SS2010
BAI2-LBP Gruppe 1 Team 07
Entwurf zu Aufgabe 2

R. C. Ladiges, D. Fast

22. April 2010

Inhaltsverzeichnis

2 Aufgabe 2 **3**
2.1 Sich mit dem Programmpaket vertraut machen . . . 3
2.1.1 Aufgabenstellung . . . 3
2.1.2 Entwurf . . . 3
2.2 Anwendung des Paketes für bestimmte Fragestellungen . . . 3
2.2.1 Aufgabenstellung . . . 3
2.2.2 Entwurf . . . 3
2.3 Veränderung der Syntax und der Semantik . . . 3
2.3.1 Aufgabenstellung . . . 3
2.3.2 Entwurf . . . 3
2.4 Sammeln syntaktischer Informationen . . . 4
2.4.1 Aufgabenstellung . . . 4
2.4.2 Entwurf . . . 4
2.5 Sammeln semantischer Informationen . . . 4
2.5.1 Aufgabenstellung . . . 4
2.5.2 Entwurf . . . 4
2.6 Erweiterung der Verarbeitungsmöglichkeiten des Pakets . . . 5
2.6.1 Aufgabenstellung . . . 5
2.6.2 Entwurf . . . 5

2 Aufgabe 2

2.1 Sich mit dem Programmpaket vertraut machen

2.1.1 Aufgabenstellung

Machen Sie sich mit dem Programmpaket vertraut. Lesen Sie die Definitionen der Prädikate durch, Probieren Sie die Prädikate aus und überzeugen Sie sich, dass Sie wissen, warum was passiert.

2.1.2 Entwurf

Ansehen aller Prädikate der vier Module und falls möglich in den fünf anderen Aufgaben verwenden.

Zur Lösung:
Auflistung aller Prädikate mit einem Beschreibungstext in eigenen Worten. Hiermit ist nicht die ausführliche Beschreibung der Funktionsweise der Prädikate gemeint.

2.2 Anwendung des Paketes für bestimmte Fragestellungen

2.2.1 Aufgabenstellung

Ergänzen Sie Aussagensymbole und Beispielformeln (Datei: examples.pl). (**Anwendung** des Paketes für bestimmte Fragestellungen)

2.2.2 Entwurf

Ergänzung des Prädikates `examples_F` in `examples.pl` um eigene Fakten, welche als zweites Argument Formeln haben. Diese sind selbst ausgedachte simple Formeln, sowie weitere komplexere Formeln aus der Vorlesung und/oder Literatur.

2.3 Veränderung der Syntax und der Semantik

2.3.1 Aufgabenstellung

Verändern Sie die Sprache (**Veränderung der Syntax und der Semantik**): Ergänzen Sie die Sprache um mindestens einen Junktor (xor) und erzeugen Sie dazu Beispielformeln (Dateien: formulae.pl, modelCheckerAL.pl, examples.pl)

2.3.2 Entwurf

Für die Umsetzung der Kontravalenz können (weil in Prolog bereits deklariert) wir zuerst das *„Symbol“* des Junktors $\veebar$ am Anfang der drei Dateien, so wie die anderen Junktoren auch, mit dem Prädikat `op` als Operator **`xor`** definieren.

In `formulae.pl` ergänzen wir das Prädikat `wff(Formel)` um einen weiteren Fall für das Auftreten des **`xor`** Operators. Was, bis auf den Operator, syntaktisch gleich mit den anderen Junktoren (mit Stelligkeit 2) ist.

In `modelCheckerAL.pl` ergänzen wir das Prädikat `evaluateAsF` für den **`xor`** Operator um zwei Regeln. Die zwei Regeln bestimmen die Logik der Kontravalenz, durch die zwei Fälle, bei der sie zu Wahr evaluiert:
1. Fall: Wenn Â(A)=1 und Â(B)=0
2. Fall: Wenn Â(A)=0 und Â(B)=1
für die Formel: `A` **`xor`** `B`

In `examples.pl` fügen wir noch, wie in [2.2], einige Beispiele zur Kontravalenz hinzu.

2.4 Sammeln syntaktischer Informationen

2.4.1 Aufgabenstellung

Ergänzen Sie einfache rekursive Prädikate (**Sammeln syntaktischer Informationen**): zählen Sie die Junktoren einer Formel und bilden Sie eine Liste aller Teilformeln.

2.4.2 Entwurf

Zählen der Junktoren:
Mit einem modifiziertem `wff` Prädikat, welches eine Ausgabe besitzt gehen wir die Formel rekursiv durch.
Literale sind der Rekursionsabbruch. Ein Aufruf des Prädikats mit einem Literal gibt 0 zurück.
Wenn wir einen Junktor mit der Stelligkeit 1 (Negation $\neg$) haben, addieren wir 1 auf den rekursiven Aufruf der Formel(ohne Junktor) und geben die Summe aus.
Bei Junktoren der Stelligkeit 2 (Konjunktion $\wedge$, Disjunktion $\vee$, Kontravalenz $\veebar$, Subjunktion $\rightarrow$ und Äquivalenz $\leftrightarrow$) geben wir die Summer der rekursiven Aufrufe der Formeln und 1 aus.

Liste aller Teilformeln:
Wie beim Zählen der Junktoren verwenden wir ein modifiziertes `wff` Prädikat mit Ausgabe.
Bei einem Literal wird nur es selbst in einer einelementigen Liste ausgegeben.
Bei einem Junktor wird die Formel selbst, zusammen mit der/den Ausgabe(n) des/der rekursiven Aufrufe(s), in die Ausgabeliste getan.

2.5 Sammeln semantischer Informationen

2.5.1 Aufgabenstellung

Ergänzen Sie Prädikate zur Berechnung semantischer Eigenschaften und Relationen (**Sammeln semantischer Informationen**) unter Betrachtung der möglichen Belegungen: Implementieren Sie Prädikate zur Bestimmung der Äquivalenz von Formeln (`aequiv(Formel1,Formel2)`, gilt, wenn beide Formeln bei der gleichen Belegung den gleichen Wahrheitsgehalt haben) und Prädikate zur Bestimmung der Folgerung (Implikation) von Formeln aus Formeln (`impl(Formel1,Formel2)`, gilt nicht, wenn `Formel1` wahr und `Formel2` bei der gleichen Belegung falsch ist, sonst gilt sie immer).

2.5.2 Entwurf

Wir definieren uns die beiden Prädikate welche wahr werden, wenn die gegebenen Formeln mit dem jeweiligen Junktor verknüpft eine Tautologie ergibt. Dazu benutzen wir das gegebene `tautologie(F)` Prädikat.

2.6 Erweiterung der Verarbeitungsmöglichkeiten des Pakets

2.6.1 Aufgabenstellung

Ergänzen Sie Prädikate zur Bestimmung der **Erfüllbarkeit von Formelmengen** (repräsentiert als Listen) und zur **Äquivalenz** und **Folgerung** von Formeln aus Formellisten (**Erweiterung der Verarbeitungsmöglichkeiten** des Pakets). Testen Sie die Prädikate auch mit der leeren Liste. Eine Formelliste wird dabei als konjunktive Verknüpfung aller ihrer Formeln aufgefasst.

2.6.2 Entwurf

Wir definieren uns ein Unterprädikat, welches Rekursiv eine gegebene Liste aus Formeln verundet (Konjunktion aller Elemente zu einer Formel) und ausgibt.

Mit dem Prädikat `erfuellbar` testen wir ob die neue Formel erfüllbar ist.

Ob die zusammengebaute Formel mit der übergebenen Formel äquivalent ist überprüfen wir mit dem Prädikat `aequiv` aus [2.5].

Dass, aus der zusammengebauten Formel, die übergebene Formel folgt, fragen wir über das `impl` Prädikat aus [2.5] ab.

Informationen zur Signatur

	**Unterzeichner**	EMAILADDRESS=robin.ladiges@haw-hamburg.de, CN=Robin Christopher Ladiges
	Datum/Zeit	Sun Jun 27 00:04:15 CEST 2010
	Austeller-Zertifikat	CN=CAcert Class 3 Root, OU=http://www.CAcert.org, O=CAcert Inc.
	Serien-Nr.	44727
	Methode	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signatur)