Entwicklung eines Beweises in Minlog

Stephan Kulla (CC-BY)

15.05.2011

1 Einleitung

Diesen Artikel habe ich als Ergänzung für einen Vortrag im Seminar Rechnerischer Gehalt von Beweisen geschrieben. Ich werde in diesem Artikel den Beweis für den Satz, dass eine injektive Abbildung $f : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ surjektiv ist, schrittweise entwickeln. Ziel ist der Beweis dieser Aussage in Minlog, einem in Scheme geschriebenes Programm zur Beweisführung.

Zunächst werde ich den Beweis des Satzes, dass injektive Abbildungungen $f : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ surjektiv sind, so führen, wie man es in gängigen Lehrbüchern zur Mathematik tun würde. Danach folgt ein ausführlicher Beweis, der sich eng an dem Beweis anlehnt, den ich in Minlog geführt habe. So soll es dem Leser erleichtert werden, den Beweis in Minlog nachzuvollziehen. Im letzten Schritt werde ich den Beweis in Minlog führen.

Die Idee zum Vortrag stammt aus dem Artikel The development of a text in AUT-QE. In diesem Artikel entwickelt L.S. van Benthem Jutting den Beweis zum obigen Satz in vier Schritten, wobei sein Ziel der Beweis in Automath ist. Automath ist eine formale Sprache zur Beweisführung, welche von Nicolaas Govert de Bruijn entwickelt wurde. In diesem Artikel werde ich aber nicht weiter auf Automath eingehen.

2 Einfacher Beweis

Zu beweisende Aussage: Jede injektive Abbildung $f : \{0, 1, \ldots, n\} \to \{0, 1, \ldots, n\}$ ist surjektiv.

Induktionsanfang: Zu beweisende Aussage ist für $n = 0$ trivial.

Induktionsschritt: Sei $f : \{1, \ldots, n, n+1\} \to \{1, \ldots, n, n+1\}$ eine injektive Funtkion.

Fall 1: $f(n+1) = n+1$
Für $m = n+1$ ist $n+1$ Urbild von m unter f. Sei nun $m \leq n$. Wegen $f(n+1) = n+1$ und der Injektivität von f ist f eingeschränkt auf die Definitionsmenge $\{1, \ldots, n\}$ eine injektive Abbildung $\{1, \ldots, n\} \to \{1, \ldots, n\}$. Auf diese Abbildung kann die Induktionsvoraussetzung angewandt werden und man erhält die Existenz eines Urbildes von m.

Fall 2: $f(n+1) \neq n+1$

Für $m = f(n+1)$ ist $n+1$ das Urbild von m unter f. Sei nun $m \neq f(n+1)$. Wir definieren die neue Funktion g mit

$$g : \{1, \ldots, n\} \to \{1, \ldots, n\} : x \mapsto \begin{cases} f(n+1) & ; f(x) = n+1 \\ f(x) & ; f(x) \neq n+1 \end{cases}$$

Auf g kann die Induduktionsvoraussetzung angewandt werden, womit g surjektiv ist. Sei zunächst $m = n+1$. Aus der Surjektivität von g folgt sie Existenz eines k_0 mit $g(k_0) = f(n+1)$. Dieses k_0 erfüllt aber nach Definition von g die Gleichung $f(k_0) = n+1$ und damit ist k_0 das Urbild von m unter f.

Sei nun $m \neq n+1$ und damit $m \leq n$. Außerdem haben wir bereits $m \neq f(n+1)$ vorausgesetzt. Wegen Surjektivität von g gibt es ein k_0 mit $g(k_0) = m \neq f(n+1)$. Aus der Definition von g folgt dann $g(k_0) = f(k_0)$, also $f(k_0) = m$. Damit ist k_0 Urbild von m unter f.

3 Ausführlicher Beweis

Zu beweisende Aussage:

$$\begin{aligned} &\forall n \in \mathbb{N}, f \in \{g \mid g \text{ ist eine Abbildung } \mathbb{N} \to \mathbb{N}\} \, (\\ &\qquad \forall i \in \mathbb{N} \, (i \leq n \to f(i) \leq n) \\ &\quad \to \quad \forall i \in \mathbb{N}, j \in \mathbb{N} \, (i \leq n \to j \leq n \to f(i) = f(j) \to i = j) \\ &\quad \to \quad \forall m \in \mathbb{N} \, (m \leq n \to \exists k \in \mathbb{N} (k \leq n \wedge f(k) = m)) \\ ) \end{aligned}$$

Induktionsanfang:
Sei $f : \mathbb{N} \to \mathbb{N}$ eine Abbildung mit $\forall i \in \mathbb{N} \, (i \leq 0 \to f(i) \leq 0)$ und $\forall i \in \mathbb{N}, j \in \mathbb{N} \, (i \leq 0 \to j \leq 0 \to f(i) = f(j) \to i = j)$. Sei $m \in \mathbb{N}$ mit $m \leq 0$ und damit $m = 0$. Aus $\forall i \in \mathbb{N} \, (i \leq 0 \to f(i) \leq 0)$ folgt für $i = 0$, dass $f(0) \leq 0$ und somit $f(0) = 0$ ist. Damit ist 0 das Urbild von $m = 0$.

Induktionsschluss:
Sei $f : \mathbb{N} \to \mathbb{N}$ eine Abbildung mit $\forall i \in \mathbb{N} \, (i \leq n+1 \to f(i) \leq n+1)$ und $\forall i \in \mathbb{N}, j \in \mathbb{N} \, (i \leq n+1 \to j \leq n+1 \to f(i) = f(j) \to i = j)$. Sei $m \in \mathbb{N}$ mit $m \leq n+1$. Es ist zu beweisen, dass m unter f ein Urbild besitzt.

Fall 1: $f(n+1) = n+1$

Fall 1.1: $f(n+1) = n+1$ und $m = n+1$
Aus $f(n+1) = n+1$ und $m = n+1$ folgt $f(n+1) = m$. Damit ist $n+1$ das Urbild von m unter f.

Fall 1.2: $f(n+1) = n+1$ und $m \neq n+1$
Aus $m \neq n+1$ und $m \leq m+1$ folgt $m \leq n$. Außerdem folgt aus $f(n+1) = n+1$ und der Injektivität von f, dass $f(i) \leq n$ für $i \leq n$ ist. Damit lässt sich die Induktionsvoraussetzung für f und $m \leq n$ anwenden. Hieraus erhält man, dass m ein Urbild unter f besitzt.

Fall 2: $f(n+1) \neq n+1$

Fall 2.1: $f(n+1) \neq n+1$ und $m = f(n+1)$
Aus $m = f(n+1)$ folgt, dass $n+1$ das Urbild von m unter f ist.

Fall 2.2: $f(n+1) \neq n+1$ und $m \neq f(n+1)$
Sei $g : \mathbb{N} \to \mathbb{N}$ definiert durch $g(x) = \begin{cases} f(n+1) & ; f(x) = n+1 \\ f(x) & ; f(x) \neq n+1 \end{cases}$. $g : \{1, \ldots, n\} \to \{1, \ldots, n\}$ ist surjektiv nach Induktionsvoraussetzung. Dass sich auf g die Induktionsvoraussetzung anwenden lässt, wird später bewiesen*.

Fall 2.2.1: $f(n+1) \neq n+1$, $m \neq f(n+1)$ und $m = n+1$
Aus $f(n+1) \neq n+1$ und $f(n+1) \leq n+1$ folgt $f(n+1) \leq n$ und damit gibt es ein $k_0 \in \mathbb{N}$ mit $k_0 \leq n$ und $g(k_0) = f(n+1)$.

Fall 2.2.1.1: $f(n+1) \neq n+1$, $m \neq f(n+1)$, $m = n+1$ und $f(k_0) = n+1$
Wegen $m = n+1$ und $f(k_0) = n+1$ ist $f(k_0) = m$ und somit k_0 Urbild von m unter f.

Fall 2.2.1.2: $f(n+1) \neq n+1$, $m \neq f(n+1)$, $m = n+1$ und $f(k_0) \neq n+1$
Wegen $f(k_0) \neq n+1$ ist $g(k_0) = f(k_0)$ und damit $f(k_0) = f(n+1)$. Aus der Injektivität von f folgt $k_0 = n+1$. $k_0 = n+1$ steht aber im Widerspruch zu $k_0 \leq n$. Somit kann dieser Fall nie auftreten.

Fall 2.2.2: $f(n+1) \neq n+1$, $m \neq f(n+1)$ und $m \neq n+1$
Aus $m \neq n+1$ und $m \leq n+1$ folgt $m \leq n$. Aus der Surjektivität von g folgt die Existenz eines $k_0 \in \mathbb{N}$ mit $k_0 \leq n$ und $g(k_0) = m$.

Fall 2.2.2.1: $f(n+1) \neq n+1$, $m \neq f(n+1)$, $m \neq n+1$ und $f(k_0) = n+1$
Aus $f(k_0) = n+1$ folgt nach Definition von g die Gleichung $g(k_0) = f(n+1)$ und damit $f(n+1) = m$. Dies steht aber im Widerspruch zu $m \neq f(n+1)$. Somit kann dieser Fall nie auftreten.

Fall 2.2.2.2: $f(n+1) \neq n+1$, $m \neq f(n+1)$, $m \neq n+1$ und $f(k_0) \neq n+1$
Wegen $f(k_0) \neq n+1$ ist $g(k_0) = f(k_0)$ und damit $f(k_0) = m$. Somit ist k_0 das Urbild von m unter f.

*** Nachträglicher Beweis:** g ist eine Abbildung $\{1, \ldots, n\} \to \{1, \ldots, n\}$
Sei $i \in \mathbb{N}$ mit $i \leq n$. Zu beweisen ist $g(i) \leq n$.

Fall 1: $f(i) = n+1$
Wegen $f(i) = n+1$ ist $g(i) = f(n+1)$. Mit $f(n+1) \neq n+1$ und $f(n+1) \leq n+1$ ist $f(n+1) \leq n$ und damit $g(i) \leq n$.

Fall 2: $f(i) \neq n+1$
Wegen $f(i) \neq n+1$ ist $g(i) = f(i)$. Mit $f(i) \neq n+1$ und $f(i) \leq n+1$ ist $f(i) \leq n$ und damit $g(i) \leq n$.

*** Nachträglicher Beweis:** g ist auf $\{1, \ldots, n\}$ injektiv
Sei $i, j \in \mathbb{N}$ mit $i \leq n$ und $j \leq n$. Sei außerdem $g(i) = g(j)$. Zu beweisen ist $i = j$.

Fall 1: $f(i) = n+1$

Fall 1.1: $f(i) = n+1$ und $f(j) = n+1$
Im Fall $f(i) = n+1$ und $f(j) = n+1$ ist $f(i) = f(j)$ und damit wegen Injektivität von f auch $i = j$.

Fall 1.2: $f(i) = n+1$ und $f(j) \neq n+1$
Im Fall $f(i) = n+1$ und $f(j) \neq n+1$ folgt aus $g(i) = g(j)$ die Gleichung $f(n+1) = f(j)$. Hieraus folgt wegen Injektivität von f die Gleichung $n+1 = j$, die im Widerspruch zu $j \leq n$ steht.

Fall 2: $f(i) \neq n+1$

Fall 2.1: $f(i) \neq n+1$ und $f(j) = n+1$
Im Fall $f(i) \neq n+1$ und $f(j) = n+1$ folgt aus $g(i) = g(j)$ die Gleichung $f(i) = f(n+1)$. Hieraus folgt wegen Injektivität von f die Gleichung $i = n+1$, die im Widerspruch zu $i \leq n$ steht.

Fall 2.2: $f(i) \neq n+1$ und $f(j) \neq n+1$
Im Fall $f(i) \neq n+1$ und $f(j) \neq n+1$ folgt aus $g(i) = g(j)$ die Gleichung $f(i) = f(j)$. Hieraus folgt wegen Injektivität von f die zu beweisende Gleichung $i = j$.

4 Beweisführung in Minlog

Die Beweisführung von Minlog wird in zwei Dateien vorgenommen: `beweis.scm` und `lemmas.scm`. In der Datei `lemmas.scm` werden alle für den Beweis notwendigen Variablen deklariert und Hilfsätze bewiesen. In der Datei `beweis.scm` wird der eigentliche Beweis geführt. Beide Dateien sind ausführlich dokumentiert, so dass eine ausführliche Erklärung dieser Dateien an dieser Stelle unnötig ist.

5 Extraktion eines Programms zur Urbildberechnung aus Minlog

Nachdem der Beweis komplett ausgeführt wurde, kann aus ihm ein Programm extrahiert werden, der das Urbild von Funktionswerten injektiver Funktionen $f : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ berechnet. Dieses Programm kann mit folgendem Befehl extrahiert werden:

```
(define eterm (nt (proof-to-extracted-term (current-proof))))
```

Dabei wird mit `proof-to-extracted-term` das Programm extrahiert und `nt` normalisiert den extrahierten Term. Das extrahierte Programm kann man sich nun mit `pp` anschauen:

```
(pp eterm)
```

Die Ausgabe ist folgende:

```
[n0]
 (Rec nat=>(nat=>nat)=>nat=>nat)n0([f3,n4]0)
 ([n3,((nat=>nat)=>nat=>nat)_4,f5,n6]
   [if (f5(Succ n3)=Succ n3)
     [if (n6=Succ n3) (Succ n3) (((nat=>nat)=>nat=>nat)_4 f5 n6)]
     [if (n6=f5(Succ n3))
      (Succ n3)
      [if (n6=Succ n3)
```

```
(((nat=>nat)=>nat=>nat)_4
([n7][if (f5 n7=Succ n3) (f5(Succ n3)) (f5 n7)])
(f5(Succ n3)))
(((nat=>nat)=>nat=>nat)_4
([n7][if (f5 n7=Succ n3) (f5(Succ n3)) (f5 n7)])
n6)]]])
```

Man kann erkennen, dass das Programm eine rekursiv definierte Funktion ist. Diese Funktion ist vom Typ `nat=>(nat=>nat)=>nat=>nat`. Dementsprechend ordnet das Programm einer natürlichen Zahl $n \in \mathbb{N}$, einer Funktion $f : \mathbb{N} \to \mathbb{N}$ und einer natürlichen Zahl $m \in \mathbb{N}$ eine natürliche Zahl $k \in \mathbb{N}$ zu. Dabei ist k das Urbild des Funktionswert m unter f, wobei f die Menge $\{1, 2, \ldots, n\}$ auf sich selbst injektiv abbildet.

Damit der Porgrammcode lesbarer wird, fügen wir eine neue Variable `ff` vom Typ `(nat=>nat)=>nat=>nat` hinzu:

```
(add-var-name "ff" (py "(nat=>nat)=>nat=>nat"))
(define eterm (nt eterm))
```

Die Ausgabe des Programms über `pp` ist dann:

```
[n0]
 (Rec nat=>(nat=>nat)=>nat=>nat)n0([f3,n4]0)
 ([n3,ff4,f5,n6]
   [if (f5(Succ n3)=Succ n3)
     [if (n6=Succ n3) (Succ n3) (ff4 f5 n6)]
     [if (n6=f5(Succ n3))
      (Succ n3)
      [if (n6=Succ n3)
       (ff4([n7][if (f5 n7=Succ n3) (f5(Succ n3)) (f5 n7)])(f5(Succ n3)))
       (ff4([n7][if (f5 n7=Succ n3) (f5(Succ n3)) (f5 n7)])n6)]]])
```

Im Rekursionsanfang $n = 0$ wird einfach vom Programm 0 zurück gegeben. Dies ist nachvollziehbar: Schließlich ist 0 Urbild für jeden Funktionswert und jeder Funktion $f : \{0\} \to \{0\}$. Der Pseudocode für den Rekursionsschritt über $n + 1$ lautet dann:

if $f(n+1) = n+1$ **then**
  if $m = n+1$ **then**
    return $n+1$
  else
    return `urbild`(n, f, m)
  end if
else
  if $m = f(n+1)$ **then**
    return $n+1$
  else
    if $m = n+1$ **then**
      return `urbild` $\left(n, x \mapsto \left\{ \begin{array}{ll} f(n+1) & ; f(x) = n+1 \\ f(x) & ; f(x) \neq n+1 \end{array} \right\}, f(n+1)\right)$
    else
      return `urbild` $\left(n, x \mapsto \left\{ \begin{array}{ll} f(n+1) & ; f(x) = n+1 \\ f(x) & ; f(x) \neq n+1 \end{array} \right\}, m\right)$
    end if

end if
end if

Anhand des Programms erkennt man deutlich die Analogie zur Vorgehehensweise im Beweis und dem Code des Programms.

Zum Schluss kann das Programm für einige konkrete Beispiele getestet werden. Der Befehl zur Anwendung des Programms auf eine konkrete Funktion lautet `mk-term-in-app-form`. Dieser Befehl erwartet in unserem Beispiel vier Argumente: unser Programm `eterm`, die Zahl $n \in \mathbb{N}$, welche Anzahl der Elemente im Definitions- und Wertebereich angibt, die Funktion $f : \mathbb{N} \to \mathbb{N}$ und den Funktionswert $m \in \mathbb{N}$, zu dem das Urbild ausgrechnet werden soll. Danach wird der gewonnene Term mit `nt` normalisiert und mit Hilfe von `pp` ausgegeben:

```
(pp (nt (mk-term-in-app-form eterm (pt "5") (pt "[n]n") (pt "2"))))
; Ausgabe: 2

(pp (nt (mk-term-in-app-form eterm (pt "5") (pt "[n]n") (pt "0"))))
; Ausgabe: 0

(pp (nt (mk-term-in-app-form eterm (pt "5") (pt "[n]n") (pt "5"))))
; Ausgabe: 5
```

Im obigen Beispiel haben wir die Identitätsfunktion $f : \{0, 1, \ldots, 5\} \to \{0, 1, \ldots, 5\} : n \mapsto n$ mit den Funktionswerten 2, 0 und 5 als Argumente für das extrahierte Programm gewählt. Wir können das obige Beispiel auch für die Funktion $f : \{0, 1, \ldots, 5\} \to \{0, 1, \ldots, 5\} : n \mapsto \begin{cases} n+1 & ; n < 5 \\ 0 & ; n = 0 \end{cases}$ ausführen:

```
(pp (nt (mk-term-in-app-form eterm
        (pt "5")          (pt "[n][if (n<5) (Succ n) 0]") (pt "2"))))
; Ausgabe: 1

(pp (nt (mk-term-in-app-form eterm
        (pt "5") (pt "[n][if (n<5) (Succ n) 0]") (pt "0"))))
; Ausgabe: 5

(pp (nt (mk-term-in-app-form eterm
        (pt "5") (pt "[n][if (n<5) (Succ n) 0]") (pt "5"))))
; Ausgabe: 4
```