden Aussagen.

(i) Im Begriffsverband gilt $(A_1, B_1) \leq (A_2, B_2)$ genau dann, wenn $B_2 \subseteq B_1$ (also „andersum“!).

(ii) Der Begriffsverband hat einen kleinsten und einen größten Begriff. Wir nennen sie $\perp$ (*Bottom*) und $\top$ (*Top*). Es gilt $\perp = (\emptyset'', \emptyset')$ und $\top = (\emptyset', \emptyset'')$.

(iii) Zu jedem Paar von Begriffen gibt es einen (eindeutigen) kleinsten Begriff, der über beiden liegt (genannt das *Supremum* der beiden Begriffe), d.h. wenn $C_1 = (A_1, B_1)$ und $C_2 = (A_2, B_2)$ ist, dann gibt es einen kleinsten Begriff $C = (A, B)$ mit $C_1 \leq C$ und $C_2 \leq C$. Wir schreiben $C = C_1 \vee C_2$.

(iv) Zu jedem Paar C_1, C_2 von Begriffen gibt es einen (eindeutigen) größten Begriff C, der unter beiden liegt (genannt das *Infimum* der beiden Begriffe). Wir schreiben $C = C_1 \wedge C_2$.

(v) Es gilt $(A_1, B_1) \wedge (A_2, B_2) = (A_1 \cap A_2, (B_1 \cup B_2)'')$ und $(A_1, B_1) \vee (A_2, B_2) = ((A_1 \cup A_2)'', B_1 \cap B_2)$.

Aufgabe 2.13
Mache Dir die Aussagen von Satz 2.7 anhand von Beispielen klar.

Aufgabe 2.14
Schreibe Programme mit folgender Funktionalität über beliebig gegebenen formalen Kontexten (G, M, I). Die Ein-/Ausgabe soll zur Laufzeit erfolgen und nach einer Ausgabe soll immer eine neue Eingabe möglich sein. (Das Programm soll also in einer Schleife laufen.)

(a) Eingabe: Eine Merkmalsmenge B. Ausgabe: Alle Begriffe, die grösser, und alle Begriffe, die kleiner als (B', B'') sind. (Die Ausgabe kann sich auf die Intents beschränken.)

(b) Eingabe: Zwei Merkmalsmengen B_1 und B_2. Ausgabe: $(B_1', B_1'') \wedge (B_2', B_2'')$ und $(B_1', B_1'') \vee (B_2', B_2'')$.

(c) Eingabe: Ein Element $a \in G \cup M$. Ausgabe: Alle Gegenstands- und Merkmalsbegriffe, die unter dem von a erzeugten Begriff liegen. Die Ausgabe der Begriffe sollte am Besten durch Angabe des Gegenstandes bzw. Merkmals erfolgen, der den entsprechenden Begriff erzeugt.

Wir betrachten nun statt des gesamten Begriffsverbandes nur die Teilmenge, die aus Top, Bottom und allen Gegenstands- und Merkmalsbegriffen besteht. Diese Menge ist ebenfalls geordnet, und zwar durch die Ordnung, die sie vom Begriffsverband erhält. Wir nennen diese Ordnungsstruktur die *Galois Teilhierarchie* (oder auch einfach *AOC* — von *Attribute and Object Concepts*) des Begriffsverbandes. Wir können die Elemente eines AOC durch die Elemente in $G \cup M \cup \{\perp, \top\}$ bezeichnen.

Aufgabe 2.15
Finde (z.B. mit Hilfe von `ConExp`) ein Beispiel fuer einen AOC, in dem zwei Elemente nicht notwendigerweise ein Infimum und Supremum besitzen.

Da zwei Elemente des AOC nicht immer ein Supremum und Infimum besitzen, führen wir folgende Begriffe ein: Für eine Menge $\{x_1, \ldots, x_n\}$ schreiben wir $\mathsf{mub}\{x_1, \ldots, x_n\}$ für die Menge aller Elemente, die über allen $x_1, \ldots, x_n$ liegen, aber gleichzeitig kleinstmöglich mit dieser Eigenschaft sind. Anders gesagt: Wir betrachten die Menge aller Elemente, die über allen $x_1, \ldots, x_n$ liegen, und suchen dann aus dieser Menge alle die heraus, die in dieser Menge minimal sind. Die Menge $\mathsf{mub}\{x_1, \ldots, x_n\}$ heißt die *Menge aller minimalen oberen Schranken* (von *minimal upper bounds*) von $\{x_1, \ldots, x_n\}$. Genauso definieren wir auch $\mathsf{mlb}\{x_1, \ldots, x_n\}$, die *Menge aller maximalen unteren Schranken* (*maximal lower bounds*) von $\{x_1, \ldots, x_n\}$.

Aufgabe 2.16
Mache Dir die gerade eingeführten Begriffe anhand von Beispielen klar.

Aufgabe 2.17
Schreibe ein Programm, das zu einer gegebenen Menge von Elementen des AOC, d.h. zu einer gegebenen Teilmenge von $G \cup M \cup \{\bot, \top\}$ alle minimalen oberen Schranken und alle maximalen unteren Schranken findet. Die Ein- und Ausgabe soll zur Laufzeit erfolgen.

3 Logisches Schließen mit formalen Begriffen

Im Folgenden sei (G, M, I) ein (endlicher) formaler Kontext und $(D, \leq)$ der zugehörige AOC in *umgedrehter* Ordnung. D.h. fuer zwei Intenten A und B im AOC gilt $A \leq B$ genau dann, wenn $A \subseteq B$. Durch dieses „Umdrehen“ des AOC kommen wir der Intuition näher, dass größere Elemente mehr Information enthalten.

Wir betrachten Teilmengen von D, die wir *Klauseln* nennen. Intuitiv verstehen wir die Elemente einer solche Menge $\{a_1, \ldots, a_n\} \subseteq D$ als durch ein logisches *oder* verknüpft. D.h. $\{a_1, \ldots, a_n\}$ steht für „a_1 *oder* a_2 *oder* $\ldots a_n$“.

Sei $X \subseteq D$ eine Klausel. Ein Element $w \in D$ heißt ein *Modell* für X wenn es ein $x \in X$ gibt mit $x \leq w$. In einem solchen Fall schreiben wir $w \models X$.

Aufgabe 3.1
Schreibe ein Programm wie folgt: Eingabe: Eine Klausel. Ausgabe: (a) Alle Modelle der Klausel. (b) Alle minimalen Modelle der Klausel.

Eine Menge T von Klauseln nennen wir eine *Theorie*. Ein Element $w \in D$ heißt ein *Modell* für T wenn $w \models X$ für alle $X \in T$ gilt. Wir schreiben $w \models T$ in diesem Fall.

Aufgabe 3.2
Schreibe ein Programm wie folgt: Eingabe: Eine Theorie. Ausgabe: (a) Alle Modelle der Theorie. (b) Alle kleinsten Modelle der Theorie.

Wir nennen eine Klausel X eine *logische Konsequenz* einer Theorie T, wenn jedes Modell von T auch ein Modell von X ist.

Aufgabe 3.3
Schreibe ein Programm wie folgt: Eingabe: Eine Theorie T und eine Klausel X. Das Programm soll Yes *antworten, falls $T \models X$, und* No *andernfalls.*

Ein *Programm* über D ist eine Menge von *Regeln* der Form $X \leftarrow Y$, wobei X und Y Klauseln sind. Intuitiv steht $X \leftarrow Y$ für „Wenn Y, dann auch X“.

Ein Element $w \in D$ heißt ein *Modell* eines Programms P, wenn für jede Regel $X \leftarrow Y$ in P gilt:

$$\text{Jedes } w \in D \text{ mit } w \models Y \text{ erfüllt auch } w \models X.$$

Wir schreiben $w \models P$ in diesem Fall.

Aufgabe 3.4
Schreibe ein Programm, das zu gegebenem Programm die Menge aller (kleinsten) Modelle des Programms berechnet.

Eine Klausel X heisst *logische Konsequenz* eines Programms P, geschrieben $P \models X$, wenn jedes Modell von P auch Modell von X ist.

Aufgabe 3.5
Schreibe ein Programm, das als Eingabe ein Programm P und eine Klausel X akzeptiert und bestimmt, ob $P \models X$ gilt.

3.1 Bemerkung
Bei den Definitionen aus Abschnitt 3 handelt es sich um Spezialfälle von in [RZ01] eingeführter Terminologie.

4 Nichtmonotones Schließen mit formalen Begriffen

Wie in Abschnitt 3 sei (G, M, I) ein (endlicher) formaler Kontext und $(D, \leq)$ der zugehörige AOC in umgedrehter Ordnung.

Eine *erweiterte Regel* ist von der Form $X \leftarrow Y, \sim Z$, wobei X, Y, Z Klauseln sind. Intiutiv verstehen wir eine solche Regel als „Wenn Y, und wenn *nicht notwendig* Z, dann auch X“. Ein *erweitertes Programm* ist eine Menge von erweiterten Regeln.

Sei P ein erweitertes Programm und $w \in D$. Wir definieren P/w als die Menge aller Regeln $X \leftarrow Y$ für die es eine erweiterte Regel $X \leftarrow Y, \sim Z$ in P mit $w \not\models Z$ gibt. Das Programm P/w besteht aus (nicht-erweiterten) Regeln, ist also ein Programm in Sinne von Abschnitt 3.

Ein Element $w \in D$ heißt ein *Antwortmodell* eines erweiterten Programms P, wenn $w \models P/w$ gilt.

Aufgabe 4.1
Schreibe ein Prolog-Programm, das zu gegebenem erweitertem Programm P die Menge aller Antwortmodelle findet.

Ein Element $w \in D$ heißt ein *min-Antwortmodell* eines erweiterten Programms P, wenn w minimal in der Menge aller $v \in D$ mit $v \models P/w$ ist.

Aufgabe 4.2
Schreibe ein Prolog-Programm, das zu gegebenem erweitertem Programm P die Menge aller min-Antwortmodelle findet.

Eine Klausel X heißt eine *skeptische Konsequenz* eines erweiterten Programms P, wenn jedes min-Antwortmodell von P ein Modell von X ist. Wir schreiben $P \models_s X$ in diesem Fall.

Eine Klausel X heißt eine *gutgläubige Konsequenz* eines erweiterten Programms P, wenn es ein min-Antwortmodell von P gibt, das Modell von X ist. Wir schreiben $P \models_c X$ in diesem Fall.

Aufgabe 4.3
Schreibe ein Prolog-Programm, das zu gegebenem erweitertem Programm P und gegebener Klausel X entscheidet, ob $P \models_s X$ oder $P \models_c X$ gilt.

4.1 Bemerkung
Die Definitionen dieses Abschnitts sind Spezialfälle von Terminologie aus [Hit04].

Literatur

[FH05] Fritsche, Karl und Pascal Hitzler: *Automatisiertes Schließen mit formalen Begriffen: Implementierung.* In: *MINT Band 11.* 2005. In diesem Band.

[GW99] Ganter, Bernhard und Rudolf Wille: *Formal Concept Analysis — Mathematical Foundations.* Springer, Berlin, 1999.

[Hit04] Hitzler, Pascal: *Default Reasoning over Domains and Concept Hierarchies.* In: Biundo, Susanne, Thom Frühwirth und Günther Palm (Herausgeber): *Proceedings of the 27th German conference on Artificial Intelligence, KI'2004, Ulm, Germany, September 2004*, Band 3238 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 351–365. Springer, Berlin, 2004.

[RZ01] Rounds, William C. und Guo-Qiang Zhang: *Clausal Logic and Logic Programming in Algebraic Domains.* Information and Computation, 171(2):156–182, 2001.