Karlsruher Institut für Technologie
Institut für Theoretische Informatik
Prof. Dr. B. Beckert
Dr. des. Vladimir Klebanov

Formale Systeme, WS 2009/2010

Lösungen zum Übungsblatt 3

Dieses Blatt wurde in der Übung am 27.11.2009 besprochen.

Zu Aufgabe 1

Abbildung 1 zeigt Tableaubeweise für beide Teilaufgaben.

Sei φ_a die Formel aus Teilaufgabe (a). Das Tableau für $0\varphi_a$ ist geschlossen; die Allgemeingültigkeit der Formel φ_a ist damit bewiesen.

Das Tableau für die Formel $0\varphi_b$ lässt sich auch im erschöpften Zustand nicht schließen, daher ist $\neg\varphi_b$ erfüllbar, φ_b also nicht allgemeingültig.

Die Abbildungen sind mit der Vorzeichen-Variante des Kalküls aus der Vorlesung erstellt und die Regelanwendungen darin markiert. Dabei steht eine rote Kante für eine α- und eine blaue für eine β-Erweiterung. Die jeweils letzten Formeln im Tableau von (b) gehen aus der β-Formel $1B \to C$ hervor.

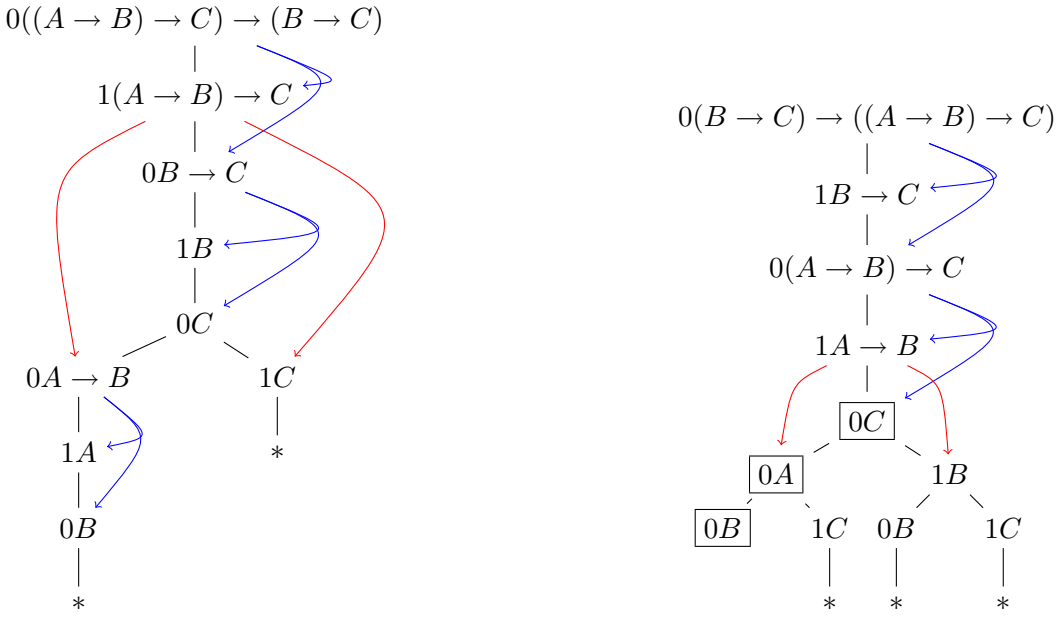

Abbildung 1: Tableaubeweise zu Aufgabe 4(a) (links) und 4(b) (rechts)

Ein erschöpftes, nicht geschlossenes Tableau hat einen Ast, der nicht geschlossen werden kann. Dieser Erlaubt das Ablesen einer Interpretation, die die Wurzelformel erfüllt: Setze für jede Variable $P \in \Sigma$

- wenn $0P$ auf dem Ast vorkommt $I(P) := F$,
- wenn $1P$ auf dem Ast vorkommt $I(P) := W$.
- andernfalls $I(P)$ beliebig.

Zur Begründung siehe Lemma 3.34 im Skript.

Insbesondere wird die Wurzel des Baumes erfüllt, das Negat der Formel, die auf Allgemeingültigkeit überprüft werden soll. Für diese Formel ist die Belegung also ein Gegenbeispiel zur Allgemeingültigkeit.

Für (b) ist das der Fall für die Belegung I mit $I(A) = I(B) = I(C) = F$. Für diese gilt: $\mathrm{val}_I(\varphi_b) = F$.

Zu Aufgabe 2

(a)

$1sh(P_1, P_2, P_3)$	
$1P_1$	$0P_1$
$1P_3$	$1P_2$

$0sh(P_1, P_2, P_3)$	
$1P_1$	$0P_1$
$0P_3$	$0P_2$

(b) Es ist zu zeigen: Für eine beliebige Interpretation I gilt

$$\mathrm{val}_I(1sh(P_1, P_2, P_3)) = W \quad \text{gdw.} \quad \begin{array}{c} \mathrm{val}_I(1P_1) = W \text{ und } \mathrm{val}_I(1P_3) = W \\ \text{oder} \\ \mathrm{val}_I(0P_1) = W \text{ und } \mathrm{val}_I(1P_2) = W \end{array}$$

Dabei ist die Richtung $\Rightarrow$ der Korrektheits- und die Richtung $\Leftarrow$ der Vollständigkeitsteil des Beweises.

Die Behauptung ergibt sich aus:

$\mathrm{val}_I(1sh(P_1, P_2, P_3)) = W$
gdw. $\mathrm{val}_I(sh(P_1, P_2, P_3)) = W$
gdw. $\mathrm{val}_I(P_1 \wedge P_3 \vee \neg P_1 \wedge P_2) = W$
gdw. $\mathrm{val}_I(P_1 \wedge P_3) = W$ oder $\mathrm{val}_I(\neg P_1 \wedge P_2) = W$
gdw. $\mathrm{val}_I(P_1) = W$ und $\mathrm{val}_I(P_3) = W$ oder $\mathrm{val}_I(\neg P_1) = W$ und $\mathrm{val}_I(P_2) = W$
gdw. $\mathrm{val}_I(1P_1) = W$ und $\mathrm{val}_I(1P_3) = W$ oder $\mathrm{val}_I(0P_1) = W$ und $\mathrm{val}_I(1P_2) = W$

Analog für ist für die zweite Regel zu zeigen, dass

$$\mathrm{val}_I(0sh(P_1, P_2, P_3)) = W \quad \text{gdw.} \quad \begin{array}{c} \mathrm{val}_I(1P_1) = W \text{ und } \mathrm{val}_I(0P_3) = W \\ \text{oder} \\ \mathrm{val}_I(0P_1) = W \text{ und } \mathrm{val}_I(0P_2) = W \end{array}$$

Das ergibt sich aus:

$\mathrm{val}_I(0sh(P_1, P_2, P_3)) = W$
gdw. $\mathrm{val}_I(sh(P_1, P_2, P_3)) = F$
gdw. $\mathrm{val}_I((P_1 \to P_3) \wedge (\neg P_1 \to P_2)) = F$
gdw. $\mathrm{val}_I((\neg P_1 \vee P_3) \wedge (P_1 \vee P_2)) = F$
gdw. $\mathrm{val}_I(\neg P_1 \vee P_3) = F$ oder $\mathrm{val}_I(P_1 \vee P_2) = F$
gdw. $\mathrm{val}_I(\neg P_1) = F$ und $\mathrm{val}_I(P_3) = F$ oder $\mathrm{val}_I(P_1) = F$ und $\mathrm{val}_I(P_2) = F$
gdw. $\mathrm{val}_I(1P_1) = W$ und $\mathrm{val}_I(0P_3) = W$ oder $\mathrm{val}_I(0P_1) = W$ und $\mathrm{val}_I(0P_2) = W$

Zu Aufgabe 3

Formalisierung:

Der Benutzer möchte den Mail-Client *mutt* installieren.	$mutt$
Der Mail-Client erfordert einen *smtp daemon*.	$mutt \rightarrow smtp\_daemon$
Ein gültiger *smtp daemon* ist entweder *sendmail*, *postfix* oder *exim*,	$smtp\_daemon \leftrightarrow (sendmail \vee postfix \vee exim)$[1]
es kann nur einer gleichzeitig installiert werden.	$\neg(sendmail \wedge postfix)$ $\neg(sendmail \wedge exim)$ $\neg(postfix \wedge exim)$
sendmail macht das installierte Legacy-Paket *sendmail-tls* obsolet,	$\neg(sendmail \wedge sendmail\text{-}tls)$
das aber nicht deinstalliert werden darf.	$sendmail\text{-}tls$

Abb. 2 auf der folgenden Seite zeigt ein voll expandiertes (vorzeichenloses) Tableau für die Konjunktion der o.g. Formeln. Die Markierung der Art $_{23[\beta(4)]}$ bedeutet, daß die so markierte Formel Nr. 23 durch die Anwendung der β-Regel auf die Formel Nr. 4 entstanden ist.

Das Tableau hat vier offene Äste, jedes davon ergibt ein mögliches Modell der Formel. Insgesamt gibt es zwei verschiedene Modelle:

$$I(mutt) = W, I(smtp\_daemon) = W, I(sendmail) = F, I(postfix) = W, I(exim) = F, I(sendmail\text{-}tls) = W$$

oder

$$I(mutt) = W, I(smtp\_daemon) = W, I(sendmail) = F, I(postfix) = F, I(exim) = W, I(sendmail\text{-}tls) = W$$

Die als wahr interpretierten Atome geben genau die Pakete an, die installiert sind oder werden müssen, um den Benutzerwunsch zu erfüllen.

[1] Es ist auch möglich, *smtp_daemon* in anderen Formeln direkt syntaktisch durch die rechte Seite zu ersetzen.

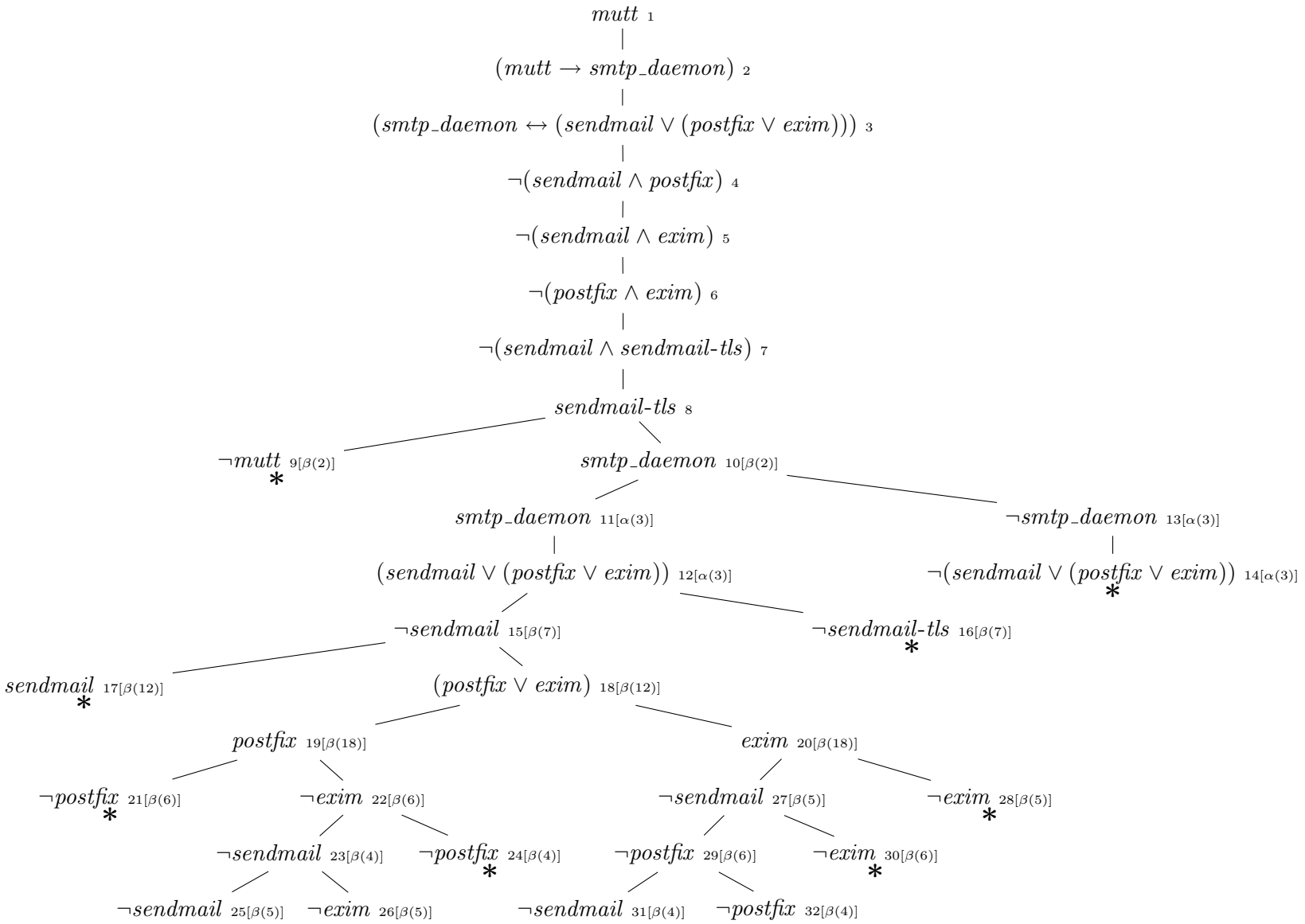

Abbildung 2: Ein voll expandiertes Tableau zur Aufgabe 3