Dr. Theo Lettmann

Paderborn, den 07. November
Abgabe 17. November 2002

Übungen zur Vorlesung

Modellierung

WS 2003/2004

Blatt 4 Musterlösungen

AUFGABE 25 :
Beweise durch vollständige Induktion:

$$\sum_{k=1}^{n}(2k-1) = n^2$$

Lösung:
Die Behauptung ist für alle natürlichen Zahlen $n \geq 1$ zu zeigen.
IA: Sei n=1.

$$\sum_{k=1}^{1}(2k-1) = (2-1) = 1^2$$

IV: Die Formel gelte für ein festes, aber beliebiges $n \geq 1$.
IS: Es ist zu zeigen

$$\sum_{k=1}^{n+1}(2k-1) = (n+1)^2.$$

Es gilt:

$$\begin{aligned}
\sum_{k=1}^{n+1}(2k-1) &= \sum_{k=1}^{n}(2k-1) + 2(n+1) - 1 \\
&= n^2 + 2(n+1) - 1 && \text{mit IV} \\
&= n^2 + 2n + 1 \\
&= (n+1)^2
\end{aligned}$$

q.e.d.

AUFGABE 26 :
Beantworte die folgenden Fragen möglichst ausführlich:

1. Was sind die Unterschiede zwischen konkreten und abstrakten Algebren?
2. Was ist die Normalform eines Terms und wie ist sie definiert?
3. Beschreibe die Funktionsweise eines Kellers (Stack) und einer Warteschlange (Queue).

Lösung:

1. Eine konkrete Algebra liefert zu einer abstrakten Algebra eine (konkrete) Trägermenge, ordnet jeder Operation eine konkrete Operation zu, wobei diese Operationen die Gesetze der abstrakten Algebra erfüllen.

2. Terme in Normalform zeichnen sich dadurch aus, dass sich durch Anwendung von Gesetzen die Anzahl von Operationen nicht mehr verringern lassen.

3. Ein Stack (dt.: Stapel) ist ein Stoss von Elementen, ganz ähnlich einem Stapel von Blättern auf einem Schreibtisch.

   Ein Stack gestattet den Zugriff immer nur auf das oberste Element. Das bedeutet, daß einerseits ein neues Element immer oben auf dem Stapel abgelegt wird und andererseits nur das oberste Element heruntergenommen werden kann. Die Elemente eines Stacks können genau in der umgekehrten Reihenfolge, in der sie eingebracht wurden, wieder entnommen werden. Ein Stack wird wegen dieser Eigenschaft auch eine Last in, First OutDatenstruktur (LIFO-Datenstruktur) genannt. Die beiden wichtigsten Operationen (Element ablegen und Element entnehmen) eines Stacks werden Push und Pop genannt. Push stapelt ein neues Element auf dem Stack, Pop holt das oberste Element herunter.

   Eine Warteschlange dagegen ist eine endliche Folge von Elementen, beispielsweise eine Schlange von Personen an der Kinokasse.

   Eine Warteschlange erlaubt nur den Zugriff auf das vorderste Element, während neue Elemente hinten an der Schlange angefügt werden. Das Löschen von Elementen erfolgt dagegen nur am Anfang der Schlange, nur das erste Element kann entfernt werden. Damit können die Elemente der Schlange nur in der gleichen Reihenfolge entfernt werden, in der sie auch in die Schlange eingefügt wurden. Eine Wartteschlange wird wegen dieser Eigenschaft auch eine First in, First OutDatenstruktur (FIFO-Datenstruktur) genannt. Die beiden wichtigsten Operationen (Element abfügen und Element entfernen) einer Warteschlange werden Enqueue und Dequeue genannt. Enqueue fügt ein neues Element an die Schlange an, Dequeue entfernt das erste Element der Schlange.

AUFGABE 27 :
Die Datenstruktur *Schlange* verwaltet Elemente in der Reihenfolge ihres Eintreffens. Operationen auf einer Schlange erlauben es, Elemente hinten anzufügen oder vorne die am längsten wartenden Elemente zu entnehmen. Schlangen arbeiten nach dem FIFO-Prinzip (First-In-First-Out). Sie sind z.B. an der Kasse im Supermarkt als Warteschlange zu finden.
Die folgende abstrakte Algebra beschreibt eine *Schlange* $= (T, \Sigma, Q)$ von natürlichen Zahlen (als Elemente) mit der **Signatur** $\Sigma = (S, F)$:

$$
\begin{array}{lllll}
S = \{ & \mathit{Queue}, Nat, \mathit{Bool} \ \} & & & (S_1) \\
F = \{ & \mathit{createQueue} : & & \rightarrow \mathit{Queue}, & (F_1) \\
 & \mathit{enqueue} : & \mathit{Queue} \times \mathit{Nat} & \rightarrow \mathit{Queue}, & (F_2) \\
 & \mathit{dequeue} : & \mathit{Queue} & \rightarrow \mathit{Queue}, & (F_3) \\
 & \mathit{front} : & \mathit{Queue} & \rightarrow \mathit{Nat}, & (F_4) \\
 & \mathit{empty} : & \mathit{Queue} & \rightarrow \mathit{Bool} \ \} & (F_5)
\end{array}
$$

Die Sorte *Queue* stellt eine Schlange dar, die Sorte Nat beschreibt Elemente der Schlange. Für die Sorte *Bool* sind die Konstanten *true* und *false* definiert.
createQueue bezeichnet eine 0-stellige Operation und ist damit eine Konstante. Sie steht für die leere Schlange. Die Operation *enqueue* fügt ein Element am Ende der Schlange an. Die Operation *front* liefert das erste Element vorne in der Schlange, *dequeue* entfernt dieses Element aus der Schlange. Ob die Schlange leer ist oder nicht, zeigt die Operation *empty* an.
Für die Menge der **Axiome** Q seien x, y Variablen der Sorte *Element* und q eine Variable der Sorte *Queue*:

$$
\begin{array}{lll}
Q = \{ & \mathit{dequeue}(\mathit{enqueue}(\mathit{createQueue}, x)) & \equiv \mathit{createQueue}, \qquad (Q_1) \\
& \mathit{dequeue}(\mathit{enqueue}(\mathit{enqueue}(q, y), x)) & \equiv \mathit{enqueue}(\mathit{dequeue}(\mathit{enqueue}(q, y)), x), \qquad (Q_2) \\
& \mathit{front}(\mathit{enqueue}(\mathit{createQueue}, x)) & \equiv x, \qquad (Q_3) \\
& \mathit{front}(\mathit{enqueue}(\mathit{enqueue}(q, x))) & \equiv \mathit{front}(\mathit{enqueue}(q, x)), \qquad (Q_4) \\
& \mathit{empty}(\mathit{createQueue}) & \equiv \mathit{true}, \qquad (Q_5) \\
& \mathit{empty}(\mathit{enqueue}(q, x)) & \equiv \mathit{false}, \qquad \} \quad (Q_6)
\end{array}
$$

Hinweis: Hinter den einzelnen Spezifikationen sind Beschriftungen in Klammern angegeben. Beziehen Sie sich auf diese in Ihrer Lösung. Im folgenden seien x, y, z Variable der Sorte *Element*.

(a) Axiome anwenden

Formen Sie die folgenden Terme mit Hilfe der Axiome so um, dass Sie als Ergebnis eine einzelne Konstante oder Variable erhalten. Notieren Sie bei jeder Umformung das benutzte Axiom.

1. $\mathit{empty}(\mathit{enqueue}(\mathit{createQueue}, y))$
2. $\mathit{front}(\mathit{dequeue}(\mathit{enqueue}(\mathit{enqueue}(\mathit{createQueue}, z), x)))$
3. $\mathit{empty}(\mathit{enqueue}(\mathit{dequeue}(\mathit{enqueue}(\mathit{createQueue}, x)), z))$
4. $\mathit{empty}(\mathit{dequeue}(\mathit{enqueue}(\mathit{createQueue}, x)))$
5. $\mathit{dequeue}(\mathit{enqueue}(\mathit{createQueue}, \mathit{front}(\mathit{enqueue}(\mathit{enqueue}(\mathit{createQueue}, x), z))))$

(b) Algebra erweitern

Erweitern Sie die abstrakte Algebra *Schlange* um die Operation *sum*, die die Summe aller Elemente der Schlage ausgibt, sowie die hierfür benötigten Operationen +, 1 und 0. Dazu ist es notwendig auch die Menge der Axiome zu erweitern.

Lösung:

(a) 1.
$$
\begin{array}{ll}
 & \mathit{empty}(\mathit{enqueue}(\mathit{createQueue}, y)) \\
\overset{Q_6}{\equiv} & \mathit{false}
\end{array}
$$

2.
$$
\begin{array}{ll}
 & \mathit{front}(\mathit{dequeue}(\mathit{enqueue}(\mathit{enqueue}(\mathit{createQueue}, z), x))) \\
\overset{Q_2}{\equiv} & \mathit{front}(\mathit{enqueue}(\mathit{dequeue}(\mathit{enqueue}(\mathit{createQueue}, z)), x)) \\
\overset{Q_1}{\equiv} & \mathit{front}(\mathit{enqueue}(\mathit{createQueue}, x))) \\
\overset{Q_3}{\equiv} & x
\end{array}
$$

3. $\mathit{empty}(\mathit{enqueue}(\mathit{dequeue}(\mathit{enqueue}(\mathit{createQueue}, x)), z))$
   $\overset{Q_1}{\equiv}$ $\mathit{empty}(\mathit{enqueue}(\mathit{createQueue}, z))$
   $\overset{Q_6}{\equiv}$ false

4. $\mathit{empty}(\mathit{dequeue}(\mathit{enqueue}(\mathit{createQueue}, x)))$
   $\overset{Q_1}{\equiv}$ $\mathit{empty}(\mathit{createQueue})$
   $\overset{Q_5}{\equiv}$ true

5. $\mathit{dequeue}(\mathit{enqueue}(\mathit{createQueue}, \mathit{front}(\mathit{enqueue}(\mathit{enqueue}(\mathit{createQueue}, x), z))))$
   $\overset{Q_4}{\equiv}$ $\mathit{dequeue}(\mathit{enqueue}(\mathit{createQueue}, \mathit{front}(\mathit{enqueue}(\mathit{createQueue}, x))))$
   $\overset{Q_3}{\equiv}$ $\mathit{dequeue}(\mathit{enqueue}(\mathit{createQueue}, x))$
   $\overset{Q_1}{\equiv}$ $\mathit{createQueue}$

(b) $F \cup \{\mathit{sum} : \mathit{Queue} \to \mathit{Nat},\ + : \mathit{Nat} \times \mathit{Nat} \to \mathit{Nat}\}$
$Q \cup \{\mathit{sum}(\mathit{createQueue}) \equiv 0,\ \mathit{sum}(\mathit{enqueue}(q, x)) \equiv x + \mathit{sum}(q)\}$

AUFGABE 28 :
Unifikation: Es seien x, y, z Variablen und $a, b, c, 0$ Konstanten. Ermitteln Sie jeweils einen Unifikator, falls ein solcher existiert.

1. $\mathit{plus}(x, 0, y)$ und $\mathit{plus}(a, y, y)$
2. $f(x, y, z)$ und $f(g(y), g(z), a)$
3. $f(x, x)$ und $f(y, g(y))$
4. $f(x, g(x))$ und $f(g(g(y)), g(z))$
5. $R(g(x), a, g(z), c)$ und $R(y, x, g(a), g(x))$

Lösung:

a)

Durchlauf	Term 1	Term 2	$\sigma -$ Erweiterung
1	$\mathit{plus}(\underline{x}, 0, y)$	$\mathit{plus}(\underline{a}, y, y)$	$[x/a]$
2	$\mathit{plus}(a, \underline{0}, y)$	$\mathit{plus}(a, \underline{y}, y)$	$[y/0]$
3	$\mathit{plus}(a, 0, 0)$	$\mathit{plus}(a, 0, 0)$	$-$

Gesamtsubstitution $\sigma = [\,][x/a][y/0] = [x/a, y/b]$

b)

Durchlauf	Term 1	Term 2	$\sigma -$ Erweiterung
1	$f(\underline{x}, y, z)$	$f(\underline{g(y)}, g(z), a)$	$[x/g(y)]$
2	$f(g(y), \underline{y}, z)$	$f(g(y), \underline{g(z)}, a)$	$[y/g(z)]$
3	$f(g(g(z)), g(z), \underline{z})$	$f(g(g(z)), g(z), \underline{a})$	$[z/a]$
4	$f(g(g(a)), g(a), a)$	$f(g(g(a)), g(a), a)$	$-$

Gesamtsubstitution $\sigma = [\,][x/g(y)][y/g(z)][z/a] = [x/g(g(a)), y/g(a), z/a]$

c)

Durchlauf	Term 1	Term 2	$\sigma -$ Erweiterung
1	$f(\underline{x}, x)$	$f(\underline{y}, g(y))$	$[x/y]$
2	$f(y, \underline{y})$	$f(y, \underline{g(y)})$	

Variable y kommt im Ersetzungsterm $g(y)$ vor, also gibt es keinen Unifikator.

a)

Durchlauf	Term 1	Term 2	$\sigma -$ Erweiterung
1	$f(\underline{x}, g(x))$	$f(\underline{g(g(y))}, g(z))$	$[x/g(g(y))]$
2	$f(g(g(y)), g(\underline{g(g(y))}))$	$f(g(g(y)), g(\underline{z}))$	$[z/g(g(y))]$
3	$f(g(g(y)), g(g(g(y))))$	$f(g(g(y)), g(g(g(y))))$	$-$

Gesamtsubstitution $\sigma = [][x/g(g(y))][z/g(g(y))] = [x/g(g(y)), y/g(g(y))]$

a)

Durchlauf	Term 1	Term 2	$\sigma -$ Erweiterung
1	$R(\underline{g(x)}, a, g(z), c)$	$R(\underline{y}, x, g(a), g(x))$	$[y/g(x)]$
2	$R(g(x), \underline{a}, g(z), c)$	$R(g(x), \underline{x}, g(a), g(x))$	$[x/a]$
3	$R(g(a), a, g(\underline{z}), c)$	$R(g(a), a, g(\underline{a}), g(a))$	$[z/a]$
4	$R(g(a), a, g(a), \underline{c})$	$R(g(a), a, g(a), \underline{g(a)})$	

Da weder c noch $g(a)$ Variable sind, gibt es keinen Unifikator.

AUFGABE 29 :
Erstelle eine abstrakte Algebra zum Datentyp *Binärbaum*, dessen Knoten Zahlen enthalten.
Ein Binärbaumknoten bzw. -blatt hat also also immer maximal 2 Nachfolger. Diese Nachfolger sind dann entweder Blätter oder weitere Binärbaume.
In dieser Aufgabe sind Teile bereit gelöst. Schreiben Sie jedoch in der Lösung die komplette Definition der Algebra. Ihre Lösung muss also enthalten:

1. Sorten
2. Operationen
3. Axiome (mind. 3)

Geben Sie zum Schluss eine Klassifikation der Operationen an (Konstruktoren, Hilfskonstruktoren, Projektionen).

Die folgende abstrakte Algebra beschreibt die Datenstruktur Binärbaum $\mathit{Binärbaum} = (T, \Sigma, Q)$ mit der **Signatur** $\Sigma = (S, F)$:

$$
\begin{array}{lllll}
S = \{ & \mathit{BinTree}, \mathit{Nat}, \mathit{Bool} \ \} & & & (S_1) \\
F = \{ & \mathit{createBinTree} : & & \rightarrow \mathit{BinTree}, & (F_1) \\
 & \mathit{bin} : & \mathit{BinTree} \times Nat \times \mathit{BinTree} & \rightarrow \mathit{BinTree}, & (F_2) \\
 & \mathit{left} : & & \rightarrow & (F_3) \\
 & \mathit{right} : & & \rightarrow & (F_4) \\
 & \mathit{value} : & & \rightarrow & (F_5) \\
 & \mathit{empty} : & & \rightarrow & (F_6)
\end{array}
$$

Die Sorte *BinTree* stellt einen Binärbaum dar, die Sorte *Nat* beschreibt ein Blatt.

Funktionsbeschreibung:

createBinTree: erzeugt einen leeren Binärbaum,
bin: erzeugt einen nicht leeren Binärbaum,
left: beschreibt den linken Teilbaum,
right: beschreibt den rechten Teilbaum,
value: beschreibt den Wert im Knoten (aus *Nat*),
empty: gibt an, ob der Baum leer ist.

Lösung:
Die folgende abstrakte Algebra beschreibt die Datenstruktur Binärbaum $\mathit{Binärbaum} = (T, \Sigma, Q)$ mit der **Signatur** $\Sigma = (S, F)$:

$$
\begin{array}{lllll}
S = \{ & \mathit{BinTree}, \mathit{Nat}, \mathit{Bool} \ \} & & & (S_1) \\
F = \{ & \mathit{createBinTree} : & & \rightarrow \mathit{BinTree}, & (F_1) \\
 & \mathit{bin} : & \mathit{BinTree} \times Nat \times \mathit{BinTree} & \rightarrow \mathit{BinTree}, & (F_2) \\
 & \mathit{left} : & \mathit{BinTree} & \rightarrow \mathit{BinTree} & (F_3) \\
 & \mathit{right} : & \mathit{BinTree} & \rightarrow \mathit{BinTree} & (F_4) \\
 & \mathit{value} : & \mathit{BinTree} & \rightarrow \mathit{Nat} & (F_5) \\
 & \mathit{empty} : & \mathit{BinTree} & \rightarrow \mathit{Bool} & (F_6)
\end{array}
$$

Axiome:
Seien t_1, t_2 Variablen der Sorte *BinTree*, $n \in$ *Nat*, dann gelten folgende Axiome:

1. $\mathit{empty}(\mathit{createBinTree}) = \mathit{true}$
2. $\mathit{empty}(\mathit{bin}(t_1, n, t_2)) = \mathit{false}$
3. $\mathit{left}(\mathit{bin}(t_1, n, t_2)) = t_1$
4. $\mathit{right}(\mathit{bin}(t_1, n, t_2)) = t_2$
5. $\mathit{value}(\mathit{bin}(t_1, n, t_2)) = n$

Einteilung der Funktionen:

- Konstruktoren sind *createBinTree* und *bin*.
- Hilfskonstruktoren sind *left* und *right*.
- Projektionen sind *empty* und *value*.

AUFGABE 30 :
Benenne die folgenden Gesetze und zeige die Gültigkeit mit Hilfe von Wahrheitstafeln:

1. $\neg(X \wedge Y) = \neg X \vee \neg Y$
2. $X \wedge Y = Y \wedge X$
3. $(X \wedge Y) \wedge Z = X \wedge (Y \wedge Z)$

Lösung:

a) + b)

X	Y	$\neg X$	$\neg Y$	$X \wedge Y$	$Y \wedge X$	$\neg(X \wedge Y)$	$\neg X \vee \neg Y$
w	w	f	f	w	w	f	f
w	f	f	w	f	f	w	w
f	w	w	f	f	f	w	w
f	f	w	w	f	f	w	w

Das Gesetz unter a) ist die Regel von DeMorgan für die Konjunktion und das Gestz unter b) ist das Kommutativgesetz für die Konjunktion.

c)

X	Y	Z	$X \wedge Y$	$Y \wedge Z$	$(X \wedge Y) \wedge Z$	$X \wedge (Y \wedge Z)$
w	w	w	w	w	w	w
w	w	f	w	f	f	f
w	f	w	f	f	f	f
w	f	f	f	f	f	f
f	w	w	f	w	f	f
f	w	f	f	f	f	f
f	f	w	f	f	f	f
f	f	f	f	f	f	f

Das Gesetz ist das Assoziativgesetz für die Konjunktion.

KORREKTURAUFGABE 31 :
Gebe die Wahrheitstafeln zu folgenden Formeln an. Als Spalten müssen hier die Variablen, sinnvolle und wichtige Einzelaussagen sowie die Gesamtaussage angegeben sein.
Beispiel: Für die Funktion $\neg(x \rightarrow (y \wedge \neg z)) \rightarrow (\neg x)$ sollten ca. 7 Spalten angeben werden $(x, y, z, \neg x, y \wedge \neg z, \neg(x \rightarrow (y \wedge \neg z)), \neg(x \rightarrow (y \wedge \neg z)) \rightarrow (\neg x))$.
Überprüfe überdies die 4 Eigenschaften der Formeln (erfüllbar, falsifizierbar, unerfüllbar, tautologisch) und forme die Formeln in die KNF und NNF um.

(1) $\neg(\neg Y \vee (Z \rightarrow (\neg X \vee Z)) \wedge \neg(X \wedge Y))$

(2) $X \rightarrow (Y \leftrightarrow Z)$

(3) $\neg((\neg X \vee \neg Y \vee \neg Z) \vee (\neg Y \rightarrow Z) \vee Y)$

Lösung:

(1) $\alpha = \neg(\neg Y \vee (Z \rightarrow (\neg X \vee Z)) \wedge \neg(X \wedge Y))$
Es sei $\alpha_1 = Z \rightarrow (\neg X \vee Z)$ und $\alpha_2 = \neg(X \wedge Y)$.

X	Y	Z	$\neg X$	$\neg X \vee Z$	α_1	$X \wedge Y$	α_2	$\alpha_1 \wedge \alpha_2$	$\neg Y$	$\neg Y \vee \alpha_1 \wedge \alpha_2$	α
0	0	0	1	1	1	0	1	1	1	1	0
0	0	1	1	1	1	0	1	1	1	1	0
0	1	0	1	1	1	0	1	1	0	1	0
0	1	1	1	1	1	0	1	1	0	1	0
1	0	0	0	0	1	0	1	1	1	1	0
1	0	1	0	1	1	0	1	1	1	1	0
1	1	0	0	0	1	1	0	0	0	0	1
1	1	1	0	1	1	1	0	0	0	0	1

Die Formel α ist damit erfüllbar (Zeile 7), falsifizierbar (Zeile 1), nicht tautologisch und nicht widerspruchsvoll.

$$
\begin{aligned}
& \neg(\neg Y \vee (Z \rightarrow (\neg X \vee Z)) \wedge \neg(X \wedge Y)) \\
\approx\ & Y \wedge \neg((Z \rightarrow (\neg X \vee Z)) \wedge \neg(X \wedge Y)) \\
\approx\ & Y \wedge (\neg(\neg Z \vee (\neg X \vee Z)) \vee \neg\neg(X \wedge Y)) \\
\approx\ & Y \wedge ((\neg\neg Z \wedge \neg(\neg X \vee Z)) \vee (X \wedge Y)) \\
\approx\ & Y \wedge ((Z \wedge X \wedge \neg Z) \vee (X \wedge Y)) \\
\approx\ & Y \wedge ((Z \wedge X \wedge \neg Z) \vee X) \wedge ((Z \wedge X \wedge \neg Z) \vee Y)) \\
\approx\ & Y \wedge (Z \vee X) \wedge (X \vee X) \wedge (\neg Z \vee X) \wedge (Z \vee Y) \wedge (X \vee Y) \wedge (\neg Z \vee Y)
\end{aligned}
$$

(2) $\beta = X \rightarrow (Y \leftrightarrow Z)$

X	Y	Z	$Y \leftrightarrow Z$	$X \rightarrow (Y \leftrightarrow Z)$
0	0	0	1	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	1
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Die Formel α ist damit erfüllbar (Zeile 1), falsifizierbar (Zeile 6), nicht tautologisch und nicht widerspruchsvoll.

$$
\begin{aligned}
& X \rightarrow (Y \leftrightarrow Z) \\
\approx\ & \neg X \vee ((Y \rightarrow Z) \wedge (Z \rightarrow Y)) \\
\approx\ & \neg X \vee ((\neg Y \vee Z) \wedge (\neg Z \vee Y)) \\
\approx\ & (\neg X \vee \neg Y \vee Z) \wedge (\neg X \vee \neg Z \vee Y))
\end{aligned}
$$

(3) $\gamma = \neg((\neg X \vee \neg Y \vee \neg Z) \vee (\neg Y \rightarrow Z) \vee Y)$
Es sei $\gamma_1 = (\neg X \vee \neg Y \vee \neg Z)$ und $\gamma_2 = (\neg X \vee \neg Y \vee \neg Z) \vee (\neg Y \rightarrow Z) \vee Y$.

X	Y	Z	$\neg X$	$\neg Y$	$\neg Z$	γ_1	$(\neg Y \rightarrow Z)$	γ_2	γ
0	0	0	1	1	1	1	0	1	0
0	0	1	1	1	0	1	1	1	0
0	1	0	1	0	1	1	1	1	0
0	1	1	1	0	0	1	1	1	0
1	0	0	0	1	1	1	0	1	0
1	0	1	0	1	0	1	1	1	0
1	1	0	0	0	1	1	1	1	0
1	1	1	0	0	0	0	1	1	0

Die Formel α ist damit falsifizierbar (Zeile 1), nicht erfüllbar, daher widerspruchsvoll und nicht tautologisch.

$$
\begin{aligned}
& \neg((\neg X \vee \neg Y \vee \neg Z) \vee (\neg Y \rightarrow Z) \vee Y) \\
\approx\ & \neg((\neg X \vee \neg Y \vee \neg Z) \vee (\neg\neg Y \vee Z) \vee Y) \\
\approx\ & \neg(\neg X \vee \neg Y \vee \neg Z) \wedge \neg(\neg\neg Y \vee Z) \wedge \neg Y \\
\approx\ & (\neg\neg X \wedge \neg\neg Y \wedge \neg\neg Z) \wedge \neg(Y \vee Z) \wedge \neg Y \\
\approx\ & (X \wedge Y \wedge Z) \wedge (\neg Y \wedge \neg Z) \wedge \neg Y \\
\approx\ & X \wedge Y \wedge Z \wedge \neg Y \wedge \neg Z \wedge \neg Y
\end{aligned}
$$