Praktikum zur Vorlesung

Angewandte Logik

Prof. Torsten Schaub, Anita Dieckhoff, Daniel Schurtzmann, Martin Gebser, Robert Engelmann, Sven Papenfuß

Universität Potsdam (Wissensverarbeitung und Informationssysteme) — Sommersemester 2013

Praktikumsaufgabe 2 (Minimale Modelle)

In dieser Praktikumsaufgabe sollen Sie ein Prädikat implementieren, dass alle *teilmengen-minimalen* Modelle einer Menge von aussagenlogischen Formeln ermitteln kann. Am einfachsten lässt sich dies an einem Beispiel wie der folgenden Formelmenge $\mathcal{F}$ erklären:

$$\mathcal{F} = \{((\neg p \wedge q) \rightarrow r), (q \vee r)\}$$

Die Modelle für $\mathcal{F}$, dargestellt als Teilmengen von $\{p, q, r\}$ mit den wahren Atomen, sind:

$$\begin{aligned} M_1 &= \{p, q\} \\ M_2 &= \{p, r\} \\ M_3 &= \{p, q, r\} \\ M_4 &= \{q, r\} \\ M_5 &= \{r\} \end{aligned}$$

Beobachten Sie, dass das Modell M_5 eine echte Teilmenge von M_2, M_3 und M_4 ist. Das bedeutet, dass die Modelle M_2, M_3 und M_4 nicht teilmengen-minimal sind. Andererseits bildet keine echte Teilmenge von M_1 oder M_5 ein Modell für $\mathcal{F}$, und somit sind $M_1 = \{p, q\}$ und $M_5 = \{r\}$ die teilmengen-minimalen Modelle für $\mathcal{F}$.

In der Folge verstehen wir den Begriff "minimal" immer bzgl. der Teilmengenbeziehung und lassen das Präfix "teilmengen-" weg.

Repräsentation in Prolog. Die in aussagenlogischen Formeln vorkommenden Atome $(a, b, c, \ldots, p, q, r, \ldots, \mathit{der}, \mathit{die}, \mathit{das}, \ldots)$ repräsentieren wir durch Prolog-Bezeichner ($\texttt{a}, \texttt{b}, \texttt{c}, \ldots, \texttt{p}, \texttt{q}, \texttt{r}, \ldots, \texttt{der}, \texttt{die}, \texttt{das}, \ldots$). Als Konnektive lassen wir den unären Junktor $\neg$ sowie die binären Junktoren $\wedge$, $\vee$, $\rightarrow$ und $\leftrightarrow$ unter ihrer Standardsemantik ($\neg^*$, $\wedge^*$, $\vee^*$, $\rightarrow^*$, $\leftrightarrow^*$; vgl. Vorlesungsfolien) zu. In Prolog repräsentieren wir diese Konnektive durch (benutzerdefinierte) Operatoren, die in der folgenden Tabelle in absteigender Reihenfolge ihrer Bindungskraft aufgelistet sind:

Junktor	Operator	Typ
$\neg$	`~`	`fy`
$\wedge$	`&`	`xfy`
$\vee$	`?`	`xfy`
$\rightarrow$	`==>`	`xfy`
$\leftrightarrow$	`<=>`	`xfy`

Auf dieser Grundlage repräsentieren wir aussagenlogische Formeln (z.B.: $((\neg p \wedge q) \rightarrow r)$) in Prolog als Terme aus durch Operatoren verbundenen Bezeichnern (z.B.: `~p & q ==> r`). Eine explizite Klammerung können wir in Prolog weglassen, wenn sie mit der durch die Bindungskraft der Operatoren implizit gegebenen Klammerung übereinstimmt. Die folgenden Unifikationsergebnisse vermitteln einen Eindruck hiervon:

```
?- (F ==> G) = (~p & q ==> r).
    F = (~p & q)    G = r
```

```
?- (F & G) = (~p & q ==> r).
    false.
```

Beobachten Sie, dass die "richtige" Klammerung einer Formel bei der Unifikation "automatisch" ermittelt wird. Wenn Sie auf vorgegebenen Formeln arbeiten, können Sie diese also mittels Unifikation geeignet in ihre Bestandteile zerlegen. (Um eine explizite Klammerung müssen Sie sich nur dann u.U. kümmern, wenn Sie selbst Formeln repräsentieren.)

Eine Interpretation stellen wir als Liste der wahren Atome dar. Z.B. repräsentiert `[p,q]` die Interpretation M_1, die p und q auf w sowie alle anderen Atome auf f abbildet. Entsprechend der Standardsemantik von Junktoren gelten $M_1 \models_a ((\neg p \wedge q) \rightarrow r)$, $M_1 \models_a (q \vee r)$ und daher $M_1 \models_a \{((\neg p \wedge q) \rightarrow r), (q \vee r)\}$.

Ihre Aufgabe ist die Implementation des folgenden Prädikats:

```
minModel(+ListOfFormulas,-MinimalModel)
```

Für eine in `ListOfFormulas` gegebene Formelmenge $\mathcal{F}$ soll der Rückgabewert `MinimalModel` einem minimalen Modell für $\mathcal{F}$ entsprechen, wobei die wahren Atome in beliebiger Reihenfolge und Duplizität aufgelistet werden können. Alle minimalen Modelle für $\mathcal{F}$ sollen über Backtracking ermittelbar sein. Für $\mathcal{F} = \{((\neg p \wedge q) \rightarrow r), (q \vee r)\}$ kann ein Aufruf die gewünschten Ergebnisse z.B. wie folgt liefern, wobei die Reihenfolge (und Wiederholungen) von minimalen Modellen irrelevant sind:

```
?- minModel([~p & q ==> r, q ? r],MinimalModel).
    MinimalModel = [r] ;
    MinimalModel = [p,q] ;
    false.
```

Natürlich muss Ihre Implementation des Prädikats `minModel/2` so allgemein gehalten werden, dass es für beliebige Formelmengen die richtigen Ergebnisse liefert.

Framework. Auf der Moodle-Seite zur Vorlesung finden Sie die Datei `minimal.pl`, in der Sie das Prädikat `minModel/2` implementieren sollen. Die Datei enthält 12 Testfälle in Form von Fakten des Prädikats `formulas(?ID,?ListOfFormulas)`. Mit folgendem Aufruf können Sie das Prädikat `minModel/2` z.B. auf den sechsten Testfall anwenden:

```
?- formulas(6,ListOfFormulas), minModel(ListOfFormulas,MinimalModel).
```

Mit der Anfrage "`?- test_all.`" können Sie die Überprüfung aller vorgegebenen Testfälle starten, wobei jeder Testfall mit der Ausgabe von "`success`" (nicht "`failure`"!) beantwortet werden muss. Wenn Ihre Implementation dieses Kriterium nicht erfüllt, dann ist sie als Lösung nicht akzeptabel. (Beim Testen auf Korrektheit verwenden wir neben den vorgegebenen Testfällen auch weitere, die Ihnen vorher unbekannt sind. Auch unsere zusätzlichen Testfälle müssen von Ihrer Implementation richtig behandelt werden.)

Formalitäten. Sie können die Praktikumsaufgabe in Gruppen von **bis zu drei** Studenten gemeinsam bearbeiten. Verschiedene Gruppen müssen verschiedene Lösungen einreichen. Bei Plagiaten wird die Praktikumsaufgabe für alle beteiligten Gruppen als "nicht

bestanden" gewertet. Reichen Sie Ihre Lösung bitte bis zum **06.06.13** über die YETI-Plattform ein. Registrieren Sie dort **alle** Ihre Gruppenmitglieder, und achten Sie darauf, dass Sie Ihre Lösung als Datei mit dem Namen `minimal.pl` einreichen, wobei der Dateiname ausschließlich Kleinbuchstaben enthält. Nach der Einreichung wird Ihre Lösung mit kurzer zeitlicher Verzögerung automatisch getestet. Dass Sie alle Testfälle erfolgreich behandelt haben, erkennen Sie an der Ausgabe "`success`" für jeden Testfall. Sollten ein oder mehrere Testfälle mit "`failure`" beantwortet werden, korrigieren Sie Ihre Implementation bitte umgehend selbstständig (bzw. in Rücksprache mit uns, falls Ihnen keiner der falsch behandelten Testfälle vorliegt).

Bei Fragen oder Problemen zu der Praktikumsaufgabe wenden Sie sich in der Folge bitte persönlich an uns (z.B. in den Sprechstunden donnerstags, 14:00–16:00 Uhr, im Poolraum 3.04.1.03) oder per Mail an:

`lp1@cs.uni-potsdam.de`

Empfehlungen und Hinweise:

- Überlegen Sie sich zunächst eine allgemeine Vorgehensweise, bevor Sie Ihre Idee in einem Prolog-Programm umsetzen. Bedenken Sie dabei, dass Formeln beliebiger Struktur über Unifikation sukzessiv in ihre Bestandteile zerlegt werden können. Dies erleichtert die Evaluation bzgl. einer vorgegebenen Interpretation, während eine Erzeugung von Modellen ausgehend von Teilformeln nicht ratsam ist. Nicht zuletzt müssen Interpretationen bzw. (minimale) Modelle für Formelmengen mit beliebigen Bezeichnern für aussagenlogische Variablen ermittelbar sein, d.h. jedwede Spezialisierung auf die Bezeichner in den vorgegebenen Testfällen ist unzulässig.

- Halten Sie Ihr Programm und die Implementation von Prädikaten so knapp wie möglich. Sehr lange Programme bzw. lange Regelkörper sind oft ein Zeichen dafür, dass die Gliederung in Teilaufgaben ungeeignet ist.

- Entwickeln Sie Ihr Prolog-Programm selbst und versuchen Sie nicht, die Aufgabe durch exzessiven Gebrauch von Google oder Built-Ins zu lösen, da Sie dadurch wenig lernen und höchstwahrscheinlich nicht zu einer eleganten Lösung kommen.

- Bei Fragen oder Problemen können Sie sich gern an uns wenden, z.B. in den Sprechstunden donnerstags, 14:00–16:00 Uhr, im Poolraum 3.04.1.03. Wir sind für alle Fragen offen und versuchen sie bestmöglich zu beantworten.

- Fangen Sie bald mit der Bearbeitung der Aufgabe an, damit Ihnen die Zeit nicht davonläuft. (Sollten Sie trotzdem Schwierigkeiten mit der Einhaltung des Termins haben, dann wenden Sie sich bitte an uns, anstatt eine beliebige Lösung zu kopieren!)

- Behalten Sie Ihren Quelltext für sich. Sie helfen Ihren Kommilitonen nicht, wenn Sie Ihren Quelltext weitergeben, da das Nachvollziehen eines fremden Prolog-Programms wesentlich schwieriger ist als ein eigenes zu entwerfen.