Kapitel 7

Formeln mit Quantoren

Wir betrachten jetzt eine logische Sprache ASSN, mit deren Formeln wir Teilmengen von $Var \to \mathbb{Z}$ beschreiben können (für eine geeignet vorgegebene Menge Var). Wenn wir $Var = Loc$ setzen, können wir mit den Formeln von ASSN Zustandsmengen für IMP beschreiben. Im nächsten Kapitel werden wir diese Eigenschaft benutzen, um mit den Formel von ASSN Eigenschaften von IMP-Programmen zu spezifizieren.

Die Formeln von ASSN können sogenannte Quantoren enthalten. Beispielsweise ist

$$\forall x \in \mathbb{Z}\ \forall y \in \mathbb{Z}\ \exists z \in \mathbb{Z}\colon\ x + z = y$$

eine Formel von ASSN, die von allen Funktionen $Var \to \mathbb{Z}$ erfüllt wird. Dagegen ist

$$\exists z \in \mathbb{Z}\colon\ x = y \cdot z \ \wedge\ z \neq 0$$

eine Formel, die nicht von allen Funktionen $Var \to \mathbb{Z}$ erfüllt wird.

ASSN ist ein typischer Vertreter der Familie der prädikatenlogischen Sprachen, die wir in einem späteren Kapitel genauer betrachten werden. Im Folgenden gehen wir daher nur auf die Eigenschaften von ASSN ein, die wir im Zusammenhang mit der im nächsten Kapitel zu behandelnden Hoare-Logik benötigen.

Das hevorstechende syntaktische Merkmal der prädikatenlogischen Sprachen sind die Quantoren $\exists$ und $\forall$, die man allgemeiner auch als Variablenbinder bezeichnet. Bisher haben wir nur Sprachen ohne Variablenbinder betrachtet. Mithilfe der Quantoren kann man innerhalb einer Formel Variablen einführen, deren Gültigkeitsbereich auf eine Teilformel beschränkt ist. Das damit verbundene Konzept von lokalen Variablen ist uns aus Programmiersprachen bekannt.

Am Beispiel von ASSN werden wir sehen, dass das Konzept der Substitution für Sprachen mit Variablenbindern um einige neue Aspekte erweitert werden muss.

Lesematerial

[Winskel, Kapitel 6]

7.1 Syntax und Semantik von ASSN

Abbildung 7.1 definiert die Syntax und Semantik von ASSN. Die Definition erfolgt relativ zu einer nichtleeren Menge *Var*, deren Elemente wir als Variablen bezeichnen. Die Syntax unterscheidet zwischen Termen und Formeln, die den arithmetischen und Booleschen Ausdrücken von IMP entsprechen. Neu ist die existenzielle Quantifizierung. Die Semantik von Termen und Formel wird denotational definiert.

Die Formeln von ASSN sind alte Bekannte. Sie formalisieren Notationen, die wir routinemäßig benutzen, wenn wir mathematische Aussagen aufschreiben. Neu ist nur, dass wir diese Notationen jetzt als logische Sprache formalisiert haben. Der mathematischen Aussage

$$\exists x \in \mathbb{Z}\colon\; 5 + x = 3$$

entspricht beispielsweise die Formel

$$\exists X(5 + X = 3)$$

Da in ASSN alle Variablen auf ganzzahlige Werte beschränkt sind, ist es bei einer Quantifizierung nicht erforderlich, den Wertebereich der quantifizierten Variablen explizit anzugeben.

Mit ASSN können wir mathematische Aussagen als Formeln formulieren. Man spricht von einer Formulierung auf *Objektebene*. Die direkte Formulierung von mathematischen Aussagen mit den üblichen mathematischen Notationen bezeichnet man dagegen als Formulierung auf der *Metaebene*.

7.2 Abkürzungen und Expressivität

Abbildung 7.2 zeigt einige notationale Abkürzungen, die wir für die Syntax von ASSN benutzen werden. Wenn wir beispielsweise die Notation

$$\forall X(\forall Y(\exists Z(X + Z = Y)))$$

$$
\begin{array}{lcll}
n & \in & \mathit{Con} = \mathbb{Z} & \textit{Konstante} \\
X, Y, Z & \in & \mathit{Var} & \textit{Variable} \\
a & \in & \mathit{Ter} = & \textit{Term} \\
 & & \quad n & \quad \textit{Konstante} \\
 & & \mid X & \quad \textit{Variable} \\
 & & \mid a_1 + a_2 & \quad \textit{Summe} \\
 & & \mid a_1 * a_2 & \quad \textit{Produkt} \\
A, B, C & \in & \mathit{Assn} = & \textit{Formel} \\
 & & \quad a_1 \leq a_2 & \quad \textit{Vergleich} \\
 & & \mid A_1 \land A_2 & \quad \textit{Konjunktion} \\
 & & \mid \neg A & \quad \textit{Negation} \\
 & & \mid \exists X A & \quad \textit{existenzielle Quantifizierung}
\end{array}
$$

$$
\begin{array}{lcll}
\sigma & \in & \Sigma = \mathit{Var} \to \mathbb{Z} & \textit{Belegung}
\end{array}
$$

$$\mathcal{T} \in \mathit{Ter} \to \Sigma \to \mathbb{Z}$$

$$
\begin{aligned}
\mathcal{T}(n)\sigma &= n \\
\mathcal{T}(X)\sigma &= \sigma X \\
\mathcal{T}(a_1 + a_2)\sigma &= \mathcal{T}(a_1)\sigma + \mathcal{T}(a_2)\sigma \\
\mathcal{T}(a_1 * a_2)\sigma &= \mathcal{T}(a_1)\sigma \cdot \mathcal{T}(a_2)\sigma
\end{aligned}
$$

$$\mathcal{D} \in \mathit{Assn} \to \Sigma \to \mathbb{B}$$

$$
\begin{aligned}
\mathcal{D}(a_1 \leq a_2)\sigma &= \text{if } \mathcal{T}(a_1)\sigma \leq \mathcal{T}(a_2)\sigma \text{ then 1 else 0} \\
\mathcal{D}(A_1 \land A_2)\sigma &= \min\,\{\mathcal{D}(A_1)\sigma,\ \mathcal{D}(A_2)\sigma\} \\
\mathcal{D}(\neg A)\sigma &= 1 - \mathcal{D}(A)\sigma \\
\mathcal{D}(\exists X A)\sigma &= \max\,\{\, \mathcal{D}(A)(\sigma[n/X]) \mid n \in \mathbb{Z}\,\}
\end{aligned}
$$

Abbildung 7.1: Syntax und Semantik der Sprache ASSN

$$
\begin{array}{rcl}
-a & \mapsto & -1 * a \\
a_1 - a_2 & \mapsto & a_1 + (-1 * a_2) \\
\mathsf{tt} & \mapsto & 0 \le 1 \\
\mathsf{ff} & \mapsto & 1 \le 0 \\
a_1 = a_2 & \mapsto & a_1 \le a_2 \wedge a_2 \le a_1 \\
a_1 < a_2 & \mapsto & (a_1 + 1) \le a_2 \\
a_1 \ge a_2 & \mapsto & a_2 \le a_1 \\
a_1 > a_2 & \mapsto & a_2 < a_1 \\
A_1 \vee A_2 & \mapsto & \neg(\neg A_1 \wedge \neg A_2) \\
A_1 \Rightarrow A_2 & \mapsto & \neg(A_1 \wedge \neg A_2) \\
A_1 \Leftrightarrow A_2 & \mapsto & A_1 \Rightarrow A_2 \wedge A_2 \Rightarrow A_1 \\
\forall X A & \mapsto & \neg\exists X(\neg A)
\end{array}
$$

Abbildung 7.2: Abkürzungen für ASSN

als eine Formel von ASSN interpretieren, meinen wir damit die Formel

$$\neg\exists X(\neg\neg\exists Y(\neg\exists Z(X + Z = Y)))$$

Alternativ hätten wir die Abkürzungen in Abbildung 7.2 auch als zusätzliche Terme und Formeln von ASSN definieren können. Diese Vorgehensweise hat aber den Nachteil, dass alle auf ASSN aufbauenden Definitionen und Beweise komplizierter werden, da sie die zusätzlichen syntaktischen Varianten behandeln müssen. Die zusätzlichen Varianten bringen aber für unsere Zwecke nichts Neues, da sich ihre Denotationen entsprechend den Abkürzungsregeln auch mit den Grundvarianten ausdrücken lassen.

Betrachten Sie die Menge

$$DA \stackrel{\text{def}}{=} \{\, \mathcal{D}(A) \mid A \in Assn \,\}$$

die aus allen Funktionen in $\Sigma \to \mathbb{B}$ besteht, die sich mit Formeln aus *Assn* beschreiben lassen. Aus mathematischer Sicht sind die Elemente von *DA* das Wesentliche und die Formeln in *Assn* sind lediglich eine Hilfskonstruktion, mit der sich die Elemente von *DA* beschreiben lassen. Wenn die Elimination einer syntaktischen Variante nicht dazu führt, dass *DA* kleiner wird, ist diese Variante aus mathematischer Sicht redundant. Die syntaktischen Varianten von ASSN (wie in Abbildung 7.1 definiert) sind in dem Sinne minimal gewählt, dass das Weglassen einer Variante stets dazu führt, dass *DA* echt kleiner wird. Die Menge *DA* kann man als *Expressivität* der Sprache ASSN bezeichnen. Weglassen von syntaktischen Varianten kann die Expressivität verkleinern, Hinzufügen kann sie vergrößern (gemäß der Inklusionsordnung auf $\mathcal{P}(\Sigma \to \mathbb{B})$.

7.3 Notationen und Sprechweisen

Für die Formeln von ASSN verwenden wir die folgenden Notationen und Sprechweisen:

$$
\begin{array}{rcl}
\sigma \models A & \stackrel{\text{def}}{\iff} \sigma \text{ erfüllt } A & \stackrel{\text{def}}{\iff} \mathcal{D}(A)\sigma = 1 \\
\models A & \stackrel{\text{def}}{\iff} A \text{ gültig} & \stackrel{\text{def}}{\iff} \forall \sigma \in \Sigma : \ \sigma \models A \\
A \models B & \stackrel{\text{def}}{\iff} A \text{ impliziert } B & \stackrel{\text{def}}{\iff} \forall \sigma \in \Sigma : \ \sigma \models A \Rightarrow \sigma \models B \\
A \models\!\!\mid B & \stackrel{\text{def}}{\iff} A \text{ äquivalent } B & \stackrel{\text{def}}{\iff} \mathcal{D}(A) = \mathcal{D}(B)
\end{array}
$$

Machen Sie sich klar, dass $\models$ eine reflexive und transitive Relation auf *Assn* ist, und dass $\models\!\!\mid$ eine Äquivalenzrelation auf *Assn* ist. Offensichtlich gilt für beliebige $A, B \in$ ASSN:

$$
A \models B \iff \models A \Rightarrow B \iff \forall \sigma \in \Sigma : \ \mathcal{D}(A)\sigma \le \mathcal{D}(B)\sigma
$$

Sei $A \in Assn$ und sei α die A entsprechende Aussage auf der Metaebene (also in normaler mathematischer Notation). Wir haben die Semantik von ASSN gerade so definiert, dass α genau dann gilt, wenn $\models A$ gilt. Die Semantik von ASSN formalisiert also das Offensichtliche.

Angesichts dieser Tatsache werden Sie sich vielleicht fragen, warum wir überhaupt den Aufwand treiben, ASSN formal als logische Sprache zu definieren. Ein Grund dafür ist, dass wir in den nächsten Kapiteln einige wichtige Eigenschaften von ASSN zeigen, die alles andere als intuitiv klar sind. Für diese Beweise benötigen wir eine solide Grundlage. Außerdem ist es so, dass der Vorrat an mathematischen Notationen sehr groß ist, und gerade für den Anfänger nur bis zu einem gewissen Grad präzise definiert ist. Dagegen haben wir mit ASSN eine kleine Sprache zur Verfügung, deren Syntax und Semantik streng formal bis ins letzte Detail definiert sind.

Die Metavariable V wird im Folgenden für eine Menge von Formeln stehen:

$$
V \in \mathcal{P}(Assn)
$$

Wir werden die folgenden Notationen und Sprechweisen benutzen:

$$
\begin{array}{rcl}
\sigma \models V & \stackrel{\text{def}}{\iff} \sigma \text{ erfüllt } V & \stackrel{\text{def}}{\iff} \forall A \in V : \ \sigma \models A \\
\models V & \stackrel{\text{def}}{\iff} V \text{ gültig} & \stackrel{\text{def}}{\iff} \forall \sigma \in \Sigma : \ \sigma \models V \\
V_1 \models V_2 & \stackrel{\text{def}}{\iff} V_1 \text{ impliziert } V_2 & \stackrel{\text{def}}{\iff} \forall \sigma \in \Sigma : \ \sigma \models V_1 \Rightarrow \sigma \models V_2 \\
V_1 \models\!\!\mid V_2 & \stackrel{\text{def}}{\iff} V_1 \text{ äquivalent } V_2 & \stackrel{\text{def}}{\iff} V_1 \models V_2 \wedge V_1 \models V_2
\end{array}
$$

Machen Sie sich klar, dass $\models$ eine reflexive und transitive Relation auf $\mathcal{P}(Assn)$ ist, und dass $\models\!\!\mid$ eine Äquivalenzrelation auf $\mathcal{P}(Assn)$ ist.

7.4 Bindungsstruktur

Quantifizierung in Formeln führt eine Bindungsstruktur zwischen Variablen ein, die wir bereits am Beispiel von Programmiersprachen kennengelernt haben. Lesen Sie dazu [Programmierung, Kapitel 3.8]. Man unterscheidet zwischen *definierenden* und *benutzenden* Auftreten von Variablen. In ASSN werden definierende Auftreten durch eine existenzielle Quantifizierung

$$\exists X\, A$$

eingeführt. Das Auftreten von X nach dem Existenzquantor $\exists$ ist definierend und bindet alle freie Auftreten von X in A. Freie Auftreten sind benutzende Auftreten, die noch durch keine Quantifizierung gebunden sind.

Die Bindungsstruktur einer Formel kann man durch Annotationen explizit machen. Zum Beispiel können wir die Bindungsstruktur der Formel

$$\exists X(X \leq Y \wedge \exists Y(X \leq Y \wedge \exists X(X \leq Y)))$$

wie folgt explizieren:

$$\exists \overline{X}_1(X_1 \leq Y \wedge \exists \overline{Y_1}(X_1 \leq Y_1 \wedge \exists \overline{X_2}(X_2 \leq Y_1)))$$

Dabei werden alle definierende Auftreten von Variablen überstrichen. Zusätzlich wird jedes definierende Auftreten einer Variablen mit einem für diese Variable eindeutigen Index versehen. Alle benutzenden Auftreten, die durch ein definierendes Auftreten gebunden sind, werden mit dem Index dieses definierenden Auftretens annotiert. Freie Variablenauftreten erkennt man nach dieser Annotation daran, dass sie keinen Index tragen. In der obigen Formel gibt es also genau ein freies Auftreten einer Variablen. Dabei handelt es sich um das erste Auftreten der Variablen Y.

Syntaktische Varianten, die ein definierendes Auftreten einer Variable einführen, heißen *Variablenbinder*. ASSN hat genau einen Variablenbinder, nähmlich existenzielle Quantifizierung. Programmiersprachen haben in der Regel viele Variablenbinder. IMP ist eine triviale Programmiersprache, die keinen Variablenbinder hat.

Durch strukturelle Rekursion kann man eine Funktion

$$VT \in Ter \to \mathcal{P}_{fin}(Var)$$

definieren, die zu jedem Term a die Menge aller Variablen liefert, die in a vorkommen. Mit der gleichen Vorgehensweise kann man eine Funktion

$$VA \in Assn \to \mathcal{P}_{fin}(Var)$$

definieren, die zu jeder Formel A die Menge aller Variablen liefert, die in A vorkommen (gebunden oder ungebunden). Wir definieren jetzt eine Funktion FV, die zu jeder Formel A die Menge aller Variablen liefert, die in A frei auftreten:

$$FV \in Assn \to \mathcal{P}_{fin}(Var)$$

$$\begin{aligned} FV(a_1 \leq a_2) &= VT(a_1) \cup VT(a_2) \\ FV(A_1 \wedge A_2) &= FV(A_1) \cup FV(A_2) \\ FV(\neg A) &= FV(A) \\ FV(\exists X A) &= FV(A) - \{X\} \end{aligned}$$

Schließlich definieren wir eine Funktion BV, die zu jeder Formel A die Menge aller Variablen liefert, die in A ein definierendes Auftreten haben:

$$BV \in Assn \to \mathcal{P}_{fin}(Var)$$

$$\begin{aligned} BV(a_1 \leq a_2) &= \emptyset \\ BV(A_1 \wedge A_2) &= BV(A_1) \cup BV(A_2) \\ BV(\neg A) &= BV(A) \\ BV(\exists X A) &= BV(A) \cup \{X\} \end{aligned}$$

Eine Formel A heißt *bereinigt* genau dann, wenn die folgenden Bedingungen erfüllt sind:

1. Keine Variable hat in A mehr als ein definierendes Auftreten.
2. Keine Variable hat in A gleichzeitg ein definierendes und ein freies Auftreten.

7.5 Substitution

Sei $A \in Assn$ eine Formel, $a \in Ter$ ein Term und $X \in Var$ eine Variable. Dann bezeichnen wir mit

$$A[a/X]$$

die Formel, die wir aus A erhalten, indem wir alle freien Auftreten der Variable X durch den Ausdruck a ersetzen (wir sagen auch *substituieren*). Hier ist ein Beispiel:

$$\begin{aligned} A &= X * Y \leq Y \\ A[Y-1/Y] &= X * (Y-1) \leq (Y-1) \end{aligned}$$

Für Sprachen ohne Variablenbinder ist Substitution ein sehr einfaches Konzept. Für Sprachen mit Variablenbindern ist Substitution jedoch etwas komplizierter, da dann das Problem des Kaperns auftritt. Betrachten Sie dazu die Formel

$$A = \exists Y(Y + 1 \le X)$$

und nehmen Sie an, wir wollen alle freien Auftreten von X durch Y ersetzen, also $A[Y/X]$ bestimmen. Wenn wir naiv vorgehen, bekommen wir die Formel

$$\exists Y(Y + 1 \le Y)$$

Diese Substitution ist jedoch unzulässig, da das von außen kommende Auftreten von Y durch den Quantor der Formel *gekapert* wird. Der Grund, dass man Substitution ohne Kapern definieren will, liegt darin, dass für Substitution aus prinzipiellen Gründen immer die sogenannte *Substitutionseigenschaft* gelten soll:

$$\begin{aligned}&\forall A \in \mathit{Assn}\ \forall a \in \mathit{Ter}\ \forall X \in \mathit{Var}\ \forall \sigma \in \Sigma :\\ &\qquad \mathcal{D}(A[a/X])\sigma = \mathcal{D}(A)(\sigma[\mathcal{T}(a)\sigma/X])\end{aligned}$$

Für das obige unzulässige Beispiel ist die Substitutionseigenschaft verletzt, da für alle $\sigma \in \Sigma$:

$$\mathcal{D}(\exists Y(Y + 1 \le Y))\sigma = 0 \ne 1 = \mathcal{D}(\exists Y(Y + 1 \le X))(\sigma[\mathcal{T}(Y)\sigma/X])$$

Wir definieren Substitution zuerst mittels struktureller Rekursion für Terme. Das ist einfach, da hier das Problem des Kaperns nicht auftritt.

$$\begin{aligned} n[a/X] &= n\\ Y[a/X] &= \text{if } Y = X \text{ then } a \text{ else } Y\\ (a_1 + a_2)[a/X] &= a_1[a/X] + a_2[a/X]\\ (a_1 * a_2)[a/X] &= a_1[a/X] * a_2[a/X]\end{aligned}$$

Proposition 7.5.1 *Seien $a, a' \in \mathit{Ter}$, $X \in \mathit{Var}$ und $\sigma \in \Sigma$. Dann:*

$$\mathcal{T}(a[a'/X])\sigma = \mathcal{T}(a)(\sigma[\mathcal{T}(a')\sigma/X])$$

Beweis Durch strukturelle Induktion über $a \in \mathit{Ter}$. Übung! □

Jetzt definieren wir mittels struktureller Rekursion eine vorläufige Substitutionsfunktion für Formeln wie folgt:

$$S \in \mathit{Assn} \times \mathit{Ter} \times \mathit{Var} \to \mathit{Assn}$$

$$\begin{aligned} S(a_1 \le a_2, a, X) &= a_1[a/X] \le a_2[a/X]\\ S(A_1 \wedge A_2, a, X) &= S(A_1, a, X) \wedge S(A_2, a, X)\\ S(\neg A, a, X) &= \neg S(A, a, X)\\ S(\exists Y A, a, X) &= \text{if } Y = X \text{ then } \exists Y A \text{ else } \exists Y(S(A, a, X))\end{aligned}$$

Proposition 7.5.2 *Seien $A \in Assn$, $a \in Ter$, $X \in Var$ und $\sigma \in \Sigma$. Dann:*

$$BV(A) \cap VT(a) = \emptyset \;\Rightarrow\; \mathcal{D}(S(A, a, X))\sigma = \mathcal{D}(A)(\sigma[\mathcal{T}(a)\sigma/X])$$

Beweis Durch strukturelle Induktion über $A \in Assn$. Wir zeigen nur den Beweisteil für existenzielle Quantifizierung.

Sei $A = \exists Y A'$ und $BV(A) \cap VT(a) = \emptyset$. Wir müssen

$$\mathcal{D}(S(A, a, X))\sigma = \mathcal{D}(A)(\sigma[\mathcal{T}(a)\sigma/X])$$

zeigen. Dazu unterscheiden wir zwei Fälle.

Sei $X = Y$. Dann:

$$\begin{array}{ll}
\mathcal{D}(S(A, a, X))\sigma & \\
= \mathcal{D}(A)\sigma & \text{Def. von } S \\
= \text{if } \exists n \in \mathbb{Z}\colon\ \mathcal{D}(A')(\sigma[n/Y]) \text{ then } 1 \text{ else } 0 & \text{Def. von } \mathcal{D} \\
= \text{if } \exists n \in \mathbb{Z}\colon\ \mathcal{D}(A')(\sigma[\mathcal{T}(a)\sigma/X][n/Y]) \text{ then } 1 \text{ else } 0 & \text{da } X = Y \\
= \mathcal{D}(\exists Y A')(\sigma[\mathcal{T}(a)\sigma/X]) & \text{Def. von } \mathcal{D} \\
= \mathcal{D}(A)(\sigma[\mathcal{T}(a)\sigma/X]) &
\end{array}$$

Sei $X \neq Y$. Dann:

$$\begin{array}{ll}
\mathcal{D}(S(A, a, X))\sigma & \\
= \mathcal{D}(\exists Y(S(A', a, X)))\sigma & \text{Def. von } S \\
= \text{if } \exists n \in \mathbb{Z}\colon\ \mathcal{D}(S(A', a, X))(\sigma[n/Y]) \text{ then } 1 \text{ else } 0 & \text{Def. von } \mathcal{D} \\
= \text{if } \exists n \in \mathbb{Z}\colon\ \mathcal{D}(A')(\sigma[n/Y][\mathcal{T}(a)(\sigma[n/Y])/X]) \text{ then } 1 \text{ else } 0 & \text{Induktionsannahme} \\
= \text{if } \exists n \in \mathbb{Z}\colon\ \mathcal{D}(A')(\sigma[n/Y][\mathcal{T}(a)\sigma/X]) \text{ then } 1 \text{ else } 0 & Y \notin VT(a) \\
= \text{if } \exists n \in \mathbb{Z}\colon\ \mathcal{D}(A')(\sigma[\mathcal{T}(a)\sigma/X][n/Y]) \text{ then } 1 \text{ else } 0 & X \neq Y \\
= \mathcal{D}(\exists Y A')(\sigma[\mathcal{T}(a)\sigma/X]) & \text{Def. von } \mathcal{D} \\
= \mathcal{D}(A)(\sigma[\mathcal{T}(a)\sigma/X]) & \square
\end{array}$$

Wir haben jetzt eine Substitutionsfunktion für Formeln, die unter gewissen Vorbedingungen die Substitionseigenschaft erfüllt. Aus technischen Gründen ist es aber wünschenswert, eine Substitutionsfunktion zu haben, für die die Substitutionseigenschaft immer gilt.

Das Problem läßt sich mittels konsistenter Umbenennung von gebundenen Variablen lösen. Beispielsweise ist

$$\exists Z(Z + 1 \leq Y)$$

eine konsistente Umbenennung von

$$\exists X(X + 1 \leq Y)$$

Wir werden zeigen, dass Formel, die bis auf konsistente Umbenennung von gebundenen Variablen gleich sind, die gleiche Denotation haben. Wenn wir also einer Substitutionsfunktion erlauben, bei Bedarf gebundene Variablen konsistent umzubenennen, kann sie die Substitutioneigenschaft uneingeschränkt erfüllen. Allerdings müssen wir noch voraussetzen, dass es unendlich viele Variablen gibt.

Wir definieren jetzt eine Relation *CR* (consistent renaming)

$$A \sim A' \overset{\text{def}}{\iff} A \textit{ Variante} \text{ von } A' \overset{\text{def}}{\iff} \langle A, A' \rangle \in CR \subseteq Assn \times Assn$$

durch die folgenden Inferenzregeln:

$$\frac{S(A, Y, X) = A' \qquad Y \notin VA(A)}{\exists X A \sim \exists Y A'}$$

$$\frac{A_1 \sim A_1' \qquad A_2 \sim A_2'}{A_1 \wedge A_2 \sim A_1' \wedge A_2'} \qquad \frac{A \sim A'}{\neg A \sim \neg A'} \qquad \frac{A \sim A'}{\exists X A \sim \exists X A'}$$

$$\frac{}{A \sim A} \qquad \frac{A' \sim A}{A \sim A'} \qquad \frac{A \sim A'' \qquad A'' \sim A'}{A \sim A'}$$

Die Relation „$A \sim A'$“ formalisiert unsere intuitive Idee von konsistenter Umbenennung in dem Sinne, dass $A \sim A'$ genau dann gilt, wenn A und A' bis auf konsistente Umbenennung von gebundenen Variablen gleich sind. Das gilt allerdings nur dann, wenn es unendlich viele Variablen gibt. Betrachten Sie dazu

$$\exists X (\exists Y (X \leq Y) \sim \exists Y (\exists X (Y \leq X)$$

Diese Aussage gilt, wenn *Var* neben X und Y mindestens eine weitere Variable enthält. Sie gilt nicht, wenn X und Y verschieden sind und *Var* nur X und Y enthält.

Proposition 7.5.3 *Die Relation „$A \sim A'$“ ist eine Äquivalenzrelation. Für alle $A, A' \in Assn$ gilt:*

$$A \sim A' \Rightarrow \mathcal{D}(A) = \mathcal{D}(A')$$

Beweis Die letzten drei Inferenzregeln erzwingen, dass „$A \sim A'$“ eine Äquivalenzrelation ist. Die zweite Behauptung folgt mit Regelinduktion und Proposition 7.5.2. □

Definition 7.5.4 *Eine* Substitutionsfunktion *für ASSN ist eine Funktion*

$$s \in Assn \times Ter \times Var \to Assn$$

die die folgenden Bedingungen erfüllt:

$$\begin{array}{l} \forall A \in \mathit{Assn}\ \forall a \in \mathit{Ter}\ \forall X \in \mathit{Var}\ \exists A' \in \mathit{Assn}: \\ \qquad A \sim A' \wedge BV(A') \cap VT(a) = \emptyset \ \Rightarrow\ s(A, a, X) \sim S(A', a, X) \end{array}$$

Proposition 7.5.5 *Wenn Var eine unendliche Menge ist, gibt es eine Substitutionsfunktion für ASSN. Jede Substitutionsfunktion s für ASSN erfüllt die Substitutionseigenschaft:*

$$\begin{array}{l} \forall A \in \mathit{Assn}\ \forall a \in \mathit{Ter}\ \forall X \in \mathit{Var}\ \forall \sigma \in \Sigma: \\ \qquad \mathcal{D}(s(A, a, X))\sigma = \mathcal{D}(A)(\sigma[\mathcal{T}(a)\sigma/X]) \end{array}$$

Wir werden im Folgenden voraussetzen, dass wir eine Substitutionsfunktion $A[a/X]$ für ASSN festgelegt haben. Welche das ist, spielt keine Rolle. Beispiele können wir wie folgt schreiben:

$$(\exists Y(Y + 1 \leq X))[Y/X] \sim \exists Z(Z + 1 \leq Y)$$

Übrigens umgeht Winskel in seinem Buch das Kaper-Problem für Substitutionen dadurch, dass er mit zwei Arten von Variablen arbeitet. Quantifizieren darf man nur über die eine Art von Variable, und die zu substituierenden Terme dürfen nur die zweite Art von Variablen enthalten.