Grundlagen der Praktischen Informatik **Sommersemester 2024**
Dr. Stefan Siemer Georg-August-Universität Göttingen
Timo Specht Institut für Informatik

Übungsblatt 07

Übung

Abgabe bis Di., 04.06., 14 Uhr

Allgemein

Die Aufgaben müssen in **Dreiergruppen** abgegeben werden. Vierergruppen sind ebenfalls möglich.

Es ist **wichtig**, dass Sie sich an folgendes **Verfahren für die Abgabe** halten.
Die Lösungen werden in geeigneter Form in der Stud.IP-Veranstaltung Ihrer Übungsgruppe über das Vips-Modul hochgeladen. Sie müssen diese Abgaben nicht mit Markdown+AsciiMath erstellen. Sie können Ihre Bearbeitungen auch mit LaTeX formatieren, es ist aber auch die direkte Eingabe von Text oder der Upload von Text- und Bilddateien in gängigen Formaten möglich.

Weitere Hinweise zur Abgabe der Lösungen finden Sie in den Aufgabenstellungen.

Aufgabe 1 – 8 Punkte

KNF

Gegeben sei die folgende aussagenlogische Formel F mit den atomaren Aussagen A, B, C, D.

$$\big((C \wedge \neg D) \wedge \neg(\neg D \implies B)\big) \vee \big((\neg A \vee \neg B) \wedge (\neg A \implies \neg C)\big)$$

Formen Sie die Formel in Konjunktive Normalform (KNF) um und geben Sie die resultierende Klauselmenge an.
(8 Punkte)

Aufgabe 2 – 13 Punkte

Resolutionskalkül

Betrachten Sie die folgende aussagenlogische Formel.

$$F = (X \vee \neg Y) \wedge (\neg X \vee \neg Y \vee \neg Z) \wedge (\neg X \vee Z) \wedge (X \vee Y \vee Z) \wedge (Y \vee \neg Z)$$

1. Bestimmen Sie die Klauselmenge $\mathcal{K}_0$ der Formel F.
   (2 Punkte)

2. F ist nicht erfüllbar. Zeigen Sie das mit dem Resolutionskalkül der Aussagenlogik.

   Geben Sie für jedem Schritt i folgendes an.

   $$\mathcal{K}_i = \text{Res}(\mathcal{K}_{i-1}) = \mathcal{K}_{i-1} \cup \{R \mid R \text{ ist Resolvente von } K', K'' \in \mathcal{K}_{i-1}\}$$

   (11 Punkte)

   Hinweis

   Der letzte Schritt ist die Bildung der Resolvente der Klauseln $\{Z\}$ und $\{\neg Z\}$.

   Beispiel aus dem Skript

   $$\mathcal{K}_0 = \Big\{\{P, S\}, \{\neg P, \neg S\}, \{\neg S, A\}, \{\neg A, P\}, \{P\}, \{S\}\Big\}$$
   $$\mathcal{K}_1 = \text{Res}(\mathcal{K}_0) = \mathcal{K}_0 \cup \Big\{\{\neg S\} \mid \text{Resolvente von } \{\neg P, \neg S\}, \{P\}\Big\}$$
   $$\mathcal{K}_2 = \text{Res}(\mathcal{K}_1) = \mathcal{K}_1 \cup \Big\{\quad \emptyset \quad \mid \text{Resolvente von } \{\neg S\}, \{S\}\Big\}$$

Aufgabe 3 – 18 Punkte

Formeln und Terme

Betrachten Sie eine einfache arithmetische Sprache über der Individuenmenge der ganzen Zahlen $\mathbb{Z}$ mit folgender Signatur. Die arithmetischen Operationen und die Größenrelationen über $\mathbb{Z}$ werden als bekannt vorausgesetzt.

- Prädikate $\mathcal{P} = \{<, >\}$.
  - Alle Prädikate habe die Stelligkeit 2.
  - Es gibt $X < Y$ ist genau dann *wahr*, wenn X echt kleiner als Y ist.
  - Es gibt $X > Y$ ist genau dann *wahr*, wenn X echt größer als Y ist.
- Funktoren $\mathcal{F} = \{+, -\}$.
  - Alle Funktoren habe die Stelligkeit 2.
  - Der Funktor $+$ entspricht der Addition in $\mathbb{Z}$.
  - Der Funktor $-$ entspricht der Subtraktion in $\mathbb{Z}$.

Welche der folgenden Ausdrücke sind Formeln, welche Terme und welche gehören nicht zur Sprache? Begründen Sie jeweils Ihre Entscheidung.

Kann für einen Formel eine Wahrheitswert angegeben werden, gegeben Sie diesen an und begründen Sie Ihre Antwort.

Kann für eine Term das bezeichnete Objekt angegeben werden, geben Sie dieses an und begründen Sie Ihre Antwort.

1. $X + (-Y)$
2. $1 + 2 - X - Y$
3. $2 < \big((2 > 3) \vee (3 \geq 2)\big)$
4. $1 > \big(\exists X : (X < 0)\big)$
5. $2 + 3 = 5$
6. $(X > 2) \wedge (X + 2)$
7. $(Z > 2) \vee \big((Z - 1) < 0\big)$
8. $\forall X : \big(X \Rightarrow \big(X \cdot (2 + 3)\big)\big)$
9. $\exists X : \Big(\exists Y : \big((X > 2) \wedge \big(\neg(Y < 3)\big)\big)\Big)$
10. $6 + 1 - 8$
11. $\forall X : \big(\big(X \wedge (2 > 3)\big) \Rightarrow X\big)$
12. $\exists : \big((Z > 2) \vee \big((Z - 1) < 0\big)\big)$

(18 Punkte)

Aufgabe 4 – 36 Punkte

Sprachen und Formeln

1. Gegeben ist eine Sprache über der Individuenmenge der rationalen Zahlen $\mathbb{Q}$ mit folgender Signatur.

   - Prädikate $\mathcal{P} = \{=\}$.

     Das Prädikat $=$ hat die Stelligkeit 2 und es gilt $X = Y$ ist genau dann *wahr*, wenn X und Y das gleiche Objekt aus $\mathbb{Q}$ bezeichnen.
   - Funktoren $\mathcal{F} = \emptyset$.

   Geben Sie den Wahrheitswert der folgenden Formeln an und begründen Sie jeweils ihre Antwort.

   a) $\forall X : \big(\exists Y : (X = Y)\big)$
   (4 Punkte)

b) $\forall X : \big(\forall Y : (X = Y)\big)$
(4 Punkte)

c) $\exists X : \big(\forall Y : (X = Y)\big)$
(4 Punkte)

d) $\exists X : \big(\exists Y : (X = Y)\big)$
(4 Punkte)

2. Gegeben ist eine Sprache über der Individuenmenge der natürlichen Zahlen (inklusive 0) $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ mit folgender Signatur. Die arithmetischen Operationen über $\mathbb{N}_0$ sind wohlbekannt.

   - Prädikate $\mathcal{P} = \{=\}$.

     Das Prädikat = hat die Stelligkeit 2 und es gilt $X = Y$ ist genau dann *wahr*, wenn X und Y das gleiche Objekt aus $\mathbb{N}_0$ bezeichnen.

   - Funktoren $\mathcal{F} = \{+, \text{dreieck}, \text{quadrat}\}$.

     Der Funktor + hat die Stelligkeit 2 und entspricht der Addition in $\mathbb{N}_0$.

     Der Funktor dreieck hat die Stelligkeit 1. Es gilt $\text{dreieck}(0) = 0$ und sonst $\text{dreieck}(X) = \sum_{i=1}^{X} i$.

     Der Funktor quadrat hat die Stelligkeit 1. Es gilt $\text{quadrat}(0) = 0$ und sonst $\text{quadrat}(X) = \sum_{i=1}^{X} (2i - 1)$.

   Geben Sie den Wahrheitswert der folgenden Formeln an. Mit Begründung.

   a) $\exists X : \big(\exists Y : \big((\neg(X = Y)) \wedge (\text{quadrat}(X) = \text{quadrat}(Y))\big)\big)$
   (5 Punkte)

   b) $\forall X : \big(\exists Y : \big((\neg(X = Y)) \wedge (\text{quadrat}(X) = \text{quadrat}(Y))\big)\big)$
   (5 Punkte)

   c) $\forall X : \big(\exists Y : \big((\text{dreieck}(X) + \text{dreieck}(Y)) = \text{quadrat}(X)\big)\big)$
   (5 Punkte)

   d) $\Big(\forall Y : \Big(\neg\big((\text{dreieck}(X) + \text{dreieck}(Y)) = \text{quadrat}(X)\big)\Big)\Big)$
   (5 Punkte)

Praktische Übung

Abgabe der Prüfsumme bis Di., 04.06., 14 Uhr
Testat Di., 04.06. ab 18 Uhr

Hilfe zum Bearbeiten der praktischen Übungen können Sie grundsätzlich jeden Tag in den Rechnerübungen bekommen. Die Testate finden ebenfalls in **Dreiergruppen** und Vierergruppen statt. Dabei sind die Gruppen identisch zu denen, die auch die theoretischen Aufgaben zusammen bearbeiten. In diesem Fall reserviert nur ein Gruppenmitglied einen Termin. Es ist ausreichend, wenn nur **eine** Person aus der Gruppe eine Prüfsumme abgibt.

Abgabe der Prüfsumme

- Siehe vorherige Übungen.
- Übermitteln Sie die Prüfsumme mit dem Test *GdPI 07 - Testat*.

Vorbereitung

1. Hilfreich für diese Übung sind die Funktionen zur Bearbeitung von Listen aus `Prelude` (siehe `https://hackage.haskell.org/package/base-4.9.0.0/docs/Prelude.html`) und `Data.List` (siehe `https://hackage.haskell.org/package/base-4.9.0.0/docs/Data-List.html`).

   Machen Sie sich mit den Funktionen `map`, `filter`, `intersperse` und `intercalate` vertraut.

2. Für eine aussagenlogische Formel

   $$F = (L_{11} \vee L_{12} \vee \ldots \vee L_{1\,m_1}) \wedge (L_{21} \vee L_{22} \vee \ldots \vee L_{2\,m_2}) \wedge \ldots \wedge (L_{n1} \vee L_{n2} \vee \ldots \vee L_{n\,m_n})$$

   in konjunktiver Normalform (KNF) mit Literalen L_{ij} sind die Mengen der Literale der einzelnen Disjunktionen die **Klausel** von F.

   $$\begin{aligned} K_1 &= \{L_{11}, L_{12}, \ldots, L_{1\,m_1}\} \\ K_2 &= \{L_{21}, L_{22}, \ldots, L_{2\,m_2}\} \\ &\ldots \\ K_n &= \{L_{n1}, L_{n2}, \ldots, L_{n\,m_n}\} \end{aligned}$$

   Die **Klauselmenge** von F ist die Menge aller Klauseln $\{K_1, K_2, \ldots, K_n\}$.

   Durch Umkehrung diese Ansatzes kann aus einer Klauselmenge eine Formel in KNF mit dieser Klauselmenge konstruiert werden.

   Bemerkung

   Verschiedene Formeln in KNF können dieselbe Klauselmenge haben, aber aufgrund von Idempotenz und Kommutativität sind Formel mit derselben Klauselmenge äquivalent. regular language Beispiel

   Die folgenden Formeln haben die Klauselmenge $\{\{A\}, \{B, \neg C\}\}$ und sind nicht identisch, aber äquivalent.

   $$\begin{aligned} F &= (A \vee A) \wedge (B \vee \neg C) \\ G &= A \wedge (\neg C \vee B) \\ H &= A \wedge (B \vee \neg C \vee B) \end{aligned}$$

Aufgabe 1 – 25 Punkte

Aussagenlogik

Ein Literal, d.h. eine atomare Formel oder die Negation einer atomaren Formel, wird wie folgt als Datentyp modelliert.

```
data Literal
  = Atom {atomChar :: Char}
  | NegA {atomChar :: Char}
  deriving Eq
```

Einen Wert diese Datentyps erzeugt man über den jeweiligen Konstruktor (`Atom`, `NegA`). Zugriff auf den Bezeichner (`Char`) der atomaren Formal erhält man, wie üblich bei *record syntax*, über die Funktion `atomChar`.

```
> atomChar (Atom 'A')
'A'
> atomChar (NegA 'B')
'B'
```

Um eine kompakte Stringrepräsentation für Literale zu erhalten

```
> Atom 'A'
A
> NegA 'B'
!B
```

wird `Literal` wir folgt zu einer Instanz der Klasse `Show`.

```
instance Show Literal where
  show (Atom a) = [a]
  show (NegA a) = ['!', a]
```

Diese Definition der Funktion `show` benutzt *pattern matching*, d.h. es gibt mehrere Definition derselben Funktion, die jeweils für unterschiedliche Muster (*pattern*) der Parameter definiert sind. Beim Funktionsaufruf wird die Definition benutzt, bei der Argumente und Muster zusammenpassen (*matching*). Gibt es kein passendes Muster ist das ein Fehler.

Zur Modellierung einer Klausel (Menge von Literalen) und einer Formel in KNF (Klauselmenge) werden die folgenden Datentypen verwendet.

```
data Clause = Clause [Literal]
data CNF = CNF [Clause]
```

1. Machen Sie `Clause` und `CNF` zu Instanzen der Klasse `Show`, sodass Stringrepräsentation wie im folgenden Beispiel erzeugt werden.

   ```
   > let a = Clause [NegA 'A', Atom 'B']
   > let b = Clause [NegA 'B', Atom 'C']
   > a
   (!A v B)
   > b
   (!B v C)
   > let c = CNF [a,b]
   > c
   (!A v B) ^ (!B v C)
   ```

(5 Punkte)

Hinweis. `String = [Char]` und `intercalate`.

2. Geben Sie Funktionen

```
fromClause :: Clause -> [Literal]
fromCNF :: CNF -> [Clause]
```

an, sodass Klausel auf die enthaltenden Listen von Literalen und Formel in KNF auf die enthaltenden Listen von Klauseln abgebildet werden.
(5 Punkte)

Hinweis. *Pattern matching.*

3. Definieren Sie die Funktion

```
alphaL :: [Literal] -> Literal -> Bool
```

die eine Funktion (`Literal -> Bool`) für eine Belegung von Literalen mit Wahrheitswerten zurückliefert. Die zurückgelieferte Funktion belegt ein Literal genau dann mit `True`, wenn es in der übergebenen Liste von Literalen enthalten ist.
(5 Punkte)

Beispiel

```
> let alpha = alphaL [NegA 'A', Atom 'B', NegA 'C']
> alpha (Atom 'B')
True
> alpha (NegA 'B')
False
```

4. Definieren Sie Funktionen

```
alphaC :: (Literal -> Bool) -> Clause -> Bool
alphaCNF :: (Literal -> Bool) -> CNF -> Bool
```

die eine übergebene Klausel (`Clausel`)/Formel in KNF (`CNF`) mit Hilfe einer übergebenen Funktion (`Literal -> Bool`) auf einen Wahrheitswert abbilden.
(5 Punkte)

Beispiel

Seien `a`, `b` und `c` wie in Aufgabenteil 1. und `alpha` wie in Aufgabenteil 3. definiert.

```
> alphaC alpha a
True
> alphaC alpha b
False
> alphaCNF alpha c
False
```

5. Wenden Sie die Belegung, die zu jeder Liste von Literalen in der Liste

```
[[a,b,c] | a <- [Atom 'A', NegA 'A'],
           b <- [Atom 'B', NegA 'B'],
           c <- [Atom 'C', NegA 'C']]
```

gehört (siehe Aufgabenteil 3.), auf die Formel `c` in KNF aus Aufgabenteil 1. an. (5 Punkte)

Hinweis

Funktionen kann man wie Operatoren und Operatoren wie Funktionen benutzen, insbesondere als Argument für `map`.

Beispiel

```
> alpha `alphaCNF` c
False
> map (+ 3) [1,2]
[4,5]
```