



Universität Karlsruhe (TH)

Institut für Innovatives Rechnen und Programmstrukturen (IPD)

Informatik I WS 2002/03

Dozent: Prof. Dr.rer.nat. G. Goos

Übungsleiter: Markus L. Noga

<http://eins.info.uni-karlsruhe.de>

goos@ipd.info.uni-karlsruhe.de

noga@ipd.info.uni-karlsruhe.de

Übungsblatt 8

Ausgabe: 6.12.2002

Abgabe: 13.12.2002 14.00 Uhr

Einwurf im Keller des Informatik-Hauptbaus (Geb. 50.34)

Aufgabe 1: Prädikatenlogik (10 T- Punkte)

1.1 Umgangssprache (2 Punkte)

Formulieren Sie diese Sätze in Prädikatenlogik:

- Jeder Barbier rasiert alle Personen, die sich nicht selbst rasieren.
- Kein Barbier rasiert jemanden, der sich selbst rasiert.

1.2 Quantoren und Variablen (4 Punkte)

Gegeben seien folgende Formeln über die Prädikate $\{R, P, Q\}$:

$$F = R(y) \vee \forall x \exists y (P(x, y) \wedge R(f(a), z)) \vee \forall z Q(g(a), x, z)$$

$$G = \exists x P(x, g(x, y)) \vee \neg \forall y Q(y, g(a, h(z)))$$

Geben Sie für jedes Vorkommen einer Variablen in F und G an, ob diese frei oder gebunden ist.

1.3 Terme und Formeln (4 Punkte)

Geben Sie sämtliche Terme an, die in der Formel F aus der vorigen Teilaufgabe enthalten sind.

Aufgabe 2: Prädikatenlogik (14 T- Punkte)

2.1 Modelle (5 Punkte)

Gegeben sei die prädikatenlogische Formel

$$F = \exists x \exists y \exists z (P(x, y, z) \wedge P(z, y, x) \wedge P(x, z, y))$$

und die Struktur $A = (I_A, U_A)$ mit $U_A = \mathbb{N}_0$ und

$$I_A(P) = \{ (m-1, m, m+1) \mid m \in U_A \}$$

Zeigen Sie oder widerlegen Sie: A ist ein Modell für F .

2.2 Umformungen (9 Punkte)

Zeigen oder widerlegen Sie die Korrektheit dieser Äquivalenzen:

$$\forall x : (\exists y : P(x, y) \wedge \exists y : Q(x, y)) \equiv \forall x \exists y : P(x, y) \wedge Q(x, y)$$

$$\exists x : (\forall y : P(x, y) \wedge \forall y : Q(x, y)) \equiv \exists x \forall y : P(x, y) \wedge Q(x, y)$$

$$\forall x : (\forall y : P(x, y) \wedge \exists y : Q(x, y)) \equiv \forall x \exists y : P(x, y) \wedge Q(x, y)$$

Aufgabe 3: Skolemnormalform (16 T- Punkte)

Bringen Sie folgende Formeln in Skolemnormalform. Bestimmen Sie dazu ggf. zunächst ihre bereinigte Pränexform.

$$\begin{aligned}
 F_1 &= \forall x \exists y \forall z \exists w : \neg P(a, w) \vee Q(f(x), y) \\
 F_2 &= \forall s \exists x : (G(s, a, b) \vee R(\neg x, s)) \wedge \exists z \forall y : (R(y) \wedge Q(z)) \vee \forall s : (\neg s) \\
 F_3 &= \forall x \forall y : T(x, y) \leftrightarrow \forall u : (R(u, x) \rightarrow R(u, y)) \\
 F_4 &= ((\forall x : P(x) \rightarrow \forall x : Q(x)) \rightarrow \forall x : (P(x) \rightarrow Q(x)))
 \end{aligned}$$

Aufgabe 4: Haskell: Kryptographie (20 P- Punkte)

Kryptographie ist die Kunst der Verschlüsselung. Professionelle Kryptographen unterdrücken Groß- und Kleinschreibung, Leerräume und Interpunktionszeichen, um die Entschlüsselung zu erschweren (zumindest taten sie das im 2. Weltkrieg noch ;-).

4.1 Datentyp (5 Punkte)

Definieren Sie einen Datentyp `CryptoChar`, der die Buchstaben A bis Z umfaßt und Mitglied der Typklassen `Eq`, `Ord`, `Enum` und `Show` ist. Definieren Sie einen Datentyp `CryptoString` für Folgen solcher Zeichen.

4.2 Konvertierung (5 Punkte)

Definieren Sie Funktionen `cryptoToString :: CryptoString -> String` und `stringToCrypto :: String -> CryptoString`, die Strings in Cryptostrings konvertieren und umgekehrt. Die Konvertierung von Strings in CryptoStrings muß Groß- und Kleinbuchstaben auf den entsprechenden CryptoChar, deutsche Sonderzeichen auf eine gängige Darstellung und Ziffernfolgen in Folgen deutscher Ziffernamen überführen (gerne auch Zahlennamen, ist aber nicht Pflicht).

4.3 Substitution (5 Punkte)

Eine Substitution ist eine zeichenweise Ersetzung. Polyalphabetische Substitutionen verwenden dafür mehrere verschiedene Ersetzungsvorschriften, monoalphabetische Substitutionen genau eine. Die einfachste monoalphabetische Substitution ist die Caesar-Verschlüsselung, welche das Alphabet zyklisch um eine feste Anzahl Buchstaben verschiebt (im Original genau drei).

Geben Sie die Signatur einer Ersetzungsvorschrift für Cryptobuchstaben an. Implementieren Sie die Caesar-Verschlüsselung mit variabler Verschiebung als Funktion `caesar` dieser Signatur. Verwenden Sie Funktionen höherer Ordnung, um Ihre Ersetzungsvorschrift auch auf `CryptoStrings` zu definieren.

Was erhalten Sie, wenn Sie diese Funktion mit Verschiebung -3 auf die Zeichenkette `ULFKWLJ` anwenden?

4.4 Transposition (5 Punkte)

Eine Transposition ist eine blockweise Verwürfelung. Die Transposition `[3,2,0,1]` hat eine Blocklänge von 4. Sie macht `ABCDEFGH` zu `DCABHGEF`.

Setzen Sie Transpositionen als Funktion `cryptoTranspose :: [Int] -> CryptoString -> CryptoString` um. Es ist zulässig, die Zeichenkette mit dem Buchstaben `X` auf die benötigte Länge aufzufüllen.

Wenden Sie die Transposition `[2,3,1,0]` und eine Caesar-Verschlüsselung mit Verschiebung -3 auf die Zeichenkette `GHZLLUHULWFKXXJX` an.