

3. Logische Grundlagen des PVS-Beweisers

- Grundbegriffe: Beweiskalkül und Beweis
- Sequenzen-Kalkül
- Mechanisierung des Sequenzen-Kalküls in PVS

Was ist ein Beweis?

Beweis: Systematische Ableitung einer Aussage (eines “Theorems”) in einem *Beweis-Kalkül*

Der Kalkül legt fest:

- die Sprache, in der Aussagen (Formeln) formuliert werden
 - mit formaler Syntax und Semantik;
- Axiome als Anfangspunkt(e) eines Beweises;
- Schlußregeln zur Ableitung neuer Formeln aus Axiomen und bereits bewiesenen Formeln

Allgemeine Form einer Schlußregel:

$$\frac{\text{Liste von Prämissen}}{\text{Schlußfolgerung}} \text{ Name der Regel (optional)}$$

Beispiel:

$$\frac{A \quad A \Rightarrow B}{B} \text{ Modus ponens}$$

Beweis-Konstruktion

Ein Beweis kann als Baum präsentiert werden; die bewiesene Formel ist die Formel an der Wurzel des Baums.

Ein Beweis wird meistens vom Ziel her, d.h. ausgehend von der zu beweisenden Formel an der Wurzel, konstruiert.

Die Beweisregeln werden dann “rückwärts” angewendet, d.h. als Generatoren von Prämissen als Unterzielen.

Beweis-Konstruktion in PVS:

- Sprache ist getypte Prädikatenlogik
(höherer Stufe und angereichert – mehr dazu später)
- Beweiskalkül ist (im wesentlichen) der *Sequenzenkalkül*

Sequenzen-Kalkül

Der Sequenzen-Kalkül bildet die formal-logische Grundlage des PVS-Beweislers.

Der Sequenzenkalkül ist einer der von dem Logiker G. Gentzen vorgeschlagenen Kalküle; daher auch die alternative Bezeichnung “Gentzen-Kalkül”.

- Das Grundelement des Kalküls ist die *Sequenz* (engl. *sequent*); der Kalkül beschreibt im wesentlichen, wie mit Sequenzen umgegangen wird.
- Eine Sequenz ist ein Ausdruck der Form $\Gamma \vdash \Delta$.
 Γ und Δ sind beliebige (endliche) Mengen (oder Multimengen) von Formeln (mit gewissen Einschränkungen; siehe unten).
 Γ heißt der *Antezedent*, Δ der *Sukzedent* der Sequenz.
- Antezedent und Sukzedent können jeweils auch leer sein.

Sequenzen – intuitiv

Intuitive Interpretation von Sequenzen:

Einer Sequenz $A_1, \dots, A_m \vdash B_1, \dots, B_n$ entspricht die Aussage, daß die Implikation

$$A_1 \wedge \dots \wedge A_m \Rightarrow B_1 \vee \dots \vee B_n$$

wahr ist

(bzw. daß $\neg A_1 \vee \dots \vee \neg A_m \vee B_1 \vee \dots \vee B_n$ wahr ist).

Bei atomaren A_i, B_j erkennt man unschwer die Nähe zur Klausel-Form.

Eine Sequenz $A_1, \dots, A_m \vdash B_1, \dots, B_n$ heißt *gültig*, wenn die Formel $A_1 \wedge \dots \wedge A_m \Rightarrow B_1 \vee \dots \vee B_n$ unter jeder Interpretation wahr ist.

Praktisches Ziel des Sequenzen-Kalküls ist die Ableitung eines Beweis für Formelmenge Δ unter der Annahme der Formeln in Γ .

Also: $\Gamma \vdash \Delta$.

Sequenzen-Kalkül: Aussagenlogik

Das einzige *Axiom* ist:

$$\Gamma, A \vdash A, \Delta$$

Jede Sequenz, in der im Antezedenten und Sukzedenten dieselbe Formel auftritt, ist eine Axiom.

Strukturelle Regeln:

$$\frac{A, A, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta} \text{contr:L} \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A} \text{contr:R}$$

$$\frac{\Gamma \vdash \Delta, A \quad A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta} \text{cut}$$

Sequenzen-Kalkül: Aussagenlogik (2)

Für die logischen Operatoren gibt es jeweils Paare von Regeln.

↷ Symmetrie zwischen linker und rechter Seite einer Sequenz

Logische Regeln:

$$\frac{\Gamma \vdash \Delta, A \quad B, \Gamma \vdash \Delta}{A \Rightarrow B, \Gamma \vdash \Delta} \Rightarrow L$$

$$\frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \Rightarrow B} \Rightarrow R$$

$$\frac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta} \wedge L$$

$$\frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B} \wedge R$$

$$\frac{A, \Gamma \vdash \Delta \quad B, \Gamma \vdash \Delta}{A \vee B, \Gamma \vdash \Delta} \vee L$$

$$\frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B} \vee R$$

$$\frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta} \neg L$$

$$\frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A} \neg R$$

Sequenzen-Kalkül: Aussagenlogik (3)

Bemerkungen:

- Antezedent und Sukzedent werden vollständig symmetrisch behandelt
- Daher jetzt immer Anwendbarkeitskriterium für die Regeln. Ausnahme: Die Schnittregel *cut*.
- Es ist nicht nötig, beim Rückwärtsbeweis eine Formel zu erraten.

Die Regeln haben folgende *Unterformel-Eigenschaft*: Tritt eine Formel in der Vorbedingung einer Regel auf, so auch in der Nachbedingung (zumindest als Unterformel).

Ausnahme: Schnittregel.

Sequenzen-Kalkül – Beispiele

Beispiel: Beweise $A \wedge B \vdash A \vee C$

1.

$$\frac{\frac{A, B \vdash A}{A \wedge B \vdash A} \wedge L}{A \wedge B \vdash A \vee C} \vee R$$

2. Es ist nicht notwendig, sich auf die Wahl von A oder C festzulegen:

$$\frac{\frac{A, B \vdash A, C}{A \wedge B \vdash A, C} \wedge L}{A \wedge B \vdash A \vee C} \vee R$$

3. Ein Beweis kann auch “Umwege” enthalten:

$$\frac{\frac{A, C \vdash C}{A \wedge C \vdash C} \wedge L \quad \frac{C \vdash C, B}{C \vdash C \vee B} \vee R}{A \wedge C \vdash C \vee B} cut$$

Elimination der Schnittregel und Anwendungen

Die Schnittregel *cut* stört einige wünschenswerten Eigenschaften.

Man kann zeigen, daß die Schnittregel redundant ist und sich jeder Beweis durch Schnittelimination “normalisieren” läßt:

Schnitt-Eliminationssatz (Gentzen):

Jeder Beweis, der einen Schnitt enthält, läßt sich in einen Beweis ohne Schnitt überführen

Dabei kann (!) die Beweislänge über-exponentiell anwachsen.

Anwendungen:

Durch die Unterformel-Eigenschaft der verbleibenden Regelmengen enthält man Entscheidungsverfahren für die klassische Aussagenlogik.

Sequenzen-Kalkül für Prädikatenlogik

Zu den aussagenlogischen Regeln kommen die Regeln zur Behandlung quantifizierter Formeln hinzu:

$$\frac{A[x \leftarrow t], \Gamma \vdash \Delta}{\forall x.A, \Gamma \vdash \Delta} \forall L$$

$$\frac{\Gamma \vdash \Delta, A[x \leftarrow y]}{\Gamma \vdash \Delta, \forall x.A} \forall R$$

$$\frac{A[x \leftarrow y], \Gamma \vdash \Delta}{\exists x.A, \Gamma \vdash \Delta} \exists L$$

$$\frac{\Gamma \vdash \Delta, A[x \leftarrow t]}{\Gamma \vdash \Delta, \exists x.A} \exists R$$

Notation: $A[x \leftarrow t]$ bezeichnet Substitution von t für jedes freie Vorkommen der Variablen x in A .

y ist jeweils eine beliebige Variable und t ein beliebiger Term, die folgenden Bedingungen genügen:

- In den Regeln $\forall L$ und $\exists R$ darf keine Variable von t von einem Quantor in A gebunden werden (“*variable capture*”); t muss ‘frei für x in A ’ sein.

Gegebenenfalls sind gebundene Variablen umzubenennen.

- In $\forall R$ und $\exists L$ darf die *Eigenvariable* y nicht frei in der Schlußfolgerung der Regel vorkommen (“Eigenvariablen-Bedingung”)

Beispiele für prädikatenlogische Beweise

1. Beweis der Sequenz $(\forall x.P(x) \wedge Q(x)) \vdash (\forall x.P(x)) \wedge (\forall x.Q(x))$

$$\frac{\frac{\frac{Pv \wedge Qv \vdash Pv}{(\forall x.Px \wedge Qx) \vdash Pv} \forall L}{(\forall x.Px \wedge Qx) \vdash \forall x.Px} \forall R \quad \frac{\frac{Pv \wedge Qv \vdash Qv}{(\forall x.Px \wedge Qx) \vdash Qv} \forall L}{(\forall x.Px \wedge Qx) \vdash \forall x.Qx} \forall R}{(\forall x.Px \wedge Qx) \vdash (\forall x.Px) \wedge (\forall x.Qx)} \wedge R$$

2. Beweis der Sequenz $P(a), \forall x.(P(x) \Rightarrow P(f(x))) \vdash P(f(f(a)))$

$$\frac{\frac{\frac{Pa, Pa \Rightarrow Pfa, Pfa \Rightarrow Pffa \vdash Pffa}{Pa, Pa \Rightarrow Pfa, (\forall x.Px \Rightarrow Pfx) \vdash Pffa} \forall L}{Pa, (\forall x.Px \Rightarrow Pfx), (\forall x.Px \Rightarrow Pfx) \vdash Pffa} \forall L}{Pa, (\forall x.Px \Rightarrow Pfx) \vdash Pffa} \text{contr:L}$$

Es kann also nötig sein, mehrmals die Kontraktionsregel *contr* anzuwenden, um eine Sequenz zu beweisen.

3. Beweis für $(\exists x \forall y. P(x, y)) \vdash (\forall y \exists x. P(x, y))$

$$\frac{\frac{\frac{P(v, w) \vdash P(v, w)}{P(v, w) \vdash \exists x. P(x, w)}{\exists R} \quad \forall L}{\forall y. P(v, y) \vdash \exists x. P(x, w)} \quad \forall R}{\forall y. P(v, y) \vdash \forall y \exists x. P(x, y)} \quad \exists L}{\exists x \forall y. P(x, y) \vdash \forall y \exists x. P(x, y)} \quad \exists L$$

Negativ-Beispiele

Die folgenden Beispiele falscher Regelanwendungen motivieren die Bedingungen für die prädikatenlogischen Regeln.

Vermeidung des Bindens freier Variablen:

Man versuche zu beweisen: $\forall x.P(x, fx) \vdash \exists y \forall x.P(x, y)$

Dies ist eine ungültige Sequenz (interpretiere z.B. P als das Prädikat “ist Vorgänger von” auf den ganzen Zahlen, f als die Nachfolger-Funktion).

$$\frac{\forall x.P(x, fx) \vdash \forall x.P(x, fx)}{\forall x.P(x, fx) \vdash \exists y \forall x.P(x, y)} \exists R$$

Bei dem Rückwärtsschritt gerät die Variable x in dem Term fx unter den Einfluß des All-Quantors.

Negativ-Beispiele [2]

Eigenvariablenbedingung:

Versuche zu beweisen: $(\forall y \exists x.P(x, y)) \vdash (\exists x \forall y.P(x, y))$

Dies ist eine ungültige Sequenz (interpretiere z.B. P als das Prädikat \geq auf den ganzen Zahlen).

$$\frac{\frac{\frac{P(v, w) \vdash P(v, w)}{\exists x.P(x, w) \vdash P(v, w)} \exists L}{\exists x.P(x, w) \vdash \forall y.P(v, y)} \forall R}{\forall y \exists x.P(x, y) \vdash \forall y.P(v, y)} \forall L}{\forall y \exists x.P(x, y) \vdash \exists x \forall y.P(x, y)} \exists R$$

Bei diesem Beweis ist in dem Schritt $\forall R$ die Eigenvariablenbedingung verletzt (w ist frei in der Nachbedingung der Regel).

Mechanisierung des Sequenzen-Kalküls in PVS

Die Beweiser-Komponente von PVS implementiert den Sequenzen-Kalkül.

Darstellung von Sequenzen in PVS: Eine Sequenz

$$A_1, A_2, \dots, A_n \vdash B_1, \dots, B_m$$

wird repräsentiert in der Form:

$$\begin{array}{l} \{-1\} \quad A_1 \\ \{-2\} \quad A_2 \\ \dots \\ [-n] \quad A_n \\ |----- \\ [1] \quad B_1 \\ \dots \\ \{m\} \quad B_m \end{array}$$

Sequenzen-Kalkül in PVS (2)

- PVS vermeidet, explizit negierte Teilformeln in einer Sequenz auftreten zu lassen. Negation als führender Operator wird im wesentlichen durch normierte Darstellung von Sequenzen eliminiert: negierte Teilformeln erscheinen als positive Teilformeln auf der jeweils anderen Seite.
- Die Index-Nummern der Teilformeln können zur Steuerung des Beweisers benutzt werden (als Argumente zu Beweiser-Befehlen).
- Formeln im Antezedent werden negativ durchnummeriert, Formeln im Sukzedent positiv.
- Die Index-Nummern von Formeln, die durch den letzten Beweiserschritt verändert oder neu hinzu gekommen sind, sind in $\{ \}$ eingeschlossen.

Beweis-Konstruktion in PVS

- Ein Beweis in PVS entspricht einem Beweisbaum, der von der “Wurzel”, der zu beweisenden Sequenz, ausgehend konstruiert wird.
- Beweise werden konstruiert durch Generierung von ein oder mehreren Unterzielen, entsprechend der “Rückwärtsanwendung” von Beweiserregeln; dadurch wird rückwärts ein Beweisbaum aufgebaut.
- Eine bestimmte Sequenz (engl. *current sequent*) ist zu jeder Zeit der Fokus des Beweisprozesse; auf sie beziehen sich jeweils die Anweisungen an den Beweiser. Wenn eine Beweiser-Anweisung neue Teilziele erzeugt, wandert der Fokus auf das “erste” neue Teilziel.
- Der Fokus kann durch Anweisungen verändert werden (*postpone*); durch Änderung des Fokus können Teilziele in beliebiger Reihenfolge bearbeitet werden.

Beweis-Konstruktion in PVS (2)

- Wird ein Teilziel bzw. ein Beweis-Ast als bewiesen erkannt (z.B. weil die Sequenz als eine Axiomeninstanz erkannt wurde), wird dieser Ast abgeschlossen, und der Fokus geht auf das nächste noch offene (d.h. unbewiesene) Teilziel.

Falls es kein solches Teilziel mehr gibt, wird der Beweis als ganzes abgeschlossen.

- Ein Beweisschritt kann explizit zurückgenommen werden durch das Kommando (undo), oder ganz abgebrochen mit dem Kommando (quit)

PVS Beweiser-Kommandos

Im Prinzip könnten die Schlußregeln des Sequenzen-Kalküls direkt als Kommandos eines Beweisers implementiert werden. Das würde die Konstruktion von Beweisen aber unnötig mühsam machen.

In PVS haben stattdessen einige Kommandos jeweils den Effekt von *Gruppen* von Schlußregeln.

Beweiser-Kommandos, die den logischen Regeln (Regeln für aussagenlogische Verknüpfungen) entsprechen:

`flatten` “disjunktive Simplifikation”

Disjunktionen im Sukzedenten, und entsprechend Konjunktionen im Antezedenten, werden aufgelöst

Entspricht (Rückwärts-)Anwendungen der Regeln $\vee R, \Rightarrow R, \wedge L$.

PVS Beweiser-Kommandos (2)

`split` “konjunktiv Aufspaltung”

Für eine konjunktive Teilformel im Sukzedenten bzw. eine disjunktive Teilformel im Antecedenten werden entsprechende Teilziele generiert.

Selektion einer Teilformel notwendig (Default ist “erste, die gefunden wird”).

Entspricht (Rückwärts-)Anwendungen der Regeln $\forall L, \Rightarrow L, \wedge R$.

`case` “Fallunterscheidung” (*case splitting*)

Für eine oder mehrere Argument-Formeln werden Teilziele generiert, in denen jeweils die Formeln als Voraussetzungen (im Antecedenten) bzw. als Schlußfolgerung (im Sukzedenten) hinzugefügt werden.

Entspricht einer oder mehrerer (Rückwärts-)Anwendungen der Regel *cut*.

`(lift-if &rest fnums [*])`: hebt bedingte Ausdrücke (in den Formeln *fnums*) auf oberste Ebene an.

PVS Beweiser-Kommandos (3)

Behandlung von Quantoren

(`skolem`) Universell quantifizierte Variablen (im Sukzedenten; im Akzedenten: existentiell quantifizierte Variablen) werden durch Skolem-Konstanten im neuen Teilziel ersetzt.

Entspricht (Rückwärts-)Anwendung der Regel $\forall R, \exists L$.

(`inst`) Instanziierung universell quantifizierter Variablen (im Antezedenten; existentiell quantifizierter Variablen im Sukzedenten)

Entspricht (Rückwärts-)Anwendung der Regel $\forall L, \exists R$.

(`lemma`) Einführung eines Lemmas als weiterer Teilformel in den Antezedenten – möglicherweise mit Instanziierung von universell quantifizierten Variablen

PVS Beweiser-Kommandos (4)

Es gibt jeweils Versionen, die explizite Referenzen auf Teilformeln und Angabe von neuen Konstanten-Namen bzw. Termen zulassen, oder geeignete Konstanten generieren (`sko1em!`) bzw. versuchen, geeignete Terme zu finden (`inst?`).

Das `lemma`-Kommando kann als eine Art “reduzierte Schnittregel” angesehen werden, die eine Voraussetzung (Beweis des eingeführten Lemmas) nicht explizit einführt.