HOL

Stefan Richter

22. 2. 2000

Proseminar Perlen der Informatik
Professor Tobias Nipkow, PhD

Betreut von Markus Wenzel

Inhaltsverzeichnis

1 Was ist HOL?

HOL ist die Abkürzung für High Order Logic. Der Begriff beantwortet die Frage eigentlich schon, denn im Kern besteht HOL tatsächlich nur in der Ausweitung der Konzepte der klassischen Logik (Aussagen- und darauf aufbauend Prädikatenlogik etc.) auf Konstrukte höherer Ordnung.

HOL ist eine formale Sprache, die ursprünglich zur Hardwarespezifikation und -verifikation entworfen wurde. Daher eignet sich HOL gut zur Formulierung der mathematischen Konzepte, die für Schlüsse über digitale Geräte benötigt werden. Dabei wurde Wert darauf gelegt, daß sie sich für die Manipulation mit Computern eignet.

Man muß hier unterscheiden zwischen HOL als tatsächlich existierende Implementierungen von semiautomatischen Beweissystemen und der zugrundeliegenden HO-Logik, auf die im Folgenden speziell eingegangen wird.

Geschichtlich gesehen entstand HOL aus Church's Simple Type Theory [1]. Im Wesentlichen sind die Erweiterungen polymorphe Typen und Hilbert's ε-Operator, der das Auswahlaxiom einbringt.

2 Syntax und Semantik

Hier sollen die syntaktischen Konstrukte der Sprache und zugleich ein ihnen entsprechendes semantisches Modell eingeführt werden.

In der HOL-Syntax können im Wesentlichen Typen und Terme dargestellt werden, die auf der semantischen Ebene für Mengen und Elemente von Mengen stehen. Daher beginnen wir mit einigen Annahmen über diese Mengen.

2.1 Grundlagen einer mengentheoretischen Semantik

Das hier vorgestellte Modell[1] basiert auf einer festen Menge von Mengen, die wir das *Universum* $\boldsymbol{U}$ nennen. $\boldsymbol{U}$ habe die folgenden Eigenschaften[2]:

Inhab : Jedes Element von $\boldsymbol{U}$ ist eine nichtleere Menge.

Sub : Wenn X $\in \boldsymbol{U}$ und $\emptyset \neq Y \subseteq X$, dann Y $\in \boldsymbol{U}$.

[1] Es existieren weitere, "non-standard", Modelle von HOL, die wir hier nicht betrachten

[2] Hier und im Folgenden werde durch Definitionen dieser Art stets die kleinste abgeschlossene Menge mit solchen Eigenschaften bezeichnet

Prod : Aus X $\in \boldsymbol{U}$ und Y $\in \boldsymbol{U}$ folgt $X \times Y \in \boldsymbol{U}$. [3]

Pow : Wenn $X \in \boldsymbol{U}$, dann ist auch die Potenzmenge $\mathcal{P}(X) = \{Y : Y \subseteq X\}$ Element von $\boldsymbol{U}$.

Infty : $\boldsymbol{U}$ enthält eine ausgewiesene unendliche Menge **I**.

Choice : Es existiert ein ausgewiesenes Element ch $\in \Pi_{X \in \boldsymbol{U}} X$.

Diese letzte Forderung entspricht dem Auswahlaxiom für das Universum. Zur Problematik dieses Axioms sei auf [2] verwiesen.

Aus diesen Regeln lassen sich weitere Eigenschaften ableiten, die für die HOL-Semantik wichtig sind:

Da sich Funktionen als Teilmengen von kartesischen Produkten darstellen lassen, folgt aus **Sub**, **Prod** und **Pow** :

Fun : Wenn $(X \in \boldsymbol{U} \wedge Y \in \boldsymbol{U})$, dann $(X \to Y \in \boldsymbol{U})$.

Durch Anwendung von **Sub** auf eine beliebige Menge in $\boldsymbol{U}$ (**Infty** garantiert die Existenz einer solchen Menge) erhält man ein- bzw. zweielementige Mengen, von denen jeweils eine bestimmte ausgewählt werde:

Unit : $\boldsymbol{U}$ enthält die ausgewählte einelementige Menge $1 = \{0\}$.

Bool : $\boldsymbol{U}$ enthält die ausgezeichnete zweielementige Menge $2 = \{0, 1\}$.

2.2 Typen

Die Typen in HOL sind Ausdrücke, die Mengen (in $\boldsymbol{U}$) beschreiben. Traditionell notiert man beliebige Typen mit σ, eventuell mit Indizes erweitert.

Prinzipiell gibt es zwei Arten von Typen in der HOL Logik:

1. **Typvariablen** stehen für beliebige Mengen im Universum. Hiermit wird die Polymorphie in HOL eingeführt, die in Church's Simple Type Theory noch Teil der Meta-Sprache ist.

[3] Dabei wird das kartesische Produkt $X \times Y$ dargestellt durch $\{(x, y) = \{\{x\}, \{x, y\}\} \ : \ x \in X \wedge y \in Y\}$.

2. **Typkonstanten** haben die Form $(\sigma_1, \ldots, \sigma_n)op$. Dabei sind $\sigma_1, \ldots, \sigma_n$ die Argumenttypen und *op* ist der *Typoperator* mit der Stelligkeit n. Typoperatoren bezeichnen Operationen zur Konstruktion von Mengen. Der Typ $(\sigma_1, \ldots, \sigma_n)op$ entspricht der Menge, die durch die Anwendung der Operation, die mit *op* bezeichnet wird, auf die Mengen, die durch $\sigma_1, \ldots, \sigma_n$ beschrieben werden, entsteht.

   So ist z.B. $(\sigma)liste$ ein Typoperator der Stelligkeit 1. Er bezeichnet die Operation, alle endlichen Listen von Elementen einer gegebenen Menge zu bilden. Ein weiteres Beispiel ist die direkte Produktbildung (Operator $(\sigma_1, \sigma_2)prod$ mit Stelligkeit 2), hierauf gehen wir später noch ausführlich ein (refprod).

   Typkonstanten der Stelligkeit 0 bezeichnet man auch als **atomare Typen** oder Basistypen. Sie bezeichnen einfache Mengen. Im Allgemeinen werden die Argumentklammern des Typoperators hier in der Schreibung weggelassen und somit Operator und Konstante identifiziert.

   Ein spezieller zweistelliger Typoperator ist $\rightarrow$. Die durch seine Anwendung enstehenden Typen bezeichnet man als **Funktionstypen**. Man schreibt diesen Operator gewöhnlich in rechtsassoziativer Infixnotation.

Abschließend kann die konkrete Syntax für Typen also etwa wie folgt in BNF wiedergegeben werden:

$$\sigma ::= \alpha \mid c \mid (\sigma_1, \ldots, \sigma_n)op \mid \sigma_1 \rightarrow \sigma_2$$

Dabei steht α für Typvariablen und c für atomare Typen.

Typkonstanten, die durch Belegung der Argumente eines Typoperators mit Typen entstehen, nennt man *Instanzen* der Typkonstanten des Operators.

2.3 Terme

HOL-Terme sind Ausdrücke, die Elemente aus Mengen bezeichnen, die von Typen beschrieben werden. Man verwendet die Meta-Variable t, um beliebige Terme zu bezeichnen.

Es gibt vier Arten von Termen in HOL. Die folgende BNF-Grammatik beschreibt sie :

$$t ::= x \mid c \mid \lambda x.t \mid t\ t'$$

Es folgt eine genauere Beschreibung dieser syntaktischen Konstrukte in der Reihenfolge ihres Auftretens:

1. **Variablen** sind Paare x_σ in Indexnotation, wobei x ein Name und σ ein Typ sei. Semantisch steht eine solche Variable für ein beliebiges Element der Menge, die durch σ bezeichnet wird.

2. **Konstanten** c_σ sind genau dann gültige Terme, wenn c für ein Element einer Menge steht, von deren Typ σ eine Instanz ist.

3. **λ-Abstraktionen** $(\lambda x_{\sigma_1}.t_{\sigma_2})_{\sigma_1\to\sigma_2}$ bezeichnen totale Funktionen $v_{\sigma_1} \to t[v/x]_{\sigma_2}$, wenn x eine Variable vom Typ σ_1 und t ein gültiger Term vom Typ σ_2 ist.

4. **Funktionsapplikationen** $(t_{\sigma'\to\sigma}\ t'_{\sigma'})_\sigma$ bezeichnen das Ergebnis der Anwendung der Funktion $t_{\sigma'\to\sigma}$ auf das Argument $t'_{\sigma'}$ (Wenn t' ein korrekter Term des Typs σ').

Ohne Einschränkung können wir annehmen, daß die Namen der Variablen disjunkt zu den Namen der Konstanten sind, und daß verschiedene Variablen stets verschiedene Namen haben (insbesondere, wenn sie verschiedenen Typs sind), weil solche Fälle durch Umbenennung (α-Konversion) stets vermieden werden können.

Funktionsapplikation sei linksassoziativ, so daß $(t\ t_1)\ t_2)$ zu $t\ t_1\ t_2$ abgekürzt werden kann.

Die Notation $\lambda x_1\ x_2.t$ steht für $\lambda x_1.(\lambda x_2.t)$.

Ein Term heißt polymorph, wenn er Typvariablen enthält. Ansonsten heißt er monomorph. Man beachte, daß ein Term polymorph sein kann, selbst wenn sein Typ monomorph ist – etwa in $(f_{\alpha\to b}\ x_\alpha)_b$, wenn b ein atomarer Typ ist.

Abschließend sei hier noch einmal die Term-BNF, diesmal aber mit vollständiger Typisierung, gegeben:

$$t_\sigma ::= x_\sigma \mid c_\sigma \mid (t_{\sigma'\to\sigma}\ t'_{\sigma'})_\sigma$$

$$t_{\alpha\to\beta} ::= (\lambda x_\alpha.t_\beta)_{\alpha\to\beta}$$

2.4 Grundlegende Annahmen

Bis jetzt war die Syntax von Typen und Termen sehr allgemein. Um die üblichen logischen Formeln zu repräsentieren, fordern wir eine spezifische Struktur.

Insbesondere muß jede Rechenstruktur[4] den atomaren Typ *bool* enthalten, der die spezielle zweielementige Menge $2 \in \boldsymbol{U}$ bezeichnen soll, gesehen als Menge von Wahrheitswerten. Logische Formeln werden dann identifiziert mit Termen des Typs *bool*.

Außerdem werden einige logische Konstanten als in allen Signaturen vorhanden angenommen:

$$\Rightarrow_{bool \to bool \to bool}$$

$$=_{\alpha \to \alpha \to bool}$$

$$\varepsilon_{(\alpha \to bool) \to \alpha}$$

Dies sind primitive Konstanten, das heißt ihre Interpretation wird als dem HOL-System bekannt angesehen, sie müssen nicht definiert werden.

Als Standardinterpretation wird folgendes angenommen, wobei hier die Konstantensymbole auch für ihre Interpretation, d.h. ihr Modell, stehen.

- $\Rightarrow$: $2 \to 2 \to 2$ wird interpretiert als die Implikationsfunktion, die $a, b \in 2$ abbildet auf

$$(a \Rightarrow b) = \begin{cases} 0 & \text{falls } a = 1 \text{ und } b = 0 \\ 1 & \text{sonst} \end{cases}$$

- $=_X$: $X \to X \to 2$ ist für jedes $X \in \boldsymbol{U}$ die Gleichheitstestfunktion, die $x, y \in X$ abbildet auf

$$(x =_X y) = \begin{cases} 1 & \text{falls } x = y \\ 0 & \text{sonst} \end{cases}$$

[4]Der Begriff Rechenstruktur stehe hier informell für ein Tupel aus einer Menge von Typen und einer Menge typisierter Konstanten. Er wird in 4 zum Begriff der Theorie erweitert.

- $\varepsilon : (X \to 2) \to X$ ist für jedes $X \in \boldsymbol{U}$ die Auswahlfunktion, die $f \in (X \to 2)$ abbildet auf

$$ch_X(f) = \begin{cases} ch(f^{-1}\{1\}) & \text{falls } f^{-1}\{1\} \neq \emptyset \\ ch(X) & \text{sonst} \end{cases}$$

  mit $f^{-1}\{1\} = \{x \in X : f(x) = 1\}$. Man beachte, daß durch die Eigenschaft **Sub** gilt: $f^{-1}\{1\} \in \boldsymbol{U}$, wenn es nichtleer ist. Die Funktion ch ist durch **Choice** gegeben.

Von nun an wird angenommen, daß alle Rechenstrukturen diese Eigenschaften erfüllen.

Bemerkung : Diese spezielle Wahl primitiver Konstanten ist beliebig und nur historisch bedingt. Die Standardauswahl logischer Konstanten enthält True, False, $\Rightarrow$, $\wedge$, $\vee$, $\neg$, $\forall$, $\exists$, $=$ usw. Diese Menge ist redundant, da sie von vielen Untermengen aus definiert (vgl. 4.2.1) werden kann. In der Praxis wird die ganze Menge logischer Konstanten benötigt, so daß die spezielle Teilmenge, die als primitiv angenommen wird, nicht wichtig ist.

In der Regel werden $=$ und $\Rightarrow$ in Infix geschrieben und ε wird als Binder verwendet wie in $\varepsilon x.t$ statt $\varepsilon(\lambda x.t)$.

3 Ableitungsregeln

Sequenzen (engl. sequents) (Γ, t) bestehen aus einer endlichen Menge von booleschen Termen (das heißt Termen vom Typ *bool*, vgl. 4) Γ, die man die *Annahmen* oder *Hypothesen* nennt, und einer *Konklusion*, dem booleschen Term t.

(Γ, t) hat die Bedeutung der Zusicherung "wenn jeder Term in Γ äquivalent zu True ist, dann auch t".

Auf einer tieferen Ebene, also innerhalb der Annahmen oder der Konklusion, schreibt man auch $\Gamma \vdash t$ für die Aussage "t ist aus Γ ableitbar".

Ein *Theorem* ist eine Sequenz, die entweder ein *Axiom* ist, also ohne Beweis als richtig gefordert wird, oder aus Theoremen durch *Ableitungsregeln* folgt.

Ableitungsregeln sind Vorschriften zur Ableitung neuer Theoreme. Formal sind die Inferenzregeln wie Sequenzen aufgebaut.

Es gibt acht primitive Regeln, alle anderen Regeln sind von ihnen und den Axiomen abgeleitet. In der hier verwendeten natürlichen Ableitungsnotation

stehen die Annahmen über dem Strich (wenn es Hypothesen gibt, sonst sind die Ergebnisse immer ableitbar), die Folgerung darunter.

Assumption introduction

$$\frac{}{t \vdash t}$$

Reflexitivity

$$\frac{}{\vdash t = t}$$

Beta – conversion

$$\frac{}{\vdash (\lambda x.t_1)t_2 = t_1[t_2/x]}$$

- Dabei dürfen in t_1 keine Variablen gebunden werden, die in t_2 frei sind, was aber automatisch durch passende α-Konversion verhindert wird.

Substitution

$$\frac{\Gamma_1 \vdash t_1 = t_1' \quad \cdots \quad \Gamma_n \vdash t_n = t_n' \qquad \Gamma \vdash t[t_1, \ldots, t_n]}{\Gamma_1 \cup \cdots \cup \Gamma_n \cup \Gamma \vdash t[t_1', \ldots, t_n']}$$

- Dabei ist $t[t_1, \ldots, t_n]$ ein Term t, in dem an einigen ausgewählten Stellen Unterterme $t_1, \ldots, t_n$ vorkommen und $t[t_1', \ldots, t_n']$ das Ergebnis der Ersetzung jedes ausgewählten Vorkommens von t_i durch t_i' (für $1 \leq i \leq n$) in t.
- Variablen werden passend umbenannt, um die Bindung freier Variablen in t_i' nach der Substitution zu verhindern.

Abstraction

$$\frac{\Gamma \vdash t_1 = t_2}{\Gamma \vdash (\lambda x.t_1) = (\lambda x.t_2)}$$

- Wenn x nicht frei in Γ

Type instantiation

$$\frac{\Gamma \vdash t}{\Gamma \vdash t[\sigma_1, \ldots, \sigma_n/\alpha_1, \ldots, \alpha_n]}$$

- Wo $t[\sigma_1, \ldots, \sigma_n/\alpha_1, \ldots, \alpha_n]$ das Ergebnis der parallelen Ersetzung der Typvariablen $\alpha_1, \ldots, \alpha_n$ durch Typen $\sigma_1, \ldots, \sigma_n$ in t ist, mit der Einschränkung, daß keine der Typvariablen $\alpha_1, \ldots, \alpha_n$ in Γ vorkommt.

Discharging an assumption

$$\frac{\Gamma \vdash t_2}{\Gamma \backslash \{t_1\} \vdash t_1 \Rightarrow t_2}$$

Modus Ponens

$$\frac{\Gamma_1 \vdash t_1 \Rightarrow t_2 \qquad \Gamma_2 \vdash t_1}{\Gamma_1 \cup \Gamma_2 \vdash t_2}$$

Man kann zeigen, daß die Ableitungsregeln korrekt (engl. sound) bezüglich der bisher aufgestellten Semantik sind. Das bedeutet, wenn die Interpretation der Hypothese einer Regel in einem Modell (v.a. in einem Standardmodell, siehe oben) wahr ist, dann erfüllt das Modell auch die Folgerung.

Der Beweis ist rein technisch und erfordert einige formale Begriffe, auf deren Einführung hier verzichtet wurde, da sie das intuitive Verstehen der HOL-Konzepte eher behindern. Er wird in [4] geführt.

4 Theorien

Das Ergebnis der Arbeit mit dem HOL-System, wenn es ein Ergebnis gibt, ist ein Objekt namens *Theorie*. Dieses Objekt ist ähnlich einer Theorie in der mathematischen Logik. Es besteht ebenfalls aus Mengen von Typen, Konstanten und Axiomen (Von denen ein Spezialfall die Definitionen sind, siehe 4.2.1).

Im Unterschied zur reinen Logik enthält eine HOL-Theorie stets eine explizite Liste von Theoremen, die bereits aus den Axiomen abgeleitet wurden. Logiker unterscheiden nicht zwischen Theoremen, die bereits bewiesen wurden,

und solchen, die beweisbar wären. Daher umfassen Theorien in der Logik im Unterschied zu HOL bereits alle Konsequenzen der Axiome.

Ein damit zusammenhängender Unterschied ist auch, daß logische Theorien statische Objekte darstellen, während sie in HOL potentiell erweiterbar sind. Während einer typischen Interaktion mit einem HOL-System werden bestehende Theorien kombiniert, einige Definitionen gemacht, einige Theoreme bewiesen und dann die neuen Ergebnisse abgespeichert.

Die Aufgabe des HOL-Systems ist die Bereitstellung von Werkzeugen zur Konstruktion wohlgeformter Theorien. Die HOL-Logik ist typisiert: Jede Theorie spezifiziert eine Rechenstruktur von Typen und Konstanten. Diese bestimmen dann die Mengen von Typen und Termen wie in den vorherigen Abschnitten beschrieben.

Alle Theoreme solcher Theorien sind logische Konsequenzen der Axiome der Theorie. Das HOL-System sichert, daß nur wohlgeformte Theorien konstruiert werden können, indem Theoreme nur durch formale Beweise[5] geschaffen werden können.

4.1 Grundlegende HOL-Theorien

Es gibt verschiedene Möglichkeiten, aus den vorgestellten Grundkonstrukten eine komplexere HOL-Theorie aufzubauen. Hier soll ein "Induktionsanfang" nach [4] illustriert werden.

4.1.1 Die Theorie MIN

Die *minimale Theorie* MIN besteht aus den den Typen *bool* und *ind* und den drei Konstanten $\Rightarrow, =$ und ε, die in 4 eingeführt wurden. Sie besitzt keine Axiome.

Die Theorie besitzt ein einheitliches Standardmodell, das in den vorangehenden Abschnitten bereits erläutert wurde. Der Typ *ind* steht dabei für die unendliche Menge $\mathbf{I}$ der Individuen aus 3.

4.1.2 Die Theorie LOG

LOG *erweitert* MIN. Das heißt, sie besitzt sämtliche Typen, Konstanten, Axiome und Theoreme (auch wenn die Menge der letzten beiden in diesem

[5] formal meint hier: korrekt im System

Fall leer ist) der *Elterntheorie* MIN. Im Allgemeinen kann eine neue Theorie durch Erweiterung einer oder (der Vereinigung) mehrerer bestehender Theorien entstehen. Ein Spezialfall wäre die "leere Erweiterung", also die bloße Vereinigung mehrerer Theorien bzw. im trivialen Fall die Kopie einer Theorie.

Die Theorie führt lediglich neue Konstanten und definitorische Axiome für diese Konstanten ein. Dies sind die üblichen logischen Konstanten, sowie Abkürzungsdefinitionen für die in 4.2.2 erklärte Typdefinition ein:

$$\vdash \mathsf{T}_{bool} = ((\lambda x_{bool}.x) = (\lambda x_{bool}.x))$$

$$\vdash \forall_{(\alpha\to bool)\to bool} = \lambda P_{\alpha\to bool}.P = (\lambda x.\mathsf{T})$$

$$\vdash \exists_{(\alpha\to bool)\to bool} = \lambda P_{\alpha\to bool}.P(\varepsilon P)$$

$$\vdash \mathsf{F}_{bool} = \forall b_{bool}.b$$

$$\vdash \neg_{bool\to bool} = \lambda b.b \Rightarrow \mathsf{F}$$

$$\vdash \wedge_{bool\to bool\to bool} = \lambda b_1 b_2.\forall b.(b_1 \Rightarrow (b_2 \Rightarrow b)) \Rightarrow b$$

$$\vdash \vee_{bool\to bool\to bool} = \lambda b_1 b_2.\forall b.(b_1 \Rightarrow b) \Rightarrow ((b_2 \Rightarrow b) \Rightarrow b)$$

$$\vdash \mathsf{One\_One}_{(\alpha\to\beta)\to bool} = \lambda f_{\alpha\to\beta}.\forall x_1 x_2.(fx_1 = fx_2) \Rightarrow (x_1 = x_2)$$

$$\vdash \mathsf{Onto}_{(\alpha\to\beta)\to bool} = \lambda f_{\alpha\to\beta}.\forall y.\exists x.y = fx$$

$$\vdash \mathsf{Type\_Definition}_{(\alpha\to bool)\to(\beta\to\alpha)\to bool} = \lambda P_{\alpha\to bool}\ rep_{\beta\to\alpha}.\mathsf{One\_One}\ rep \wedge (\forall x.Px = (\exists y.x = rep\ y))$$

$$\vdash \mathsf{Inv} = \lambda f_{\alpha\to\beta}.\lambda y.\varepsilon x.y = fx$$

Wie hier schon zu sehen war, werden in Verbindung mit diesen Konstanten einige spezielle Schreibweisen benutzt, genauer die Infixschreibweise für $\wedge$ und $\vee$, sowie Bindereigenschaften und Rechtsassoziativität für die beiden Quantoren.

Man kann zeigen, daß die hier eingeführten Konstanten durch die Standardinterpretation der MIN-Theorie ebenfalls ein eindeutig festgelegtes Modell besitzen[6], daß dem für diese Konstanten üblichen entspricht.

[6] vgl. 4.2.1

4.1.3 Die Theorie INIT

Die Theorie INIT fügt nur die folgenden fünf Axiome der Theorie LOG hinzu:

BOOL_CASES_AX	$\vdash \forall b.(b = \mathsf{T}) \vee (b = \mathsf{F})$
IMP_ANTYSIM_AX	$\vdash \forall b_1 b_2.(b_1 \Rightarrow b_2) \Rightarrow (b_2 \Rightarrow b_1) \Rightarrow (b_1 = b_2)$
ETA_AX	$\vdash \forall f_{\alpha\to\beta}.(\lambda x.fx) = f$
SELECT_AX	$\vdash \forall P_{\alpha\to bool} x.Px \Rightarrow P(\varepsilon P)$
INFINITY_AX	$\vdash \exists f_{ind\to ind}.\mathsf{One\_One}\ f \wedge \neg(\mathsf{Onto} f)$

Das Standardmodell von LOG genügt diesen fünf Axiomen und ist damit auch ein Modell von INIT.

INIT ist die *initiale Theorie* der HOL-Logik. Eine Theorie, die INIT erweitert, heißt auch *Standardtheorie*.

INIT ist konsistent in dem Sinne, daß von ihr nicht $\vdash \mathsf{F}$ abgeleitet werden kann. Dafür ist hinreichend, daß ein Standardmodell hierfür existiert, denn nach der Konsistenz der Ableitungsregeln wird jedes Theorem, das von den Axiomen der Theorie hergeleitet werden kann, von diesem Modell erfüllt, während $\vdash \mathsf{F}$ von keinem Modell erfüllt wird.

Bemerkung : Wir folgen hier aus Gründen der Übersichtlichkeit der Darstellung von [4], in der alle Axiome in der speziellen Theorie INIT eingeführt werden. Für einige der Axiome (v.a. SELECT_AX) wäre aber auch eine Definition in der MIN-Theorie möglich, da die Allquantordarstellung durch eine Darstellung mit freien Variablen (Wie in $\vdash Px \Rightarrow P(\varepsilon\ P)$) ersetzt werden kann.

4.2 Erweiterung von Theorien

Um das Wohlverhalten[7] von weitergehenden Theorien zu garantieren, sollten möglichst keine weiteren Axiome postuliert werden. Damit bleiben zur Erweiterung von Theorien im Wesentlichen die **Konstantendefinition** und die **Typdefinition**.

Es existieren theoretisch auch noch die Konstantenspezifikation und die Typspezifikation, aber diese sind nicht von so großer praktischer Bedeutung, bzw. teilweise gar nicht implementiert, und lassen sich als Sonderfall der obigen Verfahren darstellen.

[7] Im Wesentlichen heißt das hier Modellerhaltung.

4.2.1 Erweiterung durch Konstantendefinition

Eine Konstantendefinition ist eine Formel der Form $c_\sigma = t_\sigma$, so daß:

1. c ein neuer Name ist, d.h. kein Name einer bereits bekannten Konstante
2. t ein geschlossener gültiger Term ist
3. alle Typvariablen in t auch in σ vorkommen

Sind diese Bedingungen erfüllt, so ist die *definitorische Erweiterung* T' einer Theorie T durch eine solche Konstantendefinition die gleiche Theorie T, erweitert um die Konstante c_σ und das Axiom $c_\sigma = t_\sigma$.

Man kann zeigen, wenn T ein Standardmodell besitzt, dann auch T'.

4.2.2 Erweiterung durch Typdefinition

Typkonstanten (bzw. Typoperatoren) werden in HOL stets als "Teiltypen" von bereits existierenden Typen (sog. *repräsentierenden Typen* σ) definiert. Dazu geben wir eine charakteristische Funktion an (das *Teilmengenprädikat*), die die Teilmenge definiert, die wir aus der Menge der Urtyps verwenden wollen.

In HOL müssen Typen stets nichtleere Mengen darstellen. Daher ist es nur konsistent, einen Typ als isomorph zu einer durch ein Prädikat P spezifizierten Teilmenge zu definieren, wenn nachgewiesen wurde, daß es mindestens ein Element der ursprünglichen Menge gibt, für das P wahr ist, d.h. $\vdash \exists x_\sigma : Px$.

Schließlich können wir eine *Repräsentationsfunktion* einführen, die die Elemente des neu definierten Typs auf die Teilmenge des Ursprungstyps abbildet. Man muß für diese Funktion lediglich Bijektivität fordern, was wir mittels der in LOG definierten Konstante Type Definition axiomatisieren können.

Diese Funktion wird nicht genau angegeben, so daß hier die Abstraktion von der Implementierung erhalten bleibt. Die *Abstraktionsfunktion* als die Umkehrfunktion erhalten wir einfach durch Anwendung des Inv-Funktionals aus der Theorie bool.

5 Beispiel: Die Theorie PROD

Hier soll das direkte Produkt formalisiert werden. Dazu wird der binäre Typoperator $(\sigma_1, \sigma_2)prod$ eingeführt. Werte des Typs $(\sigma_1, \sigma_2)prod$ repräsentieren Paare, deren erste Komponente vom Typ σ_1 ist, und deren zweite Komponente den Typ σ_2 hat.

Wir werden einen Operator **Pair** vom Typ $\alpha \to \beta \to (\alpha, \beta)prod$ definieren, so daß wenn $t_1 : \sigma_1$ und $t_2 : \sigma_2$, dann ist (t_1, t_2) (Was lediglich eine andere Notation für **Pair** darstellt) ein Paar mit der ersten Komponente t_1 und der zweiten Komponente t_2.

Ein Paar $(t_1 : \sigma_1, t_2 : \sigma_2)$ wird repräsentiert durch die Funktion $\lambda xy.(x = t_1) \wedge (y = t_2)$ mit dem Typ $\sigma_1 \to \sigma_2 \to bool$.

Also definieren wir $(\alpha, \beta)prod$ als isomorph zu der Teilmenge von $\alpha \to \beta \to bool$, die aus den Funktionen f mit der Eigenschaft $\exists ab.f = \lambda xy.(x = a) \wedge (y = b)$ besteht.

Dazu definieren wir die folgenden Konstanten:

$\vdash$ Mk_Pair $= \lambda ab.\ \lambda xy.\ (x = a) \wedge (y = b)$

$\vdash$ Is_Pair $= \lambda f.\ \exists ab.\ f =$ Mk_Pair $a\ b$.

Dann können wir den Typ $(\alpha, \beta)prod$ wie folgt definieren:

1. Der repräsentierende Typ ist $\alpha \to \beta \to bool$.
2. Das Teilmengenprädikat ist Is_Pair.
3. $\vdash \exists f.$ Is_Pair$f$ weil $\vdash$ Is_Pair(Mk_Pair $a\ b$) für irgendein a, b.
4. Die Repräsentationsfunktion sei $\mathsf{Rep\_Pair}_{(\alpha,\beta)prod \to (\alpha \to \beta \to bool)}$.

Diese Typdefinition liefert das Axiom

$\vdash$ Type_Definition Rep_Pair

Wenn wir weiter definieren

$\vdash$ Abs_Pair $=$ Inv Rep_Pair

können wir die üblichen Operationen auf Paaren angeben durch

$\vdash$ Pair $= \lambda\ x_\alpha\ y_\beta.$ Abs_Pair (Mk_Pair $x\ y$)

$\vdash$ Fst $= \lambda p_{(\alpha,\beta)prod}.\ \varepsilon x.\ \exists y.\ p = (x, y)$

$\vdash$ Snd $= \lambda p_{(\alpha,\beta)prod}.\ \varepsilon y.\ \exists x.\ p = (x, y).$

Aus diesen Definitionen kann man ableiten

$\vdash\ \forall\ x\ y.$ Fst$(x, y) = x$

$\vdash\ \forall\ x\ y.$ Snd$(x, y) = y$

$\vdash\ \forall\ p_{(\alpha,\beta)prod}.\ p =$ (Fst p, Snd p)

Damit sind erhält man die charakteristischen Eigenschaften des zweistelligen direkten Produkts.

6 Schlußbemerkung

Wir haben gesehen, daß aus sehr wenigen Grundkonzepten und einer einfachen Syntax recht komplexe Gebilde entstehen können. In der Tat lassen sich mit einem so einfachen System wie HOL die meisten Konzepte der klassischen Mathematik darstellen.

HOL ist ein recht altes System und hat sich seit langer Zeit in der wissenschaftlichen Praxis bewährt. Es existieren sehr viele Implementierungen und Derivate.

Heute stellt HOL bzw. ähnlich aufgebaute Systeme die Grundlage vieler wesentlich komplexerer Anwendungen und Beweissysteme wie etwa Isabelle dar.

Literatur

[1] A. Church. *A Formulation of the Simple Theory of Types.* Journal of Symbolic Logic 5, 1940.

[2] Eric Schechter. *A home page for the AXIOM OF CHOICE.* `http://math.vanderbilt.edu/~schectex/ccc/choice.html`

[3] Mike Gordon. HOL — *A Machine Oriented Formulation of Higher Order Logic.* Cambridge 1985 (Revised version)

[4] Mike Gordon u.a. *Description — The HOL Logic* `http://lal.cs.byu.edu/lal/holdoc/description.html`

Diese Arbeit basiert hauptsätzlich auf [3] und [4].