Einführung in den λ-Kalkül und Semantik

Prof. Dr. Klaus Keimel

Wintersemester 2003/04

Technische Universität Darmstadt
Fachbereich Mathematik

Inhaltsverzeichnis

Literatur

(1) Historische Arbeiten

- M. SCHÖNFINKEL Über die Bausteine der mathematischen Logik. *Math. Ann.* **93** (1924), 305–316.
- H. B. CURRY Grundlagen der kombinatorischen Logik. *Amer. J. Math.* **52** (1930), 509–536 und 789–834.
- A. CHURCH A set of postulates for the foundation of logic. *Ann. Math. (2)* **33** (1932/33), 346–366.
- S. C. KLEENE and J. B. ROSSER The inconsistency of certain formal logics. *Ann. Math (2)* **36** (1935), 630–636.
- S. C. KLEENE λ-definability and recursiveness. *Duke Math. J.* **2** (1936), 340–353.
- A. M. TURING Computability and λ-definability. *J. Symbolic Logic* **2** (1937), 153–163.

(2) Beweistheorie

- G. GENTZEN Untersuchungen über das logische Schließen. *Mathematische Zeitschrift* **39** (1934), 176–210 und 405–431. (Nachdruck erschienen bei: Wissenschaftliche Buchgesellschaft Darmstadt, 1969).
- D. PRAWITZ Natural Deduction. A Proof-Theoretical Study. Almqvist & Wiksells, Uppsala (1965).

(3) Lambda-Kalkül

- R.M. AMADIO AND P.-L. CURIEN Domains and Lambda-Calculi. Cambridge University Press (1998).
- H. P. BARENDREGT The Lambda Calculus: Its Syntax and Semantics. North-Holland, revised edition (1984).
- H. P. BARENDREGT Lambda Calculi with Types. In: *Handbook of Logic in Computer Science*, vol 2 (1992), 117–309.
- J.-Y. GIRARD, Y. LAFONT, and P. TAYLOR Proofs and Types. Cambridge University Press (1989).
- J. R. HINDLEY and J. P. SELDIN Introduction to Combinators and λ-Calculus. Cambridge University Press (1986).

- J. L. Krivine Lambda-Calcul, Types et Modèles. Masson (1990). Englische Übersetzung: Lambda-Calculus, Types and Models. Masson and Ellis Horwood (1993).
- J. C. Mitchell Type Systems for Programming Languages. In: J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, North Holland (1990), 365–458.
- Ch. Hankin Lambda Calculi. A Guide for Computer Scientists. Clarendon Press (1994).

(4) Lambda Kalkül und Programmiersprachen

- A. J. T. Davie An Introduction to Functional Programming Systems Using Haskell. Cambridge University Press (1992).
- J. Fasel, P. Hudak, and J. Peterson A gentle Introduction to Haskell – Version 98. siehe: http://www.haskell.org/tutorial/
- M. J. C. Gordon Programming Language Theory and its Implementation. Prentice Hall (1988).
- P. Hudak The Haskell School of Expression. Learning Functional Programming through Multimedia. Cambridge University Press, 2000.
- G. Michaelson An Introduction to Functional Programming through Lambda Calculus. Addison-Wesley, 1989.
- L. C. Paulson ML for the Working Programmer. Cambridge University Press (1991).
- J. C. Reynolds Theories of Programming Languages. Cambridge University Press (1998).
- S. Thompson Type Theory and Functional Programming. Addison-Wesley, 1991.

(5) Semantik

- K. Alber and W. Struckmann Einführung in die Semantik von Programmiersprachen. BI Wissenschaftsverlag, 1988.
- E. Best Theorie sequentieller und paralleler Programmierung. Vieweg Verlag (1995).
- P.-L. Curien Categorical Combinators, Sequential Algorithms and Functional Programming. Pitman, 1986.
- C. A. Gunter Semantics of Programming Languages. Structures and Techniques. MIT Press, 1992.
- M. Hennessy The Semantics of Programming Languages. Wiley, 1990.
- J. C. Mitchell Foundations for Programming Languages. MIT Press (1996).
- P. Mosses Denotational Semantics. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 577–632. North Holland, 1990.

- H. R. Nielson and F. Nielson Semantics with Applications: A Formal Introduction for Computer Science. Wiley, 1991.
- G. D. Plotkin LCF Considered as a Programming Language. *Theoretical Computer Science*, 5:223–255, 1977.
- G. D. Plotkin Post-Graduate Lecture Notes in Advanced Domain Theory (incorporating the "Pisa Notes"). Dept. of Computer Science, Univ. of Edinburgh, 1981.
- D. A. Schmidt Denotational Semantics. Allyn and Bacon, 1986.
- D. S. Scott A type-theoretical alternative to ISWIM, CUCH, OWHY. Theoretical Computer Science **121** (1993), 411–440. *Hier handelt es sich um den Druck eines Manuskripts aus dem Jahre 1969, das die Entwicklung nachhaltig beeinflußt hat, jedoch vor 1993 nur als Kopie einer Kopie einer Kopie ... erhältlich war.*
- J. E. Stoy Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory. The MIT Press, 1977.
- R. D. Tennent Semantics of Programming Languages. Prentice Hall, 1991.
- Ch. Walther Semantik und Programmverifikation. B.G. Teubner Verlag, 2001.
- G. Winskel The Formal Semantics of Programming Languages. An Introduction. MIT Press (1993).

(6) Semantische Bereiche

- S. Abramsky and A. Jung Domain Theory. In: *Handbook of Logic in Computer Science* vol. 3, Oxford University Press, p. 1–169 (1994).
- G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, D. S. Scott A Compendium of Continuous Lattices. Springer Verlag (1980).
- G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, D. S. Scott Continuous Lattices and Domains. Cambridge University Press (2003).

KAPITEL I

Der typenfreie λ-Kalkül

1. Die Syntax des typenfreien λ-Kalküls

Der λ-Kalkül ist ein formales System. Man betrachtet gewisse formale Ausdrücke (λ-Terme) und führt Regeln ein, nach denen man diese Ausdrücke umformen darf. Dabei läßt man sich davon leiten, daß dieser Kalkül den Umgang mit Funktionen modellieren soll, wie in der Einleitung angedeutet. In diesem Abschnitt führen wir die Syntax des reinen typenfreien λ-Kalküls ein, in den nächsten beiden Abschnitten dann die operationale Semantik, d.h. die Umformungsregeln.

Wir geben eine abzählbar unendliche Menge $V = \{v_1, v_2, v_3, \dots\}$ von Variablen vor, die wir mit

$$x, y, z, \dots, f, g, h, \dots$$

bezeichnen. Neben Variablen verwenden wir Klammern $(\_)$ und das Zeichen $\lambda\_.$.

1.1 λ-Terme. Als λ-*Term* bezeichnen wir jede Zeichenreihe M, die sich in endlich vielen Schritten nach folgenden Regeln bilden läßt:

(1) Jede Variable x ist ein Term.

(2) Sind M und N zwei λ-Terme, dann ist auch (MN) ein λ-Term. (*Applikation*)

(3) Ist M ein Term und x eine Variable, dann ist $(\lambda x.M)$ ein λ-Term.(*Abstraktion*)

$$\frac{}{x}\,(1) \qquad\qquad \frac{M \qquad N}{(MN)}\,(2) \qquad\qquad \frac{M}{(\lambda x.M)}\,(3)$$

Abbildung 1. Die Termbildungsregeln

Mit $M, N, P, \dots$ bezeichnen wir λ-Terme. Die Menge aller λ-Terme bezeichnen wir mit Λ . Schließlich bedeute $M \equiv N$, daß M und N syntaktisch gleich sind.

Beispiel.

- $\mathsf{I} :\equiv (\lambda x.x)$
- $\mathsf{K} :\equiv (\lambda x.(\lambda y.x))$
- $\mathsf{COMP} :\equiv (\lambda z.(\lambda y.(\lambda x.(z(yx)))))$
- $\mathsf{S} :\equiv (\lambda z.(\lambda y.(\lambda x.((zx)(yx)))))$

wobei x, y, z verschiedene Variablen bezeichnen.

Den Aufbau eines Terms gemäß der Regeln (1), (2), (3) kann man durch einen *Termbaum* darstellen. Als Beispiel betrachten wir in Abbildung 2 den Termbaum von S.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z}\,(1)\quad \dfrac{}{x}\,(1)}{(zx)}\,(2)\qquad \dfrac{\dfrac{}{y}\,(1)\quad \dfrac{}{x}\,(1)}{(yx)}\,(2)}{((zx)(yx))}\,(2)}{(\lambda x.((zx)(yx)))}\,(3)}{(\lambda y.(\lambda x.((zx)(yx))))}\,(3)}{(\lambda z.(\lambda y.(\lambda x.((zx)(yx)))))}\,(3)$$

ABBILDUNG 2. Der Termbaum von $\mathsf{S} \equiv (\lambda z.(\lambda y.(\lambda x.((zx)(yx)))))$

Der Baum eines Terms ist eindeutig; denn durch die Beklammerung ist sichergestellt, daß für jeden Term M gilt: Entweder ist

(1) $M \equiv x$ für eine Variable x oder
(2) $M \equiv (MP)$ für eindeutig bestimmte λ-Terme N und P oder
(3) $M \equiv (\lambda x.N)$ für einen eindeutig bestimmten λ-Term N.

Insofern lassen sich Eigenschaften von λ-Termen rekursiv über den Termaufbau definieren und Sätze über λ-Terme induktiv über den Termaufbau beweisen (*strukturelle Induktion*), vgl. 1. Übungsblatt.

Klammerkonventionen. Um die Schreibweise von λ-Termen zu vereinfachen, führen wir folgende Abkürzungen ein, aus denen sich die vollständige Beklammerung stets rekonstruieren läßt:

- $M_1 M_2 \dots M_n :\equiv (\dots((M_1 M_2) M_3) \dots M_n)$
  Man sagt auch, die Applikation sei *linksassoziativ*.
- $\lambda x_1 x_2 \dots x_m.M :\equiv \lambda x_1.\lambda x_2.\dots\lambda x_n.M :\equiv (\lambda x_1.(\lambda x_2.\dots(\lambda x_m.M)\dots))$
  In dieser Konvention versteckt man das Phänomen, daß man modulo *curry-uncurry* (siehe **??**) eine Funktion in m Variablen $x_1, \dots, x_m$ als iterierte Funktionen einer Variablen lesen kann.
- $\lambda x.MN :\equiv \lambda x.(MN)$, d.h. Applikation bindet stärker als Abstraktion.

Beispiel. Mit diesen Konventionen kann man die obigen Beispielterme einfacher schreiben:

$$\mathsf{I} \equiv \lambda x.x \qquad \mathsf{K} \equiv \lambda xy.x \qquad \mathsf{COMP} \equiv \lambda zyx.z(yx) \qquad \mathsf{S} \equiv \lambda zyx.zx(yx)$$

Zur Übung überlege man sich eine Interpretation von K und S, so wie wir in der Einleitung I und COMP als Identität bzw. Komposition von Funktionen sehen konnten. Man kann jedoch im λ-Kalkül Terme bilden, die man nicht im klassischen Sinn durch Funktionen interpretieren kann. Es ist ja erlaubt, jeden Term auf sich selbst anzuwenden und Terme der Form MM zu bilden wie z.B.

$$\omega_2 :\equiv (\lambda x.xx)(\lambda x.xx)$$
$$\mathsf{Y} :\equiv \lambda f.VV \quad \text{mit } V :\equiv \lambda x.f(xx)$$

Solche Terme wird man schwerlich wie in der Einleitung interpretieren können, denn man kann eine Funktion f nicht auf sich selbst anwenden. Ein Ausdruck wie sin(sin) ist sinnlos. Im typenfreien λ-Kalkül kann man jeden Term sowohl als Funktion als auch als Argument betrachten. Dies wird in getypten λ-Kalkülen weitgehend ausgeschlossen, was der Intuition besser entspricht. Es ist a priori weder klar, daß der ungetypte λ-Kalkül eine vernünftige Semantik hat, noch daß er widerspruchsfrei ist.

1.2 Teilterme. Die *Teilterme* eines λ-Term M sind rekursiv wie folgt definiert:

(1) Der einzige Teilterm von $M \equiv x$ ist x selbst.

(2) Die Teilterme von $M \equiv NP$ sind die Teilterme von N, die Teilterme von P sowie M selbst.

(3) Die Teilterme von $M \equiv \lambda x.N$ sind die Teilterme von N sowie M selbst.

Insbesondere ist jeder Term ein Teilterm von sich selbst, alle anderen Teilterme nennt man echte Teilterme. Die Teilterme von M sind genau die λ-Terme, die im Termbaum von M auftreten. Man beachte, daß ein Teilterm N mehrfach in einem Term vorkommen kann.

1.3 Freie und gebundene Variablen. Bei einer Abstraktion sagen wir, die Variable x sei in $\lambda x.M$, und M sei der Bindungsbereich. Dies kann man durchaus in Analogie zur Bindung von Variablen durch Quantoren sehen. Die nicht gebundenen Variablen in einem Term nennt man *frei*. Genauer: Mit $\mathrm{FV}(M)$ wird die Menge der *freien Variablen* des λ-Terms M bezeichnet, und diese ist wie folgt rekursiv definiert:

(1) $\mathrm{FV}(x) = \{x\}$

(2) $\mathrm{FV}(NP) = \mathrm{FV}(N) \cup \mathrm{FV}(P)$

(3) $\mathrm{FV}(\lambda x.N) = \mathrm{FV}(N) \setminus \{x\}$

Entsprechend wird die Menge $\mathrm{BV}(M)$ der *gebundenen Variablen* von M definiert:

(1) $\mathrm{BV}(x) = \varnothing$

(2) $\mathrm{BV}(NP) = \mathrm{BV}(N) \cup \mathrm{BV}(P)$

(3) $\mathrm{BV}(\lambda x.N) = \mathrm{BV}(N) \cup \{x\}$

Es gilt $\mathrm{Var}(M) = \mathrm{FV}(M) \cup \mathrm{BV}(M)$ bezeichne die Menge aller in einemm Term auftretenden Variablen. Eine Variable kann in einem Term an einer Stelle frei und an einer anderen Stelle gebunden vorkommen:

Beispiel. In dem Term $M \equiv \lambda x.xz(\lambda z.xz)$ kommt x nur gebunden und z sowohl frei als auch gebunden vor.

Kombinatoren. λ-Terme ohne freie Variablen nennt man *geschlossene Terme* oder auch *Kombinatoren*. Mit Λ_0 bezeichnen wir die Menge aller geschlossenen Λ-Terme. Die oben definierten Terme I, K und S sind Kombinatoren.

1.4 α-Äquivalenz. Zur Motivation des Folgenden betrachten wir die reelle Funktion $x \mapsto xe^{tx}$ mit dem Parameter t. Wenn wir die Variable x in u umbenennen, also die Funktion $u \mapsto ue^{tu}$ betrachten, so handelt es sich um dieselbe Funktion, auch wenn man die Funktionsvariable umbenennt. In unserer intuitiven λ-Schreibweise bedeutet dies: $\lambda x.xe^{tx} = \lambda u.ue^{tu}$. Dementsprechend wollen wir erlauben, in unserem formalen Kalkül gebundene Variable umzubenennen. Dabei müssen wir aber eine gewisse Vorsicht walten lassen:

- Wir dürfen in unserem obigen Beispiel die Funktionsvariable x nicht in t umbenennen, denn dann würde der Parameter t mit der Funktionsvariablen identifiziert, und es ergäbe sich die Funktion $t \mapsto te^{t^2}$. Bei der Umbenennung der Funktionsvariablen x dürfen wir also keine in dem Ausdruck frei vorkommende Variable verwenden.
- Wir betrachten den Wert des Integrals $F(t) := \int_0^1 xe^{tx}dx$ in Abhängigkeit von t, d.h. die Funktion $t \mapsto \int_0^1 xe^{tx}dx$. In unserer intuitiven λ-Schreibweise lautet diese Funktion $F := \lambda t.\big(\int_0^1 \lambda x.xe^{tx}\big)$. Hier ist auch die Variable t gebunden. Wir dürfen t in v umbenennen, ohne die Funktion zu verändern, d.h. es gilt auch $F := \lambda v.\big(\int_0^1 \lambda x.xe^{vx}\big)$, aber wir dürfen nicht t in x umbenennen, denn es ergäbe sich $\lambda x.\big(\int_0^1 \lambda x.xe^{x^2}\big) = \lambda x.\frac{e-1}{2}$, also eine konstante Funktion. Also sollte man auch vermeiden, bei der Umbennenung für die Funktionvariable eine Variable zu verwenden, die in dem Bindungsbereich gebunden vorkommt.

Umbenennung gebundener Variabler. Unter der Nebenbedingung, daß u in dem λ-Term N weder frei noch gebunden vorkommt, erlauben wir die Umformung

$$\lambda x.N \to_\alpha \lambda u.\big(N[x := u]\big),$$

wobei $N[x := u]$ der Term ist, den man aus N erhält, wenn jedes in N frei auftretende x durch ein u ersetzt wird. Allgemeiner schreiben wir

$$M \to_\alpha M'$$

und sagen, M' sei aus M durch Umbenennung einer gebundenen Variablen hervorgangen, wenn solch eine Umbenennung in irgendeinem Teilterm der Form $\lambda x.N$ von M unter Beachtung der Nebenbedingung vorgenommen wird. Schließlich schreiben wir

$$M \equiv_\alpha M',$$

wenn M und M' durch (endlich viele) solcher Umbenennungen gebundener Variabler ineinander übergeführt werden können. Man sagt auch, M und M' seien *α-äquivalent*. Man kann nachweisen, daß $\equiv_\alpha$ in der Tat eine Äquivalenzrelation auf der Menge aller λ-Terme ist. (Obwohl dieser Sachverhalt sehr einleuchtend erscheint, ist ein formaler Beweis hierfür recht langwierig. Man kann ihn in dem Buch von Krivine nachlesen.)

Beispiel.

- $\mathsf{COMP} \equiv \lambda zyx.z(yx) \equiv_\alpha \lambda fgt.f(gt) \equiv_\alpha \lambda xyz.x(yz)$

- $\mathsf{S} \equiv \lambda zyx.zx(yx) \equiv_\alpha \lambda xyz.xz(yz)$

Da $\equiv_\alpha$ eine Äquivalenzrelation auf den λ-Termen beschreibt, werden wir im folgenden α-äquivalente Terme identifizieren: Wir schreiben einfach $M \equiv M'$ wenn M bis auf Umbenennung gebundener Variabler syntaktisch übereinstimmen, d.h. wnn $M \equiv_\alpha M'$.

Variablenkonvention. Betrachtet man in irgendeinem Zusammenhang endlich viele Terme $M_1, \ldots, M_k$, so kann man die in diesen Termen gebundenen Variablen so umbenennen, daß sie von den freien Variablen dieser Terme verschieden sind. In der Regel werden wir dies stillschweigend voraussetzen.

1.5 Substitution. Mit $M[x := N]$ wird der Term bezeichnet, der entsteht, wenn man alle freien Vorkommen von x in M durch den Term N ersetzt. Da sich ähnliche Probleme wie in 1.4 ergeben können, müssen gegebenenfalls gebundene Variable in M so umbenannt werden, daß sie in N nicht mehr frei vorkommen.

Genauer definiert man $M[x := N]$ rekursiv über den Aufbau des Termes M so:

(1) $M \equiv y$:

$$y[x := N] \equiv \begin{cases} N & \text{für } y \equiv x \\ y & \text{für } y \not\equiv x \end{cases}$$

(2) $M \equiv PQ$:

$$(PQ)[x := N] \equiv \big(P[x := N]\big)\big(Q[x := N]\big)$$

(3) $M \equiv \lambda y.P$:

$$(\lambda y.P)[x := N] \equiv \begin{cases} \lambda y.P & \text{für } y \equiv x \\ \lambda y.\big(P[x := N]\big) & \text{für } y \not\equiv x \end{cases}$$

Nebenbedingung: Kommt die Variable y in N frei vor, so muß sie vorher so umbenannt werden, daß sie in N nicht frei auftritt.

1.6 Sukzessive Substitution. Es seien M, N und L drei λ-Terme. Unter den Voraussetzungen, daß x und y zwei verschiedene Variablen sind und daß x in L nicht frei vorkommt, gilt:

$$M\big[x := N\big]\big[y := L\big] \equiv M\big[y := L\big]\big[x := N[y := L]\big]$$

Die Gültigkeit des Lemmas kann man sich anschaulich überlegen. Warum kann man nicht auf die Voraussetzung verzichten, daß x in L nicht frei vorkommt? Als Übung führe man einen induktiven Beweis über den Aufbau von M. □

Bemerkung. Das folgende Beispiel zeigt, daß einfache Vertauschung nicht zum Ziel führt. Es gilt

$$x[x := xy][y := xz] \equiv x(xz)$$

Einfache Vertauschung der Substitution liefert das „falsche“ Ergebnis

$$x[y := xz][x := xy] \equiv xy$$

Verwendet man das Lemma, so ergibt sich wieder das ursprüngliche Ergebnis

$$x\big[y := xz\big]\big[x := xy[y := xz]\big] \equiv x(xz)$$

1.7 Statisches und dynamisches Binden. Es soll nun begründet werden, wozu die Substitution gebraucht wird und warum wir bei der Substitution $M[x := N]$ vermeiden wollen, daß freie Variablen von N nach der Substitution in M gebunden werden.

Im nächsten Abschnitt wollen wir das Auswerten einer Funktion für ein Argument in unserem Kalkül formalisieren. Wollen wir die Funktion

$$f = \lambda x.\ ax^2 + bx + c$$

an der Stelle 2 auswerten, so ergibt sich $f(2) = a \cdot 2^2 + b \cdot 2 + c$. Will man f an der Stelle $x = d^2 + 1$ auswerten, so ersetzt man die Variable x in dem Ausdruck $ax^2 + bx + c$ durch $d^2 + 1$:

$$(\lambda x.\ ax^2 + bx + c)(d^2 + 1) = (ax^2 + bx + c)[x := d^2 + 1] = a(d^2 + 1) + b(d^2 + 1) + c\ .$$

Analog wollen wir in unseren Kalkül die Auswertungsregel

$$(\lambda x.M)N \to_\beta M[x := N]$$

hineinnehmen, d.h. will man eine Funktion $\lambda x.M$ für ein Argument N auswerten, so setze man in M den Ausdruck N für die Funktionsvariable x ein. Hierfür brauchen wir also die Substitution.

Die Nebenbedingung, die wir bei der Substitution beachtet haben wollen, ergibt sich aus dem Umgang mit Funktionen: Betrachten wir wie schon früher die Funktion

$$F = \lambda t.\ \int_0^1 \lambda x.\ xe^{tx}$$

Wollen wir F an der Stelle $t = 2$ berechnen, so ersetzen wir die Funktionsvariable t durch 2:

$$F(2) = \int_0^1 \lambda x.\ xe^{2x}\ .$$

Will man F an der Stelle $t = x + 1$ auswerten und setzt man naiv einfach $x + 1$ anstelle von t ein, so ergibt sich

$$F(x+1) = \int_0^1 \lambda x.\ xe^{(x+1)x}\ .$$

Dies macht keinen Sinn, denn dann wäre $F(x + 1)$ eine Konstante, die nicht von x abhängt. Es sollte aber $F(x + 1)$ bis auf eine Umbenennung der Variablen dasselbe wie $F(t + 1)$ sein. Die Ursache für den Fehler ist, daß bei dieser naiven Substitution die Funktionsvariable x in der „inneren" Funktion $\lambda x.\ xe^{tx}$ mit dem freien Parameter x in dem Argument $t = x + 1$ identifiziert wird. Vor dem Einsetzen sollte man also die Funktionsvariable der inneren Funktion umbenennen. Dies haben wir in unsere Definition der Substitution eingearbeitet.

In den ursprünglichen Versionen von LISP wurde die Substitution naiv ohne Beachtung der Nebenbedingung implementiert. Die Lambda-Ausdrücke dort darf man daher nur mit Einschränkungen als Funktionen verstehen. Man spricht dort auch von „dynamic binding". In den neueren LISP-Versionen wird die Substitution wie im Lambda-Kalkül behandelt. Man spricht dann von „static binding".

2. β-Reduktion

2.1 β-Kontraktion. Als *β-Redex* eines Terms M wird jeder Teilterm der Gestalt $(\lambda x.N)P$ bezeichnet. Der Term $N[x := P]$ heißt *β-Kontraktum* von $(\lambda x.N)P$. Wir schreiben auch

$$(\lambda x.N)P \to_\beta N[x := P] \; .$$

Allgemeiner schreiben wir $M \to_\beta M'$, falls M' aus M durch Ersetzen eines β-Redex in M durch sein Kontraktum hervorgeht. Bei einer β-Kontraktion wird also genau ein Redex in M kontrahiert.

2.2 β-Reduktion. Falls M' durch eine endliche Folge von β-Kontraktionen aus M hervorgeht (d.h. falls es Terme $M_0, M_1, \ldots, M_k$ mit $M \equiv M_0 \to_\beta M_1 \to_\beta \ldots \to_\beta M_k \equiv M'$ gibt), schreiben wir

$$M \to_\beta^* M'$$

und sagen, daß M' durch *β-Reduktion* aus M entsteht.

2.3 Kontext- und Substitutionsverträglichkeit der β-Reduktion. Die β-Reduktion ist reflexiv, transitiv, mit Kontext und der Substitution verträglich, d.h. es gelten die folgenden Implikationen:

$$(\rho) \qquad M \to_\beta^* M$$

$$(\tau) \qquad M \to_\beta^* N,\ N \to_\beta^* P \implies M \to_\beta^* P$$

$$(\text{Kontext}) \qquad N \to_\beta^* N' \implies NM \to_\beta^* N'M,\ MN \to_\beta^* MN',\ \lambda x.N \to_\beta^* \lambda x.N'$$

$$(\text{Subst}_1) \qquad M \to_\beta^* M' \implies M[x := N] \to_\beta^* M'[x := N]$$

$$(\text{Subst}_2) \qquad N \to_\beta^* N' \implies M[x := N] \to_\beta^* M[x := N']$$

Beweis. Reflexivität und Transitivität folgen direkt aus der Definition der $\beta-$Reduktion ebenso wie die Kontextverträglichkeit. Für (Subst_1) genügt es, den Fall $M \to_\beta M'$ mit $M \equiv (\lambda y.P)Q$ und $M' \equiv P[y := Q]$ zu betrachten. Durch Umbenennung von y können wir erreichen, daß y von x verschieden ist und y nicht frei in N vorkommt. Dann gilt

$$\begin{aligned} M[x := N] &\equiv \big((\lambda y.P)Q\big)[x := N] \\ &\equiv \big((\lambda y.P)[x := N]\big)Q[x := N] \\ &\equiv \big(\lambda y.P[x := N]\big)Q[x := N] \\ &\to_\beta P[x := N][y := Q[x := N]] \\ &\equiv P[y := Q][x := N] \quad \text{nach 1.6} \\ &\equiv M'[x := N] \end{aligned}$$

Die letzte Implikation wird durch Induktion über den Aufbau von N bewiesen. □

2.4 β-Normalformen. Ein Term N ist in *β-Normalform*, falls er keine β-Kontraktion erlaubt, d.h. falls er keinen β-Redex enthält. Wenn ein Term M sich durch eine β-Reduktion zu einer β-Normalform N umformen läßt, dann sagt man, N sei eine β-Normalform von M, oder auch, M sei *normalisierbar*.

Beispiele. Folgende Terme liegen in β-Normalform vor:

$$\mathsf{I} \equiv \lambda x.x \qquad \mathsf{K} \equiv \lambda xy.x \qquad \mathsf{S} \equiv \lambda xyz.xz(yz) \qquad \delta_2 \equiv \lambda x.xx \qquad L \equiv \lambda x.xxy$$

Die folgenden Terme besitzen keine β-Normalform:

$$\omega_2 \equiv \delta_2\delta_2 \equiv (\lambda x.xx)\delta_2 \to_\beta \delta_2\delta_2 \equiv \omega_2$$
$$LL \equiv (\lambda x.xxy)(\lambda x.xxy) \to_\beta (\lambda x.xxy)(\lambda x.xxy)y \equiv LLy \to_\beta LLyy \to_\beta \dots$$

Der Term $M :\equiv (\lambda x.z)(LL)$ hat die β-Normalform z, erlaubt aber trotzdem unendlich viele β-Reduktionen:

$$M \equiv (\lambda x.z)(LL) \to_\beta (\lambda x.z)(LLy) \to_\beta (\lambda x.z)(LLyy) \to_\beta \dots$$

Die Existenz einer β-Normalform sichert also nicht, daß *jede* Reduktionskette zur Normalform führt. Führt man für einen gegebenen Term M alle möglichen β-Reduktionen aus, so erhält man einen gerichteten Graphen, den *Reduktionsgraphen* von M.

Die β-Reduktion ist die wichtigste Operation im λ-Kalkül. Es erhabt sich die Frage, ob die $\beta-$Normalform eines Terms – sofern eine solche existiert – eindeutig bestimmt ist. Die positive Antwort auf diese Frage beruht auf dem folgenden Satz:

2.5 Satz von Church-Rosser. Die β-Reduktion ist konfluent, d.h. falls es von einem Term M zwei Reduktionsketten zu verschiedenen Termen N und N' gibt, dann existieren ein Term P und zwei Reduktionsketten mit $N \to_\beta^* P$ und $N' \to_\beta^* P$:

$$\begin{tikzcd} & M \arrow[dl, "*"', "\beta"] \arrow[dr, "*", "\beta"'] & \\ N \arrow[dr, dotted, "*", "\beta"'] & & N' \arrow[dl, dotted, "*"', "\beta"] \\ & P & \end{tikzcd}$$

Der Beweis dieses Satzes wird in Abschnitt 3 geführt. □

2.6 Folgerung. Wenn ein Term M eine β-Normalform N besitzt, dann ist diese bis auf α-Äquivalenz eindeutig bestimmt.

Beweis. Seien N und N' zwei β-Normalformen von M, d.h. es gibt zwei Reduktionsketten $M \to_\beta^* N$ und $M \to_\beta^* N'$. Nach 2.5 gibt es dann einen Term P mit $N \to_\beta^* P$ und $N' \to_\beta^* P$. Da jedoch N und N' als β-Normalformen keine weiteren β-Reduktionen zulassen, muß $N \equiv P \equiv N'$ gelten. □

2.7 β-Äquivalenz. Zwei Terme M und M' heißen *β-äquivalent* ($M \simeq_\beta M'$), wenn es einen Term N gibt, zu dem beide reduzieren:

$$\begin{tikzcd} M \arrow[dr, "*", "\beta"'] & & M' \arrow[dl, "*"', "\beta"] \\ & N & \end{tikzcd}$$

Die β-Äquivalenz $\simeq_\beta$ ist in der Tat eine Äquivalenzrelation, die außerdem kontext- und substitutionsverträglich ist; d.h. es gelten:

(ρ) $$M \simeq_\beta M$$

(σ) $$M \simeq_\beta N \implies N \simeq_\beta M$$

(τ) $$M \simeq_\beta N,\ N \simeq_\beta P \implies M \simeq_\beta P$$

(Kontext) $$N \simeq_\beta N' \implies NM \simeq_\beta N'M,\ MN \simeq_\beta MN',\ \lambda x.N \simeq_\beta \lambda x.N'$$

(Subst$_1$) $$M \simeq_\beta M' \implies M[x := N] \simeq_\beta M'[x := N]$$

(Subst$_2$) $$N \simeq_\beta N' \implies M[x := N] \simeq_\beta M[x := N']$$

Beweis. Reflexitivität und Symmetrie der Relation $\simeq_\beta$ sind unmittelbar klar. Zur Transitivität: Seien $M \simeq_\beta N$ und $N \simeq_\beta P$, d.h. es gibt Terme N_1 und N_2 mit

$$\begin{tikzcd} M \arrow[dr, "*", "\beta"'] & & N \arrow[dl, "*"', "\beta"] \arrow[dr, "*", "\beta"'] & & P \arrow[dl, "*"', "\beta"] \\ & N_1 & & N_2 & \end{tikzcd}$$

Da sowohl N_1 als auch N_2 aus N hervorgingen, muß es nach dem Satz von Church-Rosser einen Term N_3 geben, so daß

$$\begin{tikzcd} N_1 \arrow[dr, "*", "\beta"'] & & N_2 \arrow[dl, "*"', "\beta"] \\ & N_3 & \end{tikzcd}$$

Damit gilt auch $M \to_\beta^* N_3$ und $P \to_\beta^* N_3$, also $M \simeq_\beta P$, was die Transitivität von $\simeq_\beta$ beweist. Die Substitutions- und Kontextverträglichkeit der β-Reduktion 2.3 überträgt sich auf die β-Äquivalenz. □

Bemerkung. Die obigen Regeln zusammen mit

(α) $$\lambda x.M \simeq_\beta \lambda y.M[x := y] \text{ für } y \notin \mathrm{Var}(M)$$

(β) $$(\lambda x.M)N \simeq_\beta M[x := N]$$

ergeben ein vollständiges Axiomensystem für die β-Äquivalenz, d.h. Zwei Terme sind genau dann β-äquivalent, wenn sich dies aus den obigen Regeln herleiten läßt.

2.8 Satz. Sind M und N verschiedene β-Normalformen, dann gilt $M \not\simeq_\beta N$.

Beweis. Wären M und N β-äquivalent, dann wären sie zu einem gemeinsamen Term reduzierbar. Da M und N jedoch β-Normalformen sind, können sie nicht mehr reduziert werden. □

2.9 Satz. Die β-Äquivalenzklasse eines Terms M enthält höchstens eine Normalform N, und wenn dies der Fall ist, denn kann man jeden Term M' in der β-Äquivalenzklasse von M zu dieser Normalform N reduzieren.

Beweis. Falls zu einem Term M eine β-Normalform existiert, dann ist diese nach 2.6 eindeutig bestimmt. Da nach 2.8 zwei verschiedene β-Normalformen nicht β-äquivalent sein können, beinhaltet jede Äquivalenzklasse höchstens eine Normalform. □

2.10 Folgerung. Das Axiomensystem für $\simeq_\beta$ ist konsistent, d.h. es gibt Terme M und N, die nicht β-äquivalent sind.

Beweis. Zum Beispiel sind $\lambda x.x$ und $\lambda xy.y$ zwei inäquivalente β-Normalformen. $\square$

3. Der Beweis des Satzes von Church-Rosser

Der folgende Beweis des Satzes von Church-Rosser stammt von Tait und Martin-Löf. Andere Beweise findet man bei Barendregt (S. 279–283) und Hindley-Seldin (S. 313–320). Zur Erinnerung:

3.1 Satz von Church-Rosser. Die β-Reduktion ist konfluent, d.h. falls es von einem Term M zwei Reduktionsketten zu verschiedenen Termen N und N' gibt, dann existieren ein Term P und zwei Reduktionsketten $N \to_\beta^* P$ und $N' \to_\beta^* P$:

$$\begin{tikzcd}
& M \arrow[dl, "*", "\beta"'] \arrow[dr, "*"', "\beta"] & \\
N \arrow[dr, dotted, "*", "\beta"'] & & N' \arrow[dl, dotted, "*"', "\beta"] \\
& P &
\end{tikzcd}$$

Bevor wir in den Beweis einsteigen, betrachten wir ein Beispiel, das zeigt, daß die β-Kontraktion nicht konfluent ist, d.h. es gibt einen Term M mit zwei β-Kontraktionen $M \to_\beta P$ und $M \to_\beta Q$ derart, daß es keinen Term N mit β-Kontraktionen $P \to_\beta N$ und $Q \to_\beta N$ gibt.

Beispiel. Wir betrachten den Term $M \equiv \big(\lambda x.xx\big)\big((\lambda x.x)y\big)$. Dieser hat den folgenden Reduktionsgraphen:

$$\begin{tikzcd}
& \big(\lambda x.xx\big)\big((\lambda x.x)y\big) \arrow[dl, "\beta"] \arrow[dr, "\beta"] & \\
\big((\lambda x.x)y\big)\big((\lambda x.x)y\big) \arrow[d, dotted, "\beta"'] & & (\lambda x.xx)y \arrow[ddl, dotted, "\beta"] \\
y\big((\lambda x.x)y\big) \arrow[dr, dotted, "\beta"'] & & \\
& yy &
\end{tikzcd}$$

Auf der linken Seite benötigt man also zwei Kontraktionen, um das Diagramm konfluent zu machen.

Es ist nicht schwer, das folgende zu zeigen: Gegeben zwei β-Kontraktionen $M \to_\beta P$ und $M \to_\beta Q$, dann gibt es zwei β-Reduktionen, die das Diagramm konfluent machen:

$$\begin{tikzcd}[column sep=small] & M \arrow[dl, "\beta"'] \arrow[dr, "\beta"] & \\ P \arrow[dr, dotted, "*"', "\beta"] & & Q \arrow[dl, dotted, "*", "\beta"'] \\ & N & \end{tikzcd}$$

Diese Aussage reicht allerdings nicht aus, den Satz von Church-Rosser herzuleiten, was das Schema aus Abbildung 3 illustrieren soll. Wir wollen daher erlauben, statt einer

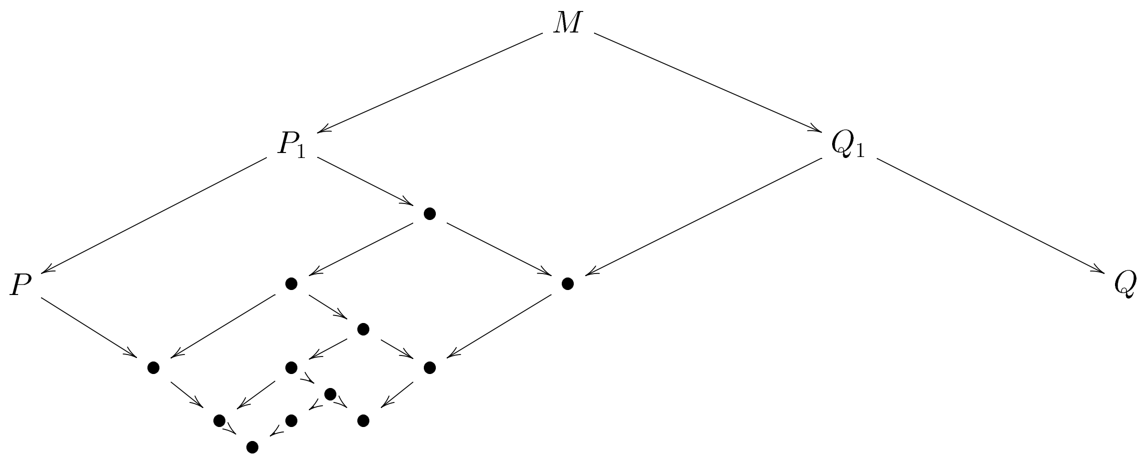


Abbildung 3

Kontraktion mehrere Kontraktionen parallel auszuführen.

3.2 Parallele Reduktion sichtbarer Redexe. Wir definieren $M \rightrightarrows N$ (und wir sagen dann M *reduziert parallel zu* N), wenn sich dies aus den folgenden Regeln in endlich vielen Schritten herleiten:

(1) $$\frac{}{M \rightrightarrows M}$$

(2) $$\frac{M \rightrightarrows M'}{\lambda x.M \rightrightarrows \lambda x.M'}$$

(3) $$\frac{M \rightrightarrows M' \qquad N \rightrightarrows N'}{MN \rightrightarrows M'N'}$$

(4) $$\frac{M \rightrightarrows M' \qquad N \rightrightarrows N'}{(\lambda x.M)N \rightrightarrows M'[x := N']}$$

Diese Regeln kann man auch so lesen:

(1) $M \rightrightarrows M$ gilt für alle λ-Terme M.

(2),(3) $M \rightrightarrows M', N \rightrightarrows N' \implies MN \rightrightarrows M'N', \lambda x.M \rightrightarrows \lambda x.M'$.

(4) $M \rightrightarrows M', N \rightrightarrows N' \implies (\lambda x.M)N \rightrightarrows M'[x := N']$.

Wegen (1) ist parallele Reduktion reflexiv. Die Regeln (2) und (3) besagen, daß parallele Reduktion kontextverträglich ist. Nimmt man noch (4) hinzu, sieht man, daß $\rightrightarrows$ die kleinste binäre Relation auf der Menge der λ-Terme ist, die reflexiv, kontextverträglich und unter β-Kontraktion (von Redexen die in dem gegebenen Term schon vorhanden sind) abgeschlossen ist.

3.3 Hilfssatz.

(1) $M \to_\beta M' \implies M \rightrightarrows M'$

(2) $M \rightrightarrows M' \implies M \to_\beta^* M'$

Jede β-Kontraktion ist also eine spezielle parallele Reduktion, und jede parallele Reduktion läßt sich auch durch eine endliche Kette von β-Kontraktionen realisieren.

Beweis. (1) Aus (1) und (4) bekommen wir für jeden Redex

$$\dfrac{\dfrac{}{M \rightrightarrows M}\,(1) \qquad \dfrac{}{N \rightrightarrows N}\,(1)}{(\lambda x.M)N \rightrightarrows M[x := N]}\,(4)$$

Da nach (2) und (3) die parallele Reduktion auch kontextverträglich ist, folgt die Behauptung.

(2) Die β-Reduktion erfüllt die Bedingungen (1), (2) und (3). Außerdem erfüllt sie (4):

$$(\lambda x.M)N \to_\beta M[x := N] \to_\beta^* M'[x := N']$$

wegen der Substitutionsverträglichkeit von $\to_\beta^*$, sofern $M \to_\beta^* M'$ und $N \to_\beta^* N'$. Da $\rightrightarrows$ die kleinste Relation auf der Menge der λ-Terme ist, die (1)–(4) erfüllt, folgt die Behauptung. □

3.4 Substitutionslemma für parallele Reduktion.

$$M \rightrightarrows M',\ N \rightrightarrows N' \implies M[x := N] \rightrightarrows M'[x := N']$$

Beweis. Wir beweisen das Lemma zuerst im Spezialfall $M \equiv M'$, d.h. wir zeigen für beliebige λ-Terme M:

$$N \rightrightarrows N' \implies M[x := N] \rightrightarrows M[x := N'] .$$

Wir führen den Beweis durch Induktion über den Aufbau des Terms M:
Induktionsanfang: M ist eine Variable. Wir unterscheiden zwei Fälle:

$M \equiv x$**:** $x[x := N] \equiv N \rightrightarrows N' \equiv x[x := N']$

$M \equiv y \not\equiv x$**:** $y[x := N] \equiv y \rightrightarrows y \equiv y[x := N']$

Nun betrachten wir einen zusammengesetzten Term M und nehmen als Induktionsvoraussetzung an, die Behauptung sei für alle Teilterme von M schon bewiesen. Wir haben die beiden Fälle zu untescheiden, ob M durch eine Applikation oder ein Abstraktion entstanden ist:

$M \equiv PQ$**:** Nach Induktionsvoraussetzung haben wir:

$$P[x := N] \rightrightarrows P[x := N'] \text{ und } Q[x := N] \rightrightarrows Q[x := N'] .$$

Mit (3) folgt

$$M[x := N] \equiv \big(P[x := N]\big)\big(Q[x := N]\big) \rightrightarrows \big(P[x := N']\big)\big(Q[x := N']\big) \equiv M[x := N'] .$$

$M \equiv \lambda y.P$**:** Nach der Induktionsvoraussetzung haben wir $P[x := N] \rightrightarrows P[x := N']$. Im Falle $y \not\equiv x$ folgt mit (2) $M[x := N] \equiv (\lambda y.P)[x := N] \equiv \lambda y.(P[x := N]) \rightrightarrows \lambda y.(P[x := N']) \equiv (\lambda y.P)[x := N'] \equiv M[x := N']$. Im Falle $y \equiv x$ ist alles einfach: $M[x := N] \equiv (\lambda x.P)[x := N] \equiv \lambda x.P \equiv (\lambda x.P)[x := N'] \equiv M[x := N']$.

Nun beweisen wir die Behauptung allgemein durch durch Induktion über die Länge der Herleitung von $M \rightrightarrows M'$ nach den Regeln (1)–(4):

(1) Induktionsanfang: $M' \equiv M$. Dieser Fall wurde gerade im vorhergehenden Abschnitt behandelt.

Für den Induktionsschluß haben wir drei Fälle zu betrachten, je nachdem, welche der Regeln als letzte angewendet wurde, um $M \rightrightarrows M'$ herzuleiten:

(2) $M \rightrightarrows M'$ entstand nach der Regel

$$\frac{P \rightrightarrows P'}{M \equiv \lambda y.P \rightrightarrows \lambda y.P' \equiv M'}$$

Sei oBdA $y \not\equiv x$, was man ggf. durch Umbenennung der in M gebundenen Variablen y erreichen kann. Da die Herleitung von $P \rightrightarrows P'$ kürzer ist als die von $M \rightrightarrows M'$, haben wir als Induktionsvoraussetzung $P[x := N] \rightrightarrows P'[x := N']$. Daraus folgt wiederum unter Benutzung von (2): $M[x := N] \equiv (\lambda y.P)[x := N] \equiv \lambda y.P[x := N] \rightrightarrows \lambda y.P'[x := N'] \equiv (\lambda y.P')[x := N'] \equiv M'[x := N']$.

(3) $M \rightrightarrows M'$ entstand nach der Regel

$$\frac{P \rightrightarrows P' \qquad Q \rightrightarrows Q'}{M \equiv PQ \rightrightarrows P'Q' \equiv M'}$$

Nach Induktionsvoraussetzung gelten $P[x := N] \rightrightarrows P'[x := N']$ und $Q[x := N] \rightrightarrows Q'[x := N']$. Dann folgt durch Anwenden der Regel $M[x := N] \equiv \big(P[x := N]\big)\big(Q[x := N]\big) \rightrightarrows \big(P'[x := N']\big)\big(Q'[x := N']\big) \equiv M'[x := N']$

(4) $M \rightrightarrows M'$ entstand nach der Regel

$$\frac{P \rightrightarrows P' \qquad Q \rightrightarrows Q'}{M \equiv (\lambda y.P)Q \rightrightarrows P'[y := Q'] \equiv M'}$$

Dabei benutzen wir folgenden Hilfssatz: Wenn $N \rightrightarrows N'$, dann $\mathrm{FV}(M) \supseteq \mathrm{FV}(M')$. (Der einfache Beweis wird induktiv über die Herleitung von $N \rightrightarrows N'$ geführt.)

Im Fall $x \not\equiv y$ und $y \notin \mathrm{FV}(N)$ gelten nach Induktionsvoraussetzung $P[x := N] \rightrightarrows P'[x := N']$ und $Q[x := N] \rightrightarrows Q'[x := N']$. Nach (4) folgt

$$\begin{aligned} M[x := N] &\equiv \big((\lambda y.P)Q\big)[x := N] && \text{da } y \notin \mathrm{FV}(N) \\ &\equiv \big(\lambda y.P[x := N]\big)\big(Q[x := N]\big) \\ &\rightrightarrows P'\big[x := N'\big]\big[y := Q'[x := N']\big] \\ &\equiv P'[y := Q'][x := N'] && \text{nach 1.6 und } y \notin \mathrm{FV}(N) \supseteq \mathrm{FV}(N') \\ &\equiv M'[x := N'] \end{aligned}$$

Falls $y \in \mathrm{FV}(N)$ muß man vorab die in $M \equiv (\lambda y.P)Q$ gebundene Variable y in eine neue Variable z umbenennen. Dann kommt diese nicht mehr frei in P vor und wir verfahren wie vorher. Der Fall $x \equiv y$ geht ähnlich wie der vorhergehende Fall und ist sogar einfacher. $\square$

Die folgende Feststellung folgt direkt aus der Definiton der parallelen Reduktion:

3.5 Lemma.

(1) Wenn $x \rightrightarrows M'$, dann $M' \equiv x$.

(2) Wenn $\lambda x.P \rightrightarrows M'$, dann existiert P' mit $M' \equiv \lambda x.P'$ und $P \rightrightarrows P'$.

(3) Wenn $PQ \rightrightarrows M'$, dann gilt entweder

- Es existieren P', Q' mit $M' \equiv P'Q'$ und $P \rightrightarrows P'$, $Q \rightrightarrows Q'$.

oder

- $P \equiv \lambda x.P_1$ und es existieren P_1', Q' mit $M' \equiv P_1'[x := Q']$ und $P_1 \rightrightarrows P_1'$, $Q \rightrightarrows Q'$.

3.6 Satz. Die parallele Reduktion ist konfluent, d.h. gegeben parallele Reduktionen $M \rightrightarrows P$ und $M \rightrightarrows Q$, dann kann man parallele Reduktionen $P \rightrightarrows N$ und $Q \rightrightarrows N$ zu einem gemeinsamen Term N finden.

$$\begin{tikzcd} & M \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\ P \arrow[dr, Rightarrow] & & Q \arrow[dl, Rightarrow] \\ & N & \end{tikzcd}$$

Beweis. Wir führen den Beweis durch Induktion über den Termaufbau von M:

$M \equiv x$**:** Dann $P \equiv x \equiv Q$ nach 3.5.1, und man kann $N \equiv x$ wählen.

$M \equiv \lambda x.M_1$**:** Nach 3.5.2 gibt es Terme P_1 und Q_1, so daß $P \equiv \lambda x.P_1$ und $Q \equiv \lambda x.Q_1$, sowie $M_1 \rightrightarrows P_1$ und $M_1 \rightrightarrows Q_1$. Nach Induktionsvoraussetzung gibt es N_1 mit $P_1 \rightrightarrows N_1$ und $Q_1 \rightrightarrows N_1$. Unter Anwendung von (2) schließen wir $P \equiv \lambda x.P_1 \rightrightarrows \lambda x.N_1 \equiv N$ und $Q \equiv \lambda x.Q_1 \rightrightarrows \lambda x.N_1 \equiv N$.

$M \equiv M_1M_2$**:** Nach 3.5.3 können für P und Q je zwei Fälle eintreten. Wir müssen also vier Fälle unterscheiden:

(1) Es gibt P_1 und P_2 mit $P \equiv P_1P_2$ und $M_1 \rightrightarrows P_1$, $M_2 \rightrightarrows P_2$, und es gibt Q_1 und Q_2 mit $Q \equiv Q_1Q_2$ und $M_1 \rightrightarrows Q_1$, $M_2 \rightrightarrows Q_2$. Nach Induktionsvoraussetzung gibt es N_1 und N_2 mit $P_1 \rightrightarrows N_1$, $P_2 \rightrightarrows N_2$, $Q_1 \rightrightarrows N_1$ und $Q_2 \rightrightarrows N_2$. Also gilt nach (3) auch $P \equiv P_1P_2 \rightrightarrows N_1N_2 \equiv N$ und $Q \equiv Q_1Q_2 \rightrightarrows N_1N_2 \equiv N$.

(2) Seien $P \equiv P_1P_2$ mit $M_1 \rightrightarrows P_1$, $M_2 \rightrightarrows P_2$, $M_1 \equiv \lambda x.M_1'$, $Q \equiv Q_1'[x := Q_1]$ mit $M_1' \rightrightarrows Q_1'$, $M_2 \rightrightarrows Q_2$. Wegen $M_1 \equiv \lambda x.M_1' \rightrightarrows P_1$ gilt $P_1 \equiv \lambda x.P_1'$ mit $M_1' \rightrightarrows P_1'$ nach 3.5.2. Nach Induktionsvoraussetzung gibt es N_1' und N_2 mit $P_1' \rightrightarrows N_1'$, $P_2 \rightrightarrows N_2$, $Q_1' \rightrightarrows N_1'$ und $Q_2 \rightrightarrows N_2$. Mit (4) folgt

$P \equiv (\lambda x.P_1')P_2 \rightrightarrows N_1'[x := N_2] \equiv N$. Nach 3.4 folgt $Q \equiv Q_1'[x := Q_2] \rightrightarrows N_1'[x := N_2] \equiv N$.

(3) ähnlich zu 2.

(4) Übung. □

Beweis des Satzes von Church-Rosser. Gegeben seien zwei β-Reduktionen $M \to_\beta^* P$ und $M \to_\beta^* Q$. Diese sind Ketten von n bzw. m β-Kontraktionen, siehe Abbildung 4. Nach 3.3.1 ist jede β-Kontraktion auch eine parallele Reduktion. Nach Satz 3.6 kann man in $n \cdot m$ Schritten das ganze Diagramm durch parallele Reduktionen konfluent machen. Da nach 3.3.2 jede parallele Reduktion auch eine β-Reduktion ist, hat man insgesamt β-Reduktionen $P \to_\beta^* N$ und $Q \to_\beta^* N$. □

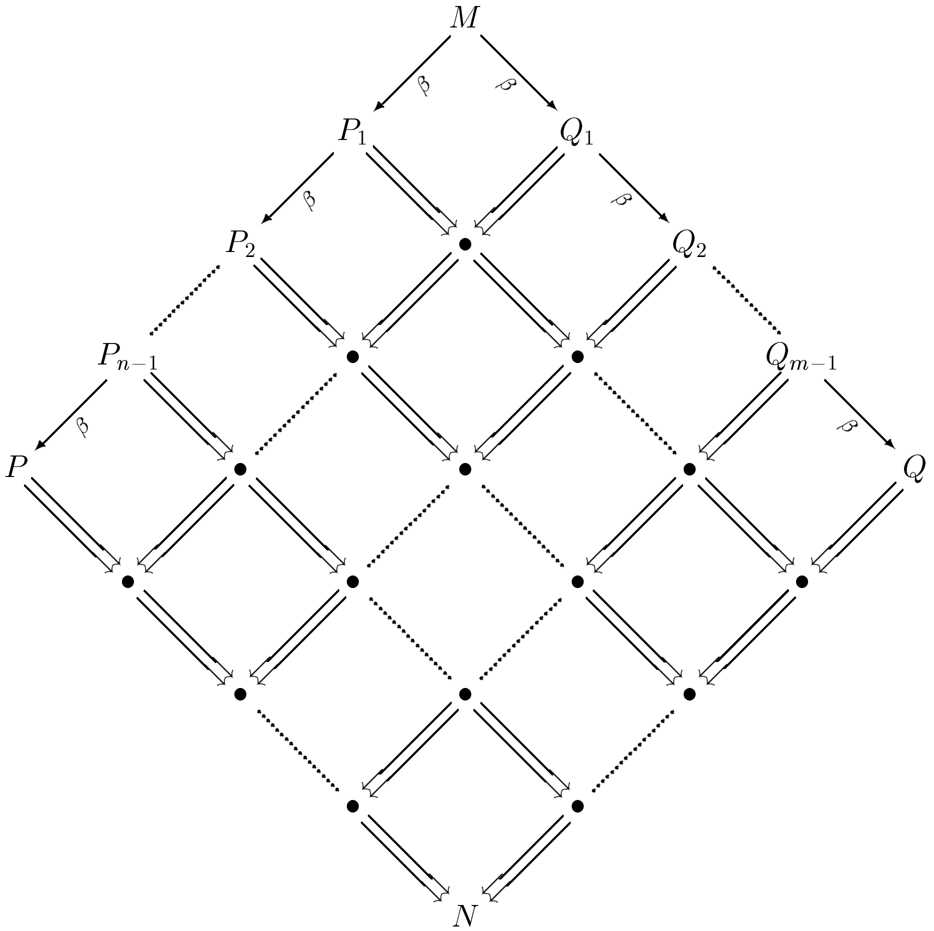


Abbildung 4

4. η-Reduktion und $\beta\eta$-Reduktion

Manchmal (aber nicht immer) will man im λ-Kalkül neben der β-Kontraktion eine weitere Kontraktionsregel verwenden, welche die Eigenschaft der *Extensionalität* von

Funktionen formalisiert. Darunter versteht man, daß zwei Funktionen genau dann gleich sind, wenn sie an jeder Stelle ihres Definitionsbereichs denselben Wert annehmen, d.h.

$$f = g \iff \big(\forall x.\ f(x) = g(x)\big)\,.$$

Dies kann man auch anders formulieren:

$$f = \big(x \mapsto f(x)\big) \quad \text{bzw.} \quad f = \lambda\!\!\lambda x.\ f(x)$$

Dies wird im λ-Kalkül durch eine zusätzliche Kontraktionsregel nachgespielt:

4.1 η-Kontraktion. Jeden Term der Form $R \equiv \lambda x.Px$, welcher der Nebenbedingung $x \notin \mathrm{FV}(P)$ genügt, nennt man einen *η-Redex* und P sein *η-Kontraktum.* Wir schreiben $\lambda x.Px \to_\eta P$.

Allgemeiner: Ist M ein λ-Term, der einen η-Redex R als Teilterm enthält, und ist N der Term, den man aus M durch η-Kontraktion von R erhält, so sagt man, N sei ein η-Kontraktum von M und man schreibt $M \to_\eta N$.

Wie schon bei der β-Reduktion steht $M \to_\eta^* N$ für eine endliche Folge sukzessiver η-Kontraktionen.

Übung. Warum fordert man, daß bei der η-Kontraktion die Variable x nicht frei in P vorkommen darf?

Beispiel. $\mathsf{E} \equiv \lambda x.(\lambda y.xy) \to_\eta \lambda x.x \equiv \mathsf{I}$, falls $x \not\equiv y$.

Der Term E ist in β-Normalform, läßt also keine β-Kontraktion zu, d.h. $\mathsf{E} \not\simeq_\beta \mathsf{I}$. Es gilt aber $\mathsf{E}\,xy \to_\beta^* xy$, d.h. $\mathsf{E}\,xy \simeq_\beta xy$, d.h. bis auf β-Äquivalenz wirkt E bei der sukzessiven Eingabe zweier λ-Terme P, Q ebenso wie I auf die Eingabe von (PQ).

4.2 η-Normalform. Jeden Term, der keine η-Kontraktion zuläßt, nennt man eine *η-Normalform.*

Bemerkung. Jeder λ-Term M, der nicht in η-Normalform ist, läßt sich durch eine endliche Folge von η-Kontraktionen in eine η-Normalform überführen.

Beweis. Da jede η-Kontraktion den gegebenen Term verkürzt, gelangt man nach endlich vielen η-Kontraktionen sicher zu einer η-Normalform. $\square$

4.3 $\beta\eta$-Reduktion. Eine endliche Folge von β- und η-Kontraktionen nennen wir eine *$\beta\eta$-Reduktion.* Wir schreiben $M \to_{\beta\eta}^* N$, falls es λ-Terme $M \equiv M_0, M_1, \ldots, M_k \equiv N$ gibt, so daß $M_{i-1} \to_\beta^* M_i$ oder $M_{i-1} \to_\eta^* M_i$ für $i = 1, \ldots, k$.

4.4 Satz. Die $\beta\eta$-Reduktion ist reflexiv, transitiv und kontextverträglich. Ferner ist sie substitutionsverträglich, d.h.

$$M \to_{\beta\eta}^* M',\ N \to_{\beta\eta}^* N' \implies M[x := N] \to_{\beta\eta}^* M'[x := N']$$

4.5 Satz von Church-Rosser. Die $\beta\eta$-Reduktion von λ-Termen ist konfluent.

$$\begin{tikzcd}
& M \arrow[dl, "*"', "\beta\eta"] \arrow[dr, "*", "\beta\eta"'] & \\
N \arrow[dr, dotted, "*", "\beta\eta"'] & & N' \arrow[dl, dotted, "*"', "\beta\eta"] \\
& P &
\end{tikzcd}$$

Für Beweise des Satzes von Church-Rosser und der folgenden Aussagen, siehe Barendregt 1984 oder Krivine (S. 15–18). Sie werden ganz analag zu den Beweisen im vorhergehenden Abschnitt für die β-Reduktion geführt. □

4.6 $\beta\eta$-Normalform. Einen λ-Term, der sowohl in β- als auch in η-Normalform ist, nennt man eine *$\beta\eta$-Normalform.*

Besitzt ein Term M eine β-Normalform N, so läßt sich diese durch η-Kontraktionen in eine η-Normalform N' überführen. Da bei einer η-Kontraktion keine β-Redexe entstehen können, ist N' eine $\beta\eta$-Normalform.

Folgerung. Hat ein Term eine β-Normalform, dann hat er auch eine $\beta\eta$-Normalform.

Die Umkehrung dieser Aussage ist ebenfalls richtig. Wir werden dies aber nicht beweisen.

Satz. Die $\beta\eta$-Normalform eines Terms ist eindeutig bestimmt, sofern sie überhaupt existiert.

Beweis. Die Eindeutigkeit folgt direkt aus 4.5. □

Der Term $\omega_2 \equiv (\lambda x.xx)(\lambda x.xx)$ ist in η-Normalform, er hat jedoch keine β-Normalform, folglich auch keine $\beta\eta$-Normalform.

4.7 $\beta\eta$-Äquivalenz. Zwei Terme P und Q werden *$\beta\eta$-äquivalent* genannt, wenn es einen Term N und $\beta\eta$-Reduktionen gibt, so daß $P \to^*_{\beta\eta} N$ und $Q \to^*_{\beta\eta} N$. Wir schreiben dann $P \simeq_{\beta\eta} Q$.

Wie für die β-Äquivalenz zeigt man:

Satz. $\simeq_{\beta\eta}$ ist eine kontextverträgliche Äquivalenzrelation. Außerdem ist $\simeq_{\beta\eta}$ substitutionsverträglich.

Eine $\beta\eta$-Äquivalenzklasse enthält höchstens eine $\beta\eta$-Normalform, denn nach dem Satz von Church-Rosser (4.5) sind verschiedene $\beta\eta$-Normalformen $\beta\eta$-inäquivalent, z.B.

$$x \not\simeq_{\beta\eta} y$$
$$\lambda y.\lambda x.x \not\simeq_{\beta\eta} \lambda y.\lambda x.yx \not\simeq_{\beta\eta} \lambda y.\lambda x.y(yx)$$

Beachte, daß in der letzten Gleichungskette der mittlere Term $\beta\eta$-äquivalent zu $\lambda y.y$ und der hintere eine $\beta\eta$-Normalform ist.

Jede $\beta\eta$-Äquivalenzklasse besteht aus β-Äquivalenzklassen, denn $M \simeq_\beta N \implies M \simeq_{\beta\eta} N$. Die $\beta\eta$-Äquivalenzklassen sind echt größer als die β-Äquivalenzklassen, denn $\mathsf{E} \equiv \lambda xy.xy$ und $\mathsf{I} \equiv \lambda x.x$ sind nicht β-äquivalent, aber $\beta\eta$-äquivalent: $\mathsf{E} \not\simeq_\beta \mathsf{I}$, aber $\mathsf{E} \simeq_{\beta\eta} \mathsf{I}$.

4.8 Satz.

(1) Ist x weder frei in M, noch frei in N, dann gilt $Nx \simeq_{\beta\eta} Mx \implies N \simeq_{\beta\eta} M$.

(2) Wenn $MP \simeq_{\beta\eta} NP$ für alle $P \in \Lambda$, dann $N \simeq_{\beta\eta} M$.

Beweis. (1) $Nx \simeq_{\beta\eta} Mx \implies \lambda x.Nx \simeq_{\beta\eta} \lambda x.Mx$. Ist nun x weder in M noch in N frei, dann ist $\lambda x.Nx \simeq_{\beta\eta} N$ und $\lambda x.Mx \simeq_{\beta\eta} M$. Folglich gilt $N \simeq_{\beta\eta} M$.

(2) Dies folgt direkt aus dem ersten Teil, denn man kann für P eine Variable wählen, die in M und N nicht vorkommt. (Dabei nutzt man aus, daß $\simeq_{\beta\eta}$ substitutionsverträglich ist.) $\square$

4.9 λ-Kalkül als formale Theorie. Für die β-Äquivalenz von λ-Termen gelten die folgenden Regeln:

(α) $$\lambda x.M \simeq \lambda y.\big(M[x := y]\big) \quad \text{, falls } y \notin \mathrm{FV}(M)$$

(β) $$(\lambda x.M)N \simeq M[x := N]$$

(ϱ) $$M \simeq M$$

(σ) $$M \simeq N \implies N \simeq M$$

(τ) $$M \simeq N,\ N \simeq P \implies M \simeq P$$

(μ) $$M \simeq M' \implies MN \simeq M'N$$

(ν) $$N \simeq N' \implies MN \simeq MN'$$

(ξ) $$M \simeq M' \implies \lambda x.M \simeq \lambda x.M'$$

Die Regeln (ϱ), (σ) und (τ) besagen, daß $=$ eine Äquivalenzrelation ist. (μ), (ν) und (ξ) drücken die Kontextverträglichkeit aus.

Umgekehrt kann man nicht allzuschwer zeigen: Zwei λ-Terme sind genau dann β-äquivalent, wenn sich ihre Äquivalenz aus den Axiomen (α), (β), (ϱ) mithilfe der Schlußregeln (ϱ), (σ), (τ), (μ), (ν) und (ξ) herleiten läßt.

Die $\beta\eta$-Äquivalenz erhält man, wenn man die Schlußregel

(η) $$\lambda x.Mx \simeq M \quad \text{, falls } x \notin \mathrm{FV}(M)$$

Man kann also den λ-Kalkül auch so einführen, daß man

- sagt, was λ-Terme sein sollen,
- die Substitution für λ-Terme erklärt,
- Gleichheit von λ-Termen dadurch definiert, daß sie sich aus den Axiomen (α), (β) und (ϱ) mithilfe der Schlußregeln
  - (ϱ), (σ), (τ), (μ), (ν) und (ξ) für den „λ_β-Kalkül" bzw.
  - (ϱ), (σ), (τ), (μ), (ν), (ξ) und (η) für den „$\lambda_{\beta\eta}$-Kalkül"

  herleiten läßt.

Eine solche Theorie wird *inkonsistent* genannt, falls man herleiten kann, daß $M \simeq N$ für alle Terme M, N gilt. Hierfür genuegt es $x \simeq y$ für zwei verschiedene Variablen herzuleiten. Ansonsten nennt man die Theorie *konsistent*.

Wir haben (letztlich durch den Satz von Church-Rosser) gezeigt, daß $x \not\simeq y$, wenn x und y verschiedene Variablen sind. Folglich gilt:

Satz. Der λ_β-Kalkül und der $\lambda_{\beta\eta}$-Kalkül sind konsistent.

Würde man bei der η-Kontraktion die Nebenbedingung weglassen, daß x nicht frei in P vorkommen soll, dann würder der $\lambda\eta$-Kalkül inkonsistent. Man hätte dann nämlich für beliebige Variablen $x = \lambda x.xx = \lambda y.yy = y$.

5. Programmieren im ungetypten λ-Kalkül

In diesem Abschnitt betrachten wir nur den λ_β-Kalkül, wollen also auf η-Reduktionen verzichten.

Wir wollen zeigen, daß der λ-Kalkül sich als Programmiersprache verstehen läßt, die in dem Sinne universell ist, daß sich alles, was berechenbar ist, in dieser Programmiersprache darstellen läßt. Es gibt verschiedene Ansätze den Begriff der Berechenbarkeit allgemein zu fassen: Turing-Maschinen, Postsche Systeme, Algorithmen im Sinne von Markov, partiell rekursive Funktionen. All diese Beechenbarkeitsbegriffe haben sich als äquivalent herausgestellt. Wir wollen auf den Begriff der partiell rekursiven Funktionen zurückgreifen und zeigen, daß es genau diese Funktionen sind, die sich im λ-Kalkül darstellen lassen.

Als erstes stellen wir die natürlichen Zahlen durch λ-Terme dar. Dafür definieren wir für beliebige λ-Terme M und N zunächst die folgende Notation (welche nicht unserer sonstigen Klammerkonvention entspricht:

$$\begin{aligned} M^0 N &:\equiv N \\ M^1 N &:\equiv MN \\ M^2 N &:\equiv M(MN) \\ &\vdots \\ M^{n+1} N &:\equiv M(M^n N) \end{aligned}$$

5.1 Darstellung der natürlichen Zahlen (Church-Zahlen). Wir setzen

$$\begin{aligned} \overline{0} &:\equiv \lambda yx.x \\ \overline{1} &:\equiv \lambda yx.yx \\ \overline{2} &:\equiv \lambda yx.y(yx) \\ &\vdots \\ \overline{n} &:\equiv \lambda yx.y^n x \quad \text{für } n \in \mathbb{N} \end{aligned}$$

Wir bemerken, daß wir die natürlichen Zahlen durch Terme in β-Normalform dargestellt haben.

5.2 Definition. Gegeben sei eine totale, d.h. überall definierte Funktion $f\colon \mathbb{N}^k \to \mathbb{N}$. Ist F ein λ-Term mit der Eigenschaft, daß

$$F\,\overline{n_1}\,\overline{n_2}\,\ldots\overline{n_k} \simeq_\beta \overline{f(n_1, n_2, \ldots, n_k)} \quad \text{für alle } (n_1, \ldots, n_k) \in \mathbb{N}^k$$

so sagen wir, F *stellt die Funktion* f *dar*.

Ist f eine partielle, d.h. nur auf einer Teilmenge $D_f \subseteq \mathbb{N}^k$ definierte Funktion mit Werten in $\mathbb{N}$, so sagt man, ein λ-Term F *stelle die Funktion* f *dar*, falls

$$F\,\overline{n_1}\,\overline{n_2}\,\ldots\overline{n_k} \begin{cases} \simeq_\beta \overline{f(n_1, n_2, \ldots, n_k)} & \text{falls } (n_1, \ldots, n_k) \in D_f \\ \text{hat keine } \beta\text{-Normalform} & \text{falls } (n_1, \ldots, n_k) \notin D_f \end{cases}$$

Statt „F stellt f dar“ sagen wir auch „F berechnet f“.

Es gilt der folgende Satz:

5.3 Satz. Jede partielle rekursive Funktion $f\colon D_f \to \mathbb{N}$ mit $D_f \subseteq \mathbb{N}$ läßt sich durch einen λ-Term F darstellen.

Man kann diesen λ-Term F als Programm betrachten, das erlaubt, die Funktion f zu berechnen: Will man für ein k-Tupel $(n_1, \ldots, n_k) \in \mathbb{N}^k$ den Funktionswert $n = f(n_1, \ldots, n_k)$ berechnen, so bilde man den λ-Term $F\,\overline{n_1}\,\overline{n_2}\,\ldots\overline{n_k}$ und bestimme dessen β-Normalform, sofern es überhaupt eine solche gibt. Wenn $F\,\overline{n_1}\,\overline{n_2}\,\ldots\overline{n_k}$ keine β-Normalform besitzt, dann ist $f(n_1, \ldots, n_k)$ nicht definiert. Ansonsten ist die β-Normalform gerade die Church-Zahl $\overline{n}$ des gesuchten Funktionswertes $n = f(n_1, \ldots, n_k)$:

$$F\,\overline{n_1}\,\overline{n_2}\,\ldots\overline{n_k} \to_\beta^* \overline{n} = \overline{f(n_1, \ldots, n_k)}$$

Wir werden 5.3 nur für totale rekursive Funktionen beweisen. Umgekehrt gilt aber auch (Kleene, 1936):

Satz. Ist eine partielle Funktion $f\colon D_f \to \mathbb{N}$ mit $D_f \subseteq \mathbb{N}^k$ durch einen λ-Term darstellbar, so ist sie rekursiv.

Dieser Satz beruht im wesentlichen darauf, daß es – wie wir im letzten Paragraphen dieses Kapitels sehen werden – einen Algorithmus gibt, der einen gegebenen Term zu seiner Normalform zu reduziert, sofern es eine solche überhaupt gibt, ansonsten nicht terminiert.

Also sind die durch λ-Terme darstellbaren partiellen Funktionen $f\colon D_f \to \mathbb{N}$ mit $D_f \subseteq \mathbb{N}^k$ genau die rekursiven. Dies ist eine Bestätigung für die CHURCHsche These, die besagt, daß die berechenbaren Funktionen genau die rekursiven seien.

Im folgenden werden wir Namen als Abkürzungen für Terme einführen, wie z.B. $\mathsf{I} :\equiv \lambda x.x$. Die Namen sind so gewählt, daß sie die intendierte Bedeutung suggerieren. Die suggestive Schreibweise sollte aber nicht darüber hinwegtäuschen, daß es sich eigentlich um λ-Terme, also rein syntaktische Wesen handelt und nicht um natürliche Zahlen, Wahrheitswerte usw.

Wie man sieht, kommen wir hier einer Programmiersprache näher. Wer sehen will, wie diese Ideen in LISP, ML oder Haskell weitergeführt werden, mag dies in den in der Literaturliste angegebenen Büchern nachlesen. Wir beginnen mit der Darstellung einfacher Funktionen:

5.4 Nachfolger und Addition. Wir setzen

$$\mathsf{SUCC} :\equiv \lambda nyx.y(nyx) \qquad\qquad \mathsf{ADD} :\equiv \lambda nmyx.ny(myx)$$

und zeigen im folgenden, daß SUCC die Nachfolgerfunktion $n \mapsto n+1$ und ADD die Addition $(n,m) \mapsto n+m$ darstellt:

$\mathsf{SUCC}\,\overline{n} \simeq_\beta \overline{n+1}$**:**

$$\begin{aligned}\mathsf{SUCC}\,\overline{n} &\equiv \big(\lambda nyx.y(nyx)\big)(\lambda yx.y^n x)\\ &\to_\beta \lambda yx.y\big((\lambda yx.y^n x)yx\big)\\ &\to_\beta \lambda yx.y(\lambda x.y^n x)x\\ &\to_\beta \lambda yx.y(y^n x) \equiv \lambda yx.y^{n+1}x \equiv \overline{n+1}\end{aligned}$$

$\mathsf{ADD}\,\overline{n}\,\overline{m} \simeq_\beta \overline{n+m}$**:**

$$\begin{aligned}\mathsf{ADD}\,\overline{n}\,\overline{m} &\equiv \big(\lambda nmyx.ny(myx)\big)\overline{n}\,\overline{m}\\ &\to_\beta \big(\lambda myx.\overline{n}y(myx)\big)\overline{m}\\ &\to_\beta \lambda yx.\overline{n}y(\overline{m}yx)\\ &\to_\beta^* \lambda yx.(\lambda x.y^n x)(y^m x)\\ &\to_\beta \lambda yx.y^n(y^m x) \equiv \lambda yx.y^{m+n}x \equiv \overline{n+m}\end{aligned}$$

Dabei haben wir die beiden folgenden Nebenrechnungen benutzt:

$$\begin{gathered}\overline{n}y \equiv (\lambda yx.y^n x)y \to_\beta \lambda x.y^n x\\ \overline{m}yx \equiv (\lambda yx.y^m x)yx \to_\beta (\lambda x.y^m x)x \to_\beta y^m x\end{gathered}$$

Bemerkung. Wir haben hier die Buchstaben n und m in zwei verschiedenen Weisen benutzt: einmal als Namen für Variablen des λ-Kalküls wie z.B. in $\lambda nyx.y(nyx)$, zum anderen als Namen für natürliche Zahlen wie z.B. in $\overline{n+m}$ oder $\lambda yx.y^n x$. Dies ist eigentlich unzulässig, denn es könnte zu Fehlern verleiten. Der vorsichtige Leser ersetze die λ-Variablen n und m durch u und v:

$$\mathsf{SUCC} :\equiv \lambda uyx.y(uyx) \qquad\qquad \mathsf{ADD} :\equiv \lambda vuyx.vy(uyx)$$

5.5 Wahrheitswerte. Die Booleschen Werte TRUE und FALSE stellen wir folgendermaßen dar:

$$\mathsf{TRUE} :\equiv \lambda xy.x \qquad\qquad \mathsf{FALSE} :\equiv \lambda xy.y$$

Bemerke: $\mathsf{TRUE} \equiv \mathsf{K}$, $\mathsf{FALSE} \equiv \overline{0}$.

Das Konditional definieren wir durch

$$\mathsf{COND}(E, E_1, E_2) :\equiv EE_1E_2 \quad \text{bzw.} \quad \mathsf{COND} :\equiv \lambda xyz.xyz$$

Wir verifizieren, daß dieser λ-Term das Gewünschte leistet, daß also gilt:

$$\mathsf{COND}\,\mathsf{TRUE}\,E_1E_2 \simeq_\beta E_1 \qquad\qquad \mathsf{COND}\,\mathsf{FALSE}\,E_1E_2 \simeq_\beta E_2$$

Beweis.

$$\mathsf{COND}\,\mathsf{TRUE}\,E_1E_2 \simeq_\beta \mathsf{TRUE}\,E_1E_2 \equiv (\lambda xy.x)E_1E_2 \to_\beta^* E_1$$
$$\mathsf{COND}\,\mathsf{FALSE}\,E_1E_2 \simeq_\beta \mathsf{FALSE}\,E_1E_2 \equiv (\lambda xy.y)E_1E_2 \to_\beta^* E_2$$

□

Den Test, ob eine natürliche Zahl gleich Null ist, können wir durch den folgenden Term simulieren:

$$\mathsf{ISNULL} :\equiv \lambda x_1.x_1(\lambda x.\,\mathsf{FALSE})\,\mathsf{TRUE}$$

Dieser Term stellt den Nulltest dar. Es gilt also

$$\mathsf{ISNULL}\,\overline{0} \simeq_\beta \mathsf{TRUE} \qquad\qquad \mathsf{ISNULL}(\mathsf{SUCC}\,\overline{n}) \simeq_\beta \mathsf{FALSE}$$

Beweis.

$$\begin{aligned}
\mathsf{ISNULL}\,\overline{0} &\equiv \big(\lambda x_1.x(\lambda x.\,\mathsf{FALSE})\,\mathsf{TRUE}\,\big)\overline{0}\\
&\to_\beta \overline{0}(\lambda x.\,\mathsf{FALSE})\,\mathsf{TRUE}\\
&\equiv (\lambda xy.y)(\lambda x.\,\mathsf{FALSE})\,\mathsf{TRUE}\\
&\to_\beta^* \mathsf{TRUE}
\end{aligned}$$

$$\begin{aligned}
\mathsf{ISNULL}(\mathsf{SUCC}\,\overline{n}) &\to_\beta^* \mathsf{ISNULL}\,\overline{n+1}\\
&\equiv \big(\lambda x_1.x_1(\lambda x.\,\mathsf{FALSE})\,\mathsf{TRUE}\,\big)(\lambda yx.y^{n+1}x)\\
&\to_\beta (\lambda yx.y^{n+1}x)(\lambda x.\,\mathsf{FALSE})\,\mathsf{TRUE}\\
&\to_\beta^* (\lambda x.\,\mathsf{FALSE})^{n+1}\,\mathsf{TRUE}\\
&\equiv (\lambda x.\,\mathsf{FALSE})\big((\lambda x.\,\mathsf{FALSE})^n\,\mathsf{TRUE}\,\big)\\
&\to_\beta \mathsf{FALSE}
\end{aligned}$$

□

5.6 Paare und n-Tupel ($n \geq 2$). Wir erklären Terme, die die Paarbildung und die Projektionen auf die erste und zweite Komponente simulieren:

$$\langle M_1, M_2\rangle :\equiv \lambda z.zM_1M_2 \qquad \mathsf{FST} :\equiv \lambda z.z(\lambda xy.x) \qquad \mathsf{SND} :\equiv \lambda z.z(\lambda xy.y)$$

Dabei sei z so gewählt, daß z in M_1 oder M_2 nicht frei vorkommt. Daß man auf diese Art und Weise Paare kodieren kann, wurde bereits in Übungaufgabe 13 nachgewiesen.

Bemerkung. Eigentlich sollte man wie in der Übung den Paarbildungsoperator definieren:

$$\mathsf{PAIR} :\equiv \lambda xyz.zxy$$

Es gilt dann: $\mathsf{PAIR}\,M_1M_2 \simeq_\beta \langle M_1, M_2\rangle$.

Durch iterierte Paarbildung gelangt man zu n-Tupeln:

$$n\text{-}\mathsf{TUPEL} :\equiv \lambda x_1 \ldots x_n.\,\mathsf{PAIR}\,x_1(\mathsf{PAIR}\,x_2(\ldots(\mathsf{PAIR}\,x_{n-1}x_n)\ldots))$$

und wir schreiben

$$\begin{aligned}
\langle M_1, M_2, \ldots, M_n\rangle &:\equiv n\text{-}\mathsf{TUPEL}\,M_1M_2\ldots M_n \simeq_\beta \langle M_1, \langle M_2, \langle \ldots, \langle M_{n-1}, M_n\rangle \ldots\rangle\rangle\rangle\\
\mathsf{PR}_1^n\,t &:\equiv \mathsf{FST}\,t
\end{aligned}$$

$$\begin{aligned}
\mathsf{PR}_2^n\,t &:\equiv \mathsf{FST}(\mathsf{SND}\,t)\\
\mathsf{PR}_i^n\,t &:\equiv \mathsf{FST}(\mathsf{SND}^{i-1}\,t) \quad \text{für } 1 < i < n\\
\mathsf{PR}_n^n\,t &:\equiv \mathsf{SND}^{n-1}\,t
\end{aligned}$$

Übung. Zeige, daß PR_i^n für $i = 1, \ldots, n$ gerade die Projektion auf die i-te Komponente eines n-Tupels darstellt, also daß gilt:

$$\mathsf{PR}_i^n\langle M_1, \ldots, M_n\rangle \simeq_\beta M_i$$

5.7 Darstellung der Vorgängerfunktion. Diese ist für die Church-Zahlen recht schwer zu simulieren, denn man muß in $\overline{n} \equiv \lambda yx.y^n x$ (für $n > 0$) ein y hinwegzaubern, um zu $\overline{n-1} \equiv \lambda yx.y^{n-1}x$ zu gelangen. Man definiert dazu folgenden Hilfsterm:

$$P :\equiv \lambda yp.\langle \mathsf{FALSE}, \mathsf{COND}(\mathsf{FST}\,p)(\mathsf{SND}\,p)(y\,\mathsf{SND}\,p)\rangle$$

Für diesen gilt:

$$\begin{aligned}
Py\langle \mathsf{TRUE}, x\rangle &\simeq_\beta \langle \mathsf{FALSE}, x\rangle\\
(Py)^2\langle \mathsf{TRUE}, x\rangle &\simeq_\beta Py\langle \mathsf{FALSE}, x\rangle \simeq_\beta \langle \mathsf{FALSE}, yx\rangle\\
(Py)^3\langle \mathsf{TRUE}, x\rangle &\simeq_\beta Py\langle \mathsf{FALSE}, yx\rangle \simeq_\beta \langle \mathsf{FALSE}, y(yx)\rangle
\end{aligned}$$

usw. Man stellt also fest:

$$(*) \qquad (Py)^n\langle \mathsf{TRUE}, x\rangle \simeq_\beta \langle \mathsf{FALSE}, y^{n-1}x\rangle \quad \text{für } n > 0$$

Nun definiere:

$$\mathsf{PRED} :\equiv \lambda nyx.\,\mathsf{SND}\,\big(n(Py)\langle \mathsf{TRUE}, x\rangle\big)$$

Dies ist ein Term, der die Vorgängerfunktion darstellt.

Beweis. Zu zeigen ist: $\mathsf{PRED}\,\overline{n} \simeq_\beta \overline{n-1}$ für alle $n > 0$.

$$\begin{aligned}
\mathsf{PRED}\,\overline{n} &\simeq_\beta \lambda yx.\,\mathsf{SND}\,\big(\overline{n}(Py)\langle \mathsf{TRUE}, x\rangle\big)\\
&\equiv \lambda yx.\,\mathsf{SND}\,\big((\lambda yx.y^n x)(Py)\langle \mathsf{TRUE}, x\rangle\big)\\
&\simeq_\beta \lambda yx.\,\mathsf{SND}\,\big((Py)^n\langle \mathsf{TRUE}, x\rangle\big)\\
&\simeq_\beta \lambda yx.\,\mathsf{SND}\langle \mathsf{FALSE}, y^{n-1}x\rangle \qquad \text{nach } (*)\\
&\simeq_\beta \lambda yx.y^{n-1}x \equiv \overline{n-1}
\end{aligned}$$

□

Um Rekursion zu programmieren werden wir wir Fixpunktkombinatoren benutzen. Wie in der Übungsaufgabe 14 wird eine geschlossener λ-Term Y ein *Fixpunktkombinator* genannt, wenn gilt

$$\mathsf{Y}\,M \simeq_\beta M(\mathsf{Y}\,M) \text{ für alle } \lambda\text{-Terme} M\,.$$

In der Übungsaufgabe 14 haben wir solche Fixpunktkombinatoren angegeben. In der Folge sei Y ein solcher.

5.8 Fixpunktsatz. Für jeden λ-Term M kann man einen λ-Term F finden, so daß $F \simeq_\beta M[y := F]$, d.h. die Gleichung $y \simeq_\beta M$ hat stets eine Lösung $y \simeq_\beta F \in \Lambda$.

Beweis. Sei Y ein Fixpunktkombinator (vgl. Übungsaufgabe 14). Setze $F \equiv \mathsf{Y}(\lambda y.M)$. Dann gilt $F \equiv \mathsf{Y}(\lambda y.M) \simeq_\beta (\lambda y.M)(\mathsf{Y}(\lambda y.M)) \equiv (\lambda y.M)F \simeq_\beta M[y := F]$. □

Verallgemeinerung. Es seien $y, x_1, \ldots, x_n$ Variablen und M ein Term, der diese Variablen möglicherweise frei enthält. Dann gibt es einen Term F, so daß $Fx_1 \ldots x_n \simeq_\beta M[y := F]$. Dabei ist vorausgesetzt, daß y von den $x_1, \ldots, x_n$ verschieden ist.

Beweis. Man setzt ähnlich wie zuvor $F \equiv \mathsf{Y}(\lambda y x_1 \ldots x_n.M)$, wobei Y wieder ein Fixpunktkombinator ist. Es gilt dann

$$\begin{aligned} F &\equiv \mathsf{Y}(\lambda y x_1 \ldots x_n.M) \\ &\simeq_\beta (\lambda y x_1 \ldots x_n.M)(\mathsf{Y}(\lambda y x_1 \ldots x_n.M)) \\ &\equiv (\lambda y x_1 \ldots x_n.M)F \\ &\to_\beta \lambda x_1 \ldots x_n.\big(M[y := F]\big) \end{aligned}$$

also $Fx_1 \ldots x_n \simeq_\beta M[y := F]$. □

Man sagt auch kurz: „Die Gleichung $yx_1 \ldots x_n \simeq_\beta M$ besitzt eine Lösung $y := F \in \Lambda$“.

5.9 Rekursive Definition der Multiplikation. In den Übungen haben wir die Multiplikation natürlicher Zahlen durch einen λ-Term dargestellt. Als Beispiel für rekursive Definition wollen wir eine andere Darstellung geben. Umgangssprachlich würde eine rekursive Definition der Multiplikation so aussehen:

$$m \cdot n = \begin{cases} 0 & \text{falls } m = 0 \\ n + (m-1) \cdot n & \text{sonst} \end{cases}$$

bzw.

$$\mathrm{mult}(m, n) = \begin{cases} 0 & \text{falls } m = 0 \\ \mathrm{add}(n, \mathrm{mult}(\mathrm{pred}(m), n)) & \text{sonst} \end{cases}$$

in Präfixschreibweise.

Wollen wir also die Multiplikation durch einen λ-Term MULT darstellen, so muß für diesen gelten

$$\mathsf{MULT}\,\overline{m}\,\overline{n} \simeq_\beta \mathsf{COND}(\mathsf{ISNULL}\,\overline{m})\,\overline{0}\,(\mathsf{ADD}\,\overline{n}(\mathsf{MULT}(\mathsf{PRED}\,\overline{m})\overline{n}))$$

oder auch

$$\mathsf{MULT} \simeq_\beta \lambda mn.\,\mathsf{COND}(\mathsf{ISNULL}\,m)\,\overline{0}\,(\mathsf{ADD}\,n(\mathsf{MULT}(\mathsf{PRED}\,m)n))$$

Nun wird MULT durch diese Gleichung nicht definiert, da MULT auch auf der rechten Seite vorkommt. Wir suchen also eine Lösung y der Gleichung

$$y \simeq_\beta \underbrace{\lambda mn.\,\mathsf{COND}(\mathsf{ISNULL}\,m)\,\overline{0}\,(\mathsf{ADD}\,n(y(\mathsf{PRED}\,m)n))}_{\equiv M}$$

Aufgrund des Fixpunktsatzes in 5.8 ist $\mathsf{MULT} :\equiv \mathsf{Y}(\lambda y.M)$ ein Term mit der gewünschten Eigenschaft.

Beweis.

$$\begin{aligned}\mathsf{MULT}\,\overline{m}\,\overline{n} &\equiv \mathsf{Y}(\lambda y.M)\overline{m}\,\overline{n}\\ &\simeq_\beta (\lambda y.M)(\mathsf{Y}(\lambda y.M))\overline{m}\,\overline{n}\\ &\to_\beta^* \mathsf{COND}(\mathsf{ISNULL}\,\overline{m})\,\overline{0}\,(\mathsf{ADD}\,\overline{n}(\mathsf{Y}(\lambda y.M)(\mathsf{PRED}\,\overline{m})\overline{n}))\\ &\to_\beta^* \begin{cases}\overline{0} & \text{falls } m=0\\ \mathsf{ADD}\,\overline{n}(\mathsf{Y}(\lambda y.M)(\mathsf{PRED}\,\overline{m})\overline{n}) \equiv \mathsf{ADD}\,\overline{n}(\mathsf{MULT}\,\overline{m-1}\,\overline{n}) & \text{falls } m>0\end{cases}\end{aligned}$$

□

Beispiel.

$$\begin{aligned}\mathsf{MULT}\,\overline{2}\,\overline{4} &\simeq_\beta \mathsf{ADD}\,\overline{4}(\mathsf{MULT}\,\overline{1}\,\overline{4})\\ &\simeq_\beta \mathsf{ADD}\,\overline{4}(\mathsf{ADD}\,\overline{4}(\mathsf{MULT}\,\overline{0}\,\overline{4}))\\ &\simeq_\beta \mathsf{ADD}\,\overline{4}(\mathsf{ADD}\,\overline{4}\,\overline{0}) \simeq_\beta \mathsf{ADD}\,\overline{4}\,\overline{4} \simeq_\beta \overline{8}\end{aligned}$$

5.10 Darstellung primitiv rekursiver Funktionen. Eine Funktion wird *primitiv rekursiv* genannt, wenn sie sich in endlich vielen Schritten aus

- der konstanten Nullfunktion $\mathrm{const}_0\colon \mathbb{N}^n \to \mathbb{N},\ (x_1,\ldots,x_n) \mapsto 0$,
- der Nachfolgerfunktion $\mathrm{succ}\colon \mathbb{N} \to \mathbb{N},\ x \mapsto x+1$ und
- den kanonischen Projektionen $\mathrm{pr}_i^n\colon \mathbb{N}^n \to \mathbb{N},\ (x_1,\ldots,x_n) \mapsto x_i$

mittels Komposition und primitiver Rekursion definieren läßt. Dabei bedeuten

Komposition: Gegeben seien $g\colon \mathbb{N}^k \to \mathbb{N}$ und $h_1,\ldots,h_k\colon \mathbb{N}^\ell \to \mathbb{N}$. Wir sagen, die durch

$$f(x_1,\ldots,x_\ell) := g\big(h_1(x_1,\ldots,x_\ell),\ldots,h_k(x_1,\ldots,x_\ell)\big)$$

definierte Funktion $f\colon \mathbb{N}^\ell \to \mathbb{N}$ sei die *Komposition* $g\circ(h_1,\ldots,h_k$ von $h_1,\ldots,h_k$ und g.

Primitive Rekursion: Gegeben seien $g\colon \mathbb{N}^{k-1} \to \mathbb{N}$ und $h\colon \mathbb{N}^{k+1} \to \mathbb{N}$. Wir sagen, die durch

$$f(n,x_2,\ldots,x_k) = \begin{cases} g(x_2,\ldots,x_k) & \text{falls } n=0\\ h\big(n-1, f(n-1,x_2,\ldots,x_k),x_2,\ldots,x_k\big) & \text{falls } n>0\end{cases}$$

gegebene Funktion $f\colon \mathbb{N}^k \to \mathbb{N}$ sei durch *primitive Rekursion* aus g und h entstanden.

Im Spezialfall $k=1$ seien $m_0 \in \mathbb{N}$ und $h\colon \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ gegeben. Dann kann man $f\colon \mathbb{N} \to \mathbb{N}$ rekursiv definieren durch

$$f(n) = \begin{cases} m_0 & \text{falls } n=0\\ h\big(n-1, f(n-1)\big) & \text{falls } n>0\end{cases}$$

Beispiel.

(1) Die Addition add: $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $(x_1, x_2) \mapsto x_1 + x_2$ ist primitiv rekursiv, denn

$$\operatorname{add}(n, x_2) = \begin{cases} \operatorname{pr}_1^1(x_2) = x_2 & \text{falls } n = 0 \\ \operatorname{succ}(\operatorname{add}(n-1, x_2)) & \text{falls } n > 0 \end{cases}$$

(2) Die Multiplikation mult: $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $(x_1, x_2) \mapsto x_1 \cdot x_2$ ist primitiv rekursiv, denn

$$\operatorname{mult}(n, x_2) = \begin{cases} 0 & \text{falls } n = 0 \\ \operatorname{add}(\operatorname{mult}(n-1, x_2), x_2) & \text{falls } n > 0 \end{cases}$$

(3) Die Fakultät $!\colon \mathbb{N} \to \mathbb{N}$ ist primitiv rekursiv, denn

$$n! = \begin{cases} 1 & \text{falls } n = 0 \\ \operatorname{mult}(n, (n-1)!) & \text{falls } n > 0 \end{cases}$$

Nun beweisen wir als Teilaussage von Satz 5.3: Jede primitiv rekursive Funktion f ist durch einen λ-Term darstellbar durch Induktion über den Aufbau von f gemäß 5.10.

Die Grundbausteine const_0, Nachfolger und die kanonischen Projektionen stellen wir wie folgt durch λ-Terme dar:

- $\mathsf{ZERO}_n \equiv \lambda x_1 \dots x_n.\overline{0} \equiv \lambda x_1 \dots x_n yx.x$ (vgl. 5.1)
- $\mathsf{SUCC} \equiv \lambda nyx.y(nyx)$ (vgl. 5.4)
- $\mathsf{P}_i^n \equiv \lambda x_1 \dots x_n.x_i$

Zur Komposition: Die Funktion $f\colon \mathbb{N}^\ell \to \mathbb{N}$ sei durch Komposition von $h_1, \dots, h_k\colon \mathbb{N}^\ell \to \mathbb{N}$ mit $g\colon \mathbb{N}^k \to \mathbb{N}$ gebildet. Wenn sich $h_1, \dots, h_k$ und g durch die λ-Terme $H_1, \dots, H_k$ und G darstellen lassen, dann ist

$$F :\equiv \lambda x_1 \dots x_\ell.G(H_1 x_1 \dots x_\ell) \dots (H_k x_1 \dots x_\ell)$$

ein λ-Term, der f darstellt, denn für alle $(n_1, \dots, n_\ell) \in \mathbb{N}^\ell$ gilt:

$$\begin{aligned} F\overline{n_1} \dots \overline{n_\ell} &\to_\beta^* G(H_1\overline{n_1} \dots \overline{n_\ell}) \dots (H_k\overline{n_1} \dots \overline{n_\ell}) \\ &\simeq_\beta G\overline{h_1(n_1, \dots, n_\ell)} \dots \overline{h_k(n_1, \dots, n_\ell)} \\ &\simeq_\beta \overline{g\big(h_1(n_1, \dots, n_\ell), \dots, h_k(n_1, \dots, n_\ell)\big)} \end{aligned}$$

Zur primitiven Rekursion: Es sei $f\colon \mathbb{N}^k \to \mathbb{N}$ durch primitive Rekursion aus $g\colon \mathbb{N}^{k-1} \to \mathbb{N}$ und $h\colon \mathbb{N}^{k+1} \to \mathbb{N}$ gebildet. Sind G und H λ-Terme, welche g und h darstellen, so ist

$$F :\equiv \mathsf{Y}(\lambda y.M)$$

ein λ-Term, der f darstellt, wobei

$$M :\equiv \lambda x_1 \dots x_k.\,\mathsf{COND}\,\big(\mathsf{ISNULL}\,x_1\big)\big(Gx_2 \dots x_k\big)\big(H(\mathsf{PRED}\,x_1)(y(\mathsf{PRED}\,x_1)x_2 \dots x_k)x_2 \dots x_k\big)$$

denn für alle $(n, n_2, \dots, n_k) \in \mathbb{N}^k$ gilt:

$$\begin{aligned} F\overline{n}\,\overline{n_2} \dots \overline{n_k} &\equiv \mathsf{Y}(\lambda y.M)\overline{n}\,\overline{n_2} \dots \overline{n_k} \\ &\simeq_\beta (\lambda y.M)\,\underbrace{(\mathsf{Y}(\lambda y.M))}_{F}\,\overline{n}\,\overline{n_2} \dots \overline{n_k} \\ &\to_\beta^* \mathsf{COND}\,\big(\mathsf{ISNULL}\,\overline{n}\big)\big(G\overline{n_2} \dots \overline{n_k}\big)\big(H(\mathsf{PRED}\,\overline{n})(F(\mathsf{PRED}\,\overline{n})\overline{n_2} \dots \overline{n_k})\overline{n_2} \dots \overline{n_k}\big) \end{aligned}$$

$$\simeq_\beta \begin{cases} G\overline{n_2}\dots\overline{n_k} \simeq_\beta \overline{g(n_2,\dots,n_k)} & \text{falls } n=0 \\ H\overline{n-1}(F\overline{n-1}\,\overline{n_2}\dots\overline{n_k})\overline{n_2}\dots\overline{n_k} & \\ \simeq_\beta \overline{h(n-1,f(n-1,n_2,\dots,n_k),n_2,\dots,n_k)} & \text{falls } n>0 \end{cases}$$

5.11 Der μ-Operator und seine Darstellung. Für jede Funktion $g\colon \mathbb{N}^{k+1} \to \mathbb{N}$ definieren wir eine Funktion μg durch

$$(\mu g)(x_1,\dots,x_k) := \min\{z \in \mathbb{N} \mid g(z,x_1,\dots,x_k)=0\}\,.$$

μg ist eine partielle Funktion auf $\mathbb{N}^k$ mit Werten in $\mathbb{N}$: An der Stelle $(x_1,x_2,\dots,x_k)$ ist μg genau dann definiert, wenn es mindestens ein $z \in \mathbb{N}$ mit $g(z,x_1,\dots,x_k)=0$ gibt.

Satz. Es gibt einen λ-Term $\overline{\mu}$ derart, daß für jede Funktion $g\colon \mathbb{N}^{k+1} \to \mathbb{N}$, die durch einen λ-Term G darstellbar ist, gilt:

$$\overline{\mu}G\overline{n_1}\dots\overline{n_k} \simeq_\beta \overline{(\mu g)(n_1,\dots,n_k)}$$

für alle $(n_1,\dots,n_k)$ im Definitionsbereich von μg.

Beweis. Wir betrachten eine beliebige Funktion $g\colon \mathbb{N}^{k+1} \to \mathbb{N}$. Für jedes $\ell \in \mathbb{N}$ betrachten wir die (partielle) Hilfsfunktion

$$(\mu(\ell)g)(x_1,\dots,x_k) = \min\{z \in \mathbb{N} \mid z \geq \ell \text{ und } g(z,x_1,\dots,x_n)=0\}$$

Wir suchen einen λ-Term M derart, daß für jede Funktion $g\colon \mathbb{N}^{k+1} \to \mathbb{N}$, die durch einen λ-Term G darstellbar ist, gilt:

$$M\overline{\ell}\,G\,\overline{n_1}\dots\overline{n_k} \simeq_\beta \overline{(\mu(\ell)g)(n_1,\dots,n_k)}$$

immer dann wenn die rechte Seite definiert ist. Da $\mu g = \mu(0)g$ gilt, ist dann $\overline{\mu} :\equiv M\overline{0}$ ein Term mit der gewünschten Eigenschaft und der Satz ist bewiesen.

Der Schlüssel zum Auffinden von M ist die folgende rekursive Definition von $\mu(\ell)g$:

$$(\mu(\ell)g)(x_1,\dots,x_k) = \begin{cases} \ell & \text{falls } g(\ell,x_1,\dots,x_k)=x_1 \\ (\mu(\ell+1)g)(x_1,\dots,x_k) & \text{sonst} \end{cases}$$

d.h. wir suchen eine Lösung der Gleichung

$$y \simeq_\beta \underbrace{\lambda \ell g x_1\dots x_k.\,\mathsf{COND}\,\big(\,\mathsf{ISNULL}\,g\ell x_1\dots x_k x_1\big)\big(\ell\big)\big(y(\mathsf{SUCC}\,\ell)gx_1\dots x_k\big)}_{\equiv L}$$

Nach 5.8 ist $M :\equiv \mathsf{Y}(\lambda y.L)$ der gesuchte Term. □

Bemerkung. Der Term $\overline{\mu}$ hängt von k ab. Weiterhin kann man $\overline{\mu}$ sogar so wählen, daß $\overline{\mu}G\overline{n_1}\dots\overline{n_k}$ keine β-Normalform hat, wenn μg an der Stelle $(n_1,\dots,n_k)$ nicht definiert ist.

5.12 Darstellung beliebiger rekursiver Funktionen. Eine partielle Funktion f, die auf einer Teilmenge von $\mathbb{N}^k$ definiert ist und Werte in $\mathbb{N}$ hat, wird *partiell rekursiv* (=μ-rekursiv) genannt, wenn sie aus einer primitiv rekursiven Funktion g durch einmalige Anwendung des μ-Operators entsteht.

Es kommt natürlich vor, daß die Funktion $f = \mu g$ überall definiert ist; in diesem Fall spricht man von einer *totalen rekursiven Funktion* oder einfach von einer *rekursiven Funktion* $f\colon \mathbb{N}^k \to \mathbb{N}$.

Aus 5.10 und 5.11 schließen wir:

Satz. Jede (totale) rekursive Funktion ist durch einen λ-Term darstellbar.

Läßt man nun die Einschränkung weg, daß $f = \mu g$ überall definiert ist, d.h. ist $f\colon \mathbb{N}^k \rightharpoonup \mathbb{N}$ eine partielle rekursive Funktion, dann gibt es nach dem Vorhergehenden einen λ-Term F, so daß

$$F\overline{n_1}\ldots\overline{n_k} \simeq_\beta \overline{f(n_1,\ldots,n_k)}$$

für alle $(n_1,\ldots,n_k)$ im Definitionsbereich von f. Für den Satz 5.3 in seiner vollen Stärke müßte man jetzt noch beweisen, daß der Term $F\overline{n_1}\ldots\overline{n_k}$ keine Normalform hat, wenn $f = \mu g$ an der Stelle $(n_1,\ldots,n_k)$ nicht definiert ist. Dafür sei verwiesen auf Hindley-Seldin S. 55–57 (siehe auch Krivine S. 27-29).

Bemerkung. Wir haben Fixpunktkombinatoren Y verwendet, um primitive Rekursion und den μ-Operator darzustellen. Fixpunktkombinatoren haben keine β-Normalform, und auch die von uns erhaltenen λ-Terme F, die rekursive Funktionen darstellen, haben in der Regel *keine* β-Normalform, obwohl $F\overline{n_1}\ldots\overline{n_k} \simeq_\beta \overline{f(n_1,\ldots,n_k)}$ eine β-Normalform hat.

In den Übungen werden wir eine andere Methode kennenlernen, primitive Rekursion und den μ-Operator darzustellen. Bei dieser Methode erhält man für jede totale rekursive Funktion f einen λ-Term F, der diese darstellt und sogar eine β-Normalform hat.

Bemerkung. Unsere Definition von partiell rekursiven Funktionen ist etwas kurz geraten. Die Tatsache, daß wir den μ-Operator nur einmal am Schluß anwenden müssen, um alle partiellen rekursiven Funktionen aus den primitiv rekursiven zu erhalten, beruht auf dem nichttrivialen Normalformensatz von Kleene.

6. Unentscheidbarkeitssätze

Wir wollen unter anderem zeigen:

6.1 Satz. Die Frage, ob ein beliebiger λ-Term eine β-Normalform besitzt, ist unentscheidbar, d.h. es gibt keinen Algorithmus, der als Eingabe beliebige λ-Terme M akzeptiert und nach endlicher Zeit die Antwort ausgibt, ob M eine Normalform hat oder nicht.

Wie auch bei anderen Unentscheidbarkeitssätzen führt man den Beweis dadurch, daß man annimmt, es gäbe einen solchen Algorithmus – der in unserem Fall durch einen λ-Term dargestellt wird –, diesen dann auf sich selbst anwendet und dadurch zu einem Widerspruch kommt. (Dieses Prinzip nennt man *Diagonalisierung*.) Die Beweisidee wird in dem folgenden Satz erläutert:

6.2 Satz. Es gibt keinen λ-Term H mit der Eigenschaft, daß für alle λ-Terme M gilt:

$$(*) \qquad HM \simeq_\beta \begin{cases} \overline{1} & \text{falls } M \text{ eine } \beta\text{-Normalform besitzt} \\ \overline{0} & \text{falls } M \text{ keine } \beta\text{-Normalform besitzt} \end{cases}$$

Beweis. Nehmen wir an, es gäbe solch einen Term H. Wir wählen einen λ-Term Ω, der keine β-Normalform besitzt (z.B. $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$) und bilden den Term

$$P \equiv \mathsf{COND}\,\big(\mathsf{ISNULL}(Hy)\big)\,\overline{1}\,\Omega$$

Substituiert man für y einen Term M, so gilt

$$P[y := M] \simeq_\beta \begin{cases} \Omega & \text{falls } M \text{ eine } \beta\text{-Normalform besitzt} \\ \overline{1} & \text{falls } M \text{ keine } \beta\text{-Normalform besitzt} \end{cases}$$

Betrachten wir nun die Gleichung

$$y \simeq_\beta \mathsf{COND}\,\big(\mathsf{ISNULL}(Hy)\big)\,\overline{1}\,\Omega$$

so gibt es einen Term W, der diese Gleichung löst, nämlich $W \equiv \mathsf{Y}(\lambda y.P)$. Für diesen Term gilt aber

$$W \simeq_\beta \mathsf{COND}(\mathsf{ISNULL}(HW))\,\overline{1}\,\Omega \simeq_\beta \begin{cases} \Omega & \text{falls } W \text{ eine } \beta\text{-Normalform besitzt} \\ \overline{1} & \text{falls } W \text{ keine } \beta\text{-Normalform besitzt} \end{cases}$$

was in jedem Fall ein Widerspruch ist. Es kann also keinen Term mit der Eigenschaft $(*)$ geben. □

Gäbe es einen Term H mit der geforderten Eigenschaft (*), dann wäre die Frage entscheidbar, ob ein beliebiger Term eine β-Normalform besitzt. Daß es keinen solchen Term H gibt, ist allerdings noch kein Beweis für Satz 6.1. Es könnte ja sein, daß man diese Entscheidung durch eine geeignete Kodierung der Terme (evtl. durch andere Terme) und andere Manipulationen auf den Termen treffen könnte.

Um einen Beweis für die Unentscheidbarkeit führen zu können, muß man sich auf einen wohldefinierten Begriff von Entscheidbarkeit berufen. Ein solcher wird durch die Theorie der rekursiven Funktionen auf den natürlichen Zahlen zur Verfügung gestellt. Um diese Theorie auf unser Entscheidbarkeitsproblem anwenden zu können, müssen wir zuerst unsere λ-Terme durch natürliche Zahlen in einer effektiven Weise kodieren:

6.3 Gödelisierung. Wir wollen zunächst jedem λ-Term M eine natürliche Zahl $\gamma(M)$ zuordnen. Diese Zahl nennen wir dann *Gödelnummer* des Terms M.

Zunächst schreiben wir Terme mit ihrer vollen Beklammerung, d.h. ohne Benutzung der Klammerkonventionen, und ordnen den syntaktischen Zeichen und den Variablen $v_1, v_2, v_3, \ldots$ Zahlen zu:

a	$\lambda\_\_.$	(	)	v_1	v_2	v_3	…
$\gamma(a)$	1	2	3	4	5	6	…

Ferner sei $p_1, p_2, p_3, \ldots$ die Folge aller Primzahlen in aufsteigender Reihenfolge. Es sei bemerkt, daß die Funktion $n \mapsto p_n : \mathbb{N} \to \mathbb{N}$ rekursiv ist.

Jeder Folge von Zeichen der Form $w \equiv a_1 a_2 a_3 \ldots a_n$ mit $a_i \in \{\lambda\_\_., (,), v_1, v_2, v_3, \ldots\}$ ordnen wir die Zahl

$$\gamma(w) = p_1^{\gamma(a_1)} \cdot p_2^{\gamma(a_2)} \cdot \ldots \cdot p_n^{\gamma(a_n)}$$

zu. Da die Primfaktorzerlegung von natürlichen Zahlen eindeutig und durch einen Algorithmus durchführbar ist, kann man effektiv aus $\gamma(w)$ die Zeichenreihe w rekonstruieren. Insbesondere kann man dieses Verfahren nun auf λ-Terme anwenden:

$$\begin{array}{ll}
\mathsf{I} \equiv (\lambda v_1.v_1) & \gamma(\mathsf{I}) = p_1^2 \cdot p_2 \cdot p_3^4 \cdot p_4^4 \cdot p_5^3 \\
\overline{0} \equiv (\lambda v_2.(\lambda v_1.v_1)) & \gamma(\overline{0}) = p_1^2 \cdot p_2 \cdot p_3^5 \cdot p_4^2 \cdot p_5 \cdot p_6^4 \cdot p_7^4 \cdot p_8^3 \cdot p_9^3 \\
\overline{1} = (\lambda v_2.(\lambda v_1.(v_2 v_1))) & \gamma(\overline{1}) = p_1^2 \cdot p_2 \cdot p_3^5 \cdot p_4^2 \cdot p_5 \cdot p_6^4 \cdot p_7^2 \cdot p_8^5 \cdot p_9^4 \cdot p_{10}^3 \cdot p_{11}^3 \cdot p_{12}^3
\end{array}$$

Es ist auch entscheidbar, ob eine gegebene Zahl die Gödelnummer eines λ-Terms ist. Im folgenden brauchen wir die folgenden Eigenschaften:

(1) Die Funktion $n \mapsto \gamma(\overline{n})\colon \mathbb{N} \to \mathbb{N}$, die jeder natürlichen Zahl n die Gödelnummer der Church-Zahl $\overline{n}$ zuordnet, ist rekursiv.

(2) Es gibt eine rekursive Funktion $\tau\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, so daß
$$\gamma\big((MN)\big) = \tau\big(\gamma(M), \gamma(N)\big)$$
für beliebige λ-Terme M und N gilt.

Die erste Feststellung ist unmittelbar einsichtig. Man kann $\tau(n,m)$ in der zweiten Feststellung folgendermaßen definieren: Bestimme die Primzerlegungen für n und m:

$$n = p_{i_1}^{\alpha_1} \cdot p_{i_2}^{\alpha_2} \cdot \ldots \cdot p_{i_r}^{\alpha_r} \qquad\qquad m = p_{j_1}^{\beta_1} \cdot p_{j_2}^{\beta_2} \cdot \ldots \cdot p_{j_s}^{\beta_s}$$

wobei $p_{i_1} < p_{i_2} < \cdots < p_{i_r}$, $q_{j_1} < q_{j_2} < \cdots < q_{j_s}$, $\alpha_i > 0$, $\beta_j > 0$ gelten soll, und setze

$$\tau(n,m) = p_1^2 \cdot p_{i_1+1}^{\alpha_1} \cdot \ldots \cdot p_{i_r+1}^{\alpha_r} \cdot p_{j_1+i_r+1}^{\beta_1} \cdot \ldots \cdot p_{j_s+i_r+1}^{\beta_s} \cdot p_{j_s+i_r+2}^3$$

Nun können wir mittels Gödelisierung unser Entscheidungsproblem nach $\mathbb{N}$ übertragen: Es sei $\mathcal{A}$ die Menge aller λ-Terme, die eine β-Normalform besitzen und $\mathcal{B}$ die Menge aller λ-Terme ohne β-Normalform. Weiterhin sei $A = \gamma(\mathcal{A})$ die Menge aller Gödelnummern $\gamma(M)$ von Termen M mit β-Normalformen und $B = \gamma(\mathcal{B})$ die Menge aller Gödelnummern $\gamma(M)$ von Termen M, die keine β-Normalform haben. Beachte: $A, B \subseteq \mathbb{N}$ und $A \cap B = \emptyset$.

Unser Problem ist nun, einen Algorithmus zu finden, der für Zahlen $n \in A \cup B$ entscheidet, ob sie in A oder in B liegen. Legt man die Churchsche These zugrunde, daß die berechenbaren (totalen) Funktionen genau die rekursiven sind, kann man unser Entscheidungsproblem so umformulieren:

Gibt es eine (totale) rekursive Funktion $f\colon \mathbb{N} \to \mathbb{N}$ mit

$$f(n) = \begin{cases} 1 & \text{falls } n \in A \\ 0 & \text{falls } n \in B \end{cases} \quad ?$$

(Warum lassen wir nur totale rekursive Funktionen zu? Ob eine Zahl einen λ-Term darstellt, ist ja entscheidbar, d.h. es gibt eine rekursive Funktion f auf $\mathbb{N}$ mit $f(n) = 1$, falls $n \in A \cup B$, und $f(n) = 0$ sonst. Wenn man also eine partiell rekursive Funktion hat, die auf $A \cup B$ definiert ist und Elemente von A von denen von B trennt, dann kann mann diese zu einer totalen rekursiven Funktion mit derselben Eigenschaft machen, indem man sie außerhalb von $A \cup B$ modifiziert und dort $= 0$ setzt.)

Nun können wir unsere natürlichen Zahlen n im λ-Kalkül durch ihre Churchzahlen $\overline{n}$ darstellen, und nach Abschnitt 5 gibt es zu jeder rekursiven Funktion $f\colon \mathbb{N} \to \mathbb{N}$ einen λ-Term F, so daß $F\overline{n} \simeq_\beta \overline{f(n)}$ für alle $n \in \mathbb{N}$. Setzen wir nun

$$\ulcorner M \urcorner :\equiv \overline{\gamma(M)}$$

für jeden λ-Term M, so läßt sich unser Problem wieder in den λ-Kalkül zurückübersetzen:

Gibt es einen λ-Term H, so daß

$$(**) \qquad H\ulcorner M \urcorner \simeq_\beta \begin{cases} \overline{1} & \text{falls } M \text{ eine } \beta\text{-Normalform hat} \\ \overline{0} & \text{falls } M \text{ keine } \beta\text{-Normalform hat} \end{cases}$$

Wir wollen zeigen, daß es einen solchen Term nicht geben kann, womit dann Satz 6.1 bewiesen wäre. Dazu erst ein Hilfssatz:

6.4 Hilfssatz. Sei H ein λ-Term. Dann gibt es einen λ-Term W derart, daß für vorgegebene $P, Q \in \Lambda$ gilt:

$$(***) \qquad W \simeq_\beta \mathsf{COND}\,\big(\mathsf{ISNULL}(H\ulcorner W \urcorner)\big)PQ$$

Beweis. Wir betrachten die nach 6.3 rekursiven Funktionen

$$n \mapsto \gamma(\overline{n})\colon \mathbb{N} \to \mathbb{N} \qquad \text{und} \qquad \tau\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$

mit $\gamma\big((MN)\big) = \tau\big(\gamma(M), \gamma(N)\big)$. Nach dem Hauptsatz aus Abschnitt 5 können wir Terme S und T finden, so daß

(1) $S\overline{n} \simeq_\beta \overline{\gamma(\overline{n})} \equiv \ulcorner \overline{n} \urcorner$ und $T\overline{n}\,\overline{m} \simeq_\beta \overline{\tau(n,m)}$,

und folglich

2. $T\ulcorner N \urcorner \ulcorner M \urcorner \simeq_\beta T\overline{\gamma(N)}\,\overline{\gamma(M)} \simeq_\beta \overline{\tau\big(\gamma(N), \gamma(M)\big)} \simeq_\beta \overline{\gamma\big((NM)\big)} = \ulcorner NM \urcorner$

Setze $M :\equiv \lambda y.\, \mathsf{COND}\,\big(\mathsf{ISNULL}(H(Ty(Sy)))\big)PQ$ und $W :\equiv M\ulcorner M \urcorner$. Dann gilt

$$W :\equiv M\ulcorner M \urcorner \to_\beta \mathsf{COND}\,\Big(\mathsf{ISNULL}\,\Big(H\,\underbrace{(T\ulcorner M \urcorner\,\underbrace{(S\ulcorner M \urcorner)}_{\simeq_\beta \ulcorner\ulcorner M \urcorner\urcorner})}_{\simeq_\beta \ulcorner M \ulcorner M \urcorner\urcorner \equiv \ulcorner W \urcorner}\Big)\Big)PQ$$

nach den obigen Eigenschaften der Terme S und T. $\square$

6.5 Satz. Es gibt keinen λ-Term H, der $(**)$ erfüllt.

Beweis. Nehmen wir an, es gäbe einen λ-Term H, der $(**)$ erfüllt. Sei I eine β-Normalform (z.B. $I \equiv \lambda x.x$) und Ω ein Term ohne β-Normalform (z.B. $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$). Nach der Hilfsbehauptung gibt es einen Term W mit

$$W \simeq_\beta \mathsf{COND}\,\big(\mathsf{ISNULL}(H\ulcorner W \urcorner)\big)I\Omega$$

Es folgt aus $(**)$:

$$W \simeq_\beta \begin{cases} \Omega & \text{falls } W \text{ eine } \beta\text{-Normalform hat} \\ I & \text{falls } W \text{ keine } \beta\text{-Normalform hat} \end{cases}$$

In jedem Fall ergibt sich also ein Widerspruch. □

Nun wollen wir Satz 6.1 verallgemeinern:

6.6 Definition. Zwei Mengen $A, B \subseteq \mathbb{N}$ werden *rekursiv trennbar* genannt, wenn es eine totale rekursive Funktion $f\colon \mathbb{N} \to \mathbb{N}$ gibt, so daß

$$f(n) = \begin{cases} 1 & \text{falls } n \in A \\ 0 & \text{falls } n \in B \end{cases}$$

Zwei Mengen $\mathcal{A}, \mathcal{B}$ von λ-Termen werden rekursiv trennbar genannt, falls ihre Gödelzahlen es sind.

6.7 Unentscheidbarkeitssatz von Scott-Curry. Je zwei nicht leere Mengen $\mathcal{A}, \mathcal{B}$ von λ-Termen, die bzgl. β-Äquivalenz saturiert sind, sind nicht rekursiv trennbar.

Dabei ist mit *saturiert* gemeint, daß $(M \in \mathcal{A},\ M \simeq_\beta N) \implies N \in \mathcal{A}$.

Beweis. Seien $\mathcal{A}$ und $\mathcal{B}$ zwei nicht leere, saturierte Mengen von λ-Termen. Nehmen wir an, $\mathcal{A}$ und $\mathcal{B}$ seien rekursiv trennbar. Dann gibt es eine rekursive Funktion $f\colon \mathbb{N} \to \mathbb{N}$, so daß

$$f(\gamma(M)) = \begin{cases} 1 & \text{falls } M \in \mathcal{A} \\ 0 & \text{falls } M \in \mathcal{B} \end{cases}$$

Nach dem Hauptsatz aus Abschnitt 5 gibt es einen λ-Term H, so daß $H\overline{n} \simeq_\beta \overline{f(n)}$ für alle $n \in \mathbb{N}$. Folglich

$$(**) \qquad H\ulcorner M\urcorner = \begin{cases} \overline{1} & \text{für alle } M \in \mathcal{A} \\ \overline{0} & \text{für alle } M \in \mathcal{B} \end{cases}$$

Wähle nun $P \in \mathcal{A}$, $Q \in \mathcal{B}$. Aufgrund der Hilfsbehauptung 6.4 gibt es einen Term W, so daß

$$(***) \qquad W \simeq_\beta \mathsf{COND}\,\big(\mathsf{ISNULL}(H\ulcorner W\urcorner)\big)PQ$$

(1) $H\ulcorner W\urcorner \simeq_\beta \overline{0}$. Dann folgt $W \simeq_\beta P$ aus $(***)$. Da $P \in \mathcal{A}$ und da $\mathcal{A}$ saturiert ist, folgt dann auch $W \in \mathcal{A}$ und folglich $H\ulcorner W\urcorner \simeq_\beta \overline{1}$ nach $(**)$. Widerspruch.

(2) $H\ulcorner W\urcorner \simeq_\beta \overline{n} \not\simeq_\beta \overline{0}$. Dann folgt $W \simeq_\beta Q$ aus $(***)$. Da $Q \in \mathcal{B}$ und da $\mathcal{B}$ saturiert ist, folgt dann auch $W \in \mathcal{B}$ und folglich $H\ulcorner W\urcorner \simeq_\beta \overline{0}$ nach $(**)$. Widerspruch.

In jedem Fall gelangt man also zu einem Widerspruch. Folglich kann es keinen Term H wie angenommen geben. □

Spezialfälle.

(1) $\mathcal{A}$ sei die Menge aller Terme mit β-Normalform, $\mathcal{B}$ die Menge aller Terme ohne eine solche. Dann sind $\mathcal{A}$ und $\mathcal{B}$ nicht leer und saturiert. Somit folgt 6.1 aus 6.7.

(2) $\mathcal{A}$ sei die β-Äquivalenzklasse eines festen Terms M und $\mathcal{B} = \Lambda \setminus \mathcal{A}$. Nach 6.7 sind $\mathcal{A}$ und $\mathcal{B}$ nicht rekursiv trennbar, d.h. es ist unentscheidbar, ob ein beliebig vorgegebener λ-Term N zu M β-äquivalent ist.

7. Der Satz von Böhm und die *leftmost* Reduktionsstrategie

Den Beweis des folgenden Satzes findet man in dem Buch von Krivine, S.67–72:

7.1 Satz von Böhm. Sind M und N geschlossene λ-Terme in $\beta\eta$-Normalform und gilt $M \not\simeq_{\beta\eta} N$, dann existieren geschlossene λ-Terme $L_1, \ldots, L_m$, so daß

$$\begin{aligned} ML_1 \ldots L_m xy &\simeq_\beta x \\ NL_1 \ldots L_m xy &\simeq_\beta y \end{aligned}$$

Wie schon in 4.9 gesagt, wird eine Gleichungstheorie wird *konsistent* genannt, wenn man aus den Axiomen und den Regeln der Theorie nicht herleiten kann, daß $P = Q$ für alle Terme P und Q gilt. Wir haben dort gesehen, daß der λ_β-Kalkül konsistent ist (denn je zwei verschiedene $\beta-$Normalformen sind nicht β-äquivalent). Die Frage ist nun, ob man zusätzliche Gleichungen zu dem $\lambda\beta-$Kalkül hinzunehmen kann, ohne die Konsistenz des Kalküls zu zerstören. Wir hatten schon gesehen, daß man die η-Gleichheit dazunehmen kann, die man durch zusätzliche Gleichung $y = \lambda x.yx$, ausdrücken kann (wobei angenommen ist, daß x und y verschiedene Variablen bezeichnen). Aus dem Satz von Böhm ergibt sich nun, daß das Hinzufügen zusätzlicher Gleichungen schnell zur Inkonsistenz führen kann:

7.2 Folgerung. Sind M und N geschlossene λ-Terme, die eine $\beta\eta$-Normalform besitzen, und gilt $M \not\simeq_{\beta\eta} N$, dann ist die Gleichung $M = N$ inkonsistent mit dem λ_β-Kalkül.

Beweis. Seien in der Tat M und N Terme mit einer $\beta\eta$-Normalform, die nicht $\beta\eta$-äquivalent sind. Dann gibt es nach dem Satz von Böhm Terme $L_1, \ldots, L_m$, so daß

$$\begin{aligned} ML_1 \ldots L_m xy &\simeq_\beta x \\ NL_1 \ldots L_m xy &\simeq_\beta y \end{aligned}$$

Nimmt man nun zusätzlich $M = N$ an, so folgt $x = y$. Da man für die Variablen x und y beliebige Terme substituieren kann, ohne die Gleichheit zu zerstören, folgt $P = Q$ für beliebige P, Q. □

Auf die Voraussetzung, daß die Terme eine $\beta\eta$-Normalform besitzen, kann man nicht verzichten, denn die Gleichung $\omega_2 \equiv (\lambda x.xx)(\lambda x.xx) = M$ ist mit dem $\lambda_\beta-$Kalkül für jeden Term M konsistent. Einen semantischen Beweis werden wir später sehen. Das folgende Beispiel zeigt, daß die Voraussetzungen des Satzes nicht notwendig sind (man bemerke, daß der Term $\omega_3 \equiv \delta_3\delta_3$ keine Normalform hat):

Behauptung. Die Gleichung $\delta_3\delta_3 = \mathsf{I}$ mit $\delta_3 \equiv \lambda x.xxx$ ist inkonsistent mit dem λ_β-Kalkül.

Beweis. Wegen $\delta_3\delta_3 \simeq_\beta (\delta_3\delta_3)\delta_3$, folgt aus der Gleichung $\mathsf{I} = \delta_3\delta_3$ die Gleichung $\mathsf{I} = \mathsf{I}\,\delta_3 \simeq_\beta \delta_3$, also $\mathsf{I} = \delta_3$. Da I und δ_3 zwei verschiedene $\beta\eta$-Normalformen sind, ist diese Gleichung inkonsistent mit dem λ_β-Kalkül nach Folgerung 7.2. □

Nun wenden wir uns einer anderen Frage zu, nämlich der Frage nach einer Reduktionsstrategie, die für einen beliebig vorgegebenen Term die β-Normalform liefert, sofern eine solche existiert. Zuvor brauchen wir einige Bezeichnungen für spezielle Arten von Reduktionen.

7.3 Kopfnormalform. Wie wir in Übungsaufgabe 8(b) gesehen haben, hat jeder λ-Term die Gestalt

$$\text{(KNF)} \qquad \lambda x_1 \dots x_n.xM_1 \dots M_m$$

oder

$$(\neg\text{KNF}) \qquad \lambda x_1 \dots x_n.RM_1 \dots M_m \quad \text{mit } R \equiv (\lambda x.M)M_0$$

wobei solch eine Darstellung eindeutig ist. Sowohl n als auch m können dabei den Wert 0 annehmen; d.h. die Abstraktionen $\lambda x_1 \dots x_n.$ am Anfang des Terms und ebenso die $M_1 \dots M_m$ am Ende des Terms brauchen nicht vorhanden zu sein.

Kopfreduktion und innere Reduktion. Der Teilterm R in ($\neg$KNF) heißt *Kopfredex* des Terms, jeder andere Redex heißt *innerer Redex*. Eine *Kopfkontraktion* $M \to_k N$ kommt zustande durch eine Kontraktion des Kopfredex von M. Eine Folge $M \equiv M_0 \to_k M_1 \to_k \cdots \to_k M_n \equiv N$ von Kopfkontraktionen heiße *Kopfreduktion* und wird mit $M \to_k^* N$ abgekürzt, wobei auch hier $n = 0$ zulässig ist. Analog werden mit $M \to_i N$ und $M \to_i^* N$ *innere Kontraktionen* und *innere Reduktionen* bezeichnet.

Man sagt, ein λ-Term sei in *Kopfnormalform*, wenn er die Gestalt (KNF) hat. Gibt es eine β-Reduktion $M \to_\beta^* N$ von M in einen Term N in Kopfnormalform, dann wird N eine Kopfnormalform von M genannt, und man sagt, der Term M habe eine Kopfnormalform.

Man beachte jedoch, daß die Kopfnormalform eines Terms, sofern es eine solche gibt, nicht eindeutigzu sein braucht. Ein Term kann mehrere Kopfnormalformen haben, wie der Fixpunktkombinator in den nachfolgenden Beispielen zeigt.

Beispiel.

(1) $\omega_2 \equiv (\lambda x.xx)(\lambda x.xx) \to_k \omega_2 \to_k \dots$ hat keine Kopfnormalform.

(2) Der Fixpunktkombinator

$$Y_C \equiv \lambda f.\, \underbrace{(\lambda x.f(xx))\, \underbrace{(\lambda x.f(xx))}_{V}}_{R}$$

ist keine KNF. Führt man jedoch eine Kopfkontraktion durch, so erhält man $\lambda f.f(VV)$ – eine Kopfnormalform. Eine weitere Kopfkontraktion liefert wieder eine Kopfnormalform, nämlich $\lambda f.f(f(VV))$.

Bemerkungen. (1) Ein Term $M \cong \lambda x_1 \dots x_n.xM_1 \dots M_m$ in Kopfnormalform erlaubt nur innere Reduktionen und das Ergebnis ist wiederum ein Term $N \cong \lambda x_1 \dots x_n.xN_1 \dots N_m$ in Kopfnormalform mit $M_1 \to_\beta^* N_1, \dots, M_m \to_\beta^* N_m$.

(2) Ist M ein Term, der nicht in Kopfnormalform ist, also $M \cong \lambda x_1 \ldots x_n.RM_1 \ldots M_m$ mit $R \equiv (\lambda x.M)M_0$, und führt man hier nur innere Reduktionen aus, so bleibt der Kopfredex R erhalten und es entstehen nur Nicht-Kopfnormalformen der Gestalt $N \cong \lambda x_1 \ldots x_n.RM_1 \ldots M_m$ mit $M_1 \to_\beta^* N_1, \ldots, M_m \to_\beta^* N_m$.

(3) Entsteht umgekehrt ein Term $N \cong \lambda x_1 \ldots x_n.xN_1 \ldots N_m$ in Kopfnormalform aus einem Term M durch interne Reduktionen so muß M ebenfalls eine Kopfnormalform der Gestalt $M \cong \lambda x_1 \ldots x_n.xM_1 \ldots M_m$ mit $M_1 \to_\beta^* N_1, \ldots, M_m \to_\beta^* N_m$ gewesen sein.

7.4 Leftmost reduction. Der am weitesten links beginnende Redex in einem λ-Term M wird *leftmost-Redex* von M genannt. Schreibe $M \to_\ell M'$, wenn M' aus M durch Kontraktion des leftmost-Redex hervorgeht. Ein gegebener Term M hat genau eine *leftmost-Reduktionskette* $M \to_\ell M_1 \to_\ell \cdots \to_\ell M_n \to_\ell \ldots$, diese Kette kann endlich oder unendlich sein. Eine endliche Reduktionskette wird wie üblich mit $M \to_\ell^* M_n$ notiert. Eine Kopfreduktion ist leftmost, aber nicht umgekehrt.

Die folgenden Sätze besagen, daß man eine KNF eines Terms durch die Strategie der Kopfreduktionen findet und daß man eine β-Normalform durch die *leftmost*-Reduktionsstrategie findet, sofern es solche Normalformen gibt.

7.5 Satz. Wenn der λ-Term M eine Kopfnormalform besitzt, dann erreicht man eine solche durch Kopfreduktionen, d.h., es gibt eine Kopfnormalform P mit $M \to_k^* P$.

7.6 Satz. Hat ein λ-Term M eine β-Normalform N, so erreicht man diese durch leftmost reduction, d.h. $M \to_\ell^* N$.

Beide Sätze folgen aus dem folgenden Satz:

7.7 Satz (Mitschke, Darmstadt 1978). Wenn $M \to_\beta^* N$, dann existiert ein P, so daß $M \to_k^* P \to_i^* N$.

Beweis von 7.5. Sei $M \to_\beta^* N$ mit N in Kopfnormalform. Nach 7.7 existiert ein P mit $M \to_k^* P \to_i^* M$. Der Term P ist schon in Kopfnormalform nach Bemerkung (3) am Ende von 7. □

Beweis von 7.6. Wir führen den Beweis durch Induktion über die Länge des Terms N und nehmen an die Behauptung sei schon bewiesen für alle Terme, die eine β-Normalform haben, deren Länge kleiner ist als die Länge von N. Sei nun $M \to_\beta^* N$ mit N in β-Normalform. Insbesondere ist dann N in Kopfnormalform, d.h. von der Form $N \equiv \lambda x_1 \ldots x_m.xN_1 \ldots N_n$; außerdem müssen dann auch $N_1, \ldots, N_n$ in β-Normalform sein. Nach 7.7 existiert ein P mit $M \to_k^* P \to_i^* N \equiv \lambda x_1 \ldots x_m.xN_1 \ldots N_n$. Aus der Bemerkung (3) in 7 schließen wir, daß $P \equiv \lambda x_1 \ldots x_m.xP_1 \ldots P_n$ mit $P_i \to_\beta^* N_i$ für alle i. Nach Induktionsannahme haben wir $P_i \to_\ell^* N_i$ für alle i. Also $M \to_k^* P \equiv \lambda x_1 \ldots x_m.xP_1P_2 \ldots P_n \to_\ell^* \lambda x_1 \ldots x_m.xN_1P_2 \ldots P_n \to_\ell^* \cdots \to_\ell^* \lambda x_1 \ldots x_m.xN_1N_2 \ldots N_n \equiv N$. □

Für den Beweis von 7.7 benutzen wir den Begriff der parallelen Reduktion $M \rightrightarrows N$ aus 3.2 und wir Schreiben $M \rightrightarrows_i N$, falls $M \rightrightarrows N$ und $M \rightarrow_i^* N$. Wir benötigen wir noch den folgenden Hilfssatz, dessen langwierigen Beweis durch Induktion über den Aufbau von M wir nicht bringen (siehe Krivine, S. 63–67):

7.8 Hilfssatz. Wenn $M \rightrightarrows N$, dann existiert ein P mit $M \rightarrow_k^* P \rightrightarrows_i N$.

7.9 Folgerung. Wenn $M \rightrightarrows_i P \rightarrow_k N$, dann existiert ein Q mit $M \rightarrow_k^* Q \rightrightarrows_i N$.

Beweis von 7.7. Sei $M \rightarrow_\beta^* N$, d.h. es gibt eine Reduktionskette

$$M \rightarrow_\beta M_1 \rightarrow_\beta M_2 \rightarrow_\beta \ldots \rightarrow_\beta M_n \equiv N$$

Diese β-Kontraktionen können natürlich durch parallele Reduktionen ersetzt werden, es gilt also

$$M \rightrightarrows M_1 \rightrightarrows M_2 \rightrightarrows \ldots \rightrightarrows M_n \equiv N$$

Nach 7.8 existieren $P_1, \ldots, P_n$, so daß

$$M \rightarrow_k^* P_1 \rightrightarrows_i M_1 \rightarrow_k^* P_2 \rightrightarrows_i M_2 \cdots \rightarrow_k^* P_n \rightrightarrows_i M_n \equiv N$$

Nach 7.9 kann man die inneren parallelen Reduktionen mit den Kopfreduktionen vertauschen. Faßt man dann die linksstehenden Kopfreduktionen und die rechtsstehenden inneren Reduktionen zusammen, so folgt die Behauptung von 7.7. $\square$

KAPITEL II

Kombinatorische Logik

Die kombinatorische Logik ist eine Alternative zum Lambda-Kalkül. Wie wir sehen werden, kann man sowohl die kombinatorische Logik im Lambda-Kalkül interpretieren, als auch den Lambda-Kalkül in der kombinatorischen Logik simulieren. Allerdings sind die beiden formalen System nicht ganz gleichwertig. Die kombinatorische Logik ist von einfacherer Struktur wie der Lambda-Kalkül und daher leichter zu implementieren. Allerdings ist der funktionale Charakter des Lambda-Kalküls in der kombinatorischen Logik weniger sichtbar.

Die kombinatorische Logik wurde von Schönfinkel in den 20er Jahren des letzten Jahrhunderts schon vor dem Lambda-Kalkül eingeführt (siehe Literaturverzeichnis). In dem Buch von Hindley und Seldin (Literaturverzeichnis) wird die kombinatorische Logik parallel zum Lambda-Kalkül entwickelt. Auch in den Büchern von Barendregt und Krivine findet man Kapitel darüber. Als Kurzinformation kann der Artikel von Roman E. Maeder: Combinators, *The Mathematica Programmer*, vol.5, pp. 35–41, dienen, der auch ueber das WWW erhältlich ist.

0.10 Motivation. Im Lambda-Kalkül spielen die λ-Terme

$$\mathsf{S} \equiv \lambda xyz.xz(yz) \qquad \mathsf{K} \equiv \lambda xy.x$$

eine herausragende Rolle. Sie haben die Eigenschaft, daß für beliebige λ-Terme M, N, P gilt:

$$\mathsf{S}\, MNP \to_\beta^* MP(NP) \qquad \mathsf{K}\, MN \to_\beta^* M\,.$$

Die kombinatorische Logik ist nun ein formales System, das das Arbeiten mit den Kombinatoren S und K formalisiert und auf den beiden eben genannten Reduktionsregeln aufbaut.

0.11 Die Syntax der kombinatorischen Logik. Wir beginnen mit

- abzählbar vielen *Variablen*, die wir mit $x, y, z, \ldots$ bezeichnen, und
- zwei *Konstanten* s, k.

Terme der kombinatorischen Logik (kurz *CL-Terme*) sind die Ausdrücke, die man in endlich vielen Schritten nach folgenden Regeln aufbauen kann:

(1) Die Variablen $x, y, z, \ldots$ und die Konstanten s und k sind CL-Terme (*atomare* CL-Terme).

(2) Sind M und N CL-Terme, dann ist auch (MN) ein CL-Term (*Applikation*).

Wie im Lambda-Kalkül bezeichne $M \equiv N$ die syntaktische Gleichheit von CL-Termen und wir übernehmen die Klammerkonvention

$$M_1 M_2 M_3 \ldots M_k \equiv (\cdots((M_1 M_2) M_3) \cdots M_k)\,,$$

daß ungeklammerte Ausdrücke als von links her geklammert zu lesen sind.

Wir bezeichnen mit $FV(M)$ die Menge der in dem CL-Term M auftretenden Variablen. Alle Variablen sind frei, denn in der kombinatorischen Logik gib es keine gebundenen Variablen.

Ein CL-Term, der keine freien Variablen enthält, wird *geschlossener CL-Term* oder auch *Kombinator* genannt. Ein solcher Term ist nur aus s und k zusammengesetzt.

0.12 Schwache Reduktion von CL-Termen. Als *schwache Redexe* bezeichnen wir alle CL-Terme der Gestalt

$$\mathsf{s}\,MNP \qquad \text{und} \qquad \mathsf{k}\,MN\,,$$

wobei M und N beliebig CL-Terme sein dürfen. Motiviert durch 0.10 erlauben wir für schwache Redexe die folgenden *schwachen Kontraktionen*

$$\mathsf{s}\,MNP \to_w MP(NP) \qquad \text{und} \qquad \mathsf{k}\,MN \to_w M\,,$$

und wir schreiben $M \to_w N$, wenn der CL-Term N aus M durch schwache Kontraktion eines in M als Teilterm enthaltenen schwachen Redexes hervorgeht.

Eine endliche Kette schwacher Kontraktionen nennen wir eine *schwache Reduktion* und, wenn es eine Kontraktionskette

$$M \equiv M_0 \to_w M_1 \to_w M_2 \to_w \ldots \to_w M_k \equiv N$$

gibt, dann schreiben wir

$$M \to_w^* N\,.$$

0.13 Substitution. Da wir in der kombinatorischen Logik keine gebundenen Varaiblen haben, ist die Substitution unproblematisch. Sind M und P CL-Terme, dann sei $M[x := P]$ der CL-Term, den man aus M erhält, indem man an jeder Stelle in M, an der die Variable x auftritt, diese durch den Term P ersetzt. Daß hierdurch wieder ein CL-Term entsteht, sieht man am einfachsten über die rekursive Definition der Substitution:

- Atomare Terme: $x[x := P] \equiv P$
  $y[x := P] \equiv y$, falls $x \neq y$,
  $\mathsf{s}[x := P] \equiv \mathsf{s}$, $\mathsf{k}[x := P] \equiv \mathsf{k}$.
- $M \equiv (UV)$: $M[x := P] \equiv U[x := P]V[x := P]$.

Ganz analog definiert man die simultane Substitution $M[x_1 := P_1, \ldots, x_n := P_n]$.

0.14 Lemma. Die schwache Reduktion von CL-Termen ist substitutionsverträglich, d.h.,

$$M \to_w^* N \quad \Longrightarrow \quad M[x := P] \to_w^* N[x := P] \quad \text{und} \quad P \to_w^* Q \quad \Longrightarrow \quad M[x := P] \to_w^* M[x := Q]\,.$$

Beweis durch Induktion über den Termaufbau (Übung).

0.15 Schwache Normalformen. Ein CL-Term, der keinen schwachen Redex als Teilterm besitzt, wird *schwache Normalform* genannt. Wenn man einen CL-Term M zu einer schwachen Normalform N reduzieren kann, so wird N schwache Normalform von M geanannt.

Ähnlich wie im Lambda-Kalkül gibt es CL-Terme, die keine schwache Normalform haben.

0.16 Satz von Church-Rosser. Die schwache Reduktion von CL-Termen ist konfluent.

Man kann nun ähnliche Folgerungen wie für die β-Reduktion im Lambda-Kalkül ziehen. Insbesondere kann man eine schwache Äquivalenzrelation für CL-terme dadurch definieren, daß man zwei Terme M und N schwach äquivalent nennt, wenn Sie durch schwache Reduktionen zu ein und demselben Term P reduziert werden können. Wir schreiben dann $M \cong_w N$.

Übung. Sei

$$\mathsf{b} :\equiv \mathsf{s}(\mathsf{k}\,\mathsf{s})\,\mathsf{k} \qquad \text{und} \qquad \mathsf{c} :\equiv \mathsf{s}(\mathsf{b}\,\mathsf{b}\,\mathsf{s})(\mathsf{k}\,\mathsf{k})\,.$$

Finde die schwachen Normalformen von $\mathsf{b}\,xyz$ und von $\mathsf{c}\,xyz$.

0.17 Übersetzung der kombinatorischen Logik in den Lambda-Kalkül. Es ist ganz einfach, die kombinatorische Logik im Lambda-Kalkül zu interpretieren. Für jeden Term M der kombinatorischen Logik, erklären wir einen Term M_λ des Lambda-Kalküls rekursiv über den Termaufbau. Dabei nehmen wir an, daß wir in der kombinatorischen Logik dieselben Variablen verwende wie im Lambda-Kalkül:

- Atomare CL-Terme: Ist x eine Variable so sei $x_\lambda \equiv x$;
  die CL-Konstanten s und k übersetzen wir durch
  $\mathsf{s}_\lambda \equiv \mathsf{S} \equiv \lambda xyz.xz(yz)$ und $\mathsf{k}_\lambda \equiv \mathsf{K} \equiv \lambda xy.x$.
- Ist $M \equiv (UV)$ so übersetzen wir $M_\lambda \equiv U_\lambda V_\lambda$.

Die schwache Reduktion vn CL-Termenwar nun gerade so definiert, daß für beliebige CL-Terme M und N gilt:

$$M \to_w^* N \quad \Longrightarrow \quad M \to_\beta^* N \quad \text{und } M \equiv_w N \quad \Longrightarrow \quad M \equiv_\beta N\,.$$

Bevor wir umgekehrt den Lambda-Kalkül in die kombinatorische Logik übersetzen können, müssen wir zuerst die β-Kontraktion in der kombinatorischen Logik simulieren:

0.18 Simulation der Lambda-Abstraktion in der kombinatorischen Logik. Jedem CL-Term M ordnen wir nun einen anderen CL-Term $\lambda^*x.M$ zu durch eine rekursive Definition über den Termaufbau:

(1) $\lambda^*x.M \equiv \mathsf{k}\,M$ falls die Variable x in M nicht vorkommt

(2) $\lambda^*x.x \equiv \mathsf{i} :\equiv \mathsf{s}\,\mathsf{k}\,\mathsf{k}$

(3) $\lambda^*x.Ux \equiv U$ falls die Variable x in U nicht vorkommt

(4) $\lambda^*x.UV \equiv \mathsf{s}(\lambda^*x.U)(\lambda^*x.V)$ falls (1) und (2) nicht zutreffen

Der folgende Hilfssatz zeigt, daß wir tatsächlich die β-Reduktion in der kombinatorischen Logik simulieren können:

0.19 Lemma. Seien M und P beliebige CL-Terme. Dann kommt x in dem CL-Term $\lambda^*x.M$ nicht mehr vor und es gilt

$$(\lambda^*x.M)P \to_w^* M[x := P]\,.$$

Beweis. Statt der angegebenen Reduktion genügt es zu zeigen

$$(\lambda^*x.M)x \to_w^* M\,;$$

denn macht man auf beiden Seiten die Substitution $x := P$, so ergibt sich die gewünschte Behauptung aus dem Substitutionslemma 0.13. Dabei benutzt man, daß x in $\lambda^*x.M$ nicht vorkommt. Die vereinfachte Behauptung beweist man nun durch Induktion über den Termaufbau. $\square$

Nun können wir ein Übersetzung des Lambda-Kalküls in die kombinatorische Logik angeben:

0.20. Für Lambda-Terme M definieren wir rekursiv über den Termaufbau eine Übersetzung in einen CL-Term M_{CL}, wobe wir daran erinnern, daß wir in beiden Kalkülen dieselben Variablen verwenden:

- Für eine Variable x sei $x_{CL} = x$.
- Für $M \cong \lambda x.U$ sei $M_{CL} \cong \lambda^*x.(M_{CL}$.
- Für $M \cong UV$, sei $M_{CL} \cong U_{CL}V_{CL}$.

Beispiel. Wir übersetzen den Lambda-Term $\mathsf{W} \cong \lambda xy.x(yy)$ in die kombinatorische Logik. Zunächst haben wir $\mathsf{W}_\lambda \cong \lambda^*x.\lambda^*y.x(yy)$ und dies führt zu:

$$\begin{array}{rcl}
\lambda^*y.x(yy) & \cong & \mathsf{s}(\lambda^*y.x)(\lambda^*y.yy) \\
 & \cong & \mathsf{s}(\mathsf{k}\,x)\big(\mathsf{s}(\lambda^*y.y)(\lambda^*y.y)\big) \\
 & \cong & \mathsf{s}(Kombkx)(\mathsf{s\,i\,i}) \\
\lambda^*x\lambda^*y.x(yy) & \cong & \lambda^*x.\,\mathsf{s}(Kombkx)(\mathsf{s\,i\,i}) \\
 & \cong & \mathsf{s}\,\big(\lambda^*x.\,\mathsf{s}(Kombkx)\big)(\lambda^*x.\,\mathsf{s\,i\,i}) \\
 & \cong & \mathsf{s}\,\big(\mathsf{s}(\lambda^*x.\,\mathsf{s})(\lambda^*x.Kombkx)\big)\big(\mathsf{k}(\mathsf{s\,i\,i})\big) \\
 & \cong & \mathsf{s}\,\big(\mathsf{s}(\mathsf{k\,s})\,\mathsf{k}\,\big)\big(\mathsf{k}(\mathsf{s\,i\,i})\big)
\end{array}$$

Der übersetzte Term ist nur aus den Kombinatoren s und k aufgebaut, allerdings viel länger als der ursprüngliche Term. Wir stellen nun fest, daß $\mathsf{W}_{CL}\,xy$ schwach zu $x(yy)$ reduziert:

$$\begin{array}{rcl}
\mathsf{s}\,\big(\mathsf{s}(\mathsf{k\,s})\,\mathsf{k}\,\big)\big(\mathsf{k}(\mathsf{s\,i\,i})\big)xy & \to_w & \mathsf{s}(\mathsf{k\,s})\,\mathsf{k}\,x\big(\mathsf{k}(\mathsf{s\,i\,i})x\big)y \\
 & \to_w & \mathsf{k\,s}\,x(\mathsf{k}\,x)\big(\mathsf{k}(\mathsf{s\,i\,i})x\big)y \\
 & \to_w & \mathsf{s}(\mathsf{k}\,x)\big(\mathsf{k}(\mathsf{s\,i\,i})x\big)y \\
 & \to_w & \mathsf{k}\,xy\big(\mathsf{k}(\mathsf{s\,i\,i})xy\big) \\
 & \to_w & x(\mathsf{s\,i\,i}\,y) \to_w x(\mathsf{i}\,y(\mathsf{i}\,y)) \to_w x(yy)
\end{array}$$

Übersetzen wir dies zurück in den Lambda-Kalkül, so erhalten wir:

$$\mathsf{W}_{CL\lambda} \cong \mathsf{S}\big(\,\mathsf{S}(\mathsf{K}\,\mathsf{S})\,\mathsf{K}\,\big)\big(\,\mathsf{K}(\mathsf{S}\,\mathsf{I}\,\mathsf{I})\big)\,.$$

Der rückübersetzte Term ist nur aus den Kombinatoren S und K, angewendet auf x, y hat er aber dieselbe Wirkung wie W, das heißt, die beiden Terme sind $\beta\eta$-äquivalent.

Allgemein gilt nun, daß unser Verfahren erlaubt, durch Übersetzung in die kombinatorische Logik und Rückübersetzung jedem Lambda-Term M einen $\beta\eta$-äquivalenten Lambda-Term $M_{CL\lambda}$ zuzuordnen, der nur aus aus den Kombinatoren S und K sowie den freie Variablenvon M aufgebaut ist:

0.21 Satz. Für jeden Lambda-Term M gilt $M_{CL\lambda} \simeq_{\beta\eta} M$.

Diese Tatsache wird ausgenutzt zum Kompilieren von Lambda-Termen.

KAPITEL III

Der einfach typisierte λ-Kalkül

1. Ein einfaches Typsystem

Wir knüpfen nun an die Einleitung an, in der wir ausgehend von Grundtypen $\alpha_1, \alpha_2, \ldots$, die wir uns durch Grundmengen $D_{\alpha_1}, D_{\alpha_2}, \ldots$ interpretiert denken, Typen $\sigma \to \tau$ von Funktionen höherer Ordnung betrachteten, die durch die Menge $D_{\sigma\to\tau} = (D_\sigma \to D_\tau)$ aller Funktion von D_σ in D_τ interpretiert werden.

1.1 Einfache Typen. Wir geben eine endliche oder abzählbare Menge $\mathcal{A}$ von *atomaren Typen* $\alpha_1, \alpha_2, \ldots$ vor und definieren induktiv die Menge Σ aller einfachen Typen:

(1) Jeder atomare Typ α ist ein Typ.

(2) Sind σ und τ Typen, dann ist auch $(\sigma \to \tau)$ ein Typ.

Die so definierten Typen nennen wir auch *einfache Typen.* $\sigma \equiv \tau$ bezeichne die syntaktische Gleichheit von Typen.

Klammerkonvention. Man vereinbart

$$\sigma_1 \to \sigma_2 \to \cdots \to \sigma_n \to \tau :\equiv (\sigma_1 \to (\sigma_2 \to (\cdots \to (\sigma_n \to \tau)\ldots)))$$

Manchmal schreibt man hierfür auch

$$\sigma_1, \sigma_2, \ldots, \sigma_n \to \tau$$

Diese Konvention wird oft auch etwas schlapsig damit umschrieben, daß Typen rechtsassoziativ seien.

Bemerkung. Jeder Typ τ läßt sich in eindeutiger Weise in der Form

$$\tau \equiv \sigma_1 \to \sigma_2 \to \cdots \to \sigma_n \to \alpha$$

mit $n \geq 0$ schreiben, wobei α ein atomarer Typ ist.

Beweis durch Induktion über den Aufbau des Typs τ.

- Ist $\tau \equiv \alpha$ atomar, so gilt die Behauptung trivialerweise mit $n = 0$.
- Ansonsten ist $\tau \equiv \sigma_1 \to \tau_1$, wobei die Typen σ_1 und τ_1 eindeutig bestimmt sind. Nach Induktionsvoraussetzung ist $\tau_1 \equiv \sigma_2 \to \ldots \to \sigma_n \to \alpha$ für eindeutig bestimmte $\sigma_2, \ldots \sigma_n, \alpha$. $\square$

1.2 Länge eines Typs. Als *Länge* eines Typs σ bezeichnen wir die Anzahl $\ell(\sigma)$ der in σ vorkommenden atomaren Typen, wobei jeder atomare Typ mit seiner Vielfachheit gezählt wird. Zum Beispiel ist $\ell(\alpha \to \alpha \to \beta) = 3$. Rekursiv über den Typaufbau kann man die Länge eines Typs so definieren:

(1) $\ell(\alpha) = 1$,

(2) $\ell(\sigma \to \tau) = \ell(\sigma) + \ell(\tau)$.

1.3 Ordnung eines Typs. Wir definieren induktiv die Ordnung $\mathrm{ord}(\tau)$ eines Typs:

- Die Ordnung eines atomaren Typs $\tau \equiv \alpha$ sei $\mathrm{ord}(\alpha) = 0$.
- Ist τ nicht atomar, dann hat τ nach der vorhergehenden Bemerkung eine eindeutige Darstellung der Form $\tau \equiv \sigma_1 \to \sigma_2 \to \cdots \to \sigma_n \to \alpha$ und wir definieren
$$\mathrm{ord}(\tau) = \max_i \mathrm{ord}(\sigma_i) + 1 .$$

Beispielsweise sind $\alpha \to \alpha$, $\alpha \to \alpha \to \alpha$ Typen der Ordnung 1, $(\alpha \to \alpha) \to \alpha$, $(\alpha \to \alpha) \to \alpha \to \alpha$ Typen der Ordnung 2, usw.

In diesem Kapitel behalten wir stets dieses einfache Typsystem bei.

2. Einfach getypte λ-Terme (à là Church)

2.1 Einfach getypte λ-Terme. Für jeden Typ σ geben wir uns abzählbar viele Variablen $v_1^\sigma, v_2^\sigma, v_3^\sigma, \ldots$ vor, die wir auch mit $x^\sigma, y^\sigma, \ldots$ bezeichnen.

Ein getypter λ-Term ist nun ein λ-Term M, der mit seinem Typ versehen ist. Wir schreiben $M : \sigma$ und sagen „M ist ein Term vom Typ σ“.

Getypte Terme werden induktiv nach folgenden Regeln aufgebaut, siehe auch Abbildung 1:

(1) Jede Variable x^σ ist ein Term vom Typ σ.

(2) Für jede Variable x^σ und jeden Term M vom Typ τ ist $(\lambda x^\sigma.M)$ ein Term vom Typ $\sigma \to \tau$.

(3) Ist M ein Term vom Typ $\sigma \to \tau$ und N ein Term vom Typ σ, dann ist (MN) ein Term vom Typ τ.

Mit $s \equiv t$ wird die syntaktische Gleichheit getypter Terme bezeichnet. Für getypte Terme übernehmen wir die Klammerkonventionen wie im ungetypten Fall.

$$\frac{}{x^\sigma : \sigma}\,(1) \qquad \frac{M : \tau}{(\lambda x^\sigma.M) : \sigma \to \tau}\,(2) \qquad \frac{M : \sigma \to \tau \qquad N : \sigma}{(MN) : \tau}\,(3)$$

Abbildung 1. Induktiver Aufbau getypter λ-Terme nach Church

Beispiel.

$$\begin{aligned}
\mathsf{I}_\sigma &\equiv \lambda x^\sigma.x^\sigma : \sigma \to \sigma \\
\mathsf{K}_{\sigma,\tau} &\equiv \lambda x^\sigma y^\tau.x^\sigma : \sigma \to \tau \to \sigma \\
& \lambda x^\sigma y^\tau.y^\tau : \sigma \to \tau \to \tau \\
\mathsf{COMP}_{\varrho,\sigma,\tau} &\equiv \lambda x^{\sigma\to\tau} y^{\varrho\to\sigma} z^\varrho.x^{\sigma\to\tau}(y^{\varrho\to\sigma} z^\varrho) : (\sigma \to \tau) \to (\varrho \to \sigma) \to \varrho \to \tau
\end{aligned}$$

Dagegen lassen sich der Term xx und verwandte Terme wie $\lambda x.xx$, Y_C und Y_T in keiner Weise als einfach getypte Terme auffassen. Dies liegt daran, daß eine Variable nicht gleichzeitig einen Typ σ und einen Typ $\sigma \to \tau$ haben kann.

2.2 Substitution. Die Substitution $M[x^\sigma := N]$ ist für getypte Terme genauso definiert wie im ungetypten Fall, allerdings mit der Einschränkung, daß für eine Variable x^σ vom Typ σ nur ein Term N von demselben Typ σ substituiert werden darf. Beachte, daß M und $M[x^\sigma := N]$ denselben Typ haben. Natürlich dürfen auch bei eventuell notwendigen Umbenennungen gebundener Variabler nur solche vom selben Typ genommen werden.

2.3 β-Reduktion und β-Äquivalenz. Diese werden ebenso wie im ungetypten Fall definiert. Man bemerke, daß eine β-Kontraktion $(\lambda x^\sigma.M)N \to_\beta M[x^\sigma := N]$ den Typ nicht verändert. Denn ist M vom Typ τ, dann ist $\lambda x^\sigma.M$ vom Typ $\sigma \to \tau$. Also muß N vom Typ σ sein, und $(\lambda x^\sigma.M)N$ ist dann vom Typ τ, ebenso wie $M[x^\sigma := N]$.

Es folgt, daß sich bei beliebigen β-Reduktionen der Typ nicht verändert und daß β-äquivalente getypte Terme immer von demselben Typ sein müssen.

Da β-Reduktionen im getypten Fall genau so vor sich gehen wie im ungetypten Fall, gilt auch hier der Satz von Church-Rosser.

Bemerkung. Auch η-Kontraktionen $\lambda x^\sigma.Mx^\sigma \to_\eta M$ ändern den Typ nicht.

Im einfach getypten Fall kann man weniger Terme als im ungetypten Fall bilden. Dafür gewinnt man aber, daß jeder einfach getypte Term eine β-Normalform besitzt. Wie sich später herausstellen wird, gilt sogar noch mehr.

2.4 Definition. Ein Term M (getypt oder ungetypt) wird *stark normalisierbend* genannt, wenn er keine unendliche β-Reduktionskette besitzt, d.h. wenn man mit jeder Reduktionsstrategie nach endlich vielen Schritten zu einer β-Normalform gelangt.

Bemerkung. Es gilt sogar: Ist M stark normalisierend, so gibt es eine natürliche Zahl n_0, so daß jede β-Reduktionskette aus höchstens n_0 β-Kontraktionen besteht, d.h., gleichgültig wie man vorgeht, nach spätestens n_0 Kontraktionsschritten ist man bei der β-Normalform angelangt. (Dies ist ein Spezialfall des Lemmas von König aus der Graphentheorie.)

Beweis. Nehmen wir an, jede β-Reduktionskette von M sei endlich, aber es gäbe Reduktionsketten beliebiger Länge. Dann gibt es insbesondere unendlich viele verschiedene Reduktionsketten. Im ersten Kontraktionsschritt hat man nur endlich viele Möglichkeiten zu beginnen, da M nur endlich viele β-Redexe enthält. Also muß es eine β-Kontraktion $M \to_\beta M_1$ geben, die in unendlich vielen dieser Reduktionsketten vorkommt. Also hat auch M_1 Reduktionsketten beliebiger Länge. Indem man diese Vorgehensweise wiederholt, findet man per Induktion eine unendlich lange Reduktionskette $M \to_\beta M_1 \to_\beta M_2 \to_\beta \dots$, im Widerspruch zur Voraussetzung. $\square$

Es gilt nun der folgende Satz, der in 6 bewiesen werden wird:

2.5 Satz. Jeder einfach getypte λ-Term ist stark normalisierend.

3. Ein alternativer Zugang: Typing à là Curry

In Abschnitt 1 hatten wir die Variablen à priori getypt; einem getypten λ-Term konnte man seinen Typ ansehen: der eindeutig bestimmte Typ ist in den Term eingebaut.

Nun wollen wir eine praktikablere Variante von getypten Termen behandeln, bei der man für die Variablen (die à prori ungetypt sind) Typen deklariert. Da in einem Term nur endlich viele Variablen vorkommen, brauchen wir Typdeklarationen nur für endlich viele Variablen betrachten.

Wir wollen *ungetypte* λ-Terme M betrachten und fragen, ob und wenn ja welche Typen σ diesem Term zugeordnet werden können. Die Schreibweise $M : \sigma$ ist dann zu lesen als: „Der ungetypte λ-Term M kann mit dem Typ σ getypt werden.“ (Wir sagen dann wieder, daß M vom Typ σ ist, meinen in diesem Fall aber das vorherige.) Insbesondere kann ein ungetypter λ-Term auf verschiedene Weisen getypt werden.

Wie man einem ungetypten λ-Term M einen Typ zuordnen kann, wird induktiv über den Termaufbau definiert. Die Typisierung von M verläuft entlang des Termbaums:

3.1 Typisierung. (1) Einer Variablen x kann man einen beliebigen Typ σ zuordnen.

(1)
$$\frac{x : \sigma}{}$$

(2) Hat man eine Typisierung von M durch den Typ τ, wobei anfangs der in M eventuell frei vorkommenmden Variablen x der Typ σ zugeordnet worden ist, dann darf man $\lambda x.M$ den Typ $\sigma \to \tau$ geben. Man darf dann Die Variablendeklaration $x : \sigma$ darf man dann streichen (man muß sie aber nicht streichen), da sie sich aus dem Typ $\sigma \to \tau$ rekonstruieren läßt.

(2)
$$\frac{\begin{matrix}[x : \sigma] \\ \vdots \\ M : \tau\end{matrix}}{(\lambda x.M) : \sigma \to \tau}\ (\to \mathrm{I})$$

(3) Hat man eine Herleitung, auf Grund von welcher M de Typ $\sigma \to \tau$ und N der Typ σ zuordnet wird, so darf man MN den Typ τ zuordnen.

(3)
$$\frac{\begin{matrix}\vdots \\ M : (\sigma \to \tau)\end{matrix} \qquad \begin{matrix}\vdots \\ N : \sigma\end{matrix}}{(MN) : \tau}\ (\to \mathrm{E})$$

Dabei ist die **Nebenbedingung** zu beachten, daß bei der Typisierung von M und N für eine gemeinsame freie Variable x anfangs derselbe Typ deklariert worden ist, sofern diese in beidn vorkommt.

Man kann sich eine Typisierung eines Terms M als Deduktionsbaum vorstellen, der dieselbe Gestalt hat wie der Baum der Teilterme von M. Die Blätter des Baums bilden als Voraussetzungen die Typdeklarationen an die Variablen, wobei Typdeklarationen gestrichen werden können, wenn die entsprechende Variable in M gebunden wird. Jeder Schritt ist dann eine Anwendung der Regeln $(\to \mathrm{E})$ bzw. $(\to \mathrm{I})$. Am Ende erhält man eine Typisierung des gegebenen Terms. Natürlich hängt der Typ von M von den anfänglichen Typdeklarationen ab.

3.2 Kontexte. Sind $x_1, \dots, x_n$ paarweise verschiedene Variablen und $\sigma_1, \dots, \sigma_n$ einfache Typen, dann nennt man

$$\Gamma := \{x_1 : \sigma_1, \dots, x_n : \sigma_n\}$$

einen *Typisierungskontext* oder einfach einen *Kontext*.

Hat man eine Typisierung $M : \tau$ und bezeichnet man mit Γ einen Kontext, der die nicht gestrichenen Typdeklarationen $x : \sigma$ an den Blättern des Typisierungsbaum von $M : \tau$ enthält, so schreiben wir

$$\Gamma \vdash M : \tau$$

Γ enthält alle Typdeklarationen für die in M frei vorkommenden Variablen, kann aber durchaus noch weitere – sozusagen überflüssige – Typdeklarationen enthalten. Dies ist zu lesen als: „Unter der Voraussetzung, daß die freien Variablen in M den in Γ angegebenen Typ zugewiesen bekommen, kann man gemäß obiger Regeln herleiten, daß man den Term M mit dem Typ τ typen kann.“ Wir sagen kurz: „Im Kontext Γ hat M den Typ τ.“

Im Fall $\Gamma = \varnothing$ schreibt man auch einfach $\vdash M : \tau$. Dieser Fall kann nur dann auftreten, wenn M ein geschlossener Term ist.

3.3 Beispiel.

- $\mathsf{K} \equiv \lambda xy.x$:

$$\dfrac{\dfrac{[x\colon \sigma]^{(2)} \qquad [y\colon \tau]^{(1)}}{(\lambda y.x) : \tau \to \sigma}\,(\to \mathrm{I})^{(1)}}{(\lambda x.\lambda y.x) : \sigma \to \tau \to \sigma}\,(\to \mathrm{I})^{(2)}$$

  Folglich gilt $\vdash K : \sigma \to \tau \to \sigma$ für beliebige Typen σ und τ.

  Man bemerke, daß die $(\to \mathrm{I})$-Regel auch angewendet werden kann, wenn die Variable, über die abstrahiert wird, sonst im Term nicht vorkommt.

- $\mathsf{S} \equiv \lambda xyz.xz(yz)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[x : \varrho \to \sigma \to \tau]^{(3)} \qquad [z : \varrho]^{(1)}}{xz : \sigma \to \tau} \qquad \dfrac{[y : \varrho \to \sigma]^{(2)} \qquad [z : \varrho]^{(1)}}{yz : \sigma}\,(\to \mathrm{E})}{xz(yz) : \tau}\,(\to \mathrm{E})}{\lambda z.xz(yz) : \varrho \to \tau}\,(\to \mathrm{I})^{(1)}}{\lambda yz.xz(yz) : (\varrho \to \sigma) \to \varrho \to \tau}\,(\to \mathrm{I})^{(2)}}{\lambda xyz.xz(yz) : (\varrho \to \sigma \to \tau) \to (\varrho \to \sigma) \to \varrho \to \tau}\,(\to \mathrm{I})^{(3)}$$

  Es gilt also für beliebige Typen ϱ, σ und τ, daß S mit dem Typ

$$(\varrho \to \sigma \to \tau) \to (\varrho \to \sigma) \to \varrho \to \tau$$

  typisierbar ist.

- $\lambda x.y(\lambda z.xz)$:

$$\dfrac{\dfrac{y : (\sigma \to \tau) \to \varrho \qquad \dfrac{\dfrac{[x : \sigma \to \tau]^{(2)} \qquad [z : \sigma]^{(1)}}{xz : \tau}}{\lambda z.xz : \sigma \to \tau}\,(\to \mathrm{I})^{(1)}}{y(\lambda z.xz) : \varrho}\,(\to \mathrm{E})}{\lambda x.y(\lambda z.xz) : (\sigma \to \tau) \to \varrho}\,(\to \mathrm{I})^{(2)}$$

Folglich gilt

$$\{y : (\sigma \to \tau) \to \varrho\} \vdash \lambda x.y(\lambda z.xz) : (\sigma \to \tau) \to \varrho$$

Die kleinen Zahlen an den Voraussetzungen und den $(\to \mathrm{I})$-Anwendungen sollen andeuten, zu welchem Zeitpunkt eine Voraussetzung gestrichen wurde.

Bemerkung. Trennt man im Typisierungsbaum eines Terms die Terme und die Typen, so erhält man einerseits den Termbaum, andererseits eine natürliche Deduktion (siehe Abschnitt 4) der Aussage $(\varrho \to \sigma \to \tau) \to (\varrho \to \sigma) \to \varrho \to \sigma$. Als Beispiel betrachten wir den Term S:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{x} \quad \overline{z}}{xz} \qquad \dfrac{\overline{y} \quad \overline{z}}{yz}}{xz(yz)}}{\lambda z.xz(yz)}}{\lambda yz.xz(yz)}}{\lambda xyz.xz(yz)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{[\varrho \to \sigma \to \tau]^{(3)}} \quad \overline{[\varrho]^{(1)}}}{\sigma \to \tau} \qquad \dfrac{\overline{[\varrho \to \sigma]^{(2)}} \quad \overline{[\varrho]^{(1)}}}{\sigma}}{\tau}}{\sigma \to \tau}\,{\scriptstyle (1)}}{(\varrho \to \sigma) \to \varrho \to \sigma}\,{\scriptstyle (2)}}{(\varrho \to \sigma \to \tau) \to (\varrho \to \sigma) \to \varrho \to \sigma}\,{\scriptstyle (3)}$$

3.4 Typisierung im Stile des Sequenzenkalküls. In 3.1 hatten wir die Schreibweise $\Gamma \vdash M : \sigma$ eingeführt. Dabei war

$$\Gamma := x_1 : \sigma_1, \ldots, x_n : \sigma_n$$

ein Kontext, M ein ungetypter λ-Term und σ ein Typ. Man sieht leicht ein, daß aus den Typisierungsregeln (1)–(3) folgt, daß

$$(1') \qquad \dfrac{}{\Gamma, x : \sigma \vdash x : \sigma}$$

$$(2') \qquad \dfrac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x\colon \sigma.M) : \sigma \to \tau}\,(\to \mathrm{I})$$

$$(3') \qquad \dfrac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma' \vdash N : \sigma}{\Gamma, \Gamma' \vdash (MN) : \tau}\,(\to \mathrm{E})$$

Dabei ist auf die **Nebenbedingung** zu achten, daß die Typdeklarationen in Γ und Γ' sich nicht widersprechen, d.h., wird für eine Variable x sowohl in Γ als auch in Γ' ein Typ deklariert, so muß das derselbe Typ sein, damit Γ, Γ' wieder ein Kontext ist. Wir verwenden die Konventionen $\Gamma, x : \sigma = \Gamma \cup \{x : \sigma\}$ und $\Gamma, \Gamma' = \Gamma \cup \Gamma'$.

Umgekehrt gilt auch: Wenn man mithilfe des Axioms (1′) und den Regeln (2′) und (3′) die *Sequenz* $\Gamma \vdash M : \sigma$ herleiten kann, dann gibt es ein Typing von $M : \sigma$ im Sinne von 3.1 mit den Voraussetzungen aus Γ.

Beispiel. Ein Typing von $\mathsf{S} \equiv \lambda xyz.xz(yz)$ sieht im Sequenzenkalkül wie folgt aus:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{x : \varrho \to \sigma \to \tau\} \vdash x : \varrho \to \sigma \to \tau} \quad \dfrac{}{\{z : \varrho\} \vdash z : \varrho}}{\{x : \varrho \to \sigma \to \tau,\ z : \varrho\} \vdash xz : \sigma \to \tau} \qquad \dfrac{\dfrac{}{\{y : \varrho \to \sigma\} \vdash y : \varrho \to \sigma} \quad \dfrac{}{\{z : \varrho\} \vdash z : \varrho}}{\{y : \varrho \to \sigma,\ z : \varrho\} \vdash yz : \sigma}}{\{x : \varrho \to \sigma \to \tau,\ y : \varrho \to \sigma,\ z : \varrho\} \vdash xz(yz) : \tau}}{\{x : \varrho \to \sigma \to \tau,\ y : \varrho \to \sigma\} \vdash \lambda z.xz(yz) : \varrho \to \tau}}{\{x : \varrho \to \sigma \to \tau\} \vdash \lambda yz.xz(yz) : (\varrho \to \sigma) \to \varrho \to \tau}}{\vdash \lambda xyz.xz(yz) : (\varrho \to \sigma \to \tau) \to (\varrho \to \sigma) \to \varrho \to \tau}
$$

Also $\vdash \mathsf{S} : (\varrho \to \sigma \to \tau) \to (\varrho \to \sigma) \to \varrho \to \tau$ für beliebige Typen ϱ, σ und τ.

4. Natürliche Deduktion für das →-Fragment der Aussagenlogik

Wir betrachten ein Fragment der Aussagenlogik: Als logische Verknüpfung lassen wir hier nur die Implikation $\to$ zu.

4.1 Aussageformen. Es seien $\alpha_1, \alpha_2, \ldots$ endlich oder abzählbar viele *atomare Aussagen* (Aussagenvariable), die wir mit $\alpha, \beta, \gamma, \ldots$ bezeichnen. *Aussageformen* werden induktiv definiert:

(1) Atomare Aussagen α sind Aussageformen.

(2) Sind σ und τ Aussageformen, dann auch $(\sigma \to \tau)$.

Formal sind also Aussageformen dieses Fragments der Aussagenlogik dasselbe wie Typen. Wir übernehmen auch die entsprechenden Klammerkonventionen.

4.2 Natürliche Deduktion. Nun wollen wir Regeln angeben, wie man aus gegebenen Aussageformen andere herleiten kann. Eine Herleitung oder Deduktion sollte man sich als einen Baum vorstellen: An den Blättern stehen die Voraussetzungen (Prämissen) und an der Wurzel die Konklusion.

Was eine Deduktion sein soll, definieren wir induktiv:

(1) Man darf jede Aussageform als Voraussetzung hinschreiben.

(2) Hat man eine Herleitung von τ, so erhält man daraus eine Herleitung von $\sigma \to \tau$, wobei man die Voraussetzung σ streichen darf (sofern σ überhaupt unter den Voraussetzungen vorkommt; die Voraussetzung σ muß aber nicht gestrichen werden).

(3) Hat man eine Herleitung von $\sigma \to \tau$ und eine Herleitung von σ, so ergibt sich daraus eine Herleitung von τ.

bzw. in Kalkülform:

(1) $$\sigma$$

(2) $$\frac{\begin{array}{c}[\sigma]^{(i)}\\ \vdots\\ \tau\end{array}}{\sigma\to\tau}\,(\to \mathrm{I})^{(i)}$$

(3) $$\frac{\begin{array}{c}\vdots\\ \sigma\to\tau\end{array}\qquad \begin{array}{c}\vdots\\ \sigma\end{array}}{\tau}\,(\to \mathrm{E})$$

Die zweite Regel nennt man die $\to$-Introduktionsregel, die dritte die $\to$-Eliminationsregel.

Bemerkung. In einer Deduktion nummeriert man die Schritte durch, bei denen die Regel 2 angewandt wird, und man markiert die jeweils gestrichenen Voraussetzungen mit derselben Nummer. Auf diese Weise läßt sich der Ablauf der Deduktion eindeutig rekonstruieren.

4.3 Beispiel.

(1)

$$\dfrac{\dfrac{[\sigma]^{(1)}}{\sigma\to\sigma}\,{\scriptstyle (1)}}{\tau\to\sigma\to\sigma}\,{\scriptstyle (2)}$$

Bei der zweiten $\to$-Introduktion wurde τ benutzt, obwohl τ nicht explizit unter den Voraussetzungen auftritt. Dies ist durchaus erlaubt.

(2)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\alpha\to\beta\to\gamma]^{(4)}\qquad [\alpha]^{(1)}}{\beta\to\gamma}\qquad\qquad \dfrac{[\alpha\to\beta]^{(3)}\qquad [\alpha]^{(2)}}{\beta}}{\gamma}}{\alpha\to\gamma}\,{\scriptstyle (1)}}{\alpha\to\alpha\to\gamma}\,{\scriptstyle (2)}}{(\alpha\to\beta)\to\alpha\to\alpha\to\gamma}\,{\scriptstyle (3)}}{(\alpha\to\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\alpha\to\gamma}\,{\scriptstyle (4)}$$

Bei der ersten $\to$-Introduktion wurde α nur einmal unter den Voraussetzungen gestrichten, obwohl es zweimal vorkommt. Dies ist durchaus zulässig.

(3)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\alpha\to\beta\to\gamma]^{(3)}\qquad[\alpha]^{(1)}}{\beta\to\gamma}\qquad\dfrac{[\alpha\to\beta]^{(2)}\qquad[\alpha]^{(1)}}{\beta}}{\gamma}}{\alpha\to\gamma}{\scriptstyle(1)}}{(\alpha\to\beta)\to\alpha\to\gamma}{\scriptstyle(2)}}{(\alpha\to\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\gamma}{\scriptstyle(3)}$$

Diese Deduktion unterscheidet sich von der vorhergehenden nur dadurch, daß bei der ersten →-Introduktion die Voraussetzung α an beiden Stellen gestrichten wird, an denen sie vorkommt.

4.4 Herleitbarkeit. Es sei Γ eine endliche Menge von Aussageformen und σ eine Aussageform. Wir schreiben

$$\Gamma\vdash\sigma$$

und sagen „σ ist aus Γ herleitbar“, wenn es eine natürliche Deduktion gibt, deren Konklusion σ ist und deren (nicht gestrichene) Voraussetzungen in Γ enthalten sind.

Im Falle von $\Gamma=\varnothing$ schreiben wir $\vdash\sigma$ und sagen dann, daß σ ein *herleitbaren Satz* ist. Jede voraussetzungsfreie Deduktion nennen wir einen *Beweis von* σ.

Beispiel.

$$\vdash\tau\to\sigma\to\sigma$$
$$\vdash(\alpha\to\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\alpha\to\gamma$$
$$\vdash(\alpha\to\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\gamma$$

Bemerkung. Jeder in diesem Sinne herleitbare Satz ist eine Tautologie im Sinne der Wahrheitswertsemantik, d.h. er bekommt den Wert WAHR für jede beliebige Belegung der in ihm vorkommenden atomaren Aussagen mit den Wahrheitswerten WAHR und FALSCH.

Jedoch gibt es Tautologien, die nicht durch eine natürliche Deduktion in obigem Sinne herleitbar sind. Eine solche ist zum Beispiel *Peirce's Law*:

$$((\alpha\to\beta)\to\alpha)\to\alpha$$

Wir haben es bei unserem System mit einem intuitionistischen Herleitungsbegriff zu tun.

4.5 Bemerkung. Ein Satz kann viele verschiedene Beweise haben, z.B. die folgenden Beweise für $\vdash(\alpha\to\alpha)\to\alpha\to\alpha$:

$$\dfrac{\dfrac{[\alpha]^{(1)}}{\alpha\to\alpha}{\scriptstyle(1)}}{(\alpha\to\alpha)\to\alpha\to\alpha}{\scriptstyle(2)}\qquad\dfrac{\dfrac{\dfrac{[\alpha\to\alpha]^{(2)}\qquad[\alpha]^{(1)}}{\alpha}}{\alpha\to\alpha}{\scriptstyle(1)}}{(\alpha\to\alpha)\to\alpha\to\alpha}{\scriptstyle(2)}\qquad\dfrac{\dfrac{\dfrac{[\alpha\to\alpha]^{(2)}\qquad\dfrac{[\alpha\to\alpha]^{(2)}\qquad[\alpha]^{(1)}}{\alpha}}{\alpha}}{\alpha\to\alpha}{\scriptstyle(1)}}{(\alpha\to\alpha)\to\alpha\to\alpha}{\scriptstyle(2)}$$

4.6 Der Sequenzenkalkül. Die drei Bildungsregeln für natürliche Deduktionen entsprechen den drei folgenden Regeln für die Herleitbarkeitsrelation $\Gamma \vdash \sigma$:

(1)
$$\frac{}{\Gamma, \sigma \vdash \sigma}$$

(2)
$$\frac{\Gamma, \sigma \vdash \tau}{\Gamma \vdash \sigma \to \tau}$$

(3)
$$\frac{\Gamma \vdash \sigma \to \tau \qquad \Gamma' \vdash \sigma}{\Gamma, \Gamma' \vdash \tau}$$

Dabei stehen Γ, σ und Γ, Γ' für $\Gamma \cup \{\sigma\}$ bzw. $\Gamma \cup \Gamma'$

Genau dann kann man σ aus Γ mit einer natürlichen Deduktion herleiten, wenn man $\Gamma \vdash \sigma$ aus den angegebenen Regeln (1)–(2) gewinnen kann. Herleitungen mit diesen Regeln nennt man den Sequenzenkalkül für das $\to$-Fragment der Aussagenlogik.

Beispiel.

(1)

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\{\alpha \to \alpha\} \vdash \alpha \to \alpha} \qquad \dfrac{\dfrac{}{\{\alpha \to \alpha\} \vdash \alpha \to \alpha} \qquad \dfrac{}{\{\alpha\} \vdash \alpha}}{\{\alpha \to \alpha,\ \alpha\} \vdash \alpha}}{\{\alpha \to \alpha,\ \alpha\} \vdash \alpha}}{\{\alpha \to \alpha\} \vdash \alpha \to \alpha}}{\vdash (\alpha \to \alpha) \to \alpha \to \alpha}$$

(2)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{\alpha \to \beta \to \gamma\} \vdash \alpha \to \beta \to \gamma} \qquad \dfrac{}{\{\alpha\} \vdash \alpha}}{\{\alpha \to \beta \to \gamma,\ ,\alpha\} \vdash \beta \to \gamma} \qquad \dfrac{\dfrac{}{\{\alpha \to \beta\} \vdash \alpha \to \beta} \qquad \dfrac{}{\{\alpha\} \vdash \alpha}}{\{\alpha \to \beta,\ ,\alpha\} \vdash \beta}}{\{\alpha \to \beta \to \gamma,\ \alpha \to \beta,\ ,\alpha\} \vdash \gamma}}{\{\alpha \to \beta \to \gamma,\ \alpha \to \beta\} \vdash \alpha \to \gamma}}{\{\alpha \to \beta \to \gamma\} \vdash (\alpha \to \beta) \to \alpha \to \gamma}}{\vdash (\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma}$$

5. Der Curry-Howard-Isomorphismus

Wir identifizieren nun einfache Typen mit Aussageformen im Sinne von 4.1.

5.1 Satz. Es gibt eine Bijektion zwischen

(1) geschlossenen λ-Termen vom Typ σ (bis auf α-Äquivivalenz)

(2) voraussetzungsfreien Beweisen von σ im Sinne von 4.2

Folgerung. Zu einem Typ σ gibt es genau dann einen geschlossenen λ-Term M, der sich mit dem Typ σ versehen läßt, wenn σ ein herleitbarer Satz im Sinne von 4.4 ist.

Bemerkung. Für nicht-geschlossene λ-Terme gibt es eine ähnliche Aussage. In der Formulierung ist dies jedoch langwieriger, da man eine Beziehung zwischen den freien Variablen und den Voraussetzungen in Deduktionen herstellen muß.

Beweisidee zu 5.1. Die wesentliche Idee ist die Beobachtung, daß die Typing-Regeln aus 3.1 in

(1) die Bildungsregeln für λ-Terme
(2) die Bildungsregeln für natürliche Deduktionen 4.2

zerfallen. Ein Typing-Baum ist also eine Überlagerung von zwei Bäumen derselben Gestalt, nämlich

(1) dem Termbaum des betrachteten Terms M
(2) ein Deduktionsbaum für den Typ σ.

Sei nun M ein geschlossener λ-Term, den man mit dem Typ σ typen kann. Betrachtet man den zugehörigen Typisierungs-Baum und vergißt die darin stehenden λ-Terme, so erhält man eine voraussetzungsfreie Deduktion von σ. (Man vergleiche die Beispiele 3.3.2 und 4.3.3.)

Sei umgekehrt eine voraussetzungsfreie Deduktion von σ gegeben. Wir bilden einen geschlossenen Term M, dessen Baum dieselbe Gestalt hat wie der gegebene Deduktionsbaum: Wir schreiben Variable an die Blätter des Baums. Dieselbe Variable benutzen wir nur dann, wenn die Voraussetzung mit demselben Index bezeichnet ist. Der gesuchte Term M mit Typ σ ergibt sich dann automatisch. $\square$

5.2 Beispiel.

$$\dfrac{\dfrac{\dfrac{[\alpha\to\alpha]^{(2)} \qquad \dfrac{[\alpha\to\alpha]^{(2)} \qquad [\alpha]^{(1)}}{\alpha}}{\alpha}}{\alpha\to\alpha}\,{\scriptstyle(1)}}{(\alpha\to\alpha)\to\alpha\to\alpha}\,{\scriptstyle(2)} \qquad\qquad \dfrac{\dfrac{\dfrac{x_2 \qquad \dfrac{x_2 \qquad x_1}{x_2x_1}}{x_2(x_2x_1)}}{\lambda x_1.x_2(x_2x_1)}}{\lambda x_2x_1.x_2(x_2x_1)}$$

Man bemerke, daß der Term M bis auf die Benennung der gebundenen Variablen eindeutig bestimmt ist.

Den Curry-Howard-Isomorphismus faßt man manchmal in dem Slogan „Ein Term ist ein Zeuge für die Beweisbarkeit seines Typs." zusammen.

5.3 Schnittelimination und β-Reduktion. In einer natürlichen Deduktion nennt man jede Konfiguration der folgenden Gestalt einen *Schnitt*:

$$\frac{\begin{array}{c}[\sigma]\\ \vdots\\ \sigma\to\tau\end{array}\qquad\begin{array}{c}\vdots\\ \sigma\end{array}}{\tau}$$

Man erlaubt nun, die obige Deduktion von τ in die nun folgende zu transformieren:

$$\begin{array}{c}\vdots\\ \sigma\\ \vdots\\ \tau\end{array}$$

Man nimmt also die Deduktion von σ und setzt diese an jeder Stelle ein, wo σ als Voraussetzung zur Deduktion von τ gebraucht wurde. Man erhält so einen direkteren Beweis von τ.

Die eben beschriebene Transformation von Deduktionen nennt man *Schnittelimination.*

Beispiel.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\alpha\to\alpha]^{(3)}\quad[\alpha]^{(1)}}{\alpha}}{\alpha\to\alpha}{\scriptstyle(1)}\qquad\dfrac{[\alpha\to\alpha]^{(3)}\quad[\alpha]^{(2)}}{\alpha}}{\alpha}}{\alpha\to\alpha}{\scriptstyle(2)}}{(\alpha\to\alpha)\to\alpha\to\alpha}{\scriptstyle(3)}\qquad\longrightarrow\qquad\dfrac{\dfrac{\dfrac{[\alpha\to\alpha]^{(2)}\qquad\dfrac{[\alpha\to\alpha]^{(2)}\quad[\alpha]^{(1)}}{\alpha}}{\alpha}}{\alpha\to\alpha}{\scriptstyle(1)}}{(\alpha\to\alpha)\to\alpha\to\alpha}{\scriptstyle(2)}$$

Via dem Curry-Howard-Isomorphismus entspricht ein Schnitt einem β-Redex. Nach Elimination des Schnitts entspricht der Term dem β-Kontraktum:

$$\frac{\dfrac{\begin{array}{c}[x:\sigma]\\ \vdots\\ M:\tau\end{array}}{\lambda x.M:\sigma\to\tau}\qquad\begin{array}{c}\vdots\\ N:\sigma\end{array}}{(\lambda x.M)N:\tau}\qquad\longrightarrow\qquad\begin{array}{c}\vdots\\ x:=N:\sigma\\ \vdots\\ M[x:=N]:\tau\end{array}$$

Wir haben damit gesehen:

5.4 Satz. Die Schnittelimination entspricht via dem Curry-Howard-Isomorphismus der β-Reduktion von Termen.

Da nach 2.5 jeder getypte Term stark normalisierend ist, folgt

Folgerung. Jede natürliche Deduktion läßt sich durch endlich viele Schnitteliminationen in eine schnittfreie Deduktion umwandeln.

Folgerung. Die schnittfreien Deduktionen entsprechen via dem Curry-Howard-Isomorphismus den (getypten) Termen in β-Normalform.

6. Beweis für die starke Normalisierbarkeit einfach getypter λ-Terme

Nach 2.4 wird ein (getypter oder ungetypter) λ-Term *stark normalisierend* genannt, wenn er keine unendliche lange β-Reduktionskette besitzt, wenn also jede β-Reduktionskette nach endlich vielen Schritten abbricht. Wir wollen in diesem Abschnitt den folgenden Satz beweisen, der zu Satz 2.5 äquivalent ist:

6.1 Satz. Jeder (ungetypte) λ-Term M, der einfach getypt werden kann, ist stark normalisierend.

Zunächst bezeichne $\mathcal{N}$ die Menge aller (ungetypten) stark normalisierenden λ-Terme. Wir betrachten einige einfache Eigenschaften dieser Menge:

6.2 Eigenschaften von $\mathcal{N}$. Seien x eine Variable und $M, M_1, \ldots, M_n$ λ-Terme.

(1) Ist M in β-Normalform, so erlaubt M keine β-Kontraktion und ist daher stark normalisierend. Insbesondere gilt daher $x \in \mathcal{N}$ für jede Variable x..

(2) $M \in \mathcal{N} \iff \lambda x.M \in \mathcal{N}$; denn jede β-Reduktionskette von $\lambda x.M$ muß sich in dem Teilterm M abspielen.

(3) $M \in \mathcal{N},\ M \to_\beta^* M' \implies M' \in \mathcal{N}$; denn hätte M' eine unendlich lange β-Reduktionskette, dann auch M.

Die Umkehrung hiervon ist nicht richtig: In den Beispielen zu Abschnitt I.2.4 hatten wir einen Term L kennengelernt, der eine unendlich lange β-Reduktionskette hat. Also ist $M \equiv (\lambda z.x)L$ nicht stark normalisierbar, aber es gilt $M \to_\beta x \in \mathcal{N}$. Wir haben jedoch:

(4) Es gilt $M \in \mathcal{N}$ genau dann, wenn $M' \in \mathcal{N}$ für jeden Term M', der aus M durch Kontraktion eines einzigen β-Redex entsteht. Die Richtung „$\Rightarrow$“ folgt aus dem vorhergehenden, für die andere Richtung nehmen wir an, M hätte eine unendlich lange β-Reduktionskette $M \to_\beta M' \to_\beta \ldots$; dann hat aber auch der Term M', der aus M durch eine einzige β-Kontraktion hervorgeht eine unendlich lange β-Reduktionskette.

(5) $M_1, \ldots, M_n \in \mathcal{N} \implies xM_1 \ldots M_n \in \mathcal{N}$; denn jeder Redex von $xM_1 \ldots M_n$ muß in einem der M_i enthalten sein. Es folgt, daß jede β-Kontraktion von $xM_1 \ldots M_n$ durch Kontraktion eines der M_i entsteht. Per Induktion folgt die Behauptung.

6.3 Hilfssatz. Aus $Q \in \mathcal{N}$ und $P[x := Q]M_1 \ldots M_n \in \mathcal{N}$ folgt, daß

$$(\lambda x.P)QM_1 \ldots M_n \in \mathcal{N}$$

Beweis. Wir beweisen den Hilfssatz durch vollständige Induktion über die Zahl $h = p + q$, wobei q bzw. p die maximalen Längen von β-Reduktionsketten von Q bzw. $P[x := Q]M_1 \ldots M_n$ sind. Wie wir in der Bemerkung zu 2.4 festgestellt hatten, haben β-Reduktionsketten stark normalisierender λ-Terme eine beschränkte Länge, so daß die natürlichen Zahlen q und p wohldefiniert sind.

Gemäß 6.2.4 genügt es zu zeigen, daß jeder Term M', der aus $M :\equiv (\lambda x.P)QM_1 \dots M_n$ durch Kontraktion eines einzigen β-Redex R entsteht, stark normalisierend ist.

Der β-Redex R kann in $M \equiv (\lambda x.P)QM_1 \dots M_n$ auf vier verschiedene Weisen auftreten:

(1) $R \equiv (\lambda x.P)Q$. In diesem Fall ist $M' \equiv P[x := Q]M_1 \dots M_n \in \mathcal{N}$ nach Voraussetzung.

Durch diesen Fall wird auch der Induktionsanfang $h = 0$ abgedeckt. Denn dann müssen $P, Q, M_1, \dots, M_n$ β-Normalformen sein, und der einzige Redex von M ist $(\lambda x.P)Q$.

(2) R ist in einem der M_i enthalten, z.B. in M_1. Dann gilt $M' \equiv (\lambda x.P)QM_1'M_2 \dots M_n$, wobei M_1' aus M_1 durch Kontraktion von R entstanden ist. Wir haben dann auch eine β-Kontraktion

$$P[x := Q]M_1M_2 \dots M_n \to_\beta P[x := Q]M_1'M_2 \dots M_n$$

Nach Voraussetzung und 6.2.3 ist auch $P[x := Q]M_1'M_2 \dots M_n$ stark normalisierend, und für die Länge der längsten β-Reduktionskette dieses Terms muß $p' \leq p - 1$ gelten. Nach Induktionsvoraussetzung folgt nun

$$M' \equiv (\lambda x.P)QM_1'M_2 \dots M_n \in \mathcal{N}$$

(3) R ist in Q enthalten. Dann gilt $M' \equiv (\lambda x.P)Q'M_1 \dots M_n$, wobei Q' aus Q durch Kontraktion des Redex R hervorgeht. Nach 6.2.3 ist dann auch $Q' \in \mathcal{N}$ und für die Länge der längsten β-Reduktionskette von Q gilt $q' \leq q - 1$. Ferner gibt es nach I.2.3 eine β-Reduktion $P[x := Q] \to_\beta^* P[x := Q']$, die genau so viele Schritte benötigt, wie die Anzahl der freien Vorkommen von x in P. Folglich gibt es auch eine β-Reduktion

$$P[x := Q]M_1M_2 \dots M_n \to_\beta^* P[x := Q']M_1M_2 \dots M_n$$

Nach Voraussetzung und 6.2.3 ist auch der letztgenannte Term stark normalisierend, und für die Länge seiner längsten β-Reduktionskette gilt $p' \leq p$. Wir haben $h' = p' + q' \leq p + q - 1$ und erhalten aus der Induktionsvoraussetzung

$$M' \equiv (\lambda x.P)Q'M_1 \dots M_n \in \mathcal{N}$$

(4) R ist in P enthalten. Argumentiere wie in Fall 2 (Übung!). □

6.4 Definition. Für Teilmengen $\mathcal{A}, \mathcal{B} \subseteq \Lambda$ von ungetypten λ-Termen sei

$$\mathcal{A} \to \mathcal{B} := \{M \in \Lambda \mid MN \in \mathcal{B} \text{ für alle } N \in \mathcal{A}\}$$

6.5 $\mathcal{N}$-saturierte Teilmengen. Eine Menge $\mathcal{B}$ von λ-Termen wird *$\mathcal{N}$-saturiert* genannt, falls gilt:

$$Q \in \mathcal{N},\ P[x := Q]M_1 \dots M_n \in \mathcal{B} \implies (\lambda x.P)QM_1 \dots M_n \in \mathcal{B}$$

Der Hilfssatz 6.3 besagt gerade, daß $\mathcal{N}$ selbst $\mathcal{N}$-saturiert ist.

Hilfssatz. Ist $\mathcal{B}$ eine $\mathcal{N}$-saturierte Menge, dann ist auch $\mathcal{A} \to \mathcal{B}$ eine $\mathcal{N}$-saturierte Menge.

Beweis. Es sei $Q \in \mathcal{N}$ und $P[x := Q]M_1 \dots M_n \in \mathcal{A} \to \mathcal{B}$. Dann gilt

$$P[x := Q]M_1 \dots M_n N \in \mathcal{B} \quad \text{für alle } N \in \mathcal{A} .$$

Da $\mathcal{B}$ eine $\mathcal{N}$-saturierte Menge ist, folgt $(\lambda x.P)QM_1 \dots M_n N \in \mathcal{B}$ für alle $N \in \mathcal{A}$ und folglich $(\lambda x.P)QM_1 \dots M_n \in \mathcal{A} \to \mathcal{B}$. □

6.6 Konstruktion. Für jeden einfachen Typ σ definieren wir eine Teilmenge $\mathcal{N}_\sigma \subseteq \Lambda$ induktiv über den Typaufbau:

(1) $\mathcal{N}_\alpha := \mathcal{N}$ für jeden atomaren Typ α

(2) $\mathcal{N}_{\sigma \to \tau} := \mathcal{N}_\sigma \to \mathcal{N}_\tau$

Nach 6.5 ist jede der Mengen $\mathcal{N}_\sigma$ eine $\mathcal{N}$-saturierte Menge.

6.7 Hilfssatz. Für jeden Typ σ gilt:

(1) Für jede Variable x und beliebige $M_1, \dots, M_n \in \mathcal{N}$ gilt $xM_1 \dots M_n \in \mathcal{N}_\sigma$. Insbesondere gilt $x \in \mathcal{N}_\sigma$.

(2) $\mathcal{N}_\sigma \subseteq \mathcal{N}$.

Beweis. Die Aussagen 1 und 2 werden simultan durch Induktion über den Aufbau von σ bewiesen. Sei zunächst $\sigma \equiv \alpha$ ein atomarer Typ. Dann ist $\mathcal{N}_\alpha = \mathcal{N}$ nach Definition und nach 6.2.5 ist $xM_1 \dots M_n \in \mathcal{N}$ für beliebige $M_1, \dots, M_n \in \mathcal{N}$. Weiterhin ist $\mathcal{N} = \mathcal{N}_\alpha$ nach 6.6.

Nehmen wir nun an, wir hätten die beiden gewünschten Eigenschaften für σ und τ bewiesen. Um zu zeigen, daß sie auch für $\sigma \to \tau$ gelten, wählen wir beliebige $M_1 \dots M_n N \in \mathcal{N}$. Für beliebiges $N \in \mathcal{N}_\sigma$ gilt dann nach Voraussetzung 2. für σ auch $N \in \mathcal{N}$ und $xM_1 \dots M_n N \in \mathcal{N}_\tau$ nach Voraussetzung 1. für τ und folglich $xM_1 \dots M_n \in \mathcal{N}_\sigma \to \mathcal{N}_\tau = \mathcal{N}_{\sigma \to \tau}$. Ist nun $M \in \mathcal{N}_{\sigma \to \tau}$, dann gilt $MN \in \mathcal{N}_\tau \subseteq \mathcal{N}$ für alle $N \in \mathcal{N}$. Wenn aber MN stark normalisierend ist, dann muß auch M stark normalisierend sein. □

6.8 Adäquatheitslemma. Ist M ein λ-Term, der mit dem Typ τ getypt werden kann, dann gilt $M \in \mathcal{N}_\tau$. Allgemeiner gilt: Falls man die Typisierung

$$(*) \qquad \{x_1 : \sigma_1, \dots, x_k : \sigma_k\} \vdash M : \tau$$

herleiten kann, dann gilt

$$M[x_1 := M_1, \dots, x_k := M_k] \in \mathcal{N}_\tau$$

für beliebige Terme $M_1 \in \mathcal{N}_{\sigma_1}, \dots, M_k \in \mathcal{N}_{\sigma_k}$.

Beweis durch Induktion über den Termaufbau von M. Wir benutzen die Abkürzung $\Gamma := \{x_1 : \sigma_1, \dots, x_k : \sigma_k\}$.

(1) $M \equiv x_i$: Wegen $(*)$ gilt $\tau \equiv \sigma_i$. Es folgt $M[x_1 := M_1, \dots, x_k := M_k] \equiv M_i \in \mathcal{N}_{\sigma_i} = \mathcal{N}_\tau$.

(2) $M \equiv PQ$: Dann hat man im letzten Schritt der Herleitung von $(*)$:

$$\Gamma \vdash P : \sigma \to \tau \qquad \Gamma \vdash Q : \sigma$$

Nach Induktionsvoraussetzung hat man

$$P[x_1 := M_1, \ldots, x_k := M_k] \in \mathcal{N}_{\sigma\to\tau} \qquad Q[x_1 := M_1, \ldots, x_k := M_k] \in \mathcal{N}_{\sigma}$$

Es folgt

$$\begin{aligned} &(PQ)[x_1 := M_1, \ldots, x_k := M_k] \\ \equiv &\big(P[x_1 := M_1, \ldots, x_k := M_k]\big)Q[x_1 := M_1, \ldots, x_k := M_k] \in \mathcal{N}_\tau \end{aligned}$$

nach der Definition von $\mathcal{N}_{\sigma\to\tau} = \mathcal{N}_\sigma \to \mathcal{N}_\tau$.

(3) $M \equiv \lambda x.P$: Durch Umbenennung von x können wir annehmen, daß x nicht frei in den $M_1, \ldots, M_k$ vorkommt und von $x_1, \ldots, x_k$ verschieden ist.

Es gilt, daß τ von der Form $\tau_1 \to \tau_2$ sein muß , und der letzte Schritt der Herleitung von $(*)$ mußte lauten

$$\Gamma, x : \tau_1 \vdash P : \tau_2$$

Nach Induktionsvoraussetzung gilt für jeden Term $N \in \mathcal{N}_{\tau_1}$:

$$P[x_1 := N_1, \ldots, x_k := M_k, x := N] \in \mathcal{N}_{\tau_2}$$

Da x in den M_i nicht frei vorkommt, gilt

$$P[x_1 := N_1, \ldots, x_k := M_k, x := N] \equiv P[x_1 := N_1, \ldots, x_k := M_k][x := N]$$

Da $N \in \mathcal{N}_{\tau_1} \subseteq \mathcal{N}$ und da $\mathcal{N}_{\tau_2}$ eine $\mathcal{N}$-saturierte Menge ist, folgt

$$\big(\lambda x.P[x_1 := N_1, \ldots, x_k := M_k]\big)N \in \mathcal{N}_{\tau_2}$$

Da dies für alle $N \in \mathcal{N}_{\tau_1}$ gilt, folgt

$$\lambda x.P[x_1 := N_1, \ldots, x_k := M_k] \in \mathcal{N}_{\tau_1} \to \mathcal{N}_{\tau_2} = \mathcal{N}_{\tau_1\to\tau_2} = \mathcal{N}_\tau \ .$$

□

Nun können wir Satz 6.1 beweisen:

Beweis von 6.1. Gegeben sei ein λ-Term M, der mit τ getypt werden kann. Nach dem Adäquatheitslemma 6.8 gilt $M \in \mathcal{N}_\tau$. Nach 6.7.2 gilt $\mathcal{N}_\tau \subseteq \mathcal{N}$, d.h. $M \in \mathcal{N}$. □

7. Allgemeiner Typ eines Term und ein Typisierungssalgorithmus

In diesem Abschnitt zeigen wir, daß die Frage der Typisierbarkeit eines λ-Terms in unserem einfachen Typsystem entscheidbar ist und daß die möglichen Typen eines typisierbaren Terms sich alle als Instanzen eines „allgemeinen“ Typs ergeben. Dazu brauchen wir zuerst einige Hilfsmittel über einfache Typen. Dabei fassen wir die atomaren Typen als Typvariable $\alpha, \beta, \gamma, \ldots$ auf, aus denen man beliebige einfache Typen rekursiv mit der folgenden Regel aufbaut: Sind σ und τ einfache Typen, dann ist auch $\sigma \to \tau$ ein einfacher Typ.

7.1 Substitution. Es seien $\alpha_1, \ldots, \alpha_n$ paarweise verschiedene Typvariable sowie $\rho_1, \ldots, \rho_n$ beliebige Typen. Wir bezeichnen mit

$$\sigma[\alpha_1 := \rho_1, \ldots, \alpha_n = \rho_n]$$

den Typ, den man erhält, indem man simultan in σ jede der Typvariablen $\alpha_1, \ldots, \alpha_n$ durch $\rho_1, \ldots \rho_n$ ersetzt.

Rekursiv über den Typaufbau kann man diese Substitution so erklären:

(1)

$$\alpha[\alpha_1 := \rho_1, \ldots, \alpha_n = \rho_n] :\equiv \begin{cases} \sigma_i & \text{falls } \alpha \equiv \alpha_i \\ \alpha & \text{falls } \alpha \not\equiv \alpha_i \text{ für alle } i \end{cases}$$

(2)

$$(\sigma \to \tau)\mathbb{S} :\equiv \sigma\mathbb{S} \to \tau\mathbb{S}$$

Dabei benutzten wir die Abkürzung $\mathbb{S} = [\alpha_1 := \rho_1, \ldots, \alpha_n := \rho_n]$ und schreiben dann auch kurz

$$\sigma\mathbb{S} \quad \text{statt} \quad \sigma[\alpha_1 := \rho_1, \ldots, \alpha_n := \rho_n]$$

7.2 Lösung von „Gleichungen" (Unifikation). Gegeben seien zwei beliebige Typen σ und τ. Wir sagen, die „Gleichung"

$$\sigma = \tau$$

habe eine *Lösung*, falls es eine Substitution $\mathbb{S}$ gibt, so daß

$$\sigma\mathbb{S} \equiv \tau\mathbb{S}\ .$$

Die Substitution $\mathbb{S} \equiv [\alpha_1 := \rho_1, \ldots, \alpha_n := \rho_n]$ wird dann eine Lösung (oder auch ein Unifikator von σ und τ) genannt.

Man nennt $\mathbb{S}$ die *allgemeine Lösung* der Gleichung $\sigma = \tau$, falls $\mathbb{S}$ eine Lösung ist und falls es zu jeder weiteren Lösung $\mathbb{S}'$ der Gleichung eine Substitution $\mathbb{T}$ so gibt, daß

$$\sigma\mathbb{S}' \equiv \sigma\mathbb{S}\mathbb{T} \quad \text{und} \quad \tau\mathbb{S}' \equiv \tau\mathbb{S}\mathbb{T}$$

7.3 Beispiel.

- Es sei $\sigma \equiv (\alpha \to \alpha) \to \beta$ und $\tau \equiv \beta \to (\gamma \to \gamma)$. Die Gleichung $\sigma = \tau$ hat als Lösung $\mathbb{S} = [\beta := \alpha \to \alpha, \gamma := \alpha]$.

  Probe: $\sigma[\beta := \alpha \to \alpha, \gamma := \alpha] \equiv (\alpha \to \alpha) \to (\alpha \to \alpha) \equiv \tau[\beta := \alpha \to \alpha, \gamma := \alpha]$

  Ist ϱ irgendein einfacher Typ, dann ist auch $\mathbb{S}' = [\beta := \varrho \to \varrho, \gamma := \varrho]$ eine Lösung unserer Gleichung. Dies ist jedoch eine Instanz der ersten Lösung $\mathbb{S}$, denn

  $$\begin{gathered}[\beta := \varrho \to \varrho, \gamma := \varrho] = [\beta := \alpha \to \alpha, \gamma := \alpha][\alpha := \varrho] \\ \mathbb{S}' = \mathbb{S}\mathbb{T}\end{gathered}$$

- Die Gleichung $\alpha = (\alpha \to \alpha) \to \beta$ hat keine Lösung, denn egal was man für α substituiert, die Länge der linken Seite bleibt kleiner als die der rechten Seite.

  Allgemeiner gilt:

- Es sei α eine Typvariable und τ ein Typ der Länge $\ell(\tau) > 1$, in dem α vorkommt. Dann hat die Gleichung $\alpha = \tau$ keine Lösung.

  In der Tat gilt für jede Substitution $\mathbb{S}$, daß $\ell(\alpha\mathbb{S}) < \ell(\tau\mathbb{S})$ und folglich $\alpha\mathbb{S} \not\equiv \tau\mathbb{S}$.

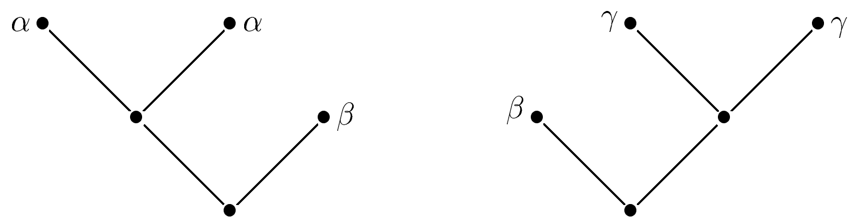
Es gilt nun der folgende Satz:

7.4 Satz. Es seien σ, τ beliebige einfache Typen. Wenn die Gleichung $\sigma = \tau$ eine Lösung besitzt, dann gibt es eine allgemeine Lösung.

Genauer: Es gibt einen Algorithmus, der für jedes Paar σ, τ einfacher Typen feststellt, ob die Gleichung $\sigma = \tau$ eine Lösung besitzt oder nicht. Im ersten Fall liefert er sogar noch eine allgemeine Lösung $\mathbb{S}$.

Der Algorithmus soll im folgenden angegeben werden. Auf den Nachweis, daß der Algorithmus korrekt ist, verzichten wir an dieser Stelle. Man findet den Beweis in jedem Buch zur Logikprogrammierung, sogar in sehr viel allgemeinerer Form unter dem Stichwort *Unifikationsalgorithmus von J. A. Robinson.*

Um den Algorithmus zu erklären, stellen wir σ und τ durch ihren Termaufbau dar. Als Beispiel wählen wir $\sigma \equiv (\alpha \to \alpha) \to \beta$ und $\tau \equiv \beta \to (\gamma \to \gamma)$:



Bei einer Substitution $\beta := \gamma$ wird der Termbaum von ϱ überall dort aufgepropft, wo β auftaucht. Um die Gleichung $\sigma = \tau$ zu lösen, muß man also eine Substitution finden, so daß aus den Bäumen für σ und τ ein- und derselbe Termbaum entsteht. Um dies zu erreichen, geht man wie folgt vor:

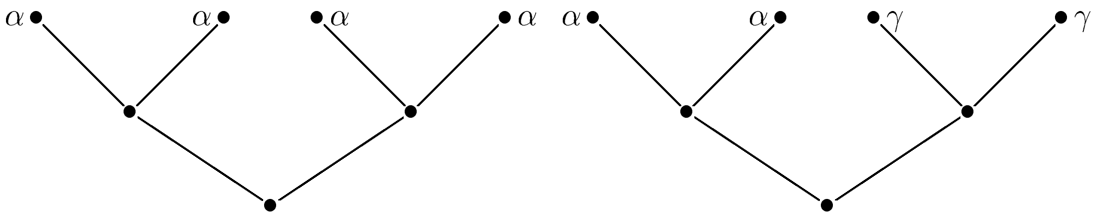
7.5 Unifikationsalgorithmus. Steige im Gleichschritt von der Wurzel ausgehend den linken Ast in σ und τ so lange hoch, bis man in einem der Bäume an der Spitze angekommen ist. Dort steht eine Typvariable α_1. In dem anderen Baum hat man noch einen Teilbaum über sich, der einem Typ σ_1 entspricht. Nun hat man zwei Fälle zu unterscheiden:

(1) Der Typ σ_1 enthält die Typvariable α_1, und es gilt $\ell(\sigma_1) > 1$. Dann hat die Gleichung $\sigma = \tau$ keine Lösung, und der Algorithmus stoppt.

(2) Ansonsten bilde $\sigma[\alpha_1 := \sigma_1]$ und $\tau[\alpha_1 := \sigma_1]$.

In den so erhaltenen Termbäumen stimmen nun die linken Äste überein. Nun läßt man die Teile von $\sigma[\alpha_1 := \sigma_1]$ und $\tau[\alpha_1 := \sigma_1]$ weg, die man zur Übereinstimmung gebracht hat, und wiederholt das Verfahren für den Restbaum.

In obigem Beispiel sieht das so aus: Wähle $\alpha_1 \equiv \beta$ und $\sigma_1 \equiv \alpha \to \alpha$. Dann erhält man die Bäume $\sigma[\beta := \alpha \to \alpha]$ und $\tau[\beta := \alpha \to \alpha]$:



Im nächsten Schritt genügt die Substitution $\gamma := \alpha$.

7.6 Verallgemeinerung. Gegeben seien einfache Typen $\sigma_1, \dots, \sigma_n$ und $\tau_1, \dots, \tau_n$. Wir sagen, das Gleichungssystem

$$(*) \qquad \sigma_1 = \tau_1, \quad \dots \quad , \sigma_n = \tau_n$$

habe eine Lösung, falls es eine Substitution

$$\mathbb{S} = [\alpha_1 := \varrho_1, \dots, \alpha_k := \varrho_k]$$

gibt, so daß

$$\sigma_1\mathbb{S} = \tau_1\mathbb{S}, \quad \dots \quad , \sigma_n\mathbb{S} = \tau_n\mathbb{S}$$

und $\mathbb{S}$ wird dann eine Lösung des Gleichungssystems genannt.

Das Problem, ein solches Gleichungssystem zu lösen, läßt sich auf das Problem zurückführen, eine einzelne Gleichung zu lösen. Setzt man nämlich

$$(**) \qquad \sigma :\equiv \sigma_1 \to \sigma_2 \to \dots \to \sigma_n \quad \text{und} \quad \tau :\equiv \tau_1 \to \tau_2 \to \dots \to \tau_n$$

dann ist $\mathbb{S}$ genau dann eine Lösung von $(*)$, wenn $\mathbb{S}$ eine Lösung von $(**)$ ist. Damit können wir den vorstehenden Algorithmus auf $\sigma \equiv \tau$ anwenden und damit feststellen, ob das Gleichungssystem $(*)$ überhaupt eine Lösung hat oder nicht, und gegebenenfalls eine „allgemeine Lösung“ bestimmen.

Die vorstehenden Überlegungen benötigt man für den folgenden Satz:

7.7 Satz. Es sei M ein geschlossener (ungetypter) λ-Term. Wenn M im Sinne von Curry überhaupt getypt werden kann, dann hat M einen „allgemeinen Typ“ σ, d.h. es gibt einen einfachen Typ σ, so daß $\vdash M : \sigma$ hergeleitet werden kann, und jeder andere Typ σ', mit dem M getypt werden kann, ist eine Instanz von σ, d.h. es gibt eine Substitution derart, daß

$$\sigma' \equiv \sigma[\alpha_1 := \sigma_1, \dots, \alpha_n := \sigma_n]$$

Genauer: Es gibt einen Algorithmus, der für jeden geschlossenen λ-Term M entscheidet, ob M getypt werden kann oder nicht, und der gegebenenfalls den „allgemeinen“ Typ von M liefert.

Eine ähnliche Aussage gilt auch für λ-Terme, die nicht geschlossen sind. Wir wollen den Algorithmus an Beispielen vorführen.

7.8 Beispiel. Bevor man die Terme betrachtet, benennt man gebundene Variablen so um, daß eine Variable in einem Term höchstens einmal gebunden wird (d.h. $(\lambda x.x)(\lambda x.x)$ müßte man in $(\lambda x.x)(\lambda y.y)$ umbenennen) und daß die gebundenen Variablen von allen freien Variablen des Terms verschieden sind.

- Der allgemeine Typ der Churchzahlen $\lambda yx.y^n x$ ist $(\alpha \to \alpha) \to (\alpha \to \alpha)$:

$$\dfrac{\dfrac{\begin{array}{c}\dfrac{y:\beta \qquad \dfrac{y:\beta \qquad x:\delta}{yx:\gamma}\,(1)}{y(yx):\alpha}\,(2)\\ \vdots \\ y^n x:\alpha\end{array}}{\lambda x.y^n x:\alpha\to\alpha}}{\lambda yx.y^n x:(\alpha\to\alpha)\to(\alpha\to\alpha)}$$

  Dabei mußten folgende Typgleichungen gelöst werden:
  (1) $\beta = \delta \to \gamma$. Diese löst man, indem man $\beta := \delta \to \gamma$ setzt.
  (2) $\delta \to \gamma = \gamma \to \alpha$. Diese löst man durch die Substitution $[\gamma := \delta, \alpha := \delta]$.

- $W \equiv \lambda xy.xyy$ hat den allgemeinen Typ $(\alpha \to \alpha \to \delta) \to \alpha \to \delta$:

$$\dfrac{\dfrac{\dfrac{\dfrac{x:\beta \qquad y:\alpha}{xy:\gamma}\,(1) \qquad y:\alpha}{xyy:\delta}\,(2)}{\lambda y.xyy:\alpha\to\delta}}{\lambda xy.xyy:(\alpha\to\alpha\to\delta)\to\alpha\to\delta}$$

  Dabei mußten folgende Typgleichungen gelöst werden:
  (1) $\beta = \alpha \to \gamma$. Diese löst man, indem man $\beta := \alpha \to \gamma$ setzt.
  (2) $\gamma = \alpha \to \delta$. Diese löst man wieder durch die entsprechende Setzung. Dabei muß man beachten, daß man dann $\beta = \alpha \to \alpha \to \delta$ erhält, denn in β kommt γ noch mal vor.

- $P \equiv \big(\lambda vxyz.v(y(vxz))\big)(\lambda w.w)$:

$$\dfrac{\dfrac{\dfrac{v:\alpha\to\delta\to\varepsilon \qquad \dfrac{y:\eta \qquad \dfrac{\dfrac{v:\beta \qquad x:\alpha}{vx:\gamma}\,(1) \qquad z:\delta}{vxz:\varepsilon}\,(2)}{y(vxz):\zeta}\,(3)}{v(y(vxz)):\delta\to\varepsilon}\,(4)}{\lambda vxyz.v(y(vxz)):(\alpha\to\delta\to\varepsilon)\to\alpha\to(\varepsilon\to\alpha)\to\delta\to\delta\to\varepsilon} \qquad \dfrac{w:\xi}{\lambda w.w:\xi\to\xi}}{\big(\lambda vxyz.v(y(vxz))\big)(\lambda w.w):(\delta\to\varepsilon)\to(\varepsilon\to\delta\to\varepsilon)\to\delta\to\delta\to\varepsilon}\,(5)$$

  Dabei mußten folgende Typgleichungen gelöst werden:
  (1) $\beta = \alpha \to \gamma$. Das geht einfach durch $\beta := \alpha \to \gamma$.
  (2) $\gamma = \delta \to \varepsilon$. Auch das geht einfach durch $\gamma := \delta \to \varepsilon$. Bemerke, daß dann $\beta = \alpha \to \delta \to \varepsilon$ gilt.

(3) $\eta = \varepsilon \to \zeta$. Wieder eine einfache Setzung ohne Auswirkungen.

(4) $\zeta = \alpha$. Da v den Typ $\beta \equiv \alpha \to \delta \to \varepsilon$ haben muß, muß man hier wieder eine Ersetzung durchführen.

(5) $\xi \to \xi = \alpha \to \delta \to \varepsilon$. Diese Gleichung löst man durch $\xi := \alpha := \delta \to \varepsilon$.

Im folgenden wird ein Algorithmus angegeben, der für jeden λ-Term M entscheidet, ob er getypt werden kann oder nicht, und der im ersten Fall die „allgemeine“ Typisierung (AT) liefert. Damit ist gemeint, daß jede andere Typisierung von M sich aus der AT durch eine geeignete Substitution $\mathbb{S} = [\alpha_1 := \sigma_1, \ldots, \alpha_n := \sigma_n]$ ergibt. Eine ausführliche Darstellung des Algorithmus samt seiner Begründung findet man ebenso wie vieles andere Wissenswerte über einfache Typen in dem Buch *Basic Simple Type Theory* von Hindley.

7.9 Typisierungsalgorithmus. Wir definieren die AT (= allgemeine Typisierung) eines ungetypten λ-Terms M rekursiv über den Termaufbau:

(1) Für $M \equiv x$ ist die AT
$$x : \alpha \vdash x : \alpha$$

(2) Ist $M \equiv \lambda x.P$ und ist $\Gamma \vdash P : \tau$ die AT von P, dann ist die AT von M gegeben durch:
$$\begin{aligned} \Gamma \setminus [x : \sigma] \vdash \lambda x.P : \sigma \to \tau \quad & \text{falls } x : \sigma \in \Gamma \\ \Gamma \vdash \lambda x.P : \alpha \to \tau \quad & \text{falls } x : \sigma \notin \Gamma \qquad (\alpha \text{ neue Typvariable}) \end{aligned}$$

(3) Ist $M \equiv PQ$ und sind $\Gamma_1 \vdash P : \sigma_1$ bzw. $\Gamma_2 \vdash Q : \sigma_2$ die ATs von P und Q, dann benenne die Typvariablen zunächst so um, daß keine in den beiden Typisierungen zugleich vorkommt. Dann betrachte das Gleichungssystem
$$\begin{aligned} \sigma_1 &= \sigma_2 \to \eta \\ \varrho_i &= \tau_i \end{aligned}$$
wobei η eine neue Typvariable ist und die Gleichungen $\varrho_i = \tau_i$ genau dann in das System aufgenommen werden, wenn es eine Variable x_i gibt, die sowohl in Γ_1 als auch in Γ_2 deklariert wird, und zwar $x_i : \varrho_i \in \Gamma_1$ und $x_i : \tau_i \in \Gamma_2$. Wenn dieses Gleichungssystem keine Lösung hat, dann kann M nicht getypt werden. Ansonsten bestimme man eine allgemeine Lösung $\mathbb{S}$ des Gleichungssystems. Die AT von M lautet dann
$$\Gamma_1\mathbb{S} \cup \Gamma_2\mathbb{S} \vdash M : \eta\mathbb{S}$$

Bemerkung. Die Variablenumbenennung im dritten Fall wird man in den Beispielen nicht immer durchführen. Sie ist nur dann erforderlich, wenn man bei den Typisierungen von P und Q ein und dieselbe Variable benutzt hat, ohne daß ein Zusammenhang zwischen beiden Benutzungen besteht.

Der Algorithmus zeigt, daß das Typisierungsproblem auf das Unifikationsproblem zurückgeführt wird.

KAPITEL IV

Denotationelle Semantik

Unser Ziel ist es, einem syntaktischem Kalkül wie dem untypisierten oder einfach typisierten λ-Kalkül ein mathematisches Modell zuzuordnen. Dabei soll jeder Term eine *Interpretation* in dem Modell bekommen. Insbesondere bekommen dann Applikation und Abstraktion eine Interpretation in dem Modell. Die Interpretation soll zumindest *korrekt* sein, d.h. z.B. daß β-äquivalente Terme durch dasselbe Element interpretiert werden. Ferner wünscht man sich *Vollständigkeit*, d.h. Terme, die in allen Modellen dieselbe Interpretation haben, sollen β-äquivalent sein.

Ein weiteres Ziel ist auch, Eigenschaften des Kalküls und von Termen semantisch, d.h. durch Überlegungen an den Modellen zu beweisen.

Allgemeine Voraussetzungen und Bezeichnungen.

- Σ sei die Menge der einfachen Typen über den atomaren Typen $\alpha_1, \alpha_2, \alpha_3, \ldots$.
- Wir betrachten einfach getypte λ-Terme im Sinne von Church III.1, d.h. Variable sind à priori mit einem Typ behaftet.
- Sind A, B Mengen, so bezeichne $(A \to B)$ die Menge aller Funktionen $f\colon A \to B$.

1. Modelle des einfach getypten λ-Kalküls

1.1 Mengenmodelle. Für jeden atomaren Typ α wählen wir eine Menge D_α. Für beliebige höhere Typen definieren wir induktiv $D_{\sigma\to\tau}$ als die Menge aller Funktionen von D_σ nach D_τ):

$$D_{\sigma\to\tau} := (D_\sigma \to D_\tau)$$

Bemerke: Nach **??** aus der Einleitung gilt

$$D_{\tau_1\to\tau_2\to\sigma} = (D_{\tau_1} \to (D_{\tau_2} \to D_\sigma)) \cong (D_{\tau_1} \times D_{\tau_2} \to D_\sigma)$$

oder allgemeiner:

$$D_{\tau_1\to\tau_2\to\cdots\to\tau_n\to\sigma} \cong (D_{\tau_1} \times D_{\tau_2} \times \cdots \times D_{\tau_n} \to D_\sigma)$$

Jeden Term M vom Typ σ wollen wir durch ein Element $[\![M]\!] \in D_\sigma$ interpretieren. Wenn M freie Variablen enthält, wird die Interpretation von M von der Interpretation dieser freien Variablen abhängen. Freie Variablen des Typs σ wollen wir durch beliebige Elemente von D_σ interpretieren:

Eine *Variablenbelegung* ist eine Funktion ϱ, die jeder Variablen x^σ vom Typ σ ein Element $\varrho(x^\sigma) \in D_\sigma$ zuordnet. Ist ϱ eine solche Variablenbelegung, dann bezeichne

$\varrho[y^\tau \mapsto a]$ die Belegung, die mit ϱ überall übereinstimmt, außer daß die Variable y^τ mit $a \in D_\tau$ belegt werden soll:

$$\big(\varrho[y^\tau \mapsto a]\big)(x^\sigma) = \begin{cases} \varrho(x^\sigma) & \text{falls } x^\sigma \not\equiv y^\tau \\ a & \text{falls } x^\sigma \equiv y^\tau \end{cases}$$

Nun definieren wir die Interpretation $[\![M]\!]_\varrho$ eines Terms M induktiv über den Termaufbau simultan für beliebige Variablenbelegungen ϱ:

(1) $[\![x^\sigma]\!]_\varrho := \varrho(x^\sigma) \in D_\sigma$

(2) Ist M vom Typ $\sigma \to \tau$, N vom Typ σ, und haben wir schon M und N in $D_{\sigma\to\tau}$ bzw. D_σ interpretiert, so definieren wir

$$[\![MN]\!]_\varrho := [\![M]\!]_\varrho\big([\![N]\!]_\varrho\big) \in D_\tau$$

(3) Ist M ein Term vom Typ τ und haben wir schon M in D_τ interpretiert, dann setzen wir:

$$[\![\lambda x^\sigma.M]\!]_\varrho := \big(a \mapsto [\![M]\!]_{\varrho[x^\sigma\mapsto a]} \colon D_\sigma \to D_\tau\big) \in D_{\sigma\to\tau}$$

Bemerkung. Der Ausdruck $[\![M]\!]_\varrho$ hängt nur von der Belegung der freien Variablen in M ab, d.h. $[\![M]\!]_\varrho = [\![M]\!]_{\varrho'}$, falls $\varrho(x_i^{\sigma_i}) = \varrho'(x_i^{\sigma i})$ für alle Variablen $x_i^{\sigma_i}$, die in M frei auftreten. (Den induktiven Beweis führe man als Übung durch.)

Insbesondere gilt: Ist M ein geschlossener Term des Typs σ, so ist $[\![M]\!]_\varrho$ ein Element von D_σ, das nicht von der Belegung ϱ abhängt. Wir können dann auch einfach $[\![M]\!]$ schreiben.

1.2 Beispiel.

- $[\![\lambda x^\sigma.x]\!] = \mathrm{id}_{D_\sigma} \colon D_\sigma \to D_\sigma$
- $[\![\lambda x^\sigma.y^\tau]\!]_\varrho = a \mapsto \varrho(y^\tau) \colon D_\sigma \to D_\tau$

  Dies ist die konstante Funktion mit dem Wert $\varrho(y^\tau)$.
- Wir berechnen die Interpretation von $\overline{n} \equiv \lambda y^{\sigma\to\sigma}x^\sigma.y^n x$:

$$\begin{aligned}
[\![\lambda y^{\sigma\to\sigma}x^\sigma.y^n x]\!] &= f \mapsto [\![\lambda x^\sigma.y^n x]\!]_{y\mapsto f} \colon (D_\sigma \to D_\sigma) \to (D_\sigma \to D_\sigma) \\
[\![\lambda x^\sigma.y^n x]\!]_{y\mapsto f} &= a \mapsto [\![y^n x]\!]_{y\mapsto f, x\mapsto a} \colon D_\sigma \to D_\sigma \\
[\![y^n x]\!]_{y\mapsto f, x\mapsto a} &= [\![y]\!]_{y\mapsto f,x\mapsto a}([\![y]\!]_{y\mapsto f,x\mapsto a}(\dots([\![y]\!]_{y\mapsto f,x\mapsto a}([\![x]\!]_{y\mapsto f,x\mapsto a}))\dots)) \\
&= f(f(\dots(f(a))\dots)) = (f\circ\dots\circ f)(a) \\
[\![\lambda x^\sigma.y^n x]\!]_{y\mapsto f} &= a \mapsto f^n(a) \colon D_\sigma \to D_\sigma = f\circ\dots\circ f \\
[\![\lambda y^{\sigma\to\sigma}x^\sigma.y^n x]\!] &= f \mapsto f^n \colon (D_\sigma \to D_\sigma) \to (D_\sigma \to D_\sigma)
\end{aligned}$$

- Als letztes Beispiel betrachten wir die Interpretation von $\mathsf{K}_{\tau,\sigma} \equiv \lambda y^\tau x^\sigma.y$:

$$\begin{aligned}
[\![\lambda y^\tau x^\sigma.y]\!] &= b \mapsto [\![\lambda x^\sigma.y]\!]_{y\mapsto b} \colon D_\tau \to D_{\sigma\to\tau} \\
&= b \mapsto (a \mapsto b) \colon D_\tau \to (D_\sigma \to D_\tau)
\end{aligned}$$

  Durch Uncurrying sieht man, daß

$$[\![\lambda y^\tau x^\sigma.y]\!] = (b,a) \mapsto b \colon D_\tau \times D_\sigma \to D_\tau$$

  durch die Projektion auf die erste Komponente interpretiert wird.

1.3 Applikative Strukturen. Ordnet man

(1) jedem Typ σ eine Menge A_σ

(2) jedem Paar von Typen σ, τ eine Abbildung $\mathrm{app}_{\sigma,\tau}\colon A_{\sigma\to\tau} \to (A_\sigma \to A_\tau)$

zu, so spricht man von einer *applikativen Struktur* $\mathcal{A} = (A_\sigma, \mathrm{app}_{\sigma,\tau})$. Eine solche wird *extensional* genannt, wenn die Abbildung $\mathrm{app}_{\sigma,\tau}$ für alle σ, τ injektiv ist.

Extensionalität bedeutet, daß zwei verschiedene Elemente $f, g \in A_{\sigma\to\tau}$ mittels $\mathrm{app}_{\sigma,\tau}$ als verschiedene Funktionen von A_σ nach A_τ wirken.

Als erstes Beispiel betrachten wir ein belieges Mengenmodell (siehe 1.1). Dort ist $D_{\sigma\to\tau} = (D_\sigma \to D_\tau)$. Man erhält eine extensionale applikative Struktur, indem man für $\mathrm{app}_{\sigma,\tau}$ stets die identische Abbildung wählt.

Bemerkung. Häufig wählt man $A_{\sigma\to\tau}$ als Teilmenge der Menge $(A_\sigma \to A_\tau)$ aller Funktionen von A_σ in A_τ. In diesem Fall wählt man für $\mathrm{app}_{\sigma,\tau}\colon A_{\sigma\to\tau} \to (A_\sigma \to A_\tau)$ die kanonische Injektion, und man erhält eine extensionale applikative Struktur.

1.4 Termmodelle.

(1) Für einen beliebigen Term M vom Typ σ bezeichnen wir mit $\langle M\rangle$ die β-Äquivalenzklasse aller zu M β-äquivalenten getypten Terme M'. Da sich nach III.2.3 der Typ eines Term bei β-Reduktion nicht verändert, haben all diese Terme M' ebenfalls den Typ σ. Wir bezeichnen mit T_σ die Menge aller β-Äquivalenzklassen von Termen vom Typ σ:

$$T_\sigma = \{\langle M\rangle \mid M \text{ ist getypter Term vom Typ } \sigma\}$$

Da $M \simeq_\beta M',\ N \simeq_\beta N' \implies MN \simeq_\beta M'N'$ nach I.2.7 gilt, können wir jedem $\langle M\rangle \in T_{\sigma\to\tau}$ eine Abbildung

$$\langle N\rangle \mapsto \langle MN\rangle\colon T_\sigma \to T_\tau$$

zuordnen, die wir $\mathrm{app}_{\sigma,\tau}\langle M\rangle$ nennen. Wir erhalten eine Abbildung

$$\mathrm{app}_{\sigma,\tau}\colon \langle M\rangle \mapsto \mathit{app}_{\sigma,\tau}\langle M\rangle\colon T_{\sigma\to\tau} \to (T_\sigma \to T_\tau)$$

Die so erhaltene applikative Struktur $\mathcal{T} = (T_\sigma, \mathrm{app}_{\sigma,\tau})$ nennen wir das *β-Termmodell.*

Diese applikative Struktur ist nicht extensional. Betrachte $Q_1 \equiv y^{\sigma\to\tau}$ und $Q_2 \equiv \lambda x^\sigma.y^{\sigma\to\tau}x^\sigma$. Dann gilt für alle $P \in D_\sigma$:

$$\big(\mathrm{app}_{\sigma,\tau}\langle y^{\sigma\to\tau}\rangle\big)(\langle P\rangle) = \langle y^{\sigma\to\tau}P\rangle$$

und

$$\big(\mathrm{app}_{\sigma,\tau}\langle \lambda x^\sigma.y^{\sigma\to\tau}x^\sigma\rangle\big)(\langle P\rangle) = \langle(\lambda x^\sigma.y^{\sigma\to\tau}x^\sigma)P\rangle = \langle y^{\sigma\to\tau}P\rangle$$

Es gilt also $\mathrm{app}_{\sigma,\tau}\langle Q_1\rangle = \mathrm{app}_{\sigma,\tau}\langle Q_2\rangle$, obwohl $\langle Q_1\rangle \neq \langle Q_2\rangle$, da es sich bei beiden Termen um β-Normalformen handelt. Die Abbildung $\mathrm{app}_{\sigma,\tau}$ ist also nicht injektiv.

(2) Das vorhergehende Beispiel kann man variieren, indem man mit $\langle M\rangle$ die $\beta\eta$-Äquivalenzklasse bezeichnet und mit S_σ die Menge aller $\beta\eta$-Äquivalenzklassen von Termen vom Typ σ. Wie vorhin erhält man eine applikative Struktur

$$\mathcal{S} = (S_\sigma, \mathrm{app}_{\sigma,\tau})$$

die wir das $\beta\eta$-Termmodell nennen. Diese applikative Stuktur ist extensional. Seien nämlich Q_1, Q_2 Terme vom Typ $\sigma \to \tau$ mit

$$\mathrm{app}_{\sigma,\tau}\langle Q_1\rangle = \mathrm{app}_{\sigma,\tau}\langle Q_2\rangle$$

Für alle Terme P vom Typ σ gilt dann $\langle Q_1\rangle\langle P\rangle = \langle Q_2\rangle\langle P\rangle$, d.h. $\langle Q_1P\rangle = \langle Q_2P\rangle$, und dies bedeutet $Q_1P \simeq_{\beta\eta} Q_2P$. Insbesondere kann man für P eine Variable x^σ wählen, die in Q_1 und Q_2 nicht frei vorkommt. Für diese gilt dann $Q_1x^\sigma \simeq_{\beta\eta} Q_2x^\sigma$. Es folgt

$$Q_1 \simeq_{\beta\eta} \lambda x^\sigma.Q_1x^\sigma \simeq_{\beta\eta} \lambda x^\sigma.Q_2x^\sigma \simeq_{\beta\eta} Q_2$$

und folglich $\langle Q_1\rangle = \langle Q_2\rangle$.

1.5 Henkin-Modelle. Es sei $(A_\sigma, \mathrm{app}\,\sigma,\tau)_{\sigma,\tau\in\Sigma}$ eine extensionale applikative Struktur. Wir versuchen nun – genau wie im Mengenmodell – eine Interpretation der getypten λ-Terme induktiv über den Termaufbau zu geben, wieder simultan für beliebige Variablenbelegungen $\varrho(x^\sigma) \in A_\sigma$:

(1) $[\![x^\sigma]\!]_\varrho := \varrho(x^\sigma)$

(2) Ist P vom Typ $\sigma \to \tau$, Q vom Typ σ, und haben wir schon P und Q in $A_{\sigma\to\tau}$ bzw. A_σ interpretiert, so definieren wir

$$[\![PQ]\!]_\varrho := \big(\mathrm{app}_{\sigma,\tau}[\![P]\!]_\varrho\big)\big([\![Q]\!]_\varrho\big) \in A_\tau$$

(3) Ist Q ein Term vom Typ τ und haben wir schon Q in A_τ interpretiert, dann setzen wir:

$$[\![\lambda x^\sigma.Q]\!]_\varrho := \big(a \mapsto [\![Q]\!]_{\varrho[x^\sigma\mapsto a]} \colon A_\sigma \to A_\tau\big)$$

Wir erhalten zwar auf diese Weise eine Interpretation des Terms $\lambda x^\sigma.Q$ als Funktion von $A_\sigma \to A_\tau$, jedoch nicht unbedingt als Element von $A_{\sigma\to\tau}$. Wenn wir aber noch verlangen:

$$(\mathrm{H}) \quad \left\{ \begin{array}{l} \text{Alle bei dieser induktiven Definition auftauchenden Funktionen} \\ \qquad\qquad a \mapsto [\![Q]\!]_{\varrho[x^\sigma\mapsto\alpha]} \colon A_\sigma \to A_\tau \\ \text{sind im Bild der Abbildung } \mathrm{app}_{\sigma,\tau} \colon A_{\sigma\to\tau} \to (A_\sigma \to A_\tau) \text{ enthalten.} \end{array} \right.$$

können wir 3 ersetzen durch:

3. $[\![\lambda x^\sigma.Q]\!]_\varrho := \mathrm{app}_{\sigma,\tau}^{-1}\big(a \mapsto [\![Q]\!]_{\varrho[x^\sigma\mapsto a]} \colon A_\sigma \to A_\tau\big)$

und wir nennen die extensionale applikative Struktur mit der Eigenschaft (H) ein *Henkin-Modell.*

Beispiel. Es ist klar daß die Mengenmodelle aus 1.1 Henkinmodelle sind. Auch das $\beta\eta$-Termmodell der $\beta\eta$-Äquivalenzklassen paßt in diesen Rahmen, wie wir zeigen wollen.

Betrachten wir also das $\beta\eta$-Termmodell $\mathcal{S} = (S_\sigma, \mathrm{app}_{\sigma,\tau})$ aus 1.4.2. Eine Variablenbelegung ϱ ordnet jeder Variablen x^σ die $\beta\eta$-Äquivalenzklasse eines Terms M vom Typ σ zu. Wir können also die Variablenbelegung ϱ so bekommen, daß wir zunächst jedem x^σ einen Term $r(x^\sigma)$ vom Typ σ zuordnen und dann $\varrho(x_\sigma) = \langle r(x^\sigma)\rangle$ setzen. Wir können nun die Interpretation eines Terms M unter der Variablenbelegung ϱ explizit angeben. Dabei bezeichnen wir mit

$$M[r] := M[x^{\sigma_1} := r(x^{\sigma_1}), \ldots, x^{\sigma_n} := r(x^{\sigma_n})]$$

den Term den wir aus M durch simultane Substitution $x^{\sigma_i} := r(x^{\sigma_i})$ für die freien Variablen x^{σ_i} in M erhalten:

1.6 Proposition. Das $\beta\eta$-Termmodell ist ein Henkin-Modell, und für alle Terme M gilt $[\![M]\!]_\varrho = \langle M[r]\rangle$.

Beweis. Wir beweisen die Proposition induktiv über den Termaufbau:

(1) $M \equiv x^\sigma$:

$$[\![x^\sigma]\!]_\varrho = \varrho(x^\sigma) = \langle r(x^\sigma)\rangle = \langle x^\sigma[x^\sigma := r(x^\sigma)]\rangle$$

(2) $M \equiv PQ$:

$$\begin{aligned} [\![PQ]\!]_\varrho &= \big(\mathrm{app}[\![P]\!]_\varrho\big)\big([\![Q]\!]_\varrho\big) \\ &= \big(\mathrm{app}\langle P[r]\rangle\big)\big(\langle Q[r]\rangle\big) && \text{nach Induktionsvoraussetzung} \\ &= \langle P[r]\ Q[r]\rangle \\ &= \langle (PQ)[r]\rangle \end{aligned}$$

(3) $M \equiv \lambda x^\sigma.Q$:
Für alle Terme N vom Typ σ gilt:

$$\begin{aligned} [\![Q]\!]_{\varrho[x^\sigma \mapsto \langle N\rangle]} &= \langle Q[r[x^\sigma := N]]\rangle && \text{nach Induktionsvoraussetzung} \\ &= \langle Q[r[x^\sigma := x^\sigma]]\ [x^\sigma := N]\rangle && (*) \\ &= \langle (\lambda x^\sigma.Q[r[x^\sigma := x^\sigma]])\ N\rangle \\ &= \langle ((\lambda x^\sigma.Q)[r])\ N\rangle \\ &= \langle (M[r])N\rangle \\ &= (\mathrm{app}\langle M[r]\rangle)(\langle N\rangle) \end{aligned}$$

Dies zeigt einerseits, daß die Funktion $\langle N\rangle \mapsto [\![Q]\!]_{\varrho[x^\sigma \mapsto \langle N\rangle]}$ im Bild von $\mathrm{app}_{\sigma,\tau}$ liegt und daß $[\![\lambda x^\sigma.Q]\!]_\varrho = \langle M[r]\rangle$. An der Stelle $(*)$ verwendeten wir, daß wir die Variable x^σ so umbenennen können, daß sie nicht mehr in den Termen $r(x^{\sigma_i})$ vorkommt, die für die freien Variablen in Q substituiert werden, so daß wir statt simultan auch sukzessive substituieren können.

2. Korrektheit und Vollständigkeit

2.1 Korrektheit. Ein Henkin-Modell wird *korrekt* genannt, wenn für getypte Terme M, N gilt:

$$M \simeq_{\beta\eta} N \implies [\![M]\!]_\varrho = [\![N]\!]_\varrho \quad \text{für alle Variablenbelegungen } \varrho$$

Korrektheitssatz. Für den $\lambda_{\beta\eta}$-Kalkül einfach getypter λ-Terme sind alle Henkin-Modelle korrekt.

Wie im Falle des ungetypten λ-Kalküls sind zwei getypte λ-Terme M und N genau dann $\beta\eta$-gleich, wenn sich dies aus den folgenden Axiomen und Regeln herleiten läßt:

$$(\alpha) \qquad \lambda x^\sigma.M = \lambda y^\sigma.\big(M[x := y]\big), \text{ falls } y^\sigma \notin \mathrm{FV}(M)$$

$$(\beta) \qquad (\lambda x^\sigma.M)N = M[x^\sigma := N]$$

$$(\eta) \qquad \lambda x^\sigma.Mx^\sigma = M, \text{ falls } x^\sigma \notin \mathrm{FV}(M)$$

$$(\varrho) \qquad M = M$$

$$(\sigma) \qquad M = N \implies N = M$$

$$(\tau) \qquad M = N,\ N = P \implies M = P$$

$$(\mu) \qquad M = M' \implies MN = M'N$$

$$(\nu) \qquad N = N' \implies MN = MN'$$

$$(\xi) \qquad M = M' \implies \lambda x^\sigma.M = \lambda x^\sigma.M'$$

Um den Korrektheitssatz zu beweisen, genügt es zu zeigen, daß diese Axiome und Regeln in jedem Henkin-Modell richtig bleiben, wenn man darin die Terme durch ihre Interpretation bei einer beliebigen Variablenbelegung ersetzt. Es ist klar, daß die Axiome (ϱ), (σ), (τ) der Gleichheit in jedem Modell gelten. Die Gültigkeit von (μ), (ν) und (ξ) folgt unmittelbar aus der Definition der Interpretation von Termen in Henkin-Modellen.

Beweis von (η).

$$\begin{aligned}
[\![\lambda x^\sigma.Mx]\!]_\varrho &= \mathrm{app}_{\sigma,\tau}^{-1}\left(a \mapsto [\![Mx]\!]_{\varrho[x\mapsto a]}\right) \\
&= \mathrm{app}_{\sigma,\tau}^{-1}\left(a \mapsto \mathrm{app}_{\sigma,\tau}([\![M]\!]_\varrho)([\![x]\!]_{x\mapsto a})\right) \quad \text{(hier benutzt man } x \notin \mathrm{FV}(M)) \\
&= \mathrm{app}_{\sigma,\tau}^{-1}\left(\mathrm{app}_{\sigma,\tau}([\![M]\!]_\varrho)\right) \\
&= [\![M]\!]_\varrho
\end{aligned}$$

□

Bevor wir die Regeln (α) und (β) beweisen können, benötigen wir noch den folgenden Hilfssatz:

2.2 Hilfssatz. Es gilt $[\![M[x := N]]\!]_\varrho = [\![M]\!]_{\varrho[x\mapsto[\![N]\!]_\varrho]}$.

Der Beweis ist eine Übungsaufgabe. □

Nun zum Beweis der beiden noch fehlenden Regeln des Korrektheitssatzes:

Beweis von (α). Sei y nicht frei in M. Dann gilt

$$\begin{aligned}
[\![\lambda y.(M[x := y])]\!]_\varrho &= \mathrm{app}_{\sigma,\tau}^{-1}\left(a \mapsto [\![M[x := y]]\!]_{\varrho[y\mapsto a]}\right) && \text{nach 2.2} \\
&= \mathrm{app}_{\sigma,\tau}^{-1}\left(a \mapsto [\![M]\!]_{\varrho[y\mapsto a][x\mapsto a]}\right) \\
&= \mathrm{app}_{\sigma,\tau}^{-1}\left(a \mapsto [\![M]\!]_{\varrho[x\mapsto a]}\right) && \text{da } y \notin \mathrm{FV}(M) \\
&= [\![\lambda x.M]\!]_{\varrho[x\mapsto a]} = [\![\lambda x.M]\!]_\varrho
\end{aligned}$$

□

Beweis von (β).

$$\begin{aligned}
[\![(\lambda x.M)N]\!]_\varrho &= (\mathrm{app}_{\sigma,\tau}[\![\lambda x.M]\!]_\varrho)([\![N]\!]_\varrho) \\
&= (\mathrm{app}_{\sigma,\tau}(\mathrm{app}_{\sigma,\tau}^{-1}(a \mapsto [\![M]\!]_{\varrho[x\mapsto a]})))([\![N]\!]_\varrho) \\
&= (a \mapsto [\![M]\!]_{\varrho[x\mapsto a]})([\![N]\!]_\varrho) \\
&= [\![M]\!]_{\varrho[x\mapsto [\![N]\!]_\varrho]} = [\![M[x := N]]\!]_\varrho && \text{nach 2.2}
\end{aligned}$$

□

Damit ist der Korrektheitssatz bewiesen.

2.3 Vollständigkeit. Ein Henkin-Modell wird *vollständig* genannt, wenn für beliebige λ-Terme M, N gilt:

$$[\![M]\!]_\varrho = [\![N]\!]_\varrho \text{ für jede Belegung } \varrho \implies M \simeq_{\beta\eta} N$$

Wenn wir ein Mengenmodell nehmen, in den ein atomarer Typ α durch eine endliche Menge D_α interpretiert wird, dann ist dieses Modell nicht vollständig, denn die Terme

$$\overline{n} := \lambda y^{\alpha\to\alpha} x^\alpha . y^n x \quad \text{mit } n = 2, 3, \ldots$$

sind in $\beta\eta$-Normalform und alle vom Typ $(\alpha \to \alpha) \to \alpha \to \alpha$. In $D_{(\alpha\to\alpha)\to\alpha\to\alpha} = ((D_\alpha \to D_\alpha) \to (D_\alpha \to D_\alpha))$ gibt es aber nur endlich viele Elemente. Also müssen unendlich viele der Terme $\overline{n}$ dieselbe Interpretation haben.

2.4 Vollständigkeitssatz. Es gibt vollständige Henkin-Modelle. Genauer:

(1) Das Termmodell 1.4.2 ist vollständig.

(2) Wählt man im Mengenmodell für jeden atomaren Typ α eine unendliche Menge D_α, dann ist dieses Mengenmodell vollständig.

Die zweite Aussage werden wir nicht beweisen. (Beweis siehe handgeschriebenes Skript.) Für den Beweis der ersten Aussage seien M und N Terme mit $[\![M]\!]_\varrho = [\![N]\!]_\varrho$ für alle Belegungen ϱ. Nach Proposition 1.6 folgt daraus $\langle M[r]\rangle = \langle N[r]\rangle$, wobei $\varrho(x^\sigma) = \langle r(x^\sigma)\rangle$ für jede Variable wie in 1.6. Wähle nun $r(x^\sigma) = x^\sigma$ für alle Variablen, d.h. $\varrho(x^\sigma) = \langle x^\sigma\rangle$) Dann ist $M[r] = M$ und $N[r] = N$. Es folgt $\langle M\rangle = \langle N\rangle$, d.h. $M \simeq_{\beta\eta} N$. □

3. Logische Prädikate und logische Relationen

Es sei $\mathcal{A} = (D_\sigma, \mathrm{app}_{\sigma,\tau})_{\sigma,\tau\in\Sigma}$ ein Henkin-Modell des einfach getypten λ-Kalküls. Im folgenden schreiben wir $\widehat{f}$ anstelle von $\mathrm{app}_{\sigma,\tau}(f)$.

3.1 Definition (Plotkin 1980). Ein *logisches Prädikat* auf $\mathcal{A}$ ist eine Familie

$$\mathcal{R} = (R_\sigma)_{\sigma\in\Sigma}$$

von Teilmengen $R_\sigma \subseteq D_\sigma$, die rekursiv wie folgt gebildet wird:

(1) R_α ist für jeden atomaren Typ α eine beliebig vorgegebene Teilmenge von D_σ.

(2) $R_{\sigma\to\tau} = \{f \in D_{\sigma\to\tau} \mid \widehat{f}(a) \in R_\tau \text{ für alle } a \in R_\sigma\}$. Das bedeutet, daß $f \in R_{\sigma\to\tau}$ genau dann, wenn $\widehat{f}(R_\sigma) \subseteq R_\tau$.

Beispiel.

- Wähle $R_\alpha = D_\alpha$ für alle atomaren Typen α. Dann gilt $R_\sigma = D_\sigma$ für alle $\sigma \in \Sigma$.
- Wähle $R_\alpha = \varnothing$ für alle atomaren Typen α. Dann wird
$$\begin{aligned} R_{\alpha\to\sigma} &= D_{\alpha\to\sigma} \\ R_{(\alpha\to\sigma)\to\beta} &= \varnothing \\ R_{\tau\to\alpha\to\sigma} &= D_{\tau\to\alpha\to\sigma} \end{aligned}$$
für beliebige Typen σ, τ und atomare Typen α, β.

3.2 Hauptsatz über logische Prädikate. Ist $\mathcal{R} = (R_\sigma)_{\sigma\in\Sigma}$ ein logisches Prädikat auf einem Henkin-Modell $\mathcal{A}$, so gilt

$$[\![M]\!] \in R_\sigma \quad \text{für jeden geschlossenen } \lambda\text{-Term des Typs } \sigma$$

Der Beweis wird durch Induktion über den Aufbau von M geführt. Zu diesem Zweck brauchen wir aber eine allgemeinere Behauptung, die Terme mit freien Variablen mitberücksichtigt. Wir werden also beweisen:

Behauptung. Für jeden λ-Term M vom Typ σ gilt $[\![M]\!]_\varrho \in R_\sigma$ für jede Variablenbelegung ϱ mit $\varrho(x^\tau) \in R_\tau$ für alle Variablen x^τ, die in M frei vorkommen.

Beweis. (1) $M \equiv x^\sigma$: Falls $\varrho(x^\sigma) \in R_\sigma$, so gilt in der Tat $[\![M]\!]_\varrho = \varrho(x^\sigma) \in R_\sigma$.

(2) $M \equiv N^{\sigma\to\tau}P^\sigma$: Sei $\varrho(x^\sigma) \in R_\sigma$ für jede in M frei vorkommende Variable x. Dann gilt
$$[\![M]\!]_\varrho = \widehat{[\![N]\!]_\varrho}\big([\![P]\!]_\varrho\big)$$
wobei $\widehat{[\![N]\!]_\varrho} \in R_{\sigma\to\tau}$ und $[\![P]\!]_\varrho \in R_\sigma$ nach Induktionsvoraussetzung. Nach der Definition von $R_{\sigma\to\tau}$ gilt dann auch $[\![M]\!]_\varrho \in R_\tau$.

(3) $M \equiv \lambda x^\sigma.N^\tau$: Sei $\varrho(x^\eta) \in R_\eta$ für alle in M frei vorkommenden Variablen x^η. Wir betrachten
$$[\![M]\!]_\varrho = \mathrm{app}_{\sigma,\tau}^{-1}\left(a \mapsto [\![N^\tau]\!]_{\varrho[x^\sigma\mapsto a]}\right) : D_\sigma \to D_\tau$$
Um zu sehen, daß $[\![M]\!]_\varrho \in R_{\sigma\to\tau}$, müssen wir zeigen, daß für alle $a \in R_\sigma$
$$\widehat{[\![M]\!]_\varrho}(a) \in R_\tau$$
gilt. Nun gilt aber für alle $a \in R_\sigma$
$$\widehat{[\![M]\!]_\varrho}(a) = [\![N^\tau]\!]_{\varrho[x^\sigma\mapsto a}$$
und letzteres ist in R_τ nach Induktionsvoraussetzung, denn $\varrho[x^\sigma \mapsto a]$ ist eine Variablenbelegung, die allen freien Variablen von N^τ, also insbesondere auch x^σ, Werte in R zuordnet.

□

Bemerkung. Den vorhergehenden Satz kann man auch so interpretieren, daß jedes logische Prädikat auf einem Henkin-Modell selbst wieder ein Henkin-Modell ist.

3.3 Folgerungen.

(1) Es gibt keinen geschlossenen λ-Term M von atomarem Typ α, denn für jeden solchen Term M müßte $[\![M]\!] \in R_\alpha$ für jede logische Relation $\mathcal{R} = (R_\sigma)_{\sigma\in\Sigma}$ gelten. Für eine solche kann man aber $R_\alpha = \varnothing$ wählen.

(2) Es gibt keinen geschlossenen Term vom Typ $(\alpha \to \sigma) \to \alpha$, denn wählt man wieder $R_\alpha = \varnothing$, so erhält man eine logische Relation mit $R_{(\alpha\to\sigma)\to\alpha} = \varnothing$.

(3) Bis auf $\beta\eta$-Äquivalenz gibt es nur einen geschlossenen λ-Term vom Typ $\alpha \to \alpha$, nämlich $\mathsf{I}_\alpha \equiv \lambda x^\alpha.x$:

Wir betrachten ein Mengenmodell, in dem die Grundmengen D_α unendlich sind. Es sei M ein geschlossener Term vom Typ $\alpha \to \alpha$ und $f := [\![M]\!]\colon D_\alpha \to D_\alpha$. Für jedes $a \in D_\alpha$ sei $\mathcal{R}^a$ die logische Relation mit $R^a_\alpha = \{a\}$. Es gilt dann

$$R^a_{\alpha\to\alpha} = \{g\colon D_\alpha \to D_\alpha \mid g(a) = a\}$$

Da $[\![M]\!] \in R^a_{\alpha\to\alpha}$, folgt $[\![M]\!](a) = a$, und dies für alle $a \in D_\alpha$. Folglich muß $[\![M]\!] = [\![\mathsf{I}_\alpha]\!]$ gelten, und aus der Vollständigkeit des Mengenmodells folgt $M \simeq_{\beta\eta} \mathsf{I}_\alpha$.

3.4 λ-Definierbarkeit. Es sei $\mathcal{A} = (D_\sigma, \mathrm{app}_{\sigma,\tau})$ ein Henkin-Modell. Ein Element $f \in D_\sigma$ wird *λ-definierbar* genannt, falls es einen geschlossenen λ-Term M vom Typ σ gibt, so daß $[\![M]\!] = f$.

Es stellt sich nun die Frage, ob man mithilfe der logischen Prädikate die λ-definierbaren Elemente von D_σ, $\sigma \in \Sigma$, kennzeichnen kann. In 3.3 haben wir mithilfe logischer Prädikate gezeigt, daß kein Element von D_α bzw. $D_{(\alpha\to\sigma)\to\alpha}$ darstellbar und in $D_{\alpha\to\alpha}$ nur das Element f mit $\widehat{f} = \mathrm{id}_{D_\alpha}$ λ-definierbar ist. Um weitere Aussagen treffen zu können, muß man zunächst den Begriff des logischen Prädikats erweitern.

3.5 Logische Relationen. Gegeben seien n Henkin-Modelle

$$\mathcal{A}^{(i)} = \left(D^{(i)}_\sigma, \mathrm{app}^{(i)}_{\sigma,\tau}\right) \quad i = 1, \ldots, n$$

Eine *logische Relation* zwischen $\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(n)}$ ist eine Familie $\mathcal{R} = (R_\sigma)_{\sigma\in\Sigma}$ von n-stelligen Relationen

$$R_\sigma \subseteq D^{(1)}_\sigma \times \cdots \times D^{(n)}_\sigma$$

die induktiv über den Typ σ wie folgt definiert werden:

(1) Wähle $R_\alpha \subseteq D^{(1)}_\alpha \times \cdots \times D^{(n)}_\alpha$ beliebig für jeden atomaren Typ α.

(2) $$\begin{aligned} R_{\sigma\to\tau} \;=\; &\{(f_1, \ldots, f_n) \in D^{(1)}_{\sigma\to\tau} \times \cdots \times D^{(n)}_{\sigma\to\tau} \\ &\mid \big(\widehat{f}_1(a_1), \ldots, \widehat{f}_n(a_n)\big) \in R_\tau \text{ für alle } (a_1, \ldots, a_n) \in R_\sigma\} \end{aligned}$$

Die logischen Prädikate sind gerade die einstelligen logischen Relationen. Im Falle $\mathcal{A}^{(1)} = \cdots = \mathcal{A}^{(n)} = \mathcal{A}$ spricht man auch von einer n-stelligen logischen Relation auf $\mathcal{A}$.

Bemerkung. Jede logische Relation $\mathcal{R}$ zwischen $\mathcal{A}^{(1)}, \ldots, A^{(n)}$ kann man als logisches Prädikat auf einem geeigneten Henkin-Modell $\mathcal{A}$ auffassen: Setze

$$D_\sigma := D^{(1)}_\sigma \times \cdots \times D^{(n)}_\sigma$$

und $\mathrm{app}_{\sigma,\tau}\colon D_{\sigma\to\tau} \to (D_\sigma \to D_\tau)$

$$D^{(1)}_{\sigma\to\tau} \times \cdots \times D^{(n)}_{\sigma\to\tau} \to (D^{(1)}_\sigma \times \cdots \times D^{(n)}_\sigma \to D^{(1)}_\tau \times \cdots \times D^{(n)}_\tau)$$

$$(f_1, \ldots, f_n) \mapsto \big((a_1, \ldots, a_n) \mapsto (\widehat{f}_1(a_1), \ldots, \widehat{f}_n(a_n))\big)$$

Insofern erhalten wir aus 3.2 sofort

3.6 Hauptsatz über logische Relationen. Ist $\mathcal{R} = (R_\sigma)_{\sigma\in\Sigma}$ eine logische Relation zwischen den Henkin-Modellen $\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(n)}$, so gilt für alle geschlossenen λ-Terme M vom Typ σ

$$\big(\llbracket M\rrbracket^{\mathcal{A}^{(1)}}, \ldots, \llbracket M\rrbracket^{\mathcal{A}^{(n)}}\big) \in R_\sigma$$

Im Falle $\mathcal{A}^{(1)} = \cdots = \mathcal{A}^{(n)} = \mathcal{A}$ bedeutet dies: Für jede n-stellige logische Relation $\mathcal{R} = (R_\sigma)_{\sigma\in\Sigma}$ auf einem Henkin-Modell $\mathcal{A}$ gilt $\big(\llbracket M\rrbracket, \ldots, \llbracket M\rrbracket\big) \in R_\sigma$ für alle geschlossenen Terme M vom Typ σ.

Daraus folgt

3.7 Notwendiges Kriterium für λ-Definierbarkeit. Sei $\mathcal{A} = (D_\sigma, \mathrm{app}_{\sigma,\tau})$ ein Henkin-Modell. Ist $f \in D_\sigma$ ein λ-definierbares Element, so gilt

$$(f, \ldots, f) \in R_\sigma$$

für jede n-stellige logische Relation $\mathcal{R} = (R_\sigma)_{\sigma\in\Sigma}$ auf $\mathcal{A}$.

Die vorstehende notwendige Bedingung ist nicht immer hinreichend für λ-Definierbarkeit. Wählt man beispielsweise ein Mengenmodell mit endlichen Grundmengen, so ist das vorstehende Kriterium sicher nicht hinreichend. In Spezialfällen kann man jedoch die Umkehrung auch zeigen. Vorab eine Definition:

3.8 Ordnung eines Typs. Die *Ordnung* $o(\sigma)$ eines Typs σ wird induktiv wie folgt erklärt:

(1) $o(\alpha) = 0$ für atomare Typen α

(2) $o(\sigma \to \tau) = \max\{o(\sigma) + 1, o(\tau)\}$

Beispiel.

- $o(\alpha \to \alpha) = 1$
- $o(\alpha \to \alpha \to \alpha) = 1$
- $o((\alpha \to \alpha) \to \alpha) = 2$

Insgesamt sind die Typen σ mit $o(\sigma) \leq 1$ von der Form

$$\sigma \equiv \alpha_1 \to \alpha_2 \to \cdots \to \alpha_n$$

wobei die α_i für $i = 1, \ldots, n$ atomare Typen sind. Entsprechend sind die Typen σ mit $o(\sigma) \leq 2$ von der Form

$$\sigma \equiv \tau_1 \to \tau_2 \to \cdots \to \tau_n \to \alpha$$

wobei die $o(\tau_i) \leq 1$ für $i = 1, \ldots, n$ und α ein atomarer Typ ist.

Der Einfachheit halber beschränken wir uns auf einen einzigen atomaren Typ α. In dem Mengenmodell mit Grundmenge D_α werden die Terme wie folgt interpretiert:

- die Terme vom Typ der Ordnung 0 als Elemente $a \in D_\alpha$,

- die Terme vom Typ $\tau \equiv \alpha \to \cdots \to \alpha \to \alpha$ als Funktionen
$$g \in (D_\alpha \to (D_\alpha \to \cdots \to (D_\alpha \to D_\alpha)\ldots))$$
bzw. in der uncurry-Version als k-stellige Operation
$$g\colon D_\alpha \times \cdots \times D_\alpha \to D_\alpha$$
- die Terme vom Typ $\sigma \equiv \tau_1 \to \cdots \to \tau_m \to \alpha$, wobei $\tau_i \equiv \alpha \to \cdots \to \alpha \to \alpha$, als „Funktionale"
$$f \in (D_{\tau_1} \to (D_{\tau_2} \to \cdots \to (D_{\tau_m} \to D_\alpha)\ldots))$$
bzw. in der uncurry-Version als
$$f\colon D_{\tau_1} \times \cdots \times D_{\tau_m} \to D_\alpha$$

Es gilt nun die folgende teilweise Umkehrung von 3.7:

3.9 Satz (Plotkin 1980). Wir betrachten einfache Typen über einem einzigen atomaren Typ α und das Mengenmodell $\mathcal{A}$ über einer *unendlichen* Grundmenge D_α. Für jeden Typ σ mit der Ordnung $o(\sigma) \leq 2$ gilt dann:

$f \in D_\sigma$ ist λ-definierbar $\iff$ $f \in P_\sigma$ für jedes logische Prädikat $\mathcal{P} = (P_\sigma)_{\sigma\in\Sigma}$ auf $\mathcal{A}$ und $(f,f) \in R_\sigma$ für jede zweistellige logische Relation $\mathcal{R} = (R_\sigma)_{\sigma\in\Sigma}$ auf $\mathcal{A}$.

Wie wir in 3.7 gesehen haben, ist „$\Rightarrow$" eine Folgerung aus den Hauptsätzen 3.2 und 3.6.

Beweis von 3.9 „$\Leftarrow$". siehe handgeschriebenes Skript □

3.10 Bemerkungen zum Beweis.

(1) Sei $\sigma \equiv \tau_1 \to \cdots \to \tau_m \to \alpha$. Dann gilt für ein logisches Prädikat $(P_\sigma)_{\sigma\to\Sigma}$:
$$f \in P_\sigma \iff fh_1\ldots h_m \in P_\alpha \quad \text{für alle } (h_1,\ldots,h_m) \in P_{\tau_1} \times \cdots \times P_{\tau_m}$$

(2) Sei $\sigma \equiv \tau_1 \to \cdots \to \tau_m \to \alpha$. Sind alle τ_i der Ordnung 1, so gibt es keinen geschlossenen Term des Typs σ, denn die Menge $\mathcal{S}_\alpha$ im Beweis ist dann leer. Es ist dann auch $P_\alpha = \varnothing$. Es kann also, wie die erste Zeile dieser Seite (???) zeigt, gar kein f geben, das die Voraussetzungen des Satzes in diesem Fall erfüllt. Ist mindestens eines der τ_i von der Ordnung 0, d.h. $\tau_i \equiv \alpha$, dann ist $y_i^{\tau_i} = y_i^\alpha \in \mathcal{S}_\alpha^i$, und $\mathcal{S}_\alpha$ ist nicht leer.

(3) Plotkin konnte seinen Satz auf Typen beliebiger Ordnung verallgemeinern. Allerdings benötigte er dazu einen verallgemeinerten Begriff der logischen Relation.

KAPITEL V

PCF – Programming Language for Computable Functionals

Der ungetypte λ-Kalkül ist ausdrucksstark: in ihm läßt sich jede rekursive Funktion simulieren. Allerdings hat man hier wenig Möglichkeiten, die Eigenschaften wie z.B. Korrektheit eines Programms zu verifizieren, da es keine auf der Hand liegende Semantik bzw. kein einfaches Modell für ihn gibt.

Der einfach getypte λ-Kalkül hat viele (natürliche) Modelle und eine recht gute Semantik, auch wenn hier nicht alle Probleme geklärt sind. Aber dieser Kalkül ist viel zu ausdrucksschwach. Es lassen sich nur ganz einfache Funktionen, wie Polynomfunktionen der Form

$$n \mapsto a_0 + a_1 n + \cdots + a_k n^k$$

darstellen, siehe Aufgabe 52 bzw. Schwichtenberg, 1976.

Um den einfach typisierten λ-Kalkül ausdrucksstärker zu machen, muß man Konstanten zusätzlich einbauen, die es erlauben, die Konstruktion zur Darstellung der rekursiven Funktionen im untypisierten Fall auch im typisierten Fall nachzuspielen. Wir folgen hier einem Vorschlag von Plotkin (1976), der im wesentlichen in einer Arbeit von Scott (1969) vorgezeichnet wurde. Diese Arbeit von Scott wurde erst 1993 veröffentlicht.

Zuvor einige allgemeine Bemerkungen: Was wir bislang betrieben haben, nennt man auch *reinen* λ-Kalkül. Im ungetypten Fall baut man Terme aus Variablen x mittels Applikation und Abstraktion zusammen:

$$M ::= x \mid MM \mid \lambda x.M$$

Im *angewandten* λ-Kalkül erlaubt man eine zusätzliche Menge C von Konstanten, die wir mit c bezeichnen. Terme werden hier aus Variablen und Konstanten mittels Applikation und Abstraktion zusammengebaut:

$$M ::= x \mid c \mid MM \mid \lambda x.M$$

Der einzige Unterschied zwischen Variablen und Konstanten ist also, daß man über Konstanten nicht abstrahieren darf.

Im einfach typisierten Fall wird man neben Variablen x^σ für jeden Typ σ auch noch – zumindest für einige Typen σ – eine Menge C_σ von Konstanten vorgeben und mit c^σ bezeichnen. Die getypten Terme werden dann nach folgendem Schema aus Variablen und Konstanten durch Applikation und Abstraktion gebildet:

$$M^\sigma ::= x^\sigma \mid c^\sigma \mid M^{\sigma\to\tau} M^\sigma : \tau \mid \lambda x^\sigma . M^\tau : \sigma \to \tau$$

Neben der β-Reduktion wird man zusätzliche Reduktionsregeln für die Konstanten fordern, in denen sich ausdrückt, welche gedachten Funktionen diese Konstanten darstellen sollen.

1. Die Syntax von PCF

1.1 PCF-Typen. PCF ist eine einfach getypte Sprache mit zwei atomaren Typen:

ν Typ der natürlichen Zahlen $\mathbb{N} = \{0, 1, 2, \dots\}$
β Typ der Wahrheitswerte $\mathbb{B} = \{\text{true}, \text{false}\}$

Die *Typen von PCF* werden wie im einfach getypten λ-Kalkül induktiv definiert:

(1) Die atomaren Typen ν und β sind Typen.

(2) Sind σ und τ Typen, dann ist auch $\sigma \to \tau$ ein Typ.

Auch die Klammerkonventionen werden übernommen.

1.2 PCF-Terme. Um Terme zu bauen, gehen wir von abzählbar vielen *Variablen* x^σ zu jedem Typ σ aus. Des weiteren geben wir uns folgende *Konstanten* vor:

$k_n : \nu$	für jedes $n \in \mathbb{N}$ eine Konstante vom Typ ν
$\mathsf{TRUE} : \beta$	zwei Konstanten vom Booleschen Typ
$\mathsf{FALSE} : \beta$	
$\mathsf{ISNULL} : \nu \to \beta$	
$\mathsf{SUCC} : \nu \to \nu$	
$\mathsf{PRED} : \nu \to \nu$	
$\mathsf{COND}_\sigma : \beta \to \sigma \to \sigma \to \sigma$	für jeden Typ σ
$\mathsf{Y}_\sigma : (\sigma \to \sigma) \to \sigma$	für jeden Typ σ

Wir benutzen die Buchstaben $c, d, \dots$ als Bezeichner für diese Konstanten.

Die *PCF-Terme* werden nun induktiv wie folgt definiert:

(1) Variablen x^σ und Konstanten c^σ sind Terme und haben den jeweils angegebenen Typ.

(2) Ist M ein Term vom Typ $\sigma \to \tau$ und N ein Term vom Typ σ, so ist (MN) ein Term vom Typ τ.

(3) Ist x^σ eine Variable und M ein Term vom Typ τ, so ist $(\lambda x^\sigma.M)$ ein Term vom Typ $\sigma \to \tau$.

Es gelten die selben Klammerkonventionen wie im ungetypten λ-Kalkül.

Beispiel.

- Sind $M : \beta$, $N : \sigma$ und $P : \sigma$ PCF-Terme, so können wir den PCF-Term $\mathsf{COND}_\sigma\, MNP : \sigma$ bilden. Dafür schreiben wir auch

  $$\textbf{if } M \textbf{ then } N \textbf{ else } P$$

- $\mathsf{ISNULL}\, k_n : \beta$
- $\mathsf{SUCC}\, k_n : \nu$
- $\mathsf{PRED}\, k_n : \nu$

2. Operationale Semantik von PCF

In den folgenden Reduktionsregeln schlägt sich die intendierte Bedeutung der Konstanten nieder. Dazu kommt die β-Reduktion für β-Redexe. Allerdings lassen wir Reduktionen nicht in beliebigen Kontexten zu: Zwei Regeln werden angeben, in welchem Kontext man reduzieren darf.

2.1 Reduktionsregeln. Im folgenden sollen alle auftretenden Terme wohlgetypt sein. Zunächst zu den Regeln, die angeben, welche (Teil-)Terme wie reduziert werden können (Typ I):

$$\begin{aligned} \mathsf{ISNULL}\, k_0 &\to \mathsf{TRUE} & \qquad \mathsf{ISNULL}\, k_{n+1} &\to \mathsf{FALSE} \\ \mathsf{SUCC}\, k_n &\to k_{n+1} & \mathsf{PRED}\, k_{n+1} &\to k_n \\ \mathsf{COND}\,\mathsf{TRUE}\, NP &\to N & \mathsf{COND}\,\mathsf{FALSE}\, NP &\to P \\ \mathsf{Y}\, M &\to M(\mathsf{Y}\, M) \\ (\lambda x.M)N &\to M[x := N] \end{aligned}$$

Allerdings darf – wie bereits erwähnt – nicht in jedem Kontext reduziert werden. An welchen Stellen eine Reduktion möglich ist, geben die Regeln zur *Kontextverträglichkeit* an (Typ II):

$$\frac{M \to M'}{MN \to M'N} \qquad\qquad \frac{M \to M'}{cM \to cM'}\ (\text{für } c = \mathsf{SUCC}, \mathsf{PRED}, \mathsf{COND}, \mathsf{ISNULL})$$

Wir schreiben $\to^*$ für die reflexive transitive Hülle von $\to$, d.h. $M \to^* N$, falls es Terme $M \equiv M_0, M_1, \ldots, M_n \equiv N$ gibt, so daß

$$M \equiv M_0 \to M_1 \to \ldots \to M_n \equiv N$$

In diesem Fall sagen wir, daß M zu N reduziert werden kann.

2.2 Bemerkungen.

- Der Typ eines Terms ändert sich durch eine Reduktion nicht.
- Ein gegebener Term erlaubt höchstens eine der Reduktionen vom Typ I oder II. Wir haben also ein deterministisches Reduktionssystem vorliegen, d.h. ein Term hat genau eine Reduktionskette, die allerdings unendlich lang sein kann, wie das folgende Beispiel zeigt:

$$\mathsf{Y}_\nu(\lambda x^\nu.\, \mathsf{SUCC}\, x) \to (\lambda x^\nu.\, \mathsf{SUCC}\, x)(\mathsf{Y}_\nu(\lambda x^\nu.\, \mathsf{SUCC}\, x)) \to \mathsf{SUCC}(\mathsf{Y}_\nu(\lambda x^\nu.\, \mathsf{SUCC}\, x))$$

  Mit $M \equiv \lambda x^\nu.\, \mathsf{SUCC}\, x$ gilt also

$$\mathsf{Y}\, M \to M(\mathsf{Y}\, M) \to \mathsf{SUCC}(\mathsf{Y}\, M) \to \mathsf{SUCC}(M(\mathsf{Y}\, M)) \to \mathsf{SUCC}(\mathsf{SUCC}(\mathsf{Y}\, M)) \to \ldots$$

  Obwohl wir also getypte Terme haben, gibt es nichtterminierende Reduktionen. Das liegt an der Reduktionsregel für Y.
- Unter den Termen, die keine Reduktion mehr zulassen, sind
  - Variablen x^σ und Konstanten c
  - Terme der Gestalt $\mathsf{COND}_\sigma\,\mathsf{TRUE}$, $\mathsf{COND}_\sigma\,\mathsf{FALSE}$, $\mathsf{COND}_\sigma\,\mathsf{TRUE}\, M$ und $\mathsf{COND}_\sigma\,\mathsf{FALSE}\, M$

– alle Terme, die mit einer Variablen beginnen

– alle Terme der Form $\lambda x^\sigma.M$

Terme dieser Gestalt werden auch *schwache Kopfnormalformen* genannt. Dies ist eine wesentlich schwächere Eigenschaft als β-Normalform oder Kopfnormalform.

- Die Reduktionsstrategie ist die sogenannte *lazy evaluation*. Sie läßt sich folgendermaßen begründen: PCF-Terme sind Programme. Wenn man Reduktionen durchführt, ist das Ziel, als Ausgabe ein Ergebnis zu bekommen. Ein Ergebnis ist eine natürliche Zahl oder ein Boolescher Wert. Es ist für die Praxis nicht das Ziel, einen Term in β-Normalform zu überführen!

  Deshalb wird man Terme der Gestalt $\lambda x^\sigma.M$ nicht reduzieren wollen, denn ein solcher Term stellt eine Funktion dar. Erst wenn die Funktion ein Argument bekommt, wird man weiter reduzieren wollen, denn dann erst besteht die Möglichkeit, als Ergebnis eine natürliche Zahl oder einen Wahrheitswert zu erhalten.

  Aus demselben Grund wird man Terme, die mit einer Variablen beginnen, nicht reduzieren wollen, denn die Variable am Anfang wird immer stehen bleiben und nie spezifiziert werden.

  Ein weiterer Grund für diese Strategie ist die Reduktionsregel für Y: Würde man nach der Reduktion $\mathsf{Y}\,M \to M(\mathsf{Y}\,M)$ im hinteren Argument weiter reduzieren, würde man niemals terminieren.

  Beim ungetypten λ-Kalkül wäre diese Strategie zu restriktiv, denn dort waren natürliche Zahlen und Boolesche Werte (also die Ergebnistypen) durch gewisse β-Normalformen simuliert. Dort muß man zumindest die leftmost Reduktionsstrategie zulassen, z.B.

$$\lambda yx.(\lambda z.z)y^2x \to_\beta \lambda yx.y^2x \equiv \overline{2}$$

  In PCF wäre diese Reduktion unzulässig.

- Für beliebe PCF-Terme M definieren wir

$$\mathrm{eval}(M) := \begin{cases} N & \text{falls man } M \text{ in endlich vielen Schritten zu } N \text{ reduzieren kann und } N \text{ keine Reduktion zuläßt} \\ \text{undefiniert} & \text{falls die Reduktionskette von } M \text{ nicht terminiert} \end{cases}$$

  Wir interessieren uns insbesondere für den Fall, daß M ein geschlossener Term vom Typ ν oder β ist. Dann gilt

$$\mathrm{eval}(M) := \begin{cases} k_n \text{ oder } \mathsf{TRUE} \text{ oder } \mathsf{FALSE} & \text{falls die Reduktion von } M \text{ terminiert} \\ \text{undefiniert} & \text{sonst} \end{cases}$$

3. Denotationelle Semantik von PCF: Vorbereitung

3.1 Naiver Ansatz. Um eine denotationelle Semantik von PCF anzugeben, versuchen wir so vorzugehen, wie wir das in Kapitel IV für den einfach getypten λ-Kalkül getan haben. Da wir für die atomaren Typen ν und β eine feste Bedeutung im Sinn haben, setzen wir

$$D_\nu = \mathbb{N} \qquad\qquad D_\beta = \mathbb{B}$$

und definieren für Typen höherer Ordnung

$$D_{\sigma\to\tau} = (D_\sigma \to D_\tau)$$

als die Menge aller Funktionen $f\colon D_\sigma \to D_\tau$.

PCF-Terme interpretieren wir wieder induktiv über den Termaufbau simultan für beliebige Variablenbelegungen ϱ. Dabei sollen die Konstanten die intendierte Bedeutung erhalten:

(1)

$$\begin{aligned}
[\![x^\sigma]\!]_\varrho &= \varrho(x^\sigma)\\
[\![k_n]\!]_\varrho &= n\\
[\![\mathsf{TRUE}]\!]_\varrho &= \text{true}\\
[\![\mathsf{FALSE}]\!]_\varrho &= \text{false}\\
[\![\mathsf{ISNULL}]\!]_\varrho &\colon \mathbb{N}\to\mathbb{B} && n \mapsto \begin{cases}\text{true} & \text{falls } n=0\\ \text{false} & \text{falls } n\neq 0\end{cases}\\
[\![\mathsf{SUCC}]\!]_\varrho &\colon \mathbb{N}\to\mathbb{N} && n\mapsto n+1\\
[\![\mathsf{COND}_\sigma]\!]_\varrho &\colon \mathbb{B}\times D_\sigma\times D_\sigma \to D_\sigma && (w,a,b)\mapsto \begin{cases} a & \text{falls } w=\text{true}\\ b & \text{falls } w = \text{false}\end{cases}
\end{aligned}$$

(2) $[\![M^{\sigma\to\tau}N^\sigma]\!]_\varrho = [\![M^{\sigma\to\tau}]\!]_\varrho\big([\![N^\sigma]\!]_\varrho\big) \in D_tau$

(3) $[\![\lambda x^\sigma.M^\tau]\!]_\varrho = (a\mapsto [\![M^\tau]\!]_{\varrho[x\mapsto a]})\colon D_\sigma \to D_\tau$

Ausgelassen haben wir hier die Interpretation von PRED und Y, denn mit diesen bekommen wir Schwierigkeiten:

Im Fall von PRED ist das Problem nicht ernsthaft: die Vorgängerfunktion

$$\text{pred}\colon \mathbb{N}\to\mathbb{N},\ \ n\mapsto n-1$$

ist eigentlich eine partielle Funktion, die für $n=0$ nicht definiert ist. In $D_{\nu\to\nu} = (D_\nu \to D_\nu)$ haben wir aber nur totale Funktionen zugelassen. Man kann das Problem dadurch lösen, daß man $\text{pred}(0) = 0$ setzt. Eine alternative Lösung wäre, in $D_{\nu\to\nu}$ auch partielle Funktionen zuzulassen. Der Nachteil ist hier, daß der Umgang mit partiellen Funktionen umständlich wird und man sehr sorgfältig hinsehen muß: Wie erklärt man Komposition partieller Funktionen, was sind partielle Funktionen von Typen höherer Ordnung etc.

3.2 Neuer Ansatz. Ein Ausweg aus dem Dilemma besteht nun darin, in der Semantik ein Zeichen $\perp$ (lies: bottom) für „nicht definiert“ einzuführen und dies den semantischen Bereichen $\mathbb{N}$ und $\mathbb{B}$ hinzuzufügen:

$$\mathbb{N}_\perp = \mathbb{N}\cup\{\perp\} \qquad\qquad \mathbb{B}_\perp = \mathbb{B}\cup\{\perp\}$$

Für jede partielle Funktion $f\colon \mathbb{N} \rightharpoonup \mathbb{N}$ mit dem Definitionsbereich $D_f \subseteq \mathbb{N}$ können wir dann die überall definierte Funktion $\tilde{f}\colon \mathbb{N}_\perp \to \mathbb{N}_\perp$ mit

$$\tilde{f}(x) = \begin{cases} f(x) & \text{falls } x \in D_f \\ \perp & \text{sonst} \end{cases}$$

benutzen. Insbesondere haben wir dann eine adäquate Interpretation für PRED:

$$[\![\mathsf{PRED}]\!]_\varrho = \mathrm{pred} \quad \text{wobei} \quad \mathrm{pred}(x) = \begin{cases} x-1 & \text{für } 0 \neq x \in \mathbb{N} \\ \perp & \text{für } x = 0 \text{ oder } x = \perp \end{cases}$$

Wenn wir nun in 3.1 bei der Interpretation der Typen $\mathbb{N}$ durch $\mathbb{N}_\perp$ und $\mathbb{B}$ durch $\mathbb{B}_\perp$ ersetzen, so geht alles andere gut. Allerdings handelt man sich mehr ein, als man zunächst will: Wählt man nämlich dann

$$D_{\nu\to\nu} = (D_\nu \to D_\nu) = (\mathbb{N}_\perp \to \mathbb{N}_\perp)$$

d.h. $D_{\nu\to\nu}$ als die Menge aller (totalen) Funktionen $g\colon \mathbb{N}_\perp \to \mathbb{N}_\perp$, so hat man sehr viel mehr als die Erweiterungen $\tilde{f}$ der partiellen Funktionen $f\colon D_f \to \mathbb{N}$. Ein Element aus $D_{\nu\to\nu}$ wäre z.B.

$$g\colon \mathbb{N}_\perp \to \mathbb{N}_\perp \quad \text{mit} \quad g(x) = \begin{cases} \perp & \text{für } x \in \mathbb{N} \\ 4 & \text{für } x = \perp \end{cases}$$

Dies wird noch drastischer, wenn man zu höheren Typen übergeht. Daher werden wir versuchen, die Menge der zulässigen Funktionen $g\colon \mathbb{N}_\perp \to \mathbb{N}_\perp$ einzuschränken.

3.3 Zur Interpretation von Y. Durch die Einführung von Y und der Reduktionsregel $\mathsf{Y}\,M \to M(\mathsf{Y}\,M)$ handelt man sich nicht terminierende Terme wie z.B. $\mathsf{Y}_\nu\,\mathsf{SUCC}$ ein. Wie sollen solche nicht terminierende Terme interpretiert werden?

Der Term $\mathsf{Y}_\nu\,\mathsf{SUCC}$ hat den Typ ν. Sicher hat es keinen Sinn, ihn durch ein $n \in \mathbb{N}$, also durch eine natürliche Zahl, zu interpretieren. Wir könnten versuchen, nicht terminierende Terme durch $\perp$ zu interpretieren, also z.B.

$$[\![\mathsf{Y}_\nu\,\mathsf{SUCC}]\!]_\varrho = \perp$$

Nur: Wie soll Y_ν interpretiert werden?

Da Y_ν vom Typ $(\nu \to \nu) \to \nu$ ist, muß $[\![\mathsf{Y}_\nu]\!] \in D_{(\nu\to\nu)\to\nu}$ sein, d.h.

$$[\![\mathsf{Y}_\nu]\!]\colon D_{(\nu\to\nu)} \to D_\nu$$

Da unsere Interpretation korrekt sein soll, heißt das aufgrund der Reduktionsregel für Y_ν für alle M vom Typ $\nu \to \nu$:

$$[\![Y_\nu]\!]\big([\![M]\!]\big) = [\![M]\!]\big([\![Y_\nu]\!]([\![M]\!])\big)$$

Sei also $f = [\![M]\!]\colon D_\nu \to D_\nu$. Dann muß $[\![Y_\nu]\!](f) = f\big([\![Y_\nu]\!](f)\big)$ gelten, d.h. $[\![Y_\nu]\!]\big(f\big)$ ist ein Fixpunkt von f. Wir suchen also ein Funktional

$$\mathbb{Y}\colon D_{\nu\to\nu} \to D_\nu$$

das jeder Funktion $f \in D_{\nu\to\nu}$ einen Fixpunkt von f zuordnet.

Dies ist sicher nicht möglich, wenn wir für $D_{\nu\to\nu}$ die Menge $(\mathbb{N}_\perp \to \mathbb{N}_\perp)$ aller Funktionen $f\colon \mathbb{N}_\perp \to \mathbb{N}_\perp$ wählen, denn sehr viele dieser Funktionen haben keinen Fixpunkt.

Daher auch hier das Problem, für $D_{\nu\to\nu}$ eine eingeschränkte Klasse von Funktionen so zu wählen, daß jede Funktion einen „kanonischen“ Fixpunkt hat.

4. Bereiche

Hier wollen wir nun die notwendigen ordnungstheoretischen Hilfsmittel bereitstellen, um eine Semantik für PCF bereitstellen zu können.

4.1 Geordnete Mengen. Sei D eine Menge. Jede reflexive, antisymmetrische und transitive Relation $\sqsubseteq$ auf D, d.h. für alle $x, y, z \in D$ gelten

$$(\varrho) \qquad x \sqsubseteq x$$

$$(\sigma) \qquad x \sqsubseteq y,\ y \sqsubseteq x \implies x = y$$

$$(\tau) \qquad x \sqsubseteq y,\ y \sqsubseteq z \implies x \sqsubseteq z$$

nennen wir eine *Ordnung* auf D. Als *geordnete Menge* (poset, partially ordered set) bezeichnen wir jedes Paar $(D, \sqsubseteq)$, bestehend aus einer Menge D und einer Ordnung $\sqsubseteq$ auf D.

Beispiel.

- *Flache Bereiche.* Für eine beliebige Menge M sei $M_\perp := M \cup \{\perp\}$, und wir definieren für $x, y \in M_\perp$

$$x \sqsubseteq y \iff x = y \text{ oder } x = \perp$$

- *Potenzmengenverbände.* Für eine beliebige Menge M sei $\mathfrak{P}(M)$ die Menge aller Teilmengen von M. Die Inklusionsbeziehung $\subseteq$ ist eine Ordnung auf $\mathfrak{P}(M)$.

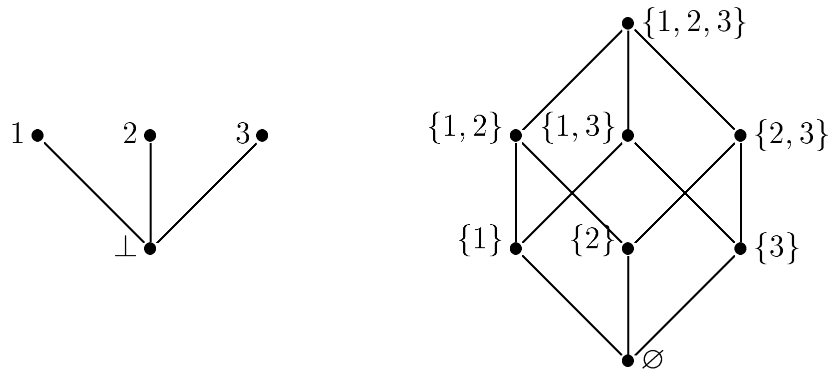


ABBILDUNG 1. $M_\perp$ und $\mathfrak{P}(M)$ für $M = \{1, 2, 3\}$

Wir benutzen den Begriff Ordnung im Sinne einer „teilweisen Ordnung“, d.h. es kann unvergleichbare Elemente geben:

Definition. Zwei Elemente a, b einer geordneten Menge werden *vergleichbar* genannt, falls $a \sqsubseteq b$ oder $b \sqsubseteq a$. Ansonsten nennt man sie *unvergleichbar*, und man schreibt manchmal $a \parallel b$.

Bemerkung. Jede Teilmenge A einer geordneten Menge D ist bezüglich der induzierten Ordnung ebenfalls geordnet. Insbesondere ist jedes System von Teilmengen einer Menge M durch $\subseteq$ geordnet.

4.2 Linear geordnete Mengen. Eine geordnete Menge D heißt *linear geordnet* oder *Kette*, falls je zwei Elemente von D vergleichbar sind.

Beispiel. $\mathbb{Z}$, $\mathbb{Q}$ und $\mathbb{R}$ mit der üblichen Ordnung $\leq$ sind Ketten.

ω-Ketten. Die Menge der natürlichen Zahlen mit der üblichen Ordnung bezeichnen wir auch mit ω. Eine aufsteigende Folge $a_0 \sqsubseteq a_1 \sqsubseteq a_2 \sqsubseteq \dots$ mit abzählbar vielen Elementen aus D nennen wir *ω-Kette*.

4.3 Gerichtete Mengen. Eine Teilmenge A einer geordneten Menge D wird *gerichtet* genannt, falls $A \neq \varnothing$ und je zwei Elemente $a, b \in A$ eine obere Schranke in A haben, d.h.

$$\forall a, b \in A.\ \exists c \in A.\ a \sqsubseteq c \text{ und } b \sqsubseteq c$$

Beispiel.

- Jede Kette ist gerichtet.
- Die Menge $\mathfrak{P}_{\text{fin}}(M)$ aller endlichen Teilmengen von M ist bzgl. $\subseteq$ gerichtet.

Bemerkung. In einer gerichteten Menge A hat jede endliche Teilmenge $F \subseteq A$ eine obere Schranke $c \in A$.

4.4 Kleinstes Element – minimale Elemente. Sei m ein Element einer geordneten Menge D. Wir sagen, m sei ein *minimales Element* von D, falls es in D kein echt kleineres Element gibt, d.h.

$$\forall x \in D.\ x \sqsubseteq d \implies x = d$$

Wenn sogar gilt, daß $m \sqsubseteq x$ für jedes $x \in D$, so nennen wir m das *kleinste Element* von D. Häufig bezeichnen wir es mit $\perp$, sofern es existiert.

Dual definiert man die Begriffe *maximales Element* und *größtes Element*.

4.5 Suprema. Sei A eine Teilmenge einer geordneten Menge D und $c \in D$. Wir sagen, c sei eine *obere Schranke* von A, falls $a \sqsubseteq c$ für alle $a \in A$.

Wir sagen, daß c die *kleinste obere Schranke* oder das *Supremum* von A ist, falls

(1) $a \sqsubseteq c$ für alle $a \in A$

(2) $a \sqsubseteq c'$ für alle $a \in A \implies c \sqsubseteq c'$

Falls das Supremum der Menge A existiert, wird es oft mit $c = \sup A$ oder $c = \bigsqcup A$ bezeichnet.

Dual definiert man die *größte untere Schranke* bzw. das *Infimum* einer Menge A. Dieses wird mit $d = \inf A$ bzw. $d = \bigsqcap A$ bezeichnet.

Ist $A = \{a_i \mid i \in I\}$, so schreibt man auch $\bigsqcup_{i \in I} a_i$ bzw. $\bigsqcap_{i \in I} a_i$.

Beispiel.

- Hat eine Teilmenge $A \subseteq D$ ein größtes Element c, so gilt $c = \bigsqcup A$.
- In $\mathfrak{P}(M)$ hat jedes System $\mathcal{A} = \{A_i \mid i \in I\}$ von Teilmengen ein Supremum, nämlich $C = \bigcup_{i \in I} A_i$.
- In der Regel muß ein Supremum nicht existieren, z.B. hat in ω keine echt aufsteigende Folge $n_1 < n_2 < n_3 < \dots$ ein Supremum.

4.6 Bereiche. Eine geordnete Menge D wird ein *ω-Bereich* genannt, falls D ein kleinstes Element $\perp$ besitzt und jede ω-Kette $a_1 \leq a_2 \leq a_3 \leq \dots$ in D ein Supremum in D besitzt.

Hat sogar jede gerichtete Teilmenge $A \subseteq D$ in D ein Supremum, nennt man D einen *Bereich*.

Bemerkung. Andere Bezeichnungen für Bereiche sind *Domain*, *cpo* (complete partial order) oder *dcpo* (directed complete partial order).

Beispiel.

- Die flachen Bereiche aus den Beispielen zu 4.1 sind Bereiche im Sinne dieser Definition.
- ω ist kein ω-Bereich, aber $\overline{\omega} = \omega \cup \{+\infty\}$ ist ein Bereich.
- $\mathfrak{P}_{\text{fin}}(M)$ ist kein ω-Bereich, sofern M unendlich ist.
- Die Menge $\mathfrak{P}_{\text{count}}(M)$ aller höchstens abzählbaren Teilmengen von M ist ein ω-Bereich. Allerdings ist $\mathfrak{P}_{\text{count}}(M)$ kein Bereich, wenn M überabzählbar ist.
- $\mathfrak{P}(M)$ ist ein Bereich.
- Jede endliche geordnete Menge mit $\perp$ ist ein Bereich.
- Jeder Bereich ist ein ω-Bereich.

Bemerkung. Manchmal schreibt man $c = \bigsqcup^{\uparrow} A$ um anzudeuten, daß c das Supremum der *gerichteten* Menge A ist.

4.7 Stetige Funktionen. Eine Funktion $f\colon D \to E$ zwischen den geordneten Mengen D und E wird *ordnungserhaltend* oder *monoton* genannt, falls

$$a \sqsubseteq b \implies f(a) \sqsubseteq f(b) \tag{1}$$

Sind D und E ω-Bereiche, so wird f *ω-stetig* genannt, falls (1) und

$$f\Big(\bigsqcup_{n \in \mathbb{N}} a_n\Big) = \bigsqcup_{n \in \mathbb{N}} f(a_n) \quad \text{für jede } \omega\text{-Kette } (a_n)_{n \in \mathbb{N}} \text{ in } D \tag{2'}$$

Sind D und E Bereiche, so wird f *stetig* genannt, falls (1) und

$$f\Big(\bigsqcup A\Big) = \bigsqcup f(A) \quad \text{für jede gerichtete Menge } A \subseteq D \tag{2}$$

Mit $(D \to E)$ bezeichnen wir die Menge aller, mit $[D \to E]$ die Menge aller stetigen Abbildungen $f\colon D \to E$.

Beispiel.

- Ist D ein endlicher oder flacher Bereich, so ist jede ordnungserhaltende Abbildung $f\colon D \to E$ stetig.
- Die stetigen Abbildungen $g\colon M_\perp \to N_\perp$ zwischen flachen Bereichen sind
  (1) die konstanten Abbildungen $\mathrm{const}_b\colon x \mapsto b$ mit $b \in N$
  (2) Abbildungen der Form
  $$x \mapsto \begin{cases} f(x) & \text{falls } x \in D_f \\ \perp & \text{sonst} \end{cases}$$
  für eine partielle Abbildung $f\colon M \rightharpoonup N$ mit Definitionsbereich $D_f \subseteq M$.
- Sei $\mathbb{2} := \begin{array}{l} \bullet\,\top \\ | \\ \bullet\,\perp \end{array}$ der zweielementige Bereich. Die Abbildung $f_1\colon \mathfrak{P}(\mathbb{R}) \to \mathbb{2}$ mit
  $$f_1(X) = \begin{cases} \top & \text{falls } X \text{ unendlich} \\ \perp & \text{falls } X \text{ endlich} \end{cases}$$
  ist ordnungserhaltend, aber nicht ω-stetig. Die Abbildung $f_2\colon \mathfrak{P}(\mathbb{R}) \to \mathbb{2}$ mit
  $$f_2(X) = \begin{cases} \top & \text{falls } X \text{ überabzählbar} \\ \perp & \text{falls } X \text{ höchstens abzählbar} \end{cases}$$
  ist ω-stetig, aber nicht stetig.

4.8 Fixpunktsatz. Es sei D ein (ω-)Bereich und $f\colon D \to D$ eine (ω-)stetige Abbildung. Dann hat diese einen kleinsten Fixpunkt $\mathbb{Y}\, f$, der sich wie folgt berechnet:
$$\mathbb{Y}\, f := \bigsqcup_{n\in\mathbb{N}} f^n(\perp)$$

Beweis. Da $\perp$ das kleinste Element von D ist, gilt $\perp \sqsubseteq f(\perp)$. Da f ordnungserhaltend ist, folgt daraus
$$f(\perp) \sqsubseteq f(f(\perp)) = f^2(\perp)$$
Durch Induktion erhält man $f^n(\perp) \sqsubseteq f^{n+1}(\perp)$ für alle $n \in \mathbb{N}$. Wir haben also eine ω-Kette
$$\perp \sqsubseteq f(\perp) \sqsubseteq f^2(\perp) \sqsubseteq \dots$$
die in D ein Supremum hat, da D ein ω-Bereich ist.

Sei $x_0 := \bigsqcup_{n\in\mathbb{N}} f^n(\perp)$. Aus der ω-Stetigkeit von f folgt
$$f(x_0) = f\Big(\bigsqcup_{n\in\mathbb{N}} f^n(\perp)\Big) = \bigsqcup_{n\in\mathbb{N}} f^{n+1}(\perp) = x_0$$
Also ist x_0 ein Fixpunkt von f. Sei x_1 ein weiterer Fixpunkt von f. Da $\perp \sqsubseteq x_1$ folgt $f(\perp) \sqsubseteq f(x_1) = x_1$ und durch Induktion $f^n(\perp) \sqsubseteq x_1$ für alle $n \in \mathbb{N}$. Daraus folgt dann
$$x_0 = \bigsqcup_{n\in\mathbb{N}} f^n(\perp) \sqsubseteq x_1$$
Folglich ist x_0 der kleinste Fixpunkt von f. $\square$

4.9 Direkte Produkte. Es sei $(D_j)_{j\in J}$ eine Familie von Bereichen. Auf der Produktmenge $\prod_{j\in J} D_j$ aller J-Tupel $x = (x_j)_{j\in J}$ mit $x_j \in D_j$ sei

$$(x_j)_{j\in J} \sqsubseteq (y_j)_{j\in J} :\Longleftrightarrow\ x_j \sqsubseteq y_j \text{ für alle } j \in J$$

Mit dieser Ordnung ist $\prod_{j\in J} D_j$ ein Bereich, denn $(\perp)_{j\in J}$ ist das kleinste Element, und für eine gerichtete Menge $\{x^i \mid i \in I\} \subseteq \prod_{j\in J} D_j$ ist

$$\bigsqcup_{i\in I} x^i = \bigsqcup_{i\in I} (x^i_j)_{j\in J} = \big(\bigsqcup_{i\in I} x^i_j\big)_{j\in J}$$

d.h. die gerichteten Suprema bestimmt man koordinatenweise.

Die kanonischen Projektionen

$$\mathrm{pr}_{j_0}\, D_j \colon \prod_{j\in J} \to D_{j_0},\ (x_j)_{j\in J} \mapsto x_{j_0}$$

sind stetig, denn für eine gerichtete Menge $\{x^i \mid i \in I\} \subseteq \prod_{j\in J} D_j$ gilt

$$\mathrm{pr}_{j_0}\big(\bigsqcup_{i\in I} x^i\big) = \mathrm{pr}_{j_0}\big(\big(\bigsqcup_{i\in I} x^i_j\big)_{j\in J}\big) = \bigsqcup_{i\in I} x^i_{j_0} = \bigsqcup \mathrm{pr}_{j_0}(x^i)$$

4.10 Der Funktionenraum. Zunächst sei E ein Bereich und D eine beliebige Menge. Auf der Menge $(D \to E)$ aller Funktion $f \colon D \to E$ definieren wir

$$f \sqsubseteq g :\Longleftrightarrow\ \forall x \in D.\ f(x) \sqsubseteq g(x)$$

Man nennt dies die *punktweise Ordnung*. Mit dieser Ordnung wird $(D \to E)$ sogar ein Bereich:

(1) Die konstante Funktion $\mathrm{const}_\perp$ ist das kleinste Element von $(D \to E)$

(2) Ist $\{f_i \mid i \in I\}$ eine gerichtete Menge von Funktionen $f_i \colon D \to E$, so ist auch $\{f_i(x) \mid i \in I\}$ für jedes $x \in D$ eine gerichtete Menge in E, und man kann für jedes $x \in D$

$$f(x) = \bigsqcup_{i\in I} f_i(x)$$

setzen. Dieses f ist ebenfalls eine Funktion von D nach E, und man sieht leicht

$$f = \bigsqcup_{i\in I} f_i$$

Es gilt also: $\bigsqcup_{i\in I} f_i$ existiert in $(D \to E)$ für jede gerichtete Familie $\{f_i \mid i \in I\}$, und es gilt

$$\big(\bigsqcup_{i\in I} f_i\big)(x) = \bigsqcup_{i\in I} f_i(x) \quad \text{für alle } x \in D$$

d.h. gerichtete Suprema werden punktweise gebildet.

Sei nun auch D ein Bereich. Dann gilt

Hilfssatz. Ist $\{f_i \mid i \in I\}$ eine gerichtete Menge stetiger Funktionen $f_i \colon D \to E$, dann ist auch das punktweise Supremum $f := \bigsqcup_{i\in I}$ stetig.

Beweis. f ist ordnungserhaltend: Sei $x \sqsubseteq y$ in D. Es gilt dann

$$f_i(x) \sqsubseteq f_i(y) \quad \text{für alle } i \in I$$

also $f(x) = \bigsqcup_{i\in I} f_i(x) \sqsubseteq \bigsqcup_{i\in I} f_i(y) = f(y)$.
Sei nun $\{x_j \mid j \in J\}$ eine gerichtete Menge in D. Wir wollen zeigen, daß

$$f\big(\bigsqcup_{j\in J} x_j\big) = \bigsqcup_{j\in J} f(x_j)$$

gilt:

$$\begin{aligned}
f\big(\bigsqcup_{j\in J} x_j\big) &= \big(\bigsqcup_{i\in I} f_i\big)\big(\bigsqcup_{j\in J} x_j\big)\\
&= \bigsqcup_{i\in I} f_i\big(\bigsqcup_{j\in J} x_j\big)\\
&= \bigsqcup_{i\in I}\big(\bigsqcup_{j\in J} f_i(x_j)\big)\\
&= \bigsqcup_{j\in J}\big(\bigsqcup_{i\in I} f_i(x_j)\big)\\
&= \bigsqcup_{j\in J} f(x_j)
\end{aligned}$$

□

Folgerung. Die Menge $[D \to E]$ aller stetigen Funktionen $f\colon D \to E$ ist ein Bereich mit der punktweise definierten Ordnung. Suprema gerichteter Familien werden punktweise gebildet.

4.11 Hilfssatz. Seien D_1, D_2 und E Bereiche. Dann ist $f\colon D_1 \times D_2 \to E$ genau dann stetig, wenn f komponentenweise stetig ist, d.h. wenn die Funktionen

$$\begin{aligned}
x &\mapsto f(x, y_0)\colon D_1 \to E \text{ für beliebiges } y_0 \in D_2\\
y &\mapsto f(x_0, y)\colon D_2 \to E \text{ für beliebiges } x_0 \in D_1
\end{aligned}$$

alle stetig sind.

Beweis. Die eine Richtung ist klar. Sei $\{x_i, y_i) \mid i \in I\}$ eine gerichtete Menge in $D_1 \times D_2$. Dann gilt

$$\begin{aligned}
f\big(\bigsqcup_{i\in I}(x_1, y_1)\big) &= f\big(\bigsqcup_{i\in I} x_i, \bigsqcup_{i\in I} y_i\big)\\
&= f\big(\bigsqcup_{i\in I} x_i, \bigsqcup_{j\in I} y_j\big)\\
&= \bigsqcup_{i\in I} f\big(x_i, \bigsqcup_{j\in I} y_j\big)
\end{aligned}$$

$$
\begin{aligned}
&= \bigsqcup_{i\in I}\bigsqcup_{j\in I} f(x_i, y_j) \\
&= \bigsqcup_{i\in I} f(x_i, y_i)
\end{aligned}
$$

Die letzte Gleichung gilt aus folgendem Grund: Da $\{(x_i, y_i \mid i \in I\}$ gerichtet ist, gibt es für alle $i, j \in I$ ein $k \in I$, so daß

$$(x_i, y_j) \sqsubseteq (x_k, y_k) \quad \text{und} \quad (x_j, y_j) \sqsubseteq (x_k, y_k)$$

und folglich $(x_i, y_j) \sqsubseteq (x_k, y_k)$. Es folgt $\bigsqcup_{i\in I} \bigsqcup j \in I f(x_i, y_j) \sqsubseteq \bigsqcup k \in I f(x_k, y_k)$. Die umgekehrte Ungleichung ist aber klar, da jedes Element $f(x_k, y_k)$ auf der rechten Seite auch auf der linken Seite vorkommt. □

Der vorhergehende Hilfssatz gilt auch für Funktionen in endlich vielen statt in zwei Variablen. Er gilt nicht für Funktionen in unendlich vielen Variablen. Man beachte auch, daß das entsprechende Analogon in der Analysis falsch ist!

4.12 Hilfssatz. Für beliebige Bereiche D und E ist die Funktion

$$\mathrm{app}\colon [D \to E] \times D \to E,\ (f, x) \mapsto f(x)$$

stetig.

Beweis. Nach 4.11 genügt es zu zeigen:

(1) Für jedes $f \in [D \to E]$ ist $x \mapsto f(x)\colon D \to E$ stetig. Dies ist aber klar, da f stetig ist und f und $x \mapsto f(x)$ die selbe Funktion beschreiben.

(2) Für jedes $x \in D$ ist $f \mapsto f(x)\colon [D \to E] \to E$ stetig. Sei also $\{f_i \mid i \in I\}$ gerichtet in $[D \to E]$. Da gerichtete Suprema in $[D \to E]$ punktweise gebildet werden, gilt

$$\big(\bigsqcup_{i\in I} f_i\big)(x) = \bigsqcup_{i\in I} f_i(x)$$

was zu beweisen war. □

4.13 Beispiele für stetige Abbildungen. Seien D, E und F Bereiche.

- id_D ist stetig. Sind $f\colon D \to E$ und $g\colon E \to F$ stetig, dann ist auch $g \circ f\colon D \to F$ stetig:

$$(g \circ f)\big(\bigsqcup_{i\in I} x_i\big) = (g\big(f\big(\bigsqcup_{i\in I} x_i\big)\big) = (g\big(\bigsqcup_{i\in I} f(x_i)\big) = \bigsqcup_{i\in I} g(f(x_i)) = \bigsqcup_{i\in I} (g \circ f)(x_i)$$

- $\mathrm{comp}\colon [D \to E] \times [E \to F] \to [D \to F],\ (f, g) \mapsto g \circ f$ ist stetig.

  Es gilt

$$g \circ \big(\bigsqcup_{i\in I} f_i\big) = \bigsqcup_{i\in I} g \circ f_i$$

  denn für alle $x \in D$ gilt

$$g\big(\big(\bigsqcup_{i\in I} f_i\big)(x)\big) = g\big(\bigsqcup_{i\in I} f_i(x)\big) = \bigsqcup_{i\in I} g(f_i(x)) = \bigsqcup_{i\in I} (g \circ f_i)(x)$$

Ebenso zeigt man $\big(\bigsqcup_{i\in I} g_i\big)\circ f = \bigsqcup_{i\in I}(g_i\circ f)$, so daß mit 4.11 die Behauptung folgt.

- $\mathrm{it}^{(2)}\colon [D\to D]\to[D\to D],\ f\mapsto f\circ f$ ist stetig, denn es ist die Hintereinanderausführung der stetigen Abbildungen

$$\begin{array}{ccccc} [D\to D] & \to & ([D\to D]\to[D\to D]) & \xrightarrow{\text{comp}} & [D\to D] \\ f & \mapsto & (f,f) & \mapsto & f\circ f \end{array}$$

- Ebenso ist $\mathrm{it}^{(n)}\colon [D\to D]\to[D\to D],\ f\mapsto f^n$ ist stetig.
- $\mathbb{Y}_D^{(n)}\colon [D\to D]\to D,\ f\mapsto f^n(\bot)$ ist stetig, denn $\mathbb{Y}_D^{(n)}$ ist die Hintereinanderausführung der stetigen Abbildungen

$$\begin{array}{ccccc} [D\to D] & \xrightarrow{\mathrm{it}^{(n)}} & [D\to D] & \to & D \\ f & \mapsto & f^n & \mapsto & f^n(\bot) \end{array}$$

- Es gilt $\mathbb{Y}_D^{(0)}\sqsubseteq\mathbb{Y}_D^{(1)}\sqsubseteq\mathbb{Y}_D^{(2)}\sqsubseteq\dots$, d.h. die $\mathbb{Y}_D^{(n)}$ bilden eine ω-Kette in $[[D\to D]\to D]$. Also ist

$$\mathbb{Y}_D := \bigsqcup_{n\in\mathbb{N}} \mathbb{Y}_D^{(n)}$$

eine stetige Funktion von $[D\to D]$ nach D.

Bemerke: $\mathbb{Y}_D(f) = \bigsqcup_{n\in\mathbb{N}}\mathbb{Y}_D^{(n)}(f) = \bigsqcup_{n\in\mathbb{N}} f^n(\bot)$ ist nach 4.8 der kleinste Fixpunkt von f. Insbesondere gilt daher

$$\mathbb{Y}_D(f) = f\big(\mathbb{Y}_D(f)\big)$$

4.14 Curry und Uncurry. Es seien D_1, D_2 und E Bereiche. In der Einleitung hatten wir zueinander inverse Bijektionen

$$(D_1\times D_2\to E) \underset{\text{uncurry}}{\overset{\text{curry}}{\rightleftarrows}} (D_1\to(D_2\to E))$$

gefunden. Diese induzieren zueinander inverse *stetige* Bijektionen

$$[D_1\times D_2\to E] \underset{\text{uncurry}}{\overset{\text{curry}}{\rightleftarrows}} [D_1\to[D_2\to E]]$$

Beweis. Der Beweis soll in Aufgabe 57(c) durchgeführt werden. □

5. Die Standard-Semantik von PCF

5.1 Interpretation der Typen. Die atomaren Typen ν und β werden durch die flachen Bereiche $\mathbb{N}_\bot$ und $\mathbb{B}_\bot$ interpretiert, die anderen Typen $D_{\sigma\to\tau}$ durch die Menge $[D_\sigma\to D_\tau]$, also durch die stetigen Abbildungen $f\colon D_\sigma\to D_\tau$.

5.2 Interpretation der Konstanten.

$$[\![k_n]\!] := n\in\mathbb{N}_\bot \qquad [\![\mathsf{TRUE}]\!] = \text{true}\in\mathbb{B}_\bot \qquad [\![\mathsf{FALSE}]\!] = \text{false}\in\mathbb{B}_\bot$$

$$[\![\mathsf{SUCC}]\!] := \mathrm{succ}\colon \mathbb{N}_\bot\to\mathbb{N}_\bot \qquad \mathrm{succ}(x) = \begin{cases} x+1 & \text{falls } x\in\mathbb{N} \\ \bot & \text{falls } x=\bot \end{cases}$$

$$[\![\mathsf{PRED}]\!] := \mathrm{pred}\colon \mathbb{N}_\perp \to \mathbb{N}_\perp \qquad \mathrm{pred}(x) = \begin{cases} x-1 & \text{falls } x \in \mathbb{N}\setminus\{0\} \\ \perp & \text{falls } x \in \{\perp, 0\} \end{cases}$$

$$[\![\mathsf{ISNULL}]\!] := \mathrm{isnull}\colon \mathbb{N}_\perp \to \mathbb{B}_\perp \qquad \mathrm{isnull}(x) = \begin{cases} \mathrm{true} & \text{falls } x = 0 \\ \mathrm{false} & \text{falls } x \in \mathbb{N}\setminus\{0\} \\ \perp & \text{falls } x = \perp \end{cases}$$

$$[\![\mathsf{COND}]\!] := \mathrm{cond}_\sigma\colon \mathbb{B}_\sigma \times D_\sigma \times D_\sigma \to D_\sigma \quad \mathrm{cond}_\sigma(x,y,z) = \begin{cases} y & \text{falls } x = \mathrm{true} \\ z & \text{falls } x = \mathrm{false} \\ \perp & \text{falls } x = \perp \end{cases}$$

$$[\![\mathsf{Y}_\sigma]\!] := \mathbb{Y}_\sigma\colon [D_\sigma \to D_\sigma] \to D_\sigma \qquad \mathbb{Y}_\sigma(f) = \bigsqcup_{n\in\mathbb{N}} f^n(\perp)$$

Nach 4.13 ist $\mathbb{Y}_\sigma$ stetig. Prüfe nach, daß auch succ, pred, isnull und cond_σ stetig sind.

5.3 Interpretation von Termen. Die PCF-Terme werden – wie schon beim Mengenmodell oder allgemeiner bei Henkin-Modellen – induktiv über den Termaufbau simultan für beliebige Variablenbelegungen ϱ interpretiert:

(1) Die PCF-Konstanten C werden wie in 5.2 interpretiert, Variablen entsprechend ihrer Belegung: $[\![x^\sigma]\!]_\varrho = \varrho(x^\sigma)$.

(2) Ist $M \equiv N^{\sigma\to\tau}P^\sigma$, und hat man Interpretationen $[\![N]\!]_\varrho \in D_{\sigma\to\tau} = [D_\sigma \to D_\tau]$ und $[\![P]\!]_\varrho \in D_\sigma$, so definiere

$$[\![M]\!]_\varrho = [\![NP]\!]_\varrho := [\![N]\!]_\varrho\big([\![P]\!]_\varrho\big) \in D_\tau$$

(3) Ist $M \equiv \lambda x^\sigma.N^\tau$, und hat man eine Interpretation $[\![N]\!]_\varrho \in D_\tau$ für beliebige ϱ, so definiere

$$[\![M]\!]_\varrho = [\![\lambda x^\sigma.N]\!]_\varrho := (a \mapsto [\![N]\!]_{\varrho[x\mapsto a]}\colon D_\sigma \to D_\tau)$$

Damit ist $[\![\lambda x^\sigma.N]\!]_\varrho$ als Funktion von $D_\sigma \to D_\tau$ interpretiert. Wir müssen noch zeigen, daß diese Funktion stetig ist, damit sie zu $D_{\sigma\to\tau} = [D_\sigma \to D_\tau]$ gehört. Zu diesem Zweck zeigen wir, daß $[\![M]\!]$ stetig von den darin vorkommenden Variablen abhängt.

Sei M^σ ein PCF-Term und $x_1^{\sigma_1}, \dots, x_k^{\sigma_k}$ verschiedene Variablen, unter denen alle freien Variablen von M vorkommen. Der Wert $[\![M]\!]$ wird nur von der Belegung ϱ der Variablen $x_1, \dots, x_k$ abhängen. Wir schreiben $[\![M]\!](a_1, \dots, a_k)$ für $[\![M]\!]_\varrho$, sofern $\varrho(x_1) = a_1, \dots, \varrho(x_k) = a_k$. Man kann also $[\![M]\!]$ als Funktion

$$(a_1, \dots, a_k) \mapsto [\![M]\!](a_1, \dots, a_k)\colon D_{\sigma_1} \times \cdots \times D_{\sigma_k} \to D_\sigma$$

auffassen. Wir zeigen nun induktiv über den Termaufbau von M, daß diese Funktion stetig ist.

(1) $M \equiv x_i^{\sigma_i}$. Es seien $x_1^{\sigma_1}, \dots, x_k^{\sigma_k}$ Variablen, unter denen $x_i^{\sigma_i}$ vorkommt. Dann ist $[\![x_i^{\sigma_i}]\!](a_1, \dots, a_i, \dots, a_k) = a_i \in D_{\sigma_i}$ und $(a_1, \dots, a_i, \dots, a_k) \mapsto a_i\colon D_{\sigma_1} \times \cdots \times D_{\sigma_k} \to D_\tau$ ist als Projektion auf den i-ten Faktor stetig.

Für Konstanten ist nichts zu zeigen, da diese keine freien Variablen besitzen.

(2) $M^\tau \equiv N^{\sigma\to\tau}P^\sigma$. Es seien $x_1^{\sigma_1},\dots,x_k^{\sigma_k}$ Variablen, unter den die freien Variablen von M – also auch die von N und P – vorkommen. Nach Induktionsvoraussetzung sind

$$(a_1,\dots,a_k)\mapsto [\![N]\!](a_1,\dots,a_k)\colon D_{\sigma_1}\times\cdots\times D_{\sigma_k}\to D_{\sigma\to\tau}$$
$$(a_1,\dots,a_k)\mapsto [\![P]\!](a_1,\dots,a_k)\colon D_{\sigma_1}\times\cdots\times D_{\sigma_k}\to D_{\sigma}$$

stetig. Also ist auch

$$(a_1,\dots,a_k)\mapsto [\![N]\!](a_1,\dots,a_k)\big([\![P]\!](a_1,\dots,a_k)\big)\colon D_{\sigma_1}\times\cdots\times D_{\sigma_k}\to D_{\tau}$$

stetig, da nach 4.12 die Funktion app: $[D_\sigma\to D_\tau]\times D_\sigma\to D_\tau$ stetig ist.

(3) $M\equiv \lambda x^\sigma.N^\tau$. Es seien $x_1^{\sigma_1},\dots,x_k^{\sigma_k}$ die freien Variablen in M. Die freien Variablen von N sind $x_1^{\sigma_1},\dots,x_k^{\sigma_k},x^\sigma$. Nach Induktionsvoraussetzung ist

$$(a_1,\dots,a_k,a)\mapsto [\![N]\!](a_1,\dots,a_k,a)\colon D_{\sigma_1}\times\cdots\times D_{\sigma_k}\times D_\sigma\to D_{\tau}$$

stetig. Faßt man die ersten k Faktoren zusammen und wendet man curry an, so erhält man nach 4.13 eine stetige Abbildung

$$(a_1,\dots,a_k)\mapsto \big(a\mapsto [\![N]\!](a_1,\dots,a_k,a)\big)\colon D_{\sigma_1}\times\cdots\times D_{\sigma_k}\to [D_\sigma\to D_{\tau}]$$

Nun ist aber gerade

$$[\![\lambda x.N^\sigma]\!](a_1,\dots,a_k)=\big(a\mapsto [\![N]\!](a_1,\dots,a_k,a)\colon D_\sigma\to D_\tau\big)$$

insbesondere also stetig.

Bemerkung. Die obige Konstruktion kann man auch verwenden, um ein Modell für den untypsierten λ-Kalkül zu erhalten: Ähnlich wie beim Mengenmodell aus Kapitel IV wähle man einen beliebigen Bereich D_α für atomare Typen und definiere induktiv für höhere Typen $D_{\sigma\to\tau}:=[D_\sigma\to D_\tau]$, der Bereich aller stetigen Funktionen von D_σ nach D_τ. Wie die Überlegungen in 5.3 zeigen, führt dies zu einem Henkin-Modell für den einfach typisierten λ-Kalkül. Diese Modelle nennt man Bereichsmodelle.

Die obige Semantik ist korrekt in dem Sinne, daß sich die Interpretation eines Terms durch Reduktion nicht verändert:

5.4 Korrektheitssatz. $M\to^* N \implies [\![M]\!]_\varrho=[\![N]\!]_\varrho$ für alle ϱ.

Beweis. Man muß dieses nur für die einzelnen Grundreduktionsregeln aus 2.1 nachprüfen. Bei den meisten Reduktionsregeln ist dies klar. Betrachten wir als Beispiel $\mathsf{Y}\,M\to M(\mathsf{Y}\,M)$:

$$[\![\mathsf{Y}\,M]\!]_\varrho=[\![\mathsf{Y}]\!]_\varrho([\![M]\!]_\varrho)=\mathbb{Y}([\![M]\!]_\varrho)$$

Nach 4.8 ist $\mathbb{Y}([\![M]\!]_\varrho)$ der kleinste Fixpunkt von $[\![M]\!]_\varrho$. Insbesondere gilt

$$\mathbb{Y}([\![M]\!]_\varrho)=[\![M]\!]_\varrho(\mathbb{Y}([\![M]\!]_\varrho)=[\![M(\mathsf{Y}\,M)]\!]_\varrho$$

□

Die Umkehrung von 5.4 ist nicht richtig. Zum einen ist die rechte Seite der Implikation symmetrisch in M und N, die linke Seite allerdings nicht. Man könnte dieses Manko beheben, indem man die Frage wie folgt umformuliert:

Wenn $[\![M]\!]_\varrho=[\![N]\!]_\varrho$, existiert dann ein P mit $M\to^* P$ und $N\to^* P$?

Aber auch dies führt nicht zum Ziel, vgl. Aufgabe 61(b).

Eine Umkehrung des Korrektheitssatzes können wir jedoch zeigen, wenn wir uns auf spezielle PCF-Terme beschränken:

5.5 Programme. Die geschlossenen PCF-Terme M vom Typ ν oder β nennen wir *Programme.*

Diese Definition ist etwas willkürlich, denn man könnte jeden PCF-Term als „Programm" bezeichnen. Daß wir uns auf geschlossene Terme von atomarem Typ beschränken, ist dadurch gerechtfertigt, daß nur solche Terme zu einem konkreten Ergebnis reduziert werden können.

Man kann sich leicht überlegen:

Bemerkung. Für jeden geschlossenen PCF-Term M vom Typ ν oder β gilt:

- Entweder $M \to M_1 \to M_2 \to M_3 \to \dots$, d.h. die Reduktion von M terminiert nicht
- oder $M \to^* k_n$, $M \to^* \mathsf{TRUE}$ bzw. $M \to^* \mathsf{FALSE}$, d.h. die Reduktion von M terminiert nach endlich vielen Schritten bei einer der Konstanten atomaren Typs.

Die einzigen Ausnahmen sind $M \to^* \mathsf{PRED}\, k_0$ und Konsequenzen hieraus, wie z.B. $M \to^* \mathsf{SUCC}(\mathsf{PRED}\, k_0)$, $M \to^* \mathsf{PRED}(\mathsf{PRED}\, k_0)$ etc.

Bezeichne C eine beliebige der Konstanten k_n, TRUE oder FALSE. Sei nun M ein Programm. Aus dem Korrektheitssatz 5.4 folgt nun

$$M \to^* \mathsf{C} \implies [\![M]\!] = [\![\mathsf{C}]\!]$$

In diesem eingeschränkten Fall gilt auch die Umkehrung der Implikation:

5.6 Adäquatheitssatz. Für Programme M gilt $M \to^* \mathsf{C} \iff [\![M]\!] = [\![\mathsf{C}]\!]$

Der Beweis dieses Satzes wird in Abschnitt 6 geführt. □

Wenn also die Interpretation $[\![M]\!]$ eines Programms eine natürliche Zahl oder ein Wahrheitswert ist, dann wird diese Zahl bzw. dieser Wahrheitswert durch PCF berechnet.

5.7 Folgerung. Für PCF-Programme M gilt

$$[\![M]\!] = \bot \iff M \text{ terminiert nicht oder } M \to^* \mathsf{PRED}\, k_0, \dots$$

6. Beweis des Adäquatheitssatzes

Wir setzen die Überlegungen aus Abschnitt 5 fort. Wesentliches Hilfsmittel ist eine „logische Relation" $\lhd$ zwischen der Standard-Semantik und den geschlossenen PCF-Termen:

Seien D_σ die semantischen Bereiche aus Abschnitt 5 und PCF_σ die Menge der geschlossenen PCF-Terme vom Typ σ. Desweiteren sei in diesem Abschnitt der Typ ι einer der beiden atomaren PCF-Typen β oder ν. Wir definieren eine zweistellige Relation $\lhd_\sigma \subseteq D_\sigma \times \mathrm{PCF}_\sigma$ induktiv über den Aufbau von σ:

(1) $\sigma \equiv \iota$: Für $d \in D_\iota$ und $M \in \mathrm{PCF}_\iota$ sei

$$d \lhd_{\nu,\beta} M :\Longleftrightarrow \ (M \to^* \mathsf{C} \text{ und } d = [\![\mathsf{C}]\!]) \quad \text{oder} \quad (d = \bot \text{ und } M \text{ beliebig})$$

(2) Für $f \in D_{\sigma\to\tau} = [D_\sigma \to D_\tau]$ und $M \in \mathrm{PCF}_{\sigma\to\tau}$ sei

$$f \lhd_{\sigma\to\tau} M :\Longleftrightarrow \ f(a) \lhd_\tau MN \text{ für alle } a \in D_\sigma,\ N \in \mathrm{PCF}_\sigma \text{ mit } a \lhd_\sigma N$$

6.1 Bemerkung. Es sei σ ein beliebiger Typ. Wie wir wissen, hat σ eine eindeutig bestimmte Darstellung der Form

$$\sigma \equiv \tau_1 \to \tau_2 \to \cdots \to \tau_k \to \iota$$

Man überlegt sich leicht, daß für alle $f \in D_\sigma$ und $M \in \mathrm{PCF}_\sigma$ gilt

$$f \lhd_\sigma M \iff \text{Für alle } a_i \in D_{\tau_i} \text{ und alle } N_i \in \mathrm{PCF}_{\tau_i} \text{ mit } a_i \lhd_{\tau_i} N_i \text{ gilt}$$
$$f(a_1, \ldots, a_k) \lhd_\iota MN_1 \ldots N_k$$

6.2 Hilfssatz. Für alle σ und alle $M \in \mathrm{PCF}_\sigma$ gilt, daß die Menge $\{d \in D_\sigma \mid d \lhd_\sigma M\}$ nicht leer und *Scott-abgeschlossen* ist, d.h.

(1) $\bot \lhd_\sigma M$

(2) $d' \sqsubseteq d,\ d \lhd_\sigma M \implies d' \lhd_\sigma M$

(3) Für jede ω-Kette $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \ldots$ mit $d_i \lhd_\sigma M$ für alle i gilt: $\bigsqcup_i d_i \lhd_\sigma M$

Beweis.

- Für $\sigma \equiv \iota$ folgen die Behauptungen direkt aus der Definition von $\lhd_\iota$.
- Die Behauptung sei für σ und τ bewiesen. Wir zeigen, daß sie auch für $\sigma \to \tau$ gilt. Es sei $M \in \mathrm{PCF}_{\sigma\to\tau}$.
  (1) Das kleinste Element von $D_{\sigma\to\tau}$ ist $\mathrm{const}_\bot$, die konstante Abbildung mit Wert $\bot$. Es gilt $\mathrm{const}_\bot(a) = \bot \in D_\tau$ für alle $a \in D_\sigma$ nach Induktionsvoraussetzung, also $\mathrm{const}_\bot \in D_{\sigma\to\tau}$.
  (2) Es sei $g \sqsupseteq f$ und $f \lhd_{\sigma\to\tau} M$. Für alle $a \in D_\sigma$ und $N \in \mathrm{PCF}_\sigma$ mit $a \lhd_\sigma N$ gilt dann $f(a) \lhd_\tau MN$. Wegen $g \sqsubseteq f$ folgt $g(a) \sqsubseteq f(a)$, also $g(a) \lhd_\tau MN$ nach Induktionsvoraussetzung. Es folgt $g \lhd_{\sigma\to\tau} M$.
  (3) Es sei $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \ldots$ eine ω-Kette mit $f_i \lhd_{\sigma\to\tau} M$ für alle i. Nach Definition gilt dann $f_i(a) \lhd_\tau MN$ für alle $a \in D_\sigma$ und $N \in \mathrm{PCF}_\sigma$ mit $a \lhd_\sigma N$. Nach Induktionsvoraussetzung folgt $\bigsqcup_i f_i(a) \lhd_\tau MN$, d.h. $\bigsqcup_i f_i \lhd_{\sigma\to\tau} M$. □

6.3 Hilfssatz. Es sei M ein geschlossener PCF-Term des Typs σ und $M \to^* M'$. Dann gilt $d \lhd_\sigma M \iff d \lhd_\sigma M'$.

Beweis. (1) Sei $\sigma \equiv \iota$. Ist $d = \bot$, so gilt $d \lhd M$ und $d \lhd M'$. Sei also $d \neq \bot$.

$\Rightarrow$: Sei $d \lhd M$. Dann gilt $d = [\![\mathsf{C}]\!]$ und $M \to^* \mathsf{C}$ nach Definition von $\lhd$. Wenn $M \to^* M'$, dann kommt M' in der Reduktionskette $M \to^* \mathsf{C}$ vor, da die Reduktionskette eindeutig ist. Folglich gilt auch $M' \to^* \mathsf{C}$ und somit $d \lhd M'$.

$\Leftarrow$: Sei $d \lhd M'$. Dann $d = [\![\mathsf{C}]\!]$ und $M' \to^* \mathsf{C}$ nach Definition von $\lhd$. Wenn $M \to^* M'$, dann gilt auch $M \to^* M' \to^* \mathsf{C}$, also $d \lhd M$.

(2) Ist σ nicht atomar, dann hat σ eine Darstellung $\sigma \equiv \tau_1 \to \cdots \to \tau_k \to \iota$. Nach 6.1 haben wir

$$f \lhd_\sigma M \iff \forall(a_i \lhd_{\tau_i} N_i).\ f(a_1, \dots, a_k) \lhd_\iota MN_1 \dots N_k$$
$$f \lhd_\sigma M' \iff \forall(a_i \lhd_{\tau_i} N_i).\ f(a_1, \dots, a_k) \lhd_\iota M'N_1 \dots N_k$$

Wenn $M \to^* M'$, dann hat man auch $MN_1 \dots N_k \to^* M'N_1 \dots N_k$ nach den PCF-Reduktionsregeln. Dann kann man 1 anwenden, um zu sehen, däs die rechten Seiten äquivalent sind. □

6.4 Hilfssatz. Für alle PCF-Konstanten C gilt $[\![\mathsf{C}]\!] \lhd_\sigma \mathsf{C}$.

Beweis. • Die Behauptung ist klar für die PCF-Konstanten k_n, TRUE und FALSE atomaren Typs.

- Um succ $\lhd_{\nu\to\nu}$ SUCC zu zeigen, müssen wir nachweisen, daß für alle $d \in \mathbb{N}_\perp$ und alle $M \in \mathrm{PCF}_\nu$ gilt

  $$d \lhd_\nu \Longrightarrow \mathrm{succ}(d) \lhd_\nu \mathsf{SUCC}\, M$$

  $d = \perp$: Es gilt $\mathrm{succ}(\perp) = \perp \lhd_\nu \mathsf{SUCC}\, M$ nach 6.2.1.

  $d = n \in \mathbb{N}$: Aus $n \lhd_\nu M$ folgt $M \to^* k_n$ nach Definition von $\lhd_\nu$, also $\mathsf{SUCC}\, M \to^* \mathsf{SUCC}\, k_n \to k_{n+1}$ gemäß den PCF-Reduktionsregeln. Also $\mathrm{succ}(n) = n + 1 \lhd_\nu \mathsf{SUCC}\, M$.

  Der Nachweis für die Fälle pred $\lhd_{\nu\to\nu}$ PRED und isnull $\lhd_{\nu\to\beta}$ ISNULL geht ähnlich.

- Um $\mathrm{cond}_\sigma \lhd_{\beta\to\sigma\to\sigma\to\sigma} \mathsf{COND}_\sigma$ nachzuweisen, muß man

  $$d \lhd_\beta M,\ a \lhd_\sigma P,\ b \lhd_\sigma Q \Longrightarrow \mathrm{cond}(d, a, b) \lhd_\sigma \mathsf{COND}\, MPQ$$

  zeigen.

  $d = \perp$: $\mathrm{cond}(\perp, a, b) = \perp \lhd_\sigma \mathsf{COND}\, MPQ$ nach 6.2.1.

  $d = \mathrm{true}$: Aus true $\lhd_\beta M$ folgt $M \to^*$ TRUE, und daraus folgt $\mathsf{COND}\, MPQ \to^* \mathsf{COND}\ \mathsf{TRUE}\, PQ \to P$ nach den Reduktionsregeln für PCF. Da $\mathrm{cond}(\mathrm{true}, a, b) = a \lhd_\sigma P$ nach Voraussetzung, folgt $\mathrm{cond}(\mathrm{true}, a, b) \lhd_\sigma \mathsf{COND}\, MPQ$.

  Der Fall $d = \mathrm{false}$ geht analog.

- Um $\mathbb{Y}_\sigma \lhd_{(\sigma\to\sigma)\to\sigma} \mathsf{Y}_\sigma$ zu zeigen, muß

  $$f \lhd_{\sigma\to\sigma} M \Longrightarrow \bigsqcup_{n\in\mathbb{N}} f^n(\perp) = \mathbb{Y}(f) \lhd_\sigma \mathsf{Y}\, M$$

  bewiesen werden. Sei also $f \lhd_{\sigma\to\sigma} M$. Nach 6.2.1 gilt $\perp \lhd_\sigma \mathsf{Y}\, M$. Es folgt $f(\perp) \lhd_\sigma M(\mathsf{Y}\, M)$. Da $\mathsf{Y}\, M \to M(\mathsf{Y}\, M)$ folgt $f(\perp) \lhd_\sigma \mathsf{Y}\, M$. Indem man dieses Argument wiederholt, folgt induktiv $f^n(\perp) \lhd_\sigma \mathsf{Y}\, M$ für alle $n \in \mathbb{N}$. Mithilfe von 6.2.3 schließen wir $\bigsqcup_{n\in\mathbb{N}} f^n(\perp) \lhd_\sigma \mathsf{Y}\, M$. □

6.5 Hauptsatz für logische Relationen. Für jeden geschlossenen PCF-Term M des Typs σ gilt $[\![M]\!] \lhd_\sigma M$.

Den Beweis wollen wir induktiv über den Termaufbau von M führen. Da hierbei auch Teilterme mit freien Variablen auftauchen, müssen wir die Behauptung des Hauptsatzes so umformulieren, daß sie auch für Terme mit freien Variablen gilt:

Verallgemeinerung von 6.5. Es sei M ein beliebiger PCF-Term des Typs $\sigma \equiv \tau_1 \to \cdots \to \tau_k \to \iota$ mit freien Variablen $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$. Dann gilt für alle $a_1 \lhd_{\sigma_1} N_1, \ldots, a_n \lhd_{\sigma_n} N_n$:

$$[\![M]\!]_{[x_1^{\sigma_1} \mapsto a_1, \ldots, x_n^{\sigma_n} \mapsto a_n]} \lhd_\sigma M[x_1^{\sigma_1} := N_1, \ldots, x_n^{\sigma_n} := N_n]$$

Beweis. • Ist M eine PCF-Konstante, so gilt die Behauptung nach 6.4. Für $M \equiv x^\sigma$ folgt mit $a \lhd_\sigma N$: $[\![x^\sigma]\!]_{[x_\sigma} \mapsto a] = a \lhd_\sigma N \equiv x^\sigma[x^\sigma := N]$. Damit ist der Fall M atomar erledigt.

- Sei $M \equiv P^{\sigma\to\tau}Q^\sigma$ und die Behauptung für P und Q bewiesen. Seien weiterhin $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$ die freien Variablen von M. Dies sind dann auch die freien Variablen von P und Q. Nun seien $a_1 \lhd_{\sigma_1} N_1, \ldots, a_n \lhd_{\sigma_n} N_n$. Nach Induktionsvoraussetzung gelten

$$\begin{aligned}[\![P]\!]_{[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n]} &\lhd_{\sigma\to\tau} P[x_1 := N_1, \ldots, x_n := N_n]\\ [\![Q]\!]_{[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n]} &\lhd_\sigma Q[x_1 := N_1, \ldots, x_n := N_n]\end{aligned}$$

Es folgt

$$\begin{aligned}[\![PQ]\!]_{[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n]} &= [\![P]\!]_{[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n]}\big([\![Q]\!]_{[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n]}\big)\\ &\lhd_\tau P[x_1 := N_1, \ldots, x_n := N_n]\, Q[x_1 := N_1, \ldots, x_n := N_n]\\ &\equiv (PQ)[x_1 := N_1, \ldots, x_n := N_n]\end{aligned}$$

was zu beweisen war.

- Sei $M \equiv \lambda x^\sigma.Q^\tau : \sigma \to \tau$ und die Behauptung für Q bewiesen. Sind $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$ die freien Variablen von M, dann sind $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}, x^\sigma$ die freien Variablen von Q. Es seien $a_1 \lhd_{\sigma_1} N_1, \ldots, a_n \lhd_{\sigma_n} N_n$. Wir wollen

$$[\![\lambda x^\sigma.P]\!]_{[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n]} \lhd_{\sigma\to\tau} (\lambda x^\sigma.P)[x_1 := N_1, \ldots, x_n := N_n]$$

zeigen. Nach der Definition von $\lhd_{\sigma\to\tau}$ genügt es zu zeigen, daß für alle $a \lhd_\sigma N$ gilt

$$\begin{aligned}[\![\lambda x^\sigma.P]\!]_{[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n, x \mapsto a]} &\lhd_\tau \big((\lambda x^\sigma.P)[x_1 := N_1, \ldots, x_n := N_n]\big)N\\ &\equiv \big(\lambda x^\sigma.P[x_1 := N_1, \ldots, x_n := N_n]\big)N\\ &\to P[x_1 := N_1, \ldots, x_n := N_n][x := N]\\ &\equiv P[x_1 := N_1, \ldots, x_n := N_n, x := N\end{aligned}$$

Letzteres gilt aber nach der Induktionsvoraussetzung über P. □

Aus 6.5 können wir nun unmittelbar den Adäquatheitssatz folgern:

Beweis von 5.6. Es sei M ein geschlossener PCF-Term des Typs ν oder β. Aus 6.5 wissen wir $[\![M]\!] \lhd_\iota M$, d.h. $[\![M]\!] = \bot$ oder ($M \to^* \mathsf{C}$ und $[\![M]\!] = [\![\mathsf{C}]\!]$) nach der Definition von $\lhd_\iota$. Wenn wir nun $[\![M]\!] \in \mathbb{N} \cup \mathbb{B}$ voraussetzen (insbesondere also $[\![M]\!] \neq \bot$), so muß der zweite Fall vorliegen, und wir haben $M \to^* \mathsf{C}$. Damit ist $\Leftarrow$ in dem Adäquatheitssatz ebenfalls bewiesen. □

7. Operationales Verhalten und Denotation beliebiger PCF-Terme

Im Adäquatheitssatz 5.6 hatten wir gesehen, daß für geschlossene PCF-Terme atomaren Typs (diese Terme hatten wir Programme genannt) das operationale Verhalten durch die Semantik, d.h. durch die Interpretation im Standard-Modell, beschrieben wird. Wir steht es mit PCF-Termen beliebigen Typs?

Einen beliebigen PCF-Term M kann man als „Unterprogramm" auffassen, das man in Programmen an einer oder mehreren Stellen verwenden kann. zu diesem Zweck müssen wir eine Bezeichnung für „Programme mit Löchern" einführen, in die man dann Terme passenden Typs einfügen kann. Solche Terme mit Löchern nennt man auch Kontexte. Formal kann man diese so einführen:

7.1 Kontexte. Es sei $[\ ]^\sigma$ eine Konstante vom Typ σ, die die Rolle eines „Platzhalters" übernimmt. Ein *Kontext* ist ein typisierter Term, der aus den Variablen, der Konstanten $[\ ]^\sigma$ sowie den übrigen PCF-Konstanten mittels Applikation und Abstraktion aufgebaut wird. (Anstatt sich auf einen Platzhalter zu beschränken, kann man auch mehrere Platzhalter $[\ ]^{\sigma_1}, \ldots, [\ ]^{\sigma_k}$ verwenden.)

Wir schreiben auch $C[\ ]^\sigma$ für einen Kontext, in dem der Platzhalter $[\ ]^\sigma$ vorkommt. Der Platzhalter kann auch mehrfach auftreten.

In solch einen Kontext $C[\ ]^\sigma$ können wir einen beliebigen PCF-Term M vom Typ σ anstelle des Platzhalters $[\ ]^\sigma$ einsetzen. Dieser Term wird dann an jeder Stelle eingefügt, an der der Platzhalter auftritt. Für den entstandenen Term schreiben wir $C[M]$. Unter dieser Einsetzung ist die sogenannte *naive Substitution* zu verstehen, d.h. gebundene Variablen werden nicht umbenannt.

Beispiel. Wir betrachten den Kontext $C[\ ]^{\nu\to\nu} \equiv \lambda y^{\nu\to\nu} x^\nu.[\ ](y([\ ]x))$:

$$C[\mathsf{SUCC}] \equiv \lambda y^{\nu\to\nu} x^\nu.\,\mathsf{SUCC}(y(\mathsf{SUCC}\,x))$$
$$C[y] \equiv \lambda y^{\nu\to\nu} x^\nu.y(y(yx))$$

Ein Kontext $C[\ ]^\sigma$ wird *Programmkontext* für M^σ genannt, falls $C[M]$ ein Programm, d.h. ein geschlossener PCF-Term atomaren Typs, ist.

7.2 Operationale Äquivalenz. Es seien M und N PCF-Terme des Typs σ. Wir sagen, M und N seien *operational äquivalent* und schreiben $M \approx_{\mathrm{op}} N$, falls für jeden Programmkontext $C[\ ]$ für M und N gilt:

$$C[M] \to^* c \iff C[N] \to^* c$$

Wir sagen, N sei *operational besser* als M und schreiben $M \lesssim_{\mathrm{op}} N$, falls für jeden Programmkontext $C[\ ]$ für M und N gilt:

$$C[M] \to^* c \implies C[N] \to^* c$$

Dabei bezeichne c eine Konstante atomaren Typs.

Operational äquivalente Terme kann man in Programmen gegeneinander austauschen, ohne daß sich am operationalen Verhalten des Programms etwas ändert.

7.3 Hilfssatz. Aus $M \lesssim_{\mathrm{op}} N$ folgt $C'[M] \lesssim_{\mathrm{op}} C'[N]$ für jeden Kontext $C'[\ ]$.

Beweis. Sei $C[\ ]$ irgendein Programmkontext für $C'[M]$ und $C'[N]$. Dann ist $C[C'[\ ]]$ ein Programmkontext für M und N, und aus $M \lesssim_{\text{op}} N$ folgt $C[C'[M]] \to^* c \implies C[C'[N]] \to^* c$. □

Die Frage ist nun, ob man die operationale Äquivalenz von Termen an ihrer Denotation in der Standard-Semantik erkennen kann.

7.4 Hilfssatz. Es sei $C[\ ]^\sigma$ ein Kontext, M und N seien PCF-Terme, die freie Variablen enthalten dürfen. Wenn $[\![M]\!]_\varrho \sqsubseteq [\![N]\!]_\varrho$ für jede Variablenbelegung ϱ gilt, so folgt $[\![C[M]]\!]_\varrho \sqsubseteq [\![C[N]]\!]_\varrho$ für jede Variablenbelegung ϱ.

Beweis durch Induktion über den Aufbau des Kontextes.

(1) $C[\ ]$ ist atomar, d.h. $C[\ ] \equiv x^\sigma$ oder $C[\ ] \equiv [\ ]$ oder $C[\ ]$ ist eine PCF-Konstante. In allen Fällen ist die Behauptung klar.

(2) $C[\ ] \equiv C_1[\ ]C_2[\ ]$. Nach Induktionsvoraussetzung gilt $[\![C_1[M]]\!]_\varrho \sqsubseteq [\![C_1[M]]\!]_\varrho$ und $[\![C_2[M]]\!]_\varrho \sqsubseteq [\![C_2[M]]\!]_\varrho$. Daraus folgt

$$[\![C[M]]\!]_\varrho = [\![C_1[M]]\!]_\varrho([\![C_2[M]]\!]_\varrho) \overset{\text{(I.A.)}}{\sqsubseteq} [\![C_1[N]]\!]_\varrho([\![C_2[N]]\!])_\varrho = [\![C[N]]\!]_\varrho$$

(3) $C[\ ] \equiv \lambda x^\sigma.C_1[\ ]$. Nach Induktionsvoraussetzung gilt $[\![C_1[M]]\!]_\varrho \sqsubseteq [\![C_1[N]]\!]_\varrho$ für alle ϱ. Also

$$[\![\lambda x^\sigma.C_1[M]]\!]_\varrho = (a \mapsto [\![C_1[M]]\!]_{\varrho[x\mapsto a]}) \overset{\text{(I.A.)}}{\sqsubseteq} (a \mapsto [\![C_1[N]]\!]_{\varrho[x\mapsto a]}) = [\![\lambda x^\sigma.C_1[N]]\!]_\varrho$$

□

7.5 Satz. Aus der semantischen Gleichheit zweier PCF-Terme folgt ihre operationale Gleichheit, genauer: Sind M und N zwei PCF-Terme gleichen Typs, dann gilt

(1) Wenn $[\![M]\!]_\varrho \sqsubseteq [\![N]\!]_\varrho$ für alle ϱ, so folgt $M \lesssim_{\text{op}} N$.

(2) Wenn $[\![M]\!]_\varrho = [\![N]\!]_\varrho$ für alle ϱ, so folgt $M \approx_{\text{op}} N$.

Beweis. Es ist klar, daß 2 aus 1 folgt. Für den Beweis von 1 setzen wir voraus, daß $[\![M]\!]_\varrho \sqsubseteq [\![N]\!]_\varrho$ für alle Variablenbelegungen ϱ gilt. Es sei $C[\ ]$ ein Programmkontext für M und N.

Nach 7.4 gilt $[\![C[M]]\!] \sqsubseteq [\![C[N]]\!]$. Wenn nun $C[M] \to^* c$, so folgt $[\![C[M]]\!] = [\![c]\!]$ und folglich $[\![c]\!] \sqsubseteq [\![C[N]]\!]$. Da $[\![c]\!]$ ein maximales Element in dem Bereich $\mathbb{B}_\perp$ bzw. $\mathbb{N}_\perp$ ist, folgt $[\![C[N]]\!] = [\![c]\!]$. Nach dem Adäquatheitssatz 5.6 schließen wir $C[N] \to^* c$. Damit gilt $M \lesssim_{\text{op}} N$. □

7.6 Volle Abstraktheit. In 7.5 haben wir gezeigt, daß im Falle des Standardmodells aus der semantischen Gleichheit zweier Terme ihre operationale Gleichheit folgt. Ideal wäre es, wenn auch die Umkehrung gelten würde, d.h. wenn operationale und semantische Gleichheit übereinstimmen würden. Ein Modell, in dem dies der Fall ist, nennt man *voll abstrakt*.

Leider gilt

7.7 Satz. Das Standardmodell für PCF ist nicht voll abstrakt.

Beweis. Der Beweis wurde in Aufgabe 60 durchgeführt. Teil (f) besagt gerade, daß das Standardmodell nicht voll abstrakt ist. □

Zur Durchführung benötigt man noch den folgenden Satz:

7.8 Satz. Es sei $\mathcal{R} = (R_\sigma)_{\sigma\in\Sigma}$ eine n-stellige logische Relation auf dem Standardmodell für PCF. Falls $\big([\![c]\!],\ldots,[\![c]\!]\big) \in R_\sigma$ für jede PCF-Konstante c vom Typ σ, dann folgt $\big([\![M]\!],\ldots,[\![M]\!]\big) \in R_\tau$ für jeden geschlossenen PCF-Term M vom Typ τ.

Beweis. Der Beweis wird induktiv über den Aufbau von M geführt. Der entsprechende Beweis aus IV.3.2 für den einfach typisierten λ-Kalkül läßt sich leicht übertragen. □

7.9 Satz von Milner. Es gibt ein voll abstraktes Modell für PCF, das bis auf Isomorphie eindeutig bestimmt ist.

Dieser Satz soll hier nicht bewiesen werden, da er den Rahmen der Vorlesung sprengen würde. Milners voll abstraktes Modell ist in der Tat sehr abstrakt. Eine einfache Beschreibung des Modells ist nicht bekannt.

8. Das Milnersche Kontextlemma

In dem Milnerschen Kontextlemma wird eine einfachere Kennzeichnung der operationalen Äquivalenz gegeben. Diese benutzten wir, um zu zeigen, daß die Terme POR-TEST und $\lambda x.\Omega$ operational äquivalent sind.

8.1 Kontextlemma. Es seien M und N geschlossene PCF-Terme vom Typ $\sigma \equiv \tau_1 \to \tau_2 \to \cdots \to \tau_k \to \iota$ mit $\iota \in \{\beta,\nu\}$. Es gilt $M \lesssim N$ genau dann, wenn für beliebige geschlossene PCF-Terme $P_1 : \tau_1, \ldots, P_k : \tau_k$ gilt:

$$MP_1 \ldots P_k \to^* c \implies NP_1 \ldots P_k \to^* c$$

Folgerung. Es gilt

$$M \approx N \iff \big(\forall P_1 : \tau_1, \ldots, P_k : \tau_k.\ MP_1 \ldots P_k \to^* c \iff NP_1 \ldots P_k \to^* c\big)$$

Im folgenden seien $M, N, \ldots$ stets geschlossene PCF-Terme vom Typ $\sigma \equiv \tau_1 \to \tau_2 \to \cdots \to \tau_k \to \iota$ mit $\iota \in \{\beta,\nu\}$. Für die Zwecke des Beweises definieren wir

$$M \lesssim_s N :\iff \text{Für alle geschlossenen Terme } P_1 : \tau_1, \ldots, P_k : \tau_k.$$
$$MP_1 \ldots P_k \to^* c \implies NP_1 \ldots P_k \to^* c\big)$$

Es ist klar, daß $M \lesssim N \implies M \lesssim_s N$, denn in der Definition von $M \lesssim_s N$ benutzen wir nur die speziellen Kontexte $C[\ ] \equiv [\ ]P_1 \ldots P_k$.

Zum Beweis der Umkehrung benutzen wir wieder die logische Relation $\lhd$ zwischen dem Standardmodell und den geschlossenen PCF-Termen. (Dieser Beweis ist wesentlich eleganter als der ursprüngliche, rein syntaktische Beweis des Kontextlemmas durch Milner.)

8.2 Hilfssatz. $f \lhd_\sigma M,\ M \lesssim_s N \implies f \lhd_\sigma N$

Beweis. Sei $f \lhd_\sigma M$. Nach 6.2 gilt für beliebige Paare $a_1 \lhd_{\tau_1} P_1, \ldots, a_k \lhd_{\tau_k} P_k$

$$f(a_1, \ldots, a_k) \lhd_\iota MP_1 \ldots P_k$$

d.h. $f(a_1, \ldots, a_k) = \perp$ (und dann auch $f(a_1, \ldots, a_k) \lhd_\iota NP_1 \ldots P_k$) oder $MP_1 \ldots P_k \to^* C$ mit $[\![C]\!] = f(a_1, \ldots, a_k)$.

Wenn nun $M \lesssim_s N$, dann gilt nach Definition auch $NP_1 \ldots P_k \to^* C$ und folglich $f(a_1, \ldots, a_k) \lhd_\iota NP_1 \ldots P_k$ auch im zweiten Fall. Also $f \lhd_\sigma N$. $\square$

8.3 Hilfssatz. $M \lesssim_s N \implies [\![M]\!] \lhd_\sigma N$

Beweis. Nach 6.5 gilt $[\![M]\!] \lhd M$. Wenn zusätzlich $M \lesssim_s N$, so folgt $[\![M]\!] \lhd_\sigma N$ aus 8.2. $\square$

8.4 Hilfssatz. $M \lesssim_s N \implies M \lesssim N$

Beweis. Wir setzen $M \lesssim_s N$ voraus. Aus 8.3 wissen wir $[\![M]\!] \lhd_\sigma N$. Nun sei $C[\ ]^\sigma$ ein beliebiger Programmkontext für M und N. Da $C[\ ]^\sigma$, M und N keine freien Variablen enthalten, können wir $[\ ]^\sigma$ als freie Variable in $C[\ ]^\sigma$ betrachten und 6.5 in seiner verallgemeinerten Form anwenden.

Wir schließen

$$\underbrace{[\![C[\ ]^\sigma]\!]_{\left[[\ ]^\sigma \mapsto [\![M]\!]\right]}}_{=[\![C[M]]\!]} \lhd_\iota \underbrace{\left(C[\ ]^\sigma\right)\left[[\ ]^\sigma := N\right]}_{=C[N]}$$

d.h. $[\![C[M]]\!] \lhd_\iota C[N]$. Wenn nun $C[M] \to^* c$, dann $[\![C[M]]\!] = [\![c]\!] \neq \perp$, und nach Definition folgt $C[N] \to^* c$. Also $M \lesssim N$. $\square$

Damit ist das Kontextlemma bewiesen. Dieser einfache semantische Beweis des Kontextlemmas rechtfertigt die Mühe, die wir uns in Abschnitt 6 mit der Relation $\lhd$ gemacht hatten.

KAPITEL VI

Modelle für den untypisierten λ-Kalkül

Wir erinnern an die Syntax des ungetypten λ-Kalküls: Es seien $v_1, v_2, \dots$ Variablen, die wir mit $x, y, \dots$ bezeichnen. λ-Terme werden induktiv definiert:

(1) x ist ein λ-Term für jede Variable x.

(2) Sind M und N λ-Terme, dann ist auch (MN) ein λ-Term.

(3) Ist x eine Variable und M ein λ-Term, dann ist auch $(\lambda x.M)$ ein λ-Term.

1. Bereichsmodelle

In Kapitel V haben wir gesehen, daß Bereiche (also geordnete Mengen mit $\perp$, in denen gerichtete Teilmengen stets ein Supremum besitzen) dafür geeignet sind, PCF zu interpretieren. Insbesondere konnte man die Y_σ durch die „kleinsten Fixpunktoperatoren"

$$\mathbb{Y}_\sigma(f) = \bigsqcup_{n \in \mathbb{N}} f^n(\perp) \quad \text{für alle } f \in [D_\sigma \to D_\sigma]$$

interpretieren. Allerdings mußte man sich dabei auf *stetige* Funktionen $f\colon D_\sigma \to D_\sigma$ beschränken, d.h. auf Funktionen, die gerichtete Suprema erhalten. Da man im ungetypten λ-Kalkül „Fixpunktkombinatoren" hat, die ähnliche operationale Eigenschaften wie die D_σ haben, liegt es nahe, den Versuch zu machen, die Terme des untypisieren λ-Kalküls in Bereichen zu interpretieren. Da wir keine Typen haben, soll die Interpretation in einem festen Bereich D erfolgen.

Sei also D ein Bereich. Jeden Term M kann man bei der Applikation sowohl als Argument als auch als Funktion benutzen. Die Interpretation $[\![M]\!]$ von M in D muß man also auch als stetige Funktion von D nach D verstehen können. Wir geben also zusätzlich eine Funktion

$$\Phi\colon D \to [D \to D]$$

vor, von der wir zunächst nur verlangen, daß sie stetig ist. Für jedes $d \in D$ ist also $\Phi(d)\colon D \to D$ stetig. Wollen wir $d \in D$ als Funktion betrachten, schreiben wir demnach $\Phi(d)$.

Umgekehrt wird die Abstraktion $\lambda x.M$ als stetige Funktion $(a \mapsto [\![M]\!]_{\varrho[x \mapsto a]})\colon D \to D$ interpretiert werden. Da wir aber alle Terme – also auch $\lambda x.M$ – in D interpretieren wollen, werden wir der stetigen Funktion $f\colon D \to D$ ein Element $\Psi(f) \in D$ zuordnen. Wir geben also zusätzlich eine Abbildung

$$\Psi\colon [D \to D] \to D$$

vor, von der wir ebenfalls zunächst nur Stetigkeit verlangen.

1.1 Interpretation von Termen. Gegeben sei in Bereich D zusammen mit zwei stetigen Abbildungen

$$D \underset{\Psi}{\overset{\Phi}{\rightleftarrows}} [D \to D]$$

Dabei ist $[D \to D]$ der Bereich der stetigen Abbildungen $f\colon D \to D$.

Die Variablen x dürfen mit beliebigen Elementen $\varrho(x) \in D$ belegt werden. Wir definieren nun induktiv über den Termaufbau simultan für beliebige Variablenbelegungen ϱ:

(1) $[\![x]\!]_\varrho := \varrho(x)$

(2) $[\![MN]\!]_\varrho := \Phi\big([\![M]\!]_\varrho\big)\big([\![N]\!]_\varrho\big)$

(3) $[\![\lambda x.M]\!]_\varrho := \Psi\big(a \mapsto [\![M]\!]_{\varrho[x\mapsto a]}\colon D \to D\big)$

Damit haben wir jedem ungetypten λ-Term M eine Interpretation $[\![M]\!] \in D$ zugeordnet, sofern die in 3 definiere Abbildung $a \mapsto [\![M]\!]_{\varrho[x\mapsto a]}\colon D \to D$ stetig ist. Um uns davon zu überzeugen, gehen wir ähnlich vor wie in V.5.3:

Sei M ein λ-Term und $x_1, \ldots, x_k$ verschiedene Variablen, unter denen alle freien Variablen von M vorkommen. Der Wert $[\![M]\!]$ wird nur von der Belegung der Variablen $x_1, \ldots, x_k$ abhängen. Wir schreiben $[\![M]\!](a_1, \ldots, a_k)$ für $[\![M]\!]_\varrho$, sofern $\varrho(x_1) = a_1, \ldots, \varrho(x_k) = a_k$.

Wir zeigen nun induktiv über den Termaufbau von M, daß $[\![M]\!](a_1, \ldots, a_k) \in D$ und daß $(a_1, \ldots, a_k) \mapsto [\![M]\!](a_1, \ldots, a_k)\colon D^k \to D$ stetig ist.

(1) $M \equiv x_i$. Es seien $x_1, \ldots, x_k$ Variablen, unter denen x_i vorkommt. Dann ist $[\![x_i]\!](a_1, \ldots, a_i, \ldots, a_k) = a_i \in D$ und $(a_1, \ldots, a_i, \ldots, a_k) \mapsto a_i\colon D^k \to D$ ist als Projektion auf den i-ten Faktor stetig.

(2) $M \equiv NP$. Es seien $x_1, \ldots, x_k$ Variablen, unter den die freien Variablen von M – also auch die von N und P – vorkommen. Nach Induktionsvoraussetzung sind

$$(a_1, \ldots, a_k) \mapsto [\![N]\!](a_1, \ldots, a_k)\colon D^k \to D$$
$$(a_1, \ldots, a_k) \mapsto [\![P]\!](a_1, \ldots, a_k)\colon D^k \to D$$

stetig. Also ist auch

$$\begin{array}{ccccc} (a_1, \ldots, a_k) & \mapsto & (\Phi([\![N]\!](a_1, \ldots, a_k)), [\![P]\!](a_1, \ldots, a_k)) & \mapsto & \Phi([\![N]\!](a_1, \ldots, a_k))([\![P]\!](a_1, \ldots, a_k)) \\ D^k & \to & [D \to D] \times D & \xrightarrow{\text{apply}} & D \end{array}$$

als Hintereinanderausführung stetiger Funktionen stetig. Dabei wird die Stetigkeit von Φ benutzt. Der letzte Term ist nach Definition gleich $[\![M]\!](a_1, \ldots, a_k)$.

(3) $M \equiv \lambda x.N$. Es seien $x_1, \ldots, x_k$ die freien Variablen in M. Die freien Variablen von N sind $x_1, \ldots, x_k$ und möglicherweise auch noch x. Nach Induktionsvoraussetzung ist

$$(a_1, \ldots, a_k, a) \mapsto [\![N]\!](a_1, \ldots, a_k, a)\colon D^k \times D \to D$$

stetig. Anwendung von curry und Komposition mit Ψ liefert eine stetige Funktion

$$\begin{array}{ccccc} (a_1, \ldots, a_k) & \mapsto & (a \mapsto [\![N]\!](a_1, \ldots, a_k, a)) & \mapsto & \Psi(a \mapsto ([\![N]\!](a_1, \ldots, a_k, a)) \\ D^k & \to & [D \to D] & \xrightarrow{\Psi} & D \end{array}$$

und letzteres ist nach Definition $[\![\lambda x.N]\!](a_1, \ldots, a_k) = [\![M]\!](a_1, \ldots, a_k)$. Man bemerke, daß hier die Stetigkeit von Ψ benutzt wurde.

Damit ist unsere Behauptung bewiesen, und wir haben alle λ-Terme M durch ein Element $[\![M]\!]_\varrho \in D$ für jede Variablenbelegung ϱ interpretiert.

1.2 Korrektheit von β-Modellen. Unsere Interpretation sollte natürlich so gewählt sein, daß β-äquivalente Terme M und N durch dasselbe Element von D interpretiert werden. Die Frage ist also:

$$(*) \qquad \text{Gilt} \quad M \simeq_\beta N \implies [\![M]\!] = [\![N]\!] \quad ?$$

Für den Beweis von $(*)$ kann man genauso wie im Beweis der Korrektheit der Henkin-Modelle des einfach typisieren λ-Kalküls vorgehen, vgl. IV.2.1. Man muß nur nachweisen, daß die Axiome und Regeln der β-Äquivalenz (siehe I.4.9) bei der Interpretation erfüllt bleiben. Für die Axiome (ϱ), (σ), (τ), (μ), (ν) und (ξ) ist dies klar. Es verbleibt zu überlegen, ob (α) und (β) bei der Interpretation gültig bleiben, d.h. ob sich $[\![M]\!]_\varrho$ bei der Umbenennung der gebundenen Variablen in M und bei β-Reduktion nicht verändert. (Die η-Regel stellen wir zurück.)

Dazu beweisen wir zunächst den folgenden Hilfssatz:

Hilfssatz. $[\![M[x := N]_\varrho]\!] = [\![M]\!]_{\varrho[x \mapsto [\![N]\!]_\varrho]}$

Der Beweis dieses Hilfssatzes ist wortwörtlich derselbe wie der des entsprechenden Hilfssatzes 2.2 in Kapitel IV. Man muß nur die dort vorkommenden Funktionen $\mathrm{app}_{\sigma,\tau}$ durch Φ und $\mathrm{app}^{-1}_{\sigma,\tau}$ durch Ψ ersetzen. □

Nun kann man zeigen, daß Umbenennung von gebundenen Variablen in M die Interpretation $[\![M]\!]_\varrho$ nicht verändert. Dies verläuft wortwörtlich wie der entsprechende Beweis für Henkin-Modelle im Abschluß an den Hilfssatz IV.2.2. Dabei ist wieder $\mathrm{app}^{-1}_{\sigma,\tau}$ durch Ψ zu ersetzen.

Den Fall der β-Reduktion wollen wir – in Analogie zu dem schon mehrfach erwähnten Hilfssatz aus Kapitel IV und dessen Folgen – genauer betrachten:

$$(*') \qquad \text{Gilt} \quad [\![(\lambda x.M)N]\!]_\varrho = [\![M[x := N]]\!]_\varrho \quad ?$$

Betrachten wir zunächst die rechte Seite von $(*')$:

$$[\![M[x := N]]\!]_\varrho = [\![M]\!]_{\varrho[x \mapsto [\![N]\!]_\varrho]}$$

nach obigem Hilfssatz. Die linke Seite sieht wie folgt aus:

$$\begin{aligned} [\![(\lambda x.M)N]\!] &= \big(\Phi([\![\lambda x.M]\!]_\varrho)\big)([\![N]\!]_\varrho) \\ &= \big(\Phi\big(\Psi(a \mapsto [\![M]\!]_{\varrho[x \mapsto a]})\big)\big)([\![N]\!]_\varrho) \end{aligned}$$

Unter unseren bisherigen Voraussetzungen gibt es keinen Grund, weshalb die linke und die rechte Seite von $(*')$ gleich sein müßten. Zu diesem Zweck fordern wir zusätzlich

$$\Phi(\Psi(f)) = f \text{ für alle } f \in [D \to D] \quad \text{bzw.} \quad \Phi \circ \Psi = \mathrm{id}_{[D \to D]}$$

Mit $f = (a \mapsto [\![M]\!]_{\varrho[x \mapsto a]})$ gilt dann nämlich

$$\begin{aligned} \text{linke Seite} = [\![(\lambda x.M)N]\!]_\varrho &= (\Phi(\Psi(f)))([\![N]\!]_\varrho) \\ &= f([\![N]\!]_\varrho) \end{aligned}$$

$$\begin{aligned} &= [\![M]\!]_{\varrho[x\mapsto[\![N]\!]_\varrho]} \\ &= [\![M[x := N]]\!]_\varrho = \text{rechte Seite} \end{aligned}$$

Unter der Zusatzvoraussetzung $\Phi \circ \Psi = \mathrm{id}_{D\to D}$ haben wir also die β-Äquivalenz von Termen korrekt interpretiert.

Zusammenfassend erhalten wir also

1.3 Satz. Es sei D ein Bereich. Haben wir stetige Abbildungen

$$D \underset{\Psi}{\overset{\Phi}{\rightleftarrows}} [D \to D]$$

so daß $\Phi \circ \Psi = \mathrm{id}_{D\to D}$ und interpretieren wir die Terme M des untypisierten λ-Kalküls als Elemente $[\![M]\!]_\varrho \in D$ für beliebige Variablenbelegungen ϱ wie in 1.1, so haben wir

$$M \simeq_\beta N \implies [\![M]\!]_\varrho = [\![N]\!]_\varrho \quad \text{für alle } \varrho$$

1.4 Definition. Jeden Bereich D zusammen mit stetigen Abbildungen

$$D \underset{\Psi}{\overset{\Phi}{\rightleftarrows}} [D \to D]$$

so daß $\Phi \circ \Psi = \mathrm{id}_{D\to D}$ gilt, nennen wir *Modell des untypisieren λ-Kalküls.*

Wir fragen uns nun, ob ein solches Modell auch für die η-Äquivalenz korrekt ist:

(∗∗) Gilt $M \simeq_\eta N \implies [\![M]\!]_\varrho = [\![N]\!]_\varrho$ für alle ϱ?

Dafür genügt es zu untersuchen, ob

(∗∗′) $[\![\lambda x.Mx]\!]_\varrho = [\![M]\!]_\varrho$für alle ϱ

gilt, sofern x in M nicht frei vorkommt. Wir bemerken:

- Ist x nicht frei in M, so hängt $[\![M]\!]_\varrho$ nicht davon ab, mit welchem Wert die Variable x durch ϱ belegt wird. Insbesondere gilt also $[\![M]\!]_\varrho = [\![M]\!]_{\varrho[x\mapsto a]}$.
- Sei $f\colon D \to D$ stetig. Es gilt $f = (a \mapsto f(a))$, denn $(a \mapsto f(a))$ ist nur eine andere Schreibweise für f.

Sei nun x nicht frei in M. Dann gilt

$$\begin{aligned} [\![Mx]\!]_{\varrho[x\mapsto a]} &= (\Phi([\![M]\!]_{\varrho[x\mapsto a]}))([\![x]\!]_{\varrho[x\mapsto a]}) \\ &= (\Phi([\![M]\!]_\varrho))(a) \end{aligned}$$

also

$$\begin{aligned} [\![\lambda x.Mx]\!]_\varrho &= \Psi(a \mapsto [\![Mx]\!]_{\varrho[x\mapsto a]}) \\ &= \Psi(a \mapsto (\Phi([\![M]\!]_\varrho))(a)) \\ &= \Psi(\Phi([\![M]\!]_\varrho)) \end{aligned}$$

Wenn wir also zusätzlich fordern, daß

$$\Psi(\Phi(d)) = d \text{ für alle } d \in D \quad \text{bzw.} \quad \Psi \circ \Phi = \mathrm{id}_D$$

dann gilt die Gleichung (∗∗′), sofern x nicht frei in M vorkommt.

1.5 Definition. Ein Modell des untypisierten λ-Kalküls wird *extensional* genannt, falls zusätzlich $\Psi \circ \Phi = \mathrm{id}_D$ gilt.

Wir haben gezeigt:

1.6 Satz. Ein extensionales Modell für den λ-Kalkül erfüllt

$$M \simeq_{\beta\eta} N \implies [\![M]\!]_\varrho = [\![N]\!]_\varrho \quad \text{für alle } \varrho$$

d.h. es ist korrekt für die $\beta\eta$-Äquivalenz untypisierter λ-Terme.

1.7 Bereichsisomorphismen. Es seien D und E Bereiche. Eine Funktion $f\colon D \to E$ wird *(Bereichs-)Isomorphismus* genannt, falls f bijektiv und stetig ist und die Umkehrabbildung $f^{-1}\colon E \to D$ ebenfalls stetig ist.

Falls eine solche Funktion existiert, werden die Bereiche D und E *isomorph* genannt, und man schreibt $D \cong E$.

Isomorphe Bereiche haben die selben ordnungstheoretischen Eigenschaften. Ist $\varphi\colon D \to E$ ein Isomorphismus, so gilt

$$\begin{aligned} a \sqsubseteq b \text{ in } D &\iff \varphi(a) \sqsubseteq \varphi(b) \text{ in } E \\ a = \bigsqcup_{i\in I} a_i \text{ in } D &\iff \varphi(a) = \bigsqcup_{i\in I} \varphi(a_i) \text{ in } E \end{aligned}$$

Beweis. Aus der Ordnungserhaltung von φ und φ^{-1} folgt

$$a \sqsubseteq b \implies \varphi(a) \sqsubseteq \varphi(b) \implies a = \varphi^{-1}(\varphi(a)) \sqsubseteq \varphi^{-1}(\varphi(b)) = b$$

und aus der Stetigkeit der Abbildungen

$$a = \bigsqcup_{i\in I} a_i \implies \varphi(a) = \varphi\big(\bigsqcup_{i\in I} a_i\big) = \bigsqcup_{i\in I} \varphi(a_i)$$

sowie

$$\varphi(a) = \bigsqcup_{i\in I} \varphi(a_i) \implies a = \varphi^{-1}(\varphi(a)) = \varphi^{-1}\big(\bigsqcup_{i\in I} \varphi(a_i)\big) = \bigsqcup_{i\in I} \varphi^{-1}(\varphi(a_i)) = \bigsqcup_{i\in I} a_i$$

□

Sind $\varphi\colon D \to E$ und $\psi\colon E \to D$ stetige Abbildungen mit $\varphi \circ \psi = \mathrm{id}_E$ und $\psi \circ \varphi = \mathrm{id}_D$, so sind φ und ψ bijektiv und $\psi = \varphi^{-1}$, d.h. φ und ψ sind zueinander inverse Isomorphismen.

Um ein extensionales Modell für den untypisierten λ-Kalkül zu finden, müssen wir also einen Bereich D finden, der zu dem Bereich $[D \to D]$ aller stetigen Funktionen $f\colon D \to D$ isomorph ist. Gibt es überhaupt solche Bereiche, und wenn ja, wie kann man solche finden?

Natürlich ist der einelementige Bereich $D = \{\bot\}$ ein Beispiel, jedoch ist dieses Beispiel für die Semantik unbrauchbar, da alle Terme dieselbe Denotation bekommen. Daß es keine „einfachen“ Beispiele gibt, wurde schon in Aufgabe 67 gezeigt. Aber ist gibt komplexere, wie z.B. die D_∞-Modelle.

Zuvor betrachten wir jedoch einige nichtextensionale Modelle.

2. Graphenmodelle als nicht-extensionale Modelle

2.1 Bezeichnungen. In diesem Abschnitt seien stets

- A eine abzählbar unendliche Menge, $\alpha, \beta, \gamma, \ldots$ Elemente von A.
- $D = \mathfrak{P}(A)$ die Menge aller Teilmengen $U, V, W, \ldots$ von A, geordnet durch die Inklusionsrelation $\subseteq$. Damit ist $(D, \subseteq)$ insbesondere ein Bereich.
- $\mathfrak{P}_{\text{fin}}(A)$ die Menge aller *endlichen* Teilnengen $a, b, c, \ldots$ von A, ebenfalls durch $\subseteq$ geordnet.
- $i\colon \mathfrak{P}_{\text{fin}}(A) \times A \to A$ eine *injektive* Abbildung

Wir wollen für jede vorgegebene Injektion $i\colon \mathfrak{P}_{\text{fin}}(A) \times A \to A$ ein Modell für den λ_β-Kalkül konstruieren. Diese Modelle sind alle nicht extensional. Das Paar (A, i) nennt man dann auch ein *Graphenmodell.*

Da A als abzählbar unendlich vorausgesetzt ist, ist auch die Menge $\mathfrak{P}_{\text{fin}}(A)$ aller endlichen Teilmengen $a \subseteq A$ abzählbar, und folglich ist auch $\mathfrak{P}_{\text{fin}}(A) \times A$ abzählbar. Darum gibt es sicherlich injektive – ja sogar bijektive – Abbildungen $i\colon \mathfrak{P}_{\text{fin}}(A) \times A \to A$. Einige Beispiele:

2.2 Beispiel.

- $\mathfrak{P}_\omega$. Es sei $A = \mathbb{N}$ die Menge aller natürlichen Zahlen. Die Abbildung $j : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ mit
  $$j(\alpha, \beta) := \tfrac{1}{2}(\alpha + \beta)(\alpha + \beta + 1) + \beta$$
  ist bijektiv. Sie entspricht der in Abbildung 1 visualisierten Abzählung der Gitterpunkte.

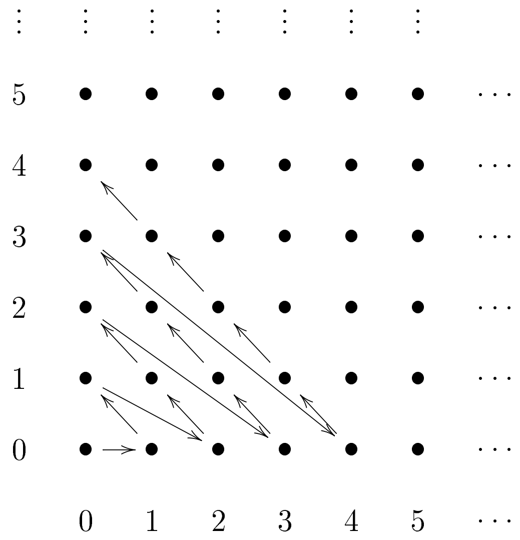


ABBILDUNG 1. Eine Abzählung von $\mathbb{N} \times \mathbb{N}$

Nun definieren wir eine Abbildung $j' \colon \mathfrak{P}_{\text{fin}}(\mathbb{N}) \to \mathbb{N}$, indem wir für jede endliche Menge $a \subseteq \mathbb{N}$ setzen:

$$j'(a) = \sum_{\alpha \in a} 2^\alpha$$

Diese Abbildung ist ebenfalls bijektiv, da jede natürliche Zahl β eine eindeutige Darstellung $\beta = \sum_{\alpha \in \mathbb{N}} a_\alpha 2^\alpha$ mit $a_\alpha \in \{0,1\}$ hat, wobei nur endlich viele $a_\alpha \neq 0$ sein dürfen. Die Umkehrabbildung ist gegeben durch

$$j'^{-1}(\beta) = \{\alpha \mid a_\alpha = 1\}$$

Insgesamt erhalten wir eine bijektive Abbildung $i_0 \colon \mathfrak{P}_{\text{fin}}(\mathbb{N}) \times \mathbb{N} \to \mathbb{N}$ durch

$$i_0(a, \alpha) = j(j'(a), \alpha)$$

- *Modell von Engeler.* Es sei P eine abzählbar unendliche Menge von „Atomen“. Wir definieren rekursiv ein Mengensystem $(A_n)_{n \in \mathbb{N}}$ durch

$$\begin{aligned}
A_0 &:= P \\
A_1 &:= P \cup (\mathfrak{P}_{\text{fin}}(A_0) \times A_0) \\
A_1 &:= P \cup (\mathfrak{P}_{\text{fin}}(A_1) \times A_1) \\
&\vdots \\
A_n &:= P \cup (\mathfrak{P}_{\text{fin}}(A_{n-1}) \times A_{n-1}) \\
&\vdots
\end{aligned}$$

Nun sei $A = \bigcup_{n \in \mathbb{N}} A_n$. Dann ist A abzählbar. und es gilt

$$A = P \cup (\mathfrak{P}_{\text{fin}}(A) \times A)$$

Dies sieht man leicht ein, wenn man zunächst $A_{n-1} \subseteq A_n$ induktiv beweist: $A_0 \subseteq A_1$ ist klar. $A_{n+1} = P \cup (\mathfrak{P}_{\text{fin}}(A_n) \times A_n) \supseteq P \cup (\mathfrak{P}_{\text{fin}}(A_{n-1}) \times A_{n-1}) = A_n$ nach Induktionsvoraussetzung.

Für $i \colon \mathfrak{P}_{\text{fin}}(A) \times A \to A$ wählt man die kanonische Injektion $i(a, \alpha) = (a, \alpha)$.

- $\mathfrak{P}_P$. Wir modifizieren das Modell von Engeler, indem wir $p \in P$ mit dem Paar $(\varnothing, p)$ identifizieren. Wir setzen also:

$$\begin{aligned}
A_0 &:= P \\
A_1 &:= P \cup \big((\mathfrak{P}_{\text{fin}}(A_0) \times A_0) \setminus (\{\varnothing\} \times A_0)\big) \\
A_2 &:= P \cup \big((\mathfrak{P}_{\text{fin}}(A_1) \times A_1) \setminus (\{\varnothing\} \times A_1)\big) \\
&\vdots \\
A_n &:= P \cup \big((\mathfrak{P}_{\text{fin}}(A_{n-1}) \times A_{n-1}) \setminus (\{\varnothing\} \times A_{n-1})\big) \\
&\vdots
\end{aligned}$$

und abschließend wieder $A = \bigcup_{n \in \mathbb{N}} A_n$. Von G. Longo wurde gezeigt, daß dieses Modell zu $\mathfrak{P}_\omega$ isomorph ist.

2.3 β-Modelle. Mit den Bezeichnungen aus 2.1 definieren wir:

- Für jede stetige Funktion $f\colon D \to D$ sei
$$\Psi(f) := \{i(a,\alpha) \mid \alpha \in f(a)\} \in D$$
- Für alle $U \in D$ definiere $\Phi(U)\colon D \to D$ durch
$$\Phi(U)(V) := \{\alpha \in A \mid \exists a \subseteq V.\ i(a,\alpha) \in U\}$$

Wir werden zeigen:

(1) $$\Phi(U)\colon D \to D \text{ ist stetig für alle } U \in D$$

(2) $$\Phi\colon D \to [D \to D] \text{ ist stetig}$$

(3) $$\Psi\colon [D \to D] \to D \text{ ist stetig}$$

(4) $$\Phi \circ \Psi = \mathrm{id}_{[D\to D]}$$

Gemäß 1.3 ist dann D ein β-Modell für den untypisierten λ-Kalkül. Weiterhin werden wir zeigen, daß dieses Modell niemals extensional sein wird.

Dazu einige Hilfssätze:

2.4 Hilfssatz. Sei $a \in \mathfrak{P}_{\mathrm{fin}}(A)$ und $(U_i)_{i\in I}$ ein gerichtetes System von Mengen $U_i \in D$. Dann gilt

(K) $$a \subseteq \bigcup_{i\in I} U_i \implies \exists i_0 \in I.\ a \subseteq U_{i_0}$$

Beweis. Sei $a = \{\alpha_1, \dots, \alpha_k\}$. Für jedes j gilt $\alpha_j \in a \subseteq \bigcup_{i\in I} U_i$. Also gibt es für jedes j ein $i_j \in I$ mit $\alpha_j \in U_{i_j}$. Da $(U_i)_{i\in I}$ gerichtet ist, gibt es ein $i_0 \in I$, so daß $U_{i_j} \in U_{i_0}$ für $i = 1, \dots, k$. Es folgt $\alpha_j \in U_{i_0}$ für alle $j = 1, \dots, k$, d.h. $a \subseteq U_{i_0}$. □

Zusatz. Ist a eine Teilmenge von A, so daß (K) für jedes gerichtete System $(U_i)_{i\in I}$ von Teilmengen $U_i \subseteq A$ gilt, dann muß a endlich sein.

Beweis. Sei $\{b_i \mid i \in I\}$ das System aller endlichen Teilmengen $b_i \subseteq a$. Dieses System ist gerichtet und $a = \bigcup_{i\in I} b_i$. Aus (K) folgt, daß es ein $i_0 \in I$ gibt, so daß $a = b_{i_0}$, und folglich ist a endlich. □

2.5 Hilfssatz. Es sei $f\colon D \to D$ ordnungserhaltend, d.h. $U \subseteq V \implies f(U) \subseteq f(V)$. Dann ist f genau dann stetig, wenn
$$f(U) = \bigcup_{a\subseteq U} f(a) \quad \text{für alle } U \subseteq A$$

Beweis. $\Rightarrow$: $U = \bigcup_{a\subseteq U} a$. Ist f stetig, so folgt $f(U) = f\bigl(\bigcup_{a\subseteq U} a\bigr) = \bigcup_{a\subseteq U} f(a)$.

$\Leftarrow$: Sei $U = \bigcup_{i\in I} U_i$. Wir wollen $f(U) = \bigcup_{i\in I}(U_i)$ zeigen. Nun gilt
$$\begin{aligned} f(U) &= \bigcup_{a\subseteq U} f(a) && \text{nach Voraussetzung}\\ &= \bigcup_{i\in I}\Bigl(\bigcup_{a\in U_i} f(a)\Bigr) && \text{denn } a \subseteq U = \bigcup U_i \text{ gdw. } a \in U_i \text{ für ein } i \in I \text{ (nach 2.4)} \end{aligned}$$

$$= \bigcup_{i\in I} f(U_i) \qquad \text{nach Voraussetzung}$$

$\square$

2.6 Hilfssatz. Jede ordnungserhaltende Abbildung $f\colon \mathfrak{P}_{\text{fin}}(A) \to D$ hat genau eine stetige Fortsetzung $\overline{f}\colon D \to D$.

Beweis. Ist $\overline{f}$ eine stetige Fortsetzung von f, so gilt für jedes $U \subseteq A$

$$\overline{f}(U) = \bigcup_{a\subseteq U} \overline{f}(a) = \bigcup_{a\in U} f(a)$$

nach 2.5, d.h. sofern f eine stetige Fortsetzung besitzt, so ist diese durch f eindeutig bestimmt.

Um die Existenz einer stetigen Fortsetzung $\overline{f}$ von f zu zeigen, setze

$$\overline{f}(U) := \bigcup_{a\subseteq U} f(a) \quad \text{für alle } U \subseteq A$$

Ist b endlich, so gilt $\overline{f}(b) = \bigcup_{a\subseteq U} f(a) = f(b)$, da $f(a) \subseteq f(b)$ für alle $a \subseteq b$, da f ordnungserhaltend ist. Also ist $\overline{f}$ eine Fortsetzung von f. Die Stetigkeit von f folgt nun direkt aus 2.5. $\square$

Zusatz. Es sei $\mathcal{M}(\mathfrak{P}_{\text{fin}}(A) \to D)$ die Menge aller ordnungserhaltenden Abbildungen $f\colon \mathfrak{P}_{\text{fin}}(A) \to D$, die durch

$$f_1 \sqsubseteq f_2 :\iff \forall a \in \mathfrak{P}_{\text{fin}}(A).\ f_1(a) \subseteq f_2(a)$$

geordnet wird. Die Abbildung

$$f \mapsto \overline{f}\colon \mathcal{M}(\mathfrak{P}_{\text{fin}}(A) \to D) \to [D \to D]$$

die jedem ordnungserhaltenden $f\colon \mathfrak{P}_{\text{fin}}(A) \to D$ die nach 2.6 eindeutig bestimmte Fortsetzung $\overline{f}\colon D \to D$ zuordnet, ist bijektiv.

Beweis. Die Umkehrabbildung besteht daraus, daß man jedem stetigen $g\colon D \to D$ die Einschränkung $g|_{\mathfrak{P}_{\text{fin}}(A)}\colon \mathfrak{P}_{\text{fin}}(A) \to D$ zuordnet. $\square$

Die Abbildung $f \mapsto \overline{f}$ ist sogar ein Ordnungsisomorphismus, denn $f_1 \sqsubseteq f_2 \iff \overline{f_1} \sqsubseteq \overline{f_2}$, wie man sofort sieht.

Zusammenfassung. Die Bereiche $\mathcal{M}(\mathfrak{P}_{\text{fin}}(A) \to D)$ und $[D \to D]$ sind isomorph.

2.7 Traces. Jeder ordnungserhaltenden Abbildung $f\colon \mathfrak{P}_{\text{fin}}(A) \to D$ ordnen wir eine Teilmenge von $\mathfrak{P}_{\text{fin}}(A) \times A$ zu, nämlich

$$\text{trace}(f) := \{(a, \alpha) \in \mathfrak{P}_{\text{fin}}(A) \times A \mid \alpha \in f(a)\}$$

Umgekehrt können wir jeder Teilmenge $R \subseteq \mathfrak{P}_{\text{fin}}(A) \times A$ eine Funktion

$$\text{fun}(R)\colon \mathfrak{P}_{\text{fin}}(A) \to D$$

zuordnen, indem wir setzen

$$\text{fun}(R)(a) := \{\alpha \in A \mid (a, \alpha) \in R\} \quad \text{für alle } a \in \mathfrak{P}_{\text{fin}}(A)$$

Es ist klar, daß f durch trace(f) eindeutig bestimmt ist. Genauer: $f = \mathrm{fun}(\mathrm{trace}(f))$

Allerdings wird nicht für jede Teilmenge $R \subseteq \mathfrak{P}_{\mathrm{fin}}(A) \times A$ die Abbildung fun(R) ordnungserhaltend. Dies ist genau dann der Fall, wenn R die folgende Bedingung erfüllt:

$$\text{(tr)} \qquad \forall a' \subseteq a.\ (a', \alpha) \in R \implies (a, \alpha) \in R$$

denn dies ist gleichbedeutend mit

$$\forall a' \subseteq a.\ \mathrm{fun}(R)(a') = \{\alpha \mid (a', \alpha) \in R\} \subseteq \{\alpha \mid (a, \alpha) \in R\} = \mathrm{fun}(R)(a)$$

Nun sei Tr die Menge aller $R \subseteq \mathfrak{P}_{\mathrm{fin}}(A) \times A$, die die Bedingung (tr) erfüllen. Man sieht leicht:

$$\varnothing \in Tr \qquad\qquad R_i \in Tr \text{ für } i \in I \implies \bigcup_{i \in I} R_i \in Tr$$

Insbesondere ist die durch $\subseteq$ geordnete Menge Tr ein Bereich. Ferner gilt für $R, R' \in Tr$:

$$\begin{aligned} R \subseteq R' &\iff \forall a \in \mathfrak{P}_{\mathrm{fin}}(A).\ \mathrm{fun}(R)(a) = \{a \mid (a, \alpha) \in R\} \subseteq \{\alpha \mid (a, \alpha) \in R'\} = \mathrm{fun}(R')(a) \\ &\iff \mathrm{fun}(R) \sqsubseteq \mathrm{fun}(R') \end{aligned}$$

Damit haben wir zueinander isomorphe Bereiche

$$Tr \underset{\text{trace}}{\overset{\text{fun}}{\rightleftarrows}} \mathcal{M}(\mathfrak{P}_{\mathrm{fin}}(A) \to D)$$

2.8 Abbilungen zwischen D und Tr. Für jede Teilmenge $R \subseteq \mathfrak{P}_{\mathrm{fin}}(A) \times A$ setzen wir nun

$$\overline{R} := \{(a, \alpha) \in \mathfrak{P}_{\mathrm{fin}}(A) \times A \mid \exists a' \subseteq a.\ (a', \alpha) \in R\}$$

$\overline{R}$ ist die kleinste Teilmenge von $\mathfrak{P}_{\mathrm{fin}}(A) \times A$, die R enthält und die Bedingung (tr) erfüllt. Es gilt $R = \overline{R} \iff R \in Tr$.

Für jede Familie R_i $(i \in I)$ von Teilmengen von $\mathfrak{P}_{\mathrm{fin}}(A) \times A$ gilt

$$\overline{\bigcup_{i \in I} R_i} = \bigcup_{i \in I} \overline{R_i}$$

denn

$$\begin{aligned} (a, \alpha) \in \overline{\bigcup_{i \in I} R_i} &\iff \exists a' \subseteq a.\ (a', \alpha) \in \bigcup_{i \in I} R_i \\ &\iff \exists a' \subseteq a.\ \exists i \in I.\ (a', \alpha) \in R_i \\ &\iff \exists i \in I.\ (a, \alpha) \in \overline{R_i} \\ &\iff (a, \alpha) \in \bigcup_{i \in I} \overline{R_i} \end{aligned}$$

Nun erklären wir Abbildungen

$$D \underset{\psi}{\overset{\varphi}{\rightleftarrows}} Tr$$

durch

$$\begin{aligned} \varphi(U) &:= \overline{i^{-1}(U)} = \{(a, \alpha \mid \exists a' \subseteq a.\ i(\alpha', \alpha) \in U\} \\ \psi(R) &:= i(R) = \{i(a, \alpha) \mid (a, \alpha) \in R\} \quad \text{für alle } R \in Tr \end{aligned}$$

Beide Abbildungen sind stetig. Für ψ ist das klar, für φ:

$$\varphi\big(\bigcup_{i\in I} U_i\big) = \overline{i^{-1}\big(\bigcup_{i\in I} U_i\big)} = \overline{\bigcup_{i\in I} i^{-1}(U_i)} = \bigcup_{i\in I} \overline{i^{-1}(U_i)} = \bigcup_{i\in I} \varphi(U_i)$$

Ferner gilt $\varphi \circ \psi = \mathrm{id}_{Tr}$, denn $\varphi(\psi(R)) = \overline{i^{-1}(i(R))} = \overline{R} = R$, da i injektiv und $R \in Tr$ ist.

2.9 Konstruktion von Φ und Ψ. Was noch bleibt, ist das Zusammensetzen der bislang konstruierten Abbildungen:

$$\begin{tikzcd}
D \arrow[r, shift left, "\varphi"] \arrow[rrr, bend left, "\Phi"] & Tr \arrow[l, shift left, dotted, "\psi"] \arrow[r, shift left, "\mathrm{fun}"] & \mathcal{M}(\mathfrak{P}_{\mathrm{fin}}(A) \to D) \arrow[l, shift left, "\mathrm{trace}"] \arrow[r, shift left, "f \mapsto \overline{f}"] & {[D \to D]} \arrow[l, shift left, "g \mapsto g|_{\mathfrak{P}_{\mathrm{fin}}(A)}"] \arrow[lll, bend left, "\Psi"]
\end{tikzcd}$$

Setze also

$$\Phi(U) := \overline{\mathrm{fun}(\varphi(U))} \quad \text{für alle } U \in D$$
$$\Psi(y) := \psi\big(\,\mathrm{trace}(g|_{\mathfrak{P}_{\mathrm{fin}}(A)})\big) \quad \text{für alle } g \in [D \to D]$$

Als Komposition stetiger Abbildungen sind Φ und Ψ stetig, und es gilt

$$\Phi \circ \Psi = \mathrm{id}_{D \to D}$$

d.h.

$$\begin{tikzcd}
D \arrow[r, shift left, "\Phi"] & {[D \to D]} \arrow[l, shift left, "\Psi"]
\end{tikzcd}$$

ist ein λ_β-Modell, denn für alle $g \in [D \to D]$ gilt

$$\begin{aligned}
\Phi(\Psi(g)) &= \overline{\mathrm{fun}(\varphi(\psi(\mathrm{trace}(g|_{\mathfrak{P}_{\mathrm{fin}}(A)}))))} && \text{nach 2.8}\\
&= \overline{\mathrm{fun}(\mathrm{trace}(g|_{\mathfrak{P}_{\mathrm{fin}}(A)}))} && \text{nach 2.7}\\
&= \overline{g|_{\mathfrak{P}_{\mathrm{fin}}(A)}} && \text{nach 2.6}\\
&= g
\end{aligned}$$

Für jede injektive Abbildung $i\colon \mathfrak{P}_{\mathrm{fin}}(A) \times A \to A$ erhalten wir also auf vorstehende Weise ein Modell für den λ_β-Kalkül. Diese Modelle sind aber *niemals* extensional: Wählt man nämlich ein $R \subseteq \mathfrak{P}_{\mathrm{fin}}(A) \times A$, das (tr) nicht erfüllt, dann gilt $R \subsetneqq \overline{R}$. Setzt man $U := i(R)$, so folgt $\psi(\varphi(U)) = i\big(\overline{i^{-1}(U)}\big) = i\big(\overline{i^{-1}(i(R))}\big) = i(\overline{R}) \supsetneqq U$. Es folgt $\Psi(\Phi(U)) \supsetneqq U$, d.h. $\Psi \circ \Phi \neq \mathrm{id}_D$.

3. D_∞-Modelle

Bevor wir ein extensionales D_∞-Modell konstruieren können, benötigen wir zunächst ein mathematisches Hilfsmittel: inverse Limites von Bereichen.

3.1 Projektions-Einbettungspaare. Seien D und E Bereiche. Ein Paar (p, e) von Abbildungen

$$\begin{tikzcd}
D \arrow[r, shift left, "p"] & E \arrow[l, shift left, dotted, "e"]
\end{tikzcd}$$

wird *Projektions-Einbettungspaar* (kurz P-E-Paar) genannt, falls p und e stetig sind und

$$p \circ e = \mathrm{id}_E \qquad\qquad e \circ p \sqsubseteq \mathrm{id}_D$$

Bemerkung. Ohne die letzte Bedingung hätte man ein *Retraktions-Einbettungspaar*, vgl. 12. Übungsblatt.

Bei einem Projektions-Einbettungspaar (p, e) ist p surjektiv und e injektiv. Aus der Surjektivität von p folgt, daß $p(\bot) = \bot$ ist, denn wäre $p(\bot) \sqsupset \bot$, so wäre $p(x) \sqsupseteq p(\bot) \sqsupset \bot$ für alle $x \in D$, und p wäre somit nicht surjektiv.

3.2 Inverse Limites. Es seien D_k $(k \in \mathbb{N})$ Bereiche. Ferner sei eine Folge von Einbettungs-Projektionspaaren gegeben:

$$\cdots \underset{\longleftarrow}{\longrightarrow} D_3 \underset{e_{3,2}}{\overset{p_{2,3}}{\rightleftarrows}} D_2 \underset{e_{2,1}}{\overset{p_{1,2}}{\rightleftarrows}} D_1 \underset{e_{1,0}}{\overset{p_{0,1}}{\rightleftarrows}} D_0$$

Für jedes solche System setzen wir

$$p_{k,k} = \mathrm{id}_{D_k} \qquad\qquad e_{k,k} = \mathrm{id}_{D_k}$$

sowie für $k < l$

$$p_{k,\ell} = p_{k,k+1} \circ \cdots \circ p_{\ell-1,\ell} \colon D_\ell \to D_k \qquad e_{\ell,k} = e_{\ell,\ell-1} \circ \cdots \circ e_{k+1,k} \colon D_k \to D_\ell$$

Man bemerke, daß $(p_{k,l}, e_{l,k})$ für alle $k < l$ Projektions-Einbettungspaare sind.

In dem direkten Produkt $\prod_{n\in\mathbb{N}} D_k$ aller $x := (x_k)_{k\in\mathbb{N}}$ mit $x_k \in D_k$ betrachten wir die Menge

$$D := \Big\{ x = (x_k)_{k\in\mathbb{N}} \in \prod_{k\in\mathbb{N}} D_k \mid \forall k \in \mathbb{N}.\ p_{k,k+1}(x_{k+1}) = x_k \Big\}$$

Wir nennen D den *inversen Limes* des Systems $(D_k, p_{k,k+1})$.

Dieser ist wieder ein Bereich, wobei die Ordnung die vom direkten Produkt induzierte ist: Wegen $p_{k,k+1}(\bot) = \bot$ gilt $(\bot_k)_{k\in\mathbb{N}} \in D$. Und ist $x^i := (x_k^i)_{k\in\mathbb{N}}$ mit $i \in I$ ein gerichtetes System von Elementen in D, so gilt $p_{k,k+1}(x_{k+1}^i) = x_k^i$ für jedes dieser Elemente. Für das koordinatenweise Supremum $\bigsqcup x^i = \big(\bigsqcup_{i\in I} x_k^i\big)$ gilt dann

$$p_{k,k+1}\big(\bigsqcup_{i\in I} x_{k+1}^i\big) = \bigsqcup_{i\in I} p_{k,k+1}(x_{k+1}^i) = \bigsqcup_{i\in I} x_k^i$$

d.h. $\bigsqcup_{i\in I} x^i$ liegt wieder in D. Somit ist D ein Bereich, den wir mit

$$\varprojlim (D_k, p_{k,k+1}, e_{k+1,k})_{k\in\mathbb{N}}$$

bezeichnen.

Indem wir die kanonische Projektion $p_n \colon \prod_{k\in\mathbb{N}} D_k \to D_n$ auf D einschränken, erhalten wir eine stetige Abbildung

$$p_n \colon D \to D_n \qquad \big((x_k)_{k\in\mathbb{N}}\big) \mapsto x_n$$

Für diese gilt $p_{n-1,n} \circ p_n = p_{n-1}$ für alle $n \in \mathbb{N}$, denn $p_{n-1,n}\big(p_n\big((x_k)_{k\in\mathbb{N}}\big)\big) = p_{n-1,n}(x_n) = x_{n-1} = p_{n-1}\big((x_k)_{k\in\mathbb{N}}\big)$.

Umgekehrt können wir auch eine stetige Abbildung

$$e_n \colon D_n \to D$$

folgendermaßen definieren: Für gegebenes $d \in D_n$ sei

$$d_k := \begin{cases} p_{k,n}(d) & \text{für } k \leq n \\ e_{k,n}(d) & \text{für } k > n \end{cases}$$

Es gilt dann

$$p_{k,k+1}(d_{k+1}) = \begin{cases} p_{k,k+1}\big(p_{k+1,n}(d_{k+1})\big) = p_{k,n}(d) = d_k & \text{für } k < n \\ p_{k,k+1}\big(e_{k+1,n}(d_{k+1})\big) & \\ = \big(\underbrace{p_{k,k+1} \circ e_{k+1,k}}_{=\mathrm{id}} \circ \underbrace{e_{k,k-1} \circ \cdots \circ e_{n+1,n}}_{=e_{k,n}}\big)(d) = d_k & \text{für } k \geq n \end{cases}$$

Also gilt $(d_k)_{k\in\mathbb{N}} \in D$, und wir können $e_n(d) = (d_k)_{k\in\mathbb{N}}$ setzen. Ähnlich wie für p_n gilt $e_n \circ e_{n,n-1} = e_{n-1}$ für alle $n \in \mathbb{N}$.

Desweiteren ist

$$D \underset{e_n}{\overset{p_n}{\rightleftarrows}} D_n$$

ein Projektions-Einbettungspaar: Es gilt $p_n\big(e_n(d)\big) = d$ für alle d und $e_n\big(p_n\big((x_k)_{k\in\mathbb{N}}\big)\big) = e_n(x_n) \sqsubseteq (x_k)_{k\in\mathbb{N}}$.

Das Diagramm in Abbildung 2 stellt die Gesamtsituation dar, ebenso wie die universelle Eigenschaft 3.4.

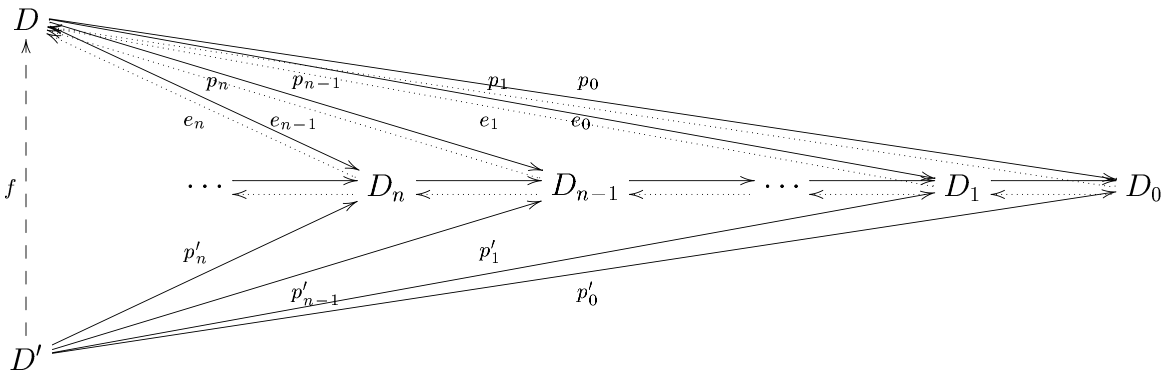


Abbildung 2. Der inverse Limes $D = \varprojlim(D_k, p_{k,k+1}, e_{k+1,k})_{k\in\mathbb{N}}$

3.3 Hilfssatz. Es gilt $\mathrm{id}_D = \bigsqcup_{n\in\mathbb{N}} e_n \circ p_n$.

Beweis. Dies sieht man, indem man für beliebiges $x = (x_n)_{n\in\mathbb{N}}$ die Beziehung $x = \big(\bigsqcup_{n\in\mathbb{N}} e_n \circ p_n\big)(x) = \bigsqcup_{n\in\mathbb{N}} e_n(x_n)$ nachprüft. □

3.4 Universelle Eigenschaft. Es sei $D = \varprojlim(D_k, p_{k,k+1}, e_{k+1,k})_{k\in\mathbb{N}}$ wie in 3.2 definiert.

Ist D' ein Bereich zusammen mit einer Folge $(p'_k)_{k\in\mathbb{N}}$ von stetigen Abbildungen $p_k\colon D' \to D_k$, so daß $p_{n-1,n} \circ p'_n = p'_{n-1}$ für alle $n \in \mathbb{N}$, dann gibt es eine eindeutig bestimmte stetige Abbildung $f\colon D' \to D$, so daß $p_n \circ f = p'_n$ für alle $n \in \mathbb{N}$.

Beweis. Eindeutigkeit von f. Sei $x' \in D'$. Aus $p_n \circ f = p'_n$ folgt für $y := f(x')$, daß $y_n = p_n(y) = p_n(f(x')) = p'_n(x')$ für jedes n, d.h. es muß $y = (y_n)_{n\in\mathbb{N}} = \big(p'_n(x')\big)$ gelten, und $y := f(x')$ ist durch die p'_n eindeutig bestimmt.

Existenz von f. Für jedes $x' \in D'$ setze $f(x') := (p'_n(x'))_{n\in\mathbb{N}}$. Nach Voraussetzung gilt $p_{n-1,n}(p'_n(x')) = p'_{n-1}(x')$, d.h. $f(x') \in D$. Es ist klar, daß $p_n(f(x')) = p'_n(x')$, d.h. $p_n \circ f = p'_n$. Ferner ist f stetig, da f koordinatenweise stetig ist. □

Bemerkung. In der Kategorientheorie wird der inverse Limes durch die Eigenschaft 3.4 gekennzeichnet. Um diese universelle Eigenschaft zu bekommen, muß man nur von einem System

$$(D_k, p_{k,k+1}\colon D_{k+1} \to D_k)_{k\in\mathbb{N}}$$

von stetigen Abbildungen $p_{k,k+1}$ ausgehen und die selbe Konstruktion wie in 3.2 durchführen. Es kann dann allerdings passieren, daß D nur aus einem einzigen Element $\perp$ besteht. Durch die Zusatzvoraussetzung, daß die $p_{k,k+1}$ Projektionen sind, erreichen wir, daß die Projektionen $p_n\colon D \to D_n$ alle surjektiv sind und die D_n vermittels der Einbettungen $e_n\colon D_n \to D$ als Teilbereiche von D betrachtet werden können.

Zusatzbemerkung. Der Bereich D ist gleichzeitig der *direkte* Limes des Systems

$$(D_k, e_{k+1,k}\colon D_k \to D_{k+1})_{k\in\mathbb{N}}$$

3.5 D_∞-Modelle. Wir betrachten nun spezielle Systeme

$$\cdots \rightleftarrows D_n \rightleftarrows D_{n-1} \rightleftarrows \cdots \rightleftarrows D_3 \rightleftarrows D_2 \rightleftarrows D_1 \rightleftarrows D_0$$

von Projektions-Einbettungspaaren:

(1) Wähle einen beliebigen Bereich D_0, und definiere induktiv $D_n = [D_{n-1}, D_{n-1}]$ für $n > 0$.

(2) Definiere ein Projektions-Einbettungspaar

$$D_1 \underset{e_{1,0}}{\overset{p_{0,1}}{\rightleftarrows}} D_0$$

durch $p_{0,1}(f) = f(\perp)$ für alle $f \in D_1 = [D_0 \to D_0]$ und $e_{1,0}(d) = \mathrm{const}_d$, der konstanten Funktion $D_0 \to D_0$ mit Wert $d \in D_0$. Definiere dann induktiv Projektions-Einbettungspaare

$$D_{n+1} \underset{e_{n+1,n}}{\overset{p_{n,n+1}}{\rightleftarrows}} D_n$$

für $n > 0$. Unter der Induktionsannahme, daß das Projektions-Einbettungspaar

$$D_n \underset{e_{n,n-1}}{\overset{p_{n-1,n}}{\rightleftarrows}} D_{n-1}$$

schon definiert ist, setze

$$p_{n,n+1}(g) = p_{n-1,n} \circ g \circ e_{n,n-1} \quad \text{für alle } g \in D_{n+1} = [D_n \to D_n]$$
$$e_{n+1,n}(f) = e_{n,n-1} \circ f \circ p_{n-1,n} \quad \text{für alle } f \in D_n = [D_{n-1} \to D_{n-1}]$$

was als Diagramm wie folgt aussieht:

$$\begin{tikzcd} D_n \arrow[r, shift left, "p_{n-1,n}"] \arrow[d, "g"'] & D_{n-1} \arrow[l, shift left, dotted, "e_{n,n-1}"] \arrow[d, "f"] \\ D_n \arrow[r, shift left, "p_{n-1,n}"] & D_{n-1} \arrow[l, shift left, dotted, "e_{n,n-1}"] \end{tikzcd}$$

Der Nachweis, daß

$$\begin{tikzcd} D_{n+1} \arrow[r, shift left, "p_{n,n+1}"] & D_n \arrow[l, shift left, dotted, "e_{n+1,n}"] \end{tikzcd}$$

wirklich Projektions-Einbettungspaare sind, wird in den Übungen durchgeführt.

(3) Sei nun D_∞ der inverse Limes des in 1 und 2 definierten Systems von Projektions-Einbettungspaaren. Es seien

$$\begin{tikzcd} D_\infty \arrow[r, shift left, "p_n"] & D_n \arrow[l, shift left, dotted, "e_n"] \end{tikzcd}$$

die kanonischen Projektions-Einbettungspaare wie in 3.2.

Im folgenden konstruieren wir nun zwei Abbildung Ψ und Φ, die sich nachher als zueinander inverse Bereichsisomorphismen zwischen den Bereichen D_∞ und $[D_\infty \to D_\infty]$ herausstellen:

3.6 Konstruktion von Ψ und Φ. Für eine beliebige stetige Abbildung $f\colon D_\infty \to D_\infty$ sei $y_{n+1} := p_n \circ f \circ e_n$ für alle $n \in \mathbb{N}$. Für $n > 0$ gilt $p_{n,n+1}(y_{n+1}) = y_n$:

$$\begin{aligned} p_{n,n+1}(y_{n+1}) &= p_{n-1,n} \circ y_{n+1} \circ e_{n,n-1} \\ &= p_{n-1,n} \circ p_n \circ f \circ e_n \circ e_{n,n-1} \\ &= p_{n-1} \circ f \circ e_{n-1} \\ &= y_n \end{aligned}$$

Es gilt also $y := (y_k)_{k\in\mathbb{N}} \in D_\infty$, wobei wir $y_0 := p_{0,1}(y_1)$ setzen, da oben y_n nur für $n > 0$ definiert war. Wir setzen nun

$$\Psi(f) := y$$

d.h. $p_{n+1} \circ \Psi(f) = y_{n+1} = p_n \circ f \circ e_n$ oder auch $\Psi(f) = \bigsqcup_{n\in\mathbb{N}} e_{n+1}(p_n \circ f \circ e_n)$. Man bemerke, daß Ψ stetig ist, denn die Komposition, e_{n+1} und die gerichteten Suprema stetiger Funktionen sind stetig.

Für $y = (y_n)_{n\in\mathbb{N}}$ setzen wir

$$\Phi(y) := \bigsqcup_{n\in\mathbb{N}} (e_n \circ y_{n+1} \circ p_n)$$

Aus den selben Gründen wie vorher Ψ ist auch Φ stetig. Wir müssen aber zunächst sehen, daß Φ wohldefiniert ist:

Es gilt $y_{n+1} \in D_{n+1} = [D_n \to D_n]$, d.h. y_{n+1} ist eine stetige Abbildung von D_n nach D_n. Die Komposition

$$D_\infty \xrightarrow{p_n} D_n \xrightarrow{y_{n+1}} D_n \xrightarrow{e_n} D_\infty$$

liefert eine stetige Abbildung $e_n \circ y_{n+1} \circ p_n\colon D_\infty \to D_\infty$. Die Folge dieser Abbildungen ist aufsteigend, denn $y_n = p_{n,n+1}(y_{n+1})$, da $y \in D_\infty$ und folglich $e_{n+1,n}(y_n) =$

$e_{n+1,n}\big(p_{n,n+1}(y_{n+1})\big) \sqsubseteq y_{n+1}$, da $e_{\_} \circ p_{\_} \sqsubseteq \mathrm{id}$. Aus $e_{n-1} = e_n \circ e_{n,n-1}$ und $p_{n-1} = p_{n-1,n} \circ p_n$ schließen wir

$$\begin{aligned} e_{n-1} \circ y_n \circ p_{n-1} &= e_n \circ e_{n,n-1} \circ y_n \circ p_{n-1,n} \circ p_n \\ &= e_n \circ e_{n+1,n}(y_n) \circ p_n \\ &\sqsubseteq e_n \circ y_{n+1} \circ p_n \end{aligned}$$

Also ist $\Psi(y)$ eine wohldefinierte stetige Abbildung von D_∞ nach D_∞.

3.7 Satz. Die in 3.6 definierten Abbildungen

$$D_\infty \underset{\Psi}{\overset{\Phi}{\rightleftarrows}} [D_\infty \to D_\infty]$$

sind zueinander inverse Bereichsisomorphismen. Insbesondere haben wir damit ein extensionales Modell für den untypisierten λ-Kalkül gefunden,

Beweis. • Es gilt $\Psi(\Phi(y)) = y$ für alle $y \in D_\infty$:

Sei $y' = \Psi(\Phi(y))$. Für die $(k+1)$-te Komponente von y' gilt nach Definition von Ψ und Φ:

$$\begin{aligned} y'_{k+1} &= p_k \circ \Phi(y) \circ e_k \\ &= p_k \circ \bigsqcup_{n\in\mathbb{N}} (e_n \circ y_{n+1} \circ p_n) \circ e_k \\ &= \bigsqcup_{n\in\mathbb{N}} (p_k \circ e_n \circ y_{n+1} \circ p_n \circ e_k) \end{aligned}$$

Für $n > k$ ist nun $p_k = p_{k,n} \circ p_n$ und $e_k = e_n \circ e_{n,k}$, also

$$\begin{aligned} &= \bigsqcup_{n>k} (p_{k,n} \circ p_n \circ e_n \circ y_{n+1} \circ p_n \circ e_n \circ e_{n,k}) \\ &= \bigsqcup_{n>k} (p_{k,n} \circ y_{n+1} \circ e_{n,k}) \\ &= y_{k+1} \end{aligned}$$

• Es gilt $\Phi(\Psi(f)) = f$ für alle $f \in [D_\infty \to D_\infty]$:

Sei $y = \Psi(f)$, d.h. $y_{n+1} = p_n \circ f \circ e_n$. Dann gilt:

$$\begin{aligned} \Phi(\Psi(f)) &= \bigsqcup_{n\in\mathbb{N}} (e_n \circ y_{n+1} \circ p_n) \\ &= \bigsqcup_{n\in\mathbb{N}} \big(e_n \circ (p_n \circ f \circ e_n) \circ p_n\big) \\ &= \bigsqcup_{n\in\mathbb{N}} (e_n \circ p_n) \circ f \circ (e_n \circ p_n) \\ &= \bigsqcup_{n\in\mathbb{N}} \bigsqcup_{m\in\mathbb{N}} (e_n \circ p_n) \circ f \circ (e_m \circ p_m) \\ &= \bigsqcup_{n\in\mathbb{N}} (e_n \circ p_n) \circ f \circ \bigsqcup_{m\in\mathbb{N}} (e_m \circ p_m) \end{aligned}$$

$$= f$$

Die letzte Gleichheit folgt wegen $\bigsqcup_{n\in\mathbb{N}} e_n \circ p_n = \mathrm{id}_{D_\infty}$ aus Hilfssatz 3.3. □

4. Bereichsgleichungen

In Abschnitt 3 hatten wir einen Bereich D konstruiert, der die „Gleichung“

$$D \cong [D \to D]$$

erfüllt. In diesem Abschnitt wollen wir diese Idee allgemeiner betrachten.

4.1 Die Kategorie $\mathcal{B}$. Die Kategorie $\mathcal{B}$ der Bereiche besteht aus

- der Klasse aller Bereiche D (Objekte der Kategorie)
- der Klasse aller stetigen Abbildungen $f\colon D \to E$ zwischen den Bereichen (Morphismen)

4.2 Funktoren. Ein *Funktor* $F\colon \mathcal{B} \to \mathcal{B}$ der Kategorie $\mathcal{B}$ der Bereiche in sich wird gegeben durch

- eine Vorschrift, nach der jedem Bereich D ein Bereich $F(D)$ zugeordnet wird
- eine Vorschrift, nach der jeder stetigen Abbildung $f\colon D \to E$ eine stetige Abbildung $F(f)\colon F(D) \to F(E)$ zugeordnet wird, so daß
  (1) $F(\mathrm{id}_D) = \mathrm{id}_{F(D)}$ für jeden Bereich D
  (2) $F(g \circ f) = F(g) \circ F(f)$ für beliebige stetige Abbildungen $f\colon D \to E$ und $g\colon E \to H$

  gelten.

Insbesondere induziert ein Funktor $F\colon \mathcal{B} \to \mathcal{B}$ für je zwei Bereiche D und E eine Abbildung

$$\begin{array}{rcl} [D \to E] & \to & [F(D) \to F(E)] \\ f & \mapsto & F(f) \end{array}$$

Wir sagen, der Funktor F sei *lokal stetig*, falls diese Abbildung zwischen den Funktionenräumen stetig ist.

4.3 Hilfssatz. Ist

$$D \underset{e}{\overset{p}{\rightleftarrows}} E$$

ein Projektions-Einbettungspaar, und ist $F\colon \mathcal{B} \to \mathcal{B}$ ein lokal stetiger Funktor, dann ist

$$F(D) \underset{F(e)}{\overset{F(p)}{\rightleftarrows}} F(E)E$$

ein Projektions-Einbettungspaar.

Beweis. Es gilt

$$F(p) \circ F(e) = F(p \circ e) = F(\mathrm{id}_E) = \mathrm{id}_{F(E)}$$

da $p \circ e = \mathrm{id}_E$. Weiterhin ist

$$F(e) \circ F(p) = F(e \circ p) \sqsubseteq F(\mathrm{id}_D) = \mathrm{id}_{F(D)}$$

da $e \circ p \sqsubseteq \mathrm{id}_D$ und F lokal stetig ist. $\square$

4.4 Hilfssatz. Es sei $(D_n, p_{n,n+1}, e_{n+1,n})_{n\in\mathbb{N}}$ ein System von Projektions-Einbettungspaaren. Wenn der Funktor $F\colon \mathcal{B} \to \mathcal{B}$ lokal stetig ist, dann gilt

$$F\big(\varprojlim(D_n, p_{n,n+1}, e_{n+1,n})\big) \cong \varprojlim\big(F(D_n), F(p_{n,n+1}), F(e_{n+1,n})\big)$$

Man sagt dann auch, der Funktor F sei stetig.

Beweis. Es sei $D := \varprojlim(D_n, p_{n,n+1}, e_{n+1,n})$. Wir haben dann das Diagramm

$$\begin{tikzcd}
D \arrow[drr, "p_n"] \arrow[drrr, "p_{n-1}"] \arrow[drrrrr, "p_1"] \arrow[drrrrrr, "p_0"] & & & & & & \\
& \cdots \arrow[r, shift left] & D_n \arrow[l, dotted, shift left] \arrow[r, shift left] \arrow[ull, dotted, bend left, "e_n"] & D_{n-1} \arrow[l, dotted, shift left] \arrow[r, shift left] \arrow[ulll, dotted, bend left, "e_{n-1}"] & \cdots \arrow[l, dotted, shift left] \arrow[r, shift left] & D_1 \arrow[l, dotted, shift left] \arrow[r, shift left] \arrow[ulllll, dotted, bend left, "e_1"] & D_0 \arrow[l, dotted, shift left] \arrow[ullllll, dotted, bend left, "e_0"]
\end{tikzcd}$$

und es gelten

(1) $D_{n+1} \underset{e_{n+1,n}}{\overset{p_{n,n+1}}{\rightleftarrows}} D_n$ und $D \underset{e_n}{\overset{p_n}{\rightleftarrows}} D_n$ sind Projektions-Einbettungspaare

(2) $$p_{n,n+1} \circ p_{n+1} = p_n \qquad e_{n+1} \circ e_{n+1,n} = e_n$$

(3) $$\bigsqcup_{n\in\mathbb{N}} e_n \circ p_n = \mathrm{id}_D \quad \text{nach Hilfssatz 3.3}$$

Nun wenden wir den Funktor F an und erhalten die obige Hälfte des nachstehenden Diagramms.

$$\begin{tikzcd}
F(D) \arrow[drr, "F(p_n)"] \arrow[drrr, "F(p_{n-1})"] \arrow[drrrrr, "F(p_1)"] \arrow[drrrrrr, "F(p_0)"] \arrow[dd, dashed, shift left, "p"] & & & & & & \\
& \cdots \arrow[r, shift left] & F(D_n) \arrow[l, dotted, shift left] \arrow[r, shift left] \arrow[ull, dotted, bend left, "F(e_n)"] & F(D_{n-1}) \arrow[l, dotted, shift left] \arrow[r, shift left] \arrow[ulll, dotted, bend left, "F(e_{n-1})"] & \cdots \arrow[l, dotted, shift left] \arrow[r, shift left] & F(D_1) \arrow[l, dotted, shift left] \arrow[r, shift left] \arrow[ulllll, dotted, bend left, "F(e_1)"] & F(D_0) \arrow[l, dotted, shift left] \arrow[ullllll, dotted, bend left, "F(e_0)"] \\
D' \arrow[uu, dashed, shift left, "e"] \arrow[urr, "F(p'_n)"] \arrow[urrr, "F(p'_{n-1})"] \arrow[urrrrr, "F(p'_1)"] \arrow[urrrrrr, "F(p'_0)"] & & & & & &
\end{tikzcd}$$

Nach 4.3 gilt

(1′)

$$F(D_{n+1}) \underset{F(e_{n+1,n})}{\overset{F(p_{n,n+1})}{\rightleftarrows}} F(D_n) \quad \text{und} \quad F(D) \underset{F(e_n)}{\overset{F(p_n)}{\rightleftarrows}} F(D_n) \quad \text{sind Projektions-Einbettungspaare}$$

(2′) $$F(p_{n+1}) \circ F(p_{n,n+1}) = F(p_n) \qquad F(e_{n+1}) \circ F(e_{n+1,n}) = F(e_n)$$

(3′) $$\bigsqcup_{n\in\mathbb{N}} F(e_n) \circ F(p_n) = \bigsqcup_{n\in\mathbb{N}} F(e_n \circ p_n) = F\Big(\bigsqcup_{n\in\mathbb{N}} (e_n \circ p_n)\Big) = F(\mathrm{id}_D) = \mathrm{id}_{F(D)}$$

da F lokal stetig ist.

Wegen $(1')$ können wir den inversen Limes

$$D' = \varprojlim \big(F(D_n), F(p_{n,n+1}), F(e_{n+1,n})\big)$$

mit den zugehörigen Projektions-Einbettungspaaren

$$D' \underset{e'_n}{\overset{p'_n}{\rightleftarrows}} F(D_n)$$

bilden, und es gilt $\bigsqcup e'_n \circ p'_n = \mathrm{id}_{D'}$ nach 3.3. Nun können wir Abbildungen $p\colon F(D) \to D'$ und $e\colon D' \to F(D)$ definieren durch

$$p := \bigsqcup_{n\in\mathbb{N}} \big(e'_n \circ F(p_n)\big) \qquad\qquad e := \bigsqcup_{n\in\mathbb{N}} \big(F(e_n) \circ p'_n\big)$$

Dazu muß man zuerst

$(4')$ $$e'_n \circ F(p_n) \sqsubseteq e'_{n+1} \circ F(p_{n+1})$$

$(5')$ $$F(e_n) \circ p_n \sqsubseteq F(e_{n+1}) \circ p'_{n+1}$$

nachweisen. Wir zeigen nur $(4')$, der Beweis für $(5')$ verläuft ähnlich.

$$\begin{aligned}
e'_n \circ F(p_n) &= e'_{n+1} \circ F(e_{n+1,n}) \circ F(p_{n,n+1}) \circ F(p_{n+1})\\
&= e'_{n+1} \circ F(e_{n+1,n} \circ p_{n,n+1}) \circ F(p_{n+1})\\
&\sqsubseteq e'_{n+1} \circ F(\mathrm{id}_{D_{n+1}}) \circ F(p_{n+1})\\
&= e'_{n+1} \circ \mathrm{id}_{F(D_{n+1})} \circ F(p_{n+1})\\
&= e'_{n+1} \circ F(p_{n+1})
\end{aligned}$$

Nun gilt

$$\begin{aligned}
e \circ p &= \Big(\bigsqcup_{n\in\mathbb{N}} F(e_n) \circ p'_n\Big) \circ \Big(\bigsqcup_{n\in\mathbb{N}} e'_n \circ F(p_n)\Big)\\
&= \bigsqcup_{n\in\mathbb{N}} \big(F(e_n) \circ p'_n \circ e'_n \circ F(p_n)\big) && \text{wegen der Stetigkeit der Komposition}\\
&= \bigsqcup_{n\in\mathbb{N}} F(e_n) \circ \mathrm{id}_{F(D_n)} \circ F(p_n)\\
&= \bigsqcup_{n\in\mathbb{N}} F(e_n) \circ F(p_n)\\
&= \bigsqcup_{n\in\mathbb{N}} F(e_n \circ p_n)\\
&= F\Big(\bigsqcup_{n\in\mathbb{N}} e_n \circ p_n\Big) && \text{da } F \text{ lokal stetig}\\
&= F(\mathrm{id}_D)\\
&= \mathrm{id}_{F(D)}
\end{aligned}$$

Ähnlich zeigt man $p \circ e = \mathrm{id}_D$. Also sind p und e zueinander inverse Bijektionen. Aus ihrer Definition folgt sofort die Stetigkeit von p und e. Also sind $p\colon F(D) \to D'$ und $e\colon D' \to F(D)$ zueinander inverse Bereichsisomorphismen. □

4.5. Seien nun ein lokal stetiger Funktor $F\colon \mathcal{B} \to \mathcal{B}$, ein Bereich D_0 und ein Projektions-Einbettungspaar

$$F(D_0) \underset{e}{\overset{p}{\rightleftarrows}} D_0$$

gegeben. Wir definieren $D_1 := F(D_0), \ldots, D_{n+1} := F(D_n)$ und Projektions-Einbettungspaare

$$D_{n+1} \underset{e_{n+1,n}}{\overset{p_{n,n+1}}{\rightleftarrows}} D_n$$

durch

$$p_{n,n+1} := \begin{cases} p & \text{für } n = 0 \\ F(p_{n-1,n}) & \text{für } n \geq 0 \end{cases} \qquad e_{n+1,n} := \begin{cases} e & \text{für } n = 0 \\ F(e_{n,n-1}) & \text{für } n \geq 0 \end{cases}$$

und betrachten die Folge

$$\cdots \rightleftarrows F^n(D_0) \rightleftarrows \cdots \underset{F^2(e)}{\overset{F^2(p)}{\rightleftarrows}} F^2(D_0) \underset{F(e)}{\overset{F(p)}{\rightleftarrows}} F(D_0) \underset{e}{\overset{p}{\rightleftarrows}} D_0$$

Es sei $D_\infty = \varprojlim \big(F^n(D_0), F^n(p), F^n(e)\big)_{n\in\mathbb{N}}$. Dann gilt

$$\begin{aligned} F(D_\infty) &= F\big(\varprojlim \big(F^n(D_0), F^n(p), F^n(e)\big)_{n\in\mathbb{N}}\big) \\ &\cong \varprojlim \big(F^{n+1}(D_0), F^{n+1}(p), F^{n+1}(e)\big)_{n\in\mathbb{N}} && \text{nach 4.4} \\ &\cong D_\infty \end{aligned}$$

Unter den oben angegebenen Voraussetzungen haben wir also eine Lösung der „Bereichs-gleichung“ $D \cong F(D)$ gefunden.

Als Startbereich D_0 kann man z.B. den trivialen Bereich $\{\perp\}$ wählen. Es gibt genau ein Projektions-Einbettungspaar

$$F(D_0) \underset{e}{\overset{p}{\rightleftarrows}} D_0$$

nämlich $p(x) = \perp$ für alle x und $e(\perp) = \perp$. Der Bereich

$$D = \varprojlim \big(F^n(\{\perp\}), F^n(p), F^n(e)\big)_{n\in\mathbb{N}}$$

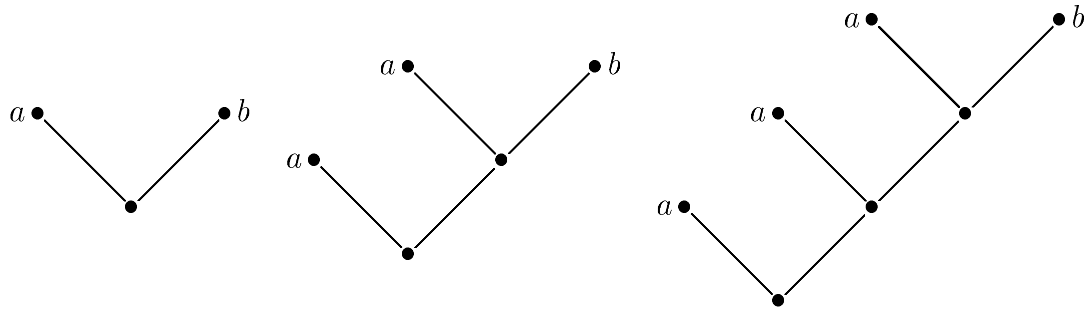
ist eine Lösung der Gleichung $D \cong F(D)$ und zwar die kleinste solche. Dies ist eine starke Analogie zu der Aussage, daß jede stetige Funktion $f\colon D \to D$ einen kleinsten Fixpunkt hat: $x_\infty = \bigsqcup_{n\in\mathbb{N}} f^n(\perp)$.

Übung. Zeigen Sie: Wenn $E \cong F(E)$, dann gibt es ein Projektions-Einbettungspaar

$$E \underset{e}{\overset{p}{\rightleftarrows}} D$$

wobei D der Bereich aus dem vorhergehenden Abschnitt ist. (In diesem Sinne ist die Aussage, D ist die kleinste Lösung für $D \cong F(D)$, zu verstehen.)

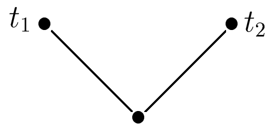
4.6 Bäume. Wir betrachten endliche binäre Bäume, deren Blätter mit Elementen a einer Menge A beschriftet sind, z.B.



Man kann die Menge $\mathcal{T}$ all dieser Bäume rekursiv wie folgt definieren:

(1) $a\bullet$ ist ein Baum für jedes $a \in A$.

(2) Sind t_1 und t_2 Bäume, dann ist auch



ein Baum.

Eine formal gleichwertige Definition ist

(1) $a \in \mathcal{T}$ für alle $a \in A$

(2) $t_1, t_2 \in \mathcal{T} \implies (t_1, t_2) \in \mathcal{T}$

Wir können $\mathcal{T}$ schrittweise so aufbauen:

$$\begin{gathered}
T_0 := A \\
T_1 := A \cup (T_0 \times T_0) \\
T_2 := A \cup (T_1 \times T_1) \\
\vdots \\
T_{n+1} := A \cup (T_n \times T_n) \\
\vdots \\
\mathcal{T} = \bigcup_{n \in \mathbb{N}} T_n
\end{gathered}$$

Man bemerke $T_0 \subseteq T_1 \subseteq T_2 \subseteq \dots$.

Setzt man $F(M) := A \cup (M \times M)$ für eine beliebige Menge M, so gilt

$$\begin{gathered}
T_0 = F(\varnothing) \\
T_1 = F(T_0) = F^2(\varnothing) \\
T_2 = F(T_1) = F^3(\varnothing) \\
\vdots \\
T_{n+1} = F(T_n) = F^{n+2}(\varnothing)
\end{gathered}$$

$$\vdots$$

$$\mathcal{T} = \bigcup_{n\in\mathbb{N}} F^n(\varnothing)$$

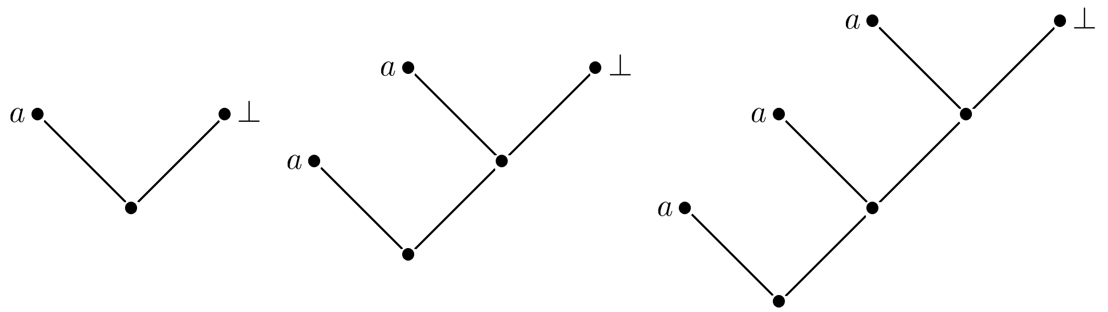
Es folgt

$$F(\mathcal{T}) = F\big(\bigcup_{n\in\mathbb{N}} F^n(\varnothing)\big) = \bigcup_{n\in\mathbb{N}} F^{n+1}(\varnothing) = \mathcal{T}$$

d.h. $\mathcal{T}$ ist eine Lösung der Gleichung $M = F(M) = A \cup (M \times M)$ in der Kategorie der Mengen und zwar wieder die kleinste Lösung.

4.7 Unendliche binäre Bäume. Nun wollen wir auch unendliche binäre Bäume zulassen, deren Blätter mit Elementen aus A bezeichnet sind. Wie kann man die Menge dieser Bäume sauber einführen?

Zu diesem Zweck führen wir *unvollständige endliche Bäume* ein:



Dabei deuten wir durch das Zeichen $\perp$ an, daß der Baum an dieser Stelle noch vervollständigt werden muß. Es ist naheliegend, eine Ordnung $s \sqsubseteq t$ einzuführen, die ausdrückt, daß der Baum t „vollständiger" ist als der Baum s. Dann kann man Baumketten $t_1 \sqsubseteq t_2 \sqsubseteq t_3 \sqsubseteq \ldots$ und deren Suprema $t = \bigsqcup_{n\in\mathbb{N}} t_n$ betrachten. Diese Ideen wollen wir nun sauber ausführen:

Es seien D und E Bereiche. Mit $D + E$ wird der Bereich bezeichnet, der entsteht, wenn man D und E in ihrem kleinsten Element zusammenklebt, die Ordnung in D und E aber unverändert beibehält. Sei weiterhin $A_\perp$ der flache Bereich $A \cup \{\perp\}$.

Für jeden Bereich D definieren wir $F(D) := A_\perp + (D \times D)$ und für jede stetige Funktion $f\colon D \to E$ sei $F(f) := \mathrm{id}_{A_\perp} + (f \times f)$. Man prüfe als Übung nach, daß F ein lokal stetiger Funktor der Kategorie der Bereiche in sich ist.

Geht man nun wie in 4.6 vor und setzt

$$D_0 := \{\perp\}$$
$$D_1 := F(\{\perp\}) = A_\perp$$
$$D_2 := F(D_1) = A_\perp + (A_\perp \times A_\perp)$$
$$\vdots$$
$$D_{n+1} := F(D_n) = A_\perp + (D_n \times D_n)$$
$$\vdots$$
$$D := \bigcup_{n\in\mathbb{N}} D_n$$

so erhält man die Menge aller endlichen (vollständigen und unvollständigen) Bäume, geordnet durch $\sqsubseteq$. Aber D ist kein Bereich, denn die aufsteigende Baumfolge, deren Anfangsglieder zu Beginn dieses Abschnittes graphisch dargestellt wurden, hat kein Supremum in D.

Gemäß 4.5 bilden wir nun: $D_0 = \{\perp\}$ und betrachten das eindeutig bestimmte Projektions-Einbettungspaar

$$F(D_0) \underset{e}{\overset{p}{\rightleftarrows}} D_0$$

und setzen

$$D_\infty = \varprojlim \big(F^n(D_0), F^n(p), F^n(e)\big)_{n\in\mathbb{N}}$$

Dann gilt $D_\infty \cong F(D_\infty)$, d.h. $D_\infty \cong A_\perp + D_\infty \times D_\infty$, und D_∞ ist die kleinste Lösung dieser Gleichung.

D_∞ ist der Bereich der endlichen und unendlichen, vollständigen und unvollständigen Bäume mit Blättern aus A. Dazu gehören insbesondere alle rekursiv definierbaren binäre Bäume.

4.8 Ein Dämpfer. Es sieht so aus, als ob die in 4.5 angegebene Methode auch den Spezialfall aus Abschnitt 3 erledigen würde, wo wir eine Lösung der Bereichsgleichung $D \cong [D \to D]$ angegeben hatten. Leider ist dies nicht so:

Definiert man $F(D) := [D \to D]$ für jeden Bereich D, dann muß man F noch zu einem Funktor machen, d.h. man müßte $F(f)\colon [D \to D] \to [E \to E]$ für jede stetige Abbildung $f\colon D \to E$ definieren, so daß F die Eigenschaften eines Funktors erfüllt. Dies ist jedoch nicht möglich. Den Grund hierfür wollen wir aufzeigen.

Dazu betrachten wir für zwei beliebige Bereiche D und D' den Bereich $[D \to D']$ aller stetigen Funktionen $\varphi\colon D \to D'$, der wie üblich mit der punktweisen Ordnung versehen ist.

- Für festes D können wir nun jedem Bereich D' den Funktionenraum $F'(D') := [D \to D']$ zuordnen. Für jede stetige Abbildung $f'\colon D' \to E'$ definieren wir

  $$F'(f')\colon [D \to D'] \to [D \to E']$$

  durch $F'(f)(\varphi) := f' \circ \varphi$. Da $\circ$ stetig ist, ist $F'(f')$ auch stetig, und es gilt

  (1) $F'(\mathrm{id}_{D'})(\varphi) = \mathrm{id}_{D'} \circ \varphi = \varphi$ für alle $\varphi \in [D \to D']$, d.h. $F'(\mathrm{id}_{D'}) = \mathrm{id}_{F(D')}$.

  (2) $F'(g' \circ f')(\varphi) = (g' \circ f') \circ \varphi = g' \circ (f' \circ \varphi) = g' \circ F'(f)(\varphi) = F'(g')(F'(f')(\varphi)) = (F'(g') \circ F'(f'))(\varphi)$ für alle $\varphi \in [D \to D']$, d.h. $F'(g' \circ f') = F'(g') \circ F'(f')$.

  Also ist $F'\colon \mathcal{B} \to \mathcal{B}$ ein Funktor, der auch lokal stetig ist.

- Analog halten wir nun D' fest und ordnen jedem Bereich den Funktionenraum $F(D) := [D \to D']$ zu. In analoger Weise ordnen wir jeder stetigen Abbildung $f\colon D \to E$ die Abbildung

  $$F(f)\colon [E \to D'] \to [D \to D']$$

  zu, die durch $F(f)(\psi) := \psi \circ f$ definiert wird. Wir bemerken, daß F die Richtung der Morphismen umdreht: $f\colon D \to E$, aber $F(f)\colon F(E) \to F(D)$. Ferner gilt

  (1) $F(\mathrm{id}_D) = \mathrm{id}_{F(D)}$ wie vorhin, aber

(2) $F(g \circ f) = F(f) \circ F(g)$, denn für alle $\psi\colon C \to D'$ gilt $F(g \circ f)(\psi) = \psi \circ (g \circ f) = (\psi \circ g) \circ f = (F(g)(\psi)) \circ f = F(f)(F(g)(\psi)) = (F(f) \circ F(g))(\psi)$.

Im Gegensatz zu den bisher betrachteten (kovarianten) Funktoren nennt man solche Funktoren *kontravariant*. Genauer:

4.9 Definition. Ordnet man jedem Bereich D einen Bereich $F(D)$ und jeder stetigen Abbildung $f\colon D \to E$ eine stetige Abbildung $F(f)\colon F(E) \to F(D)$ zu, so daß

(1) $F(\mathrm{id}_D) = \mathrm{id}_{F(D)}$

(2) $F(g \circ f) = F(f) \circ F(g)$

gelten, so nennt man F einen *kontravarianten Funktor* der Kategorie der Bereiche in sich.

4.10 Übung. (1) Definieren Sie den Begriff *lokal stetig* für kontravariante Funktoren.

(2) Gelten (nach geeigneter Modifikation) 4.3 und 4.4 auch für lokal stetige kontravariante Funktoren?

(3) Zeigen Sie, daß wie in 4.5 auch für lokal stetige kontravariante Funktoren Bereichsgleichungen der Form $D \cong F(D)$ gelöst werden können.

4.11 Zurück zu unserem Problem. Das Problem für den Funktionenraumkonstruktor $[D \to D']$ ist, daß er kontravariant im ersten und kovariant im zweiten Argument ist. Auch hier gibt es einen Ausweg: Man betrachtet Funktoren $F(D, D')$ mit zwei Argumenten, die kontravariant im ersten und kovariant im zweiten Argument sind, und sucht nach Lösungen der Gleichung $D \cong F(D, D)$. Dies läßt sich in der Tat erfolgreich durchführen. Die Bereichsgleichung $D \cong [D \to D]$ wird dann ein Spezialfall hiervon. Der allgemeine Fall, der erst vor wenigen Jahren von Freyd erfolgreich durchgeführt werden konnte, soll jedoch hier nicht ausgeführt werden.

4.12 Beispiel eines nicht lokal-stetigen kontravarianten Funktors. Sei D ein Bereich. Eine Teilmenge $I \subseteq D$ wird *Ideal* genannt, falls

(1) I gerichtet ist

(2) $x \sqsubseteq y,\ y \in I \implies x \in I$

Zeige:

(1) Die Menge $\mathcal{I}(D)$ aller Ideale von D geordnet durch $\subseteq$ ist ein Bereich.

(2) Ist $f\colon D \to E$ stetig, so ist $f^{-1}(I) \in \mathcal{I}(D)$ für alle $I \in \mathcal{I}(E)$.

(3) $\mathcal{I}(f)$ ist stetig, wobei $\mathcal{I}(f)(I) = f^{-1}(I)$ für alle $I \in \mathcal{I}(E)$.

(4) $\mathcal{I}$ ist ein kontravarianter Funktor.

(5) $\mathcal{I}$ ist nicht lokal stetig.

Index