Lambda-Kalkül und Kombinatorische Logik

Skript zur Vorlesung
von
Thomas Piecha

Dieses Skript beruht nahezu vollständig auf dem von *Michael Arndt* erstellten Skript zu der von *Peter Schroeder-Heister* im Sommersemester 1997 gehaltenen Vorlesung sowie auf meinen Mitschrieben aus ebendieser Vorlesung. Hinzugekommen sind kleinere Korrekturen, Modifikationen und Ergänzungen.

Wintersemester 2026/27

Universität Tübingen
Fachbereich Informatik

Vorwort 1997

Dies ist ein Skriptum zu einer Vorlesung, die ich zuletzt im Sommersemester 1997 gehalten habe. In den ersten beiden Teilen orientiert es sich im wesentlichen am klassischen Lehrbuch von Hindley und Seldin, in den letzten beiden Teilen an Barendregts Kapitel über den getypten λ-Kalkül im *Handbook of Logic in Computer Science* (Band II). Das Skriptum soll zur Orientierung über das technische Gerüst des Themas dienen. Dementsprechend ist es nicht bis in alle Einzelheiten ausgearbeitet. So wurde auf Stilfragen wenig Rücksicht genommen. Auch wurden elementare, aber langwierige Beweise häufig weggelassen. Erläuternde Passagen zu Sinn und Zweck des λ-Kalküls sowie einzelner Begriffsbildungen sind ebenfalls nicht aufgezeichnet. Hierzu seien Leser auf die genannten Texte verwiesen.

Ich danke Michael Arndt für die Erstellung des Skriptums. Frau Natali Alt und Herrn Reinhard Kahle danke ich für eine kritische Durchsicht des Textes. Alle verbleibenden inhaltlichen Fehler gehen natürlich zu meinen Lasten.

Peter Schroeder-Heister

Vorwort 2016

Bei dieser aktualisierten Fassung des Skripts konnte ich auch auf die zweite Auflage des Lehrbuchs von Hindley und Seldin zurückgreifen:

> J. Roger Hindley und Jonathan P. Seldin: *Lambda-Calculus and Combinators, an Introduction*, Cambridge University Press, 2008 (reprinted 2010).

Es wurde wieder darauf verzichtet, die Verwendung dieser und anderer Quellen (siehe Literaturverzeichnis) im Einzelnen immer kenntlich zu machen.

Thomas Piecha

Vorwort 2021

Es wurden kleinere Änderungen vorgenommen.

Thomas Piecha

Inhaltsverzeichnis

1 Der ungetypte λ-Kalkül **5**
- 1.1 Syntax und operationelle Semantik . . . 6
- 1.2 λ-Definierbarkeit rekursiver Funktionen . . . 24
- 1.3 Die formalen Theorien $\lambda\beta$ und $\lambda\beta\eta$ . . . 30
- 1.4 Unentscheidbarkeitsresultate . . . 33

2 Kombinatorische Logik **35**
- 2.1 Syntax und operationelle Semantik . . . 35
- 2.2 Die formale Theorie CLw . . . 38
- 2.3 Verhältnis zwischen λ-Kalkül und CL . . . 38

3 Der einfach getypte λ-Kalkül **45**
- 3.1 Implizite Typisierung . . . 45
- 3.2 Der Typisierungsalgorithmus . . . 50
- 3.3 Der Curry-Howard-Isomorphismus . . . 56

4 Der polymorph getypte λ-Kalkül **60**

Literaturverzeichnis **65**

Sachverzeichnis **67**

1 Der ungetypte λ-Kalkül

Der λ-Kalkül ist ein formales System, in dem die Bildung und Anwendung von Funktionen ausgedrückt werden kann. Im ungetypten λ-Kalkül, den wir zunächst behandeln, werden Funktionen ohne zugeordnete (Daten-)Typen betrachtet.

Motivation der Syntax

Der Ausdruck "$x - y$" kann als Definition einer Funktion f von x oder als Definition einer Funktion g von y aufgefasst werden:

$$f(x) = x - y \qquad\qquad g(y) = x - y$$

bzw.

$$f : x \mapsto x - y \qquad\qquad g : y \mapsto x - y$$

Unter Verwendung des Hilfszeichens λ kann für jeden Ausdruck, in dem eine Variable x vorkommt, systematisch ein Ausdruck für die entsprechende Funktion von x angegeben werden:

$$f = \lambda x.x - y \qquad\qquad g = \lambda y.x - y$$

Aus den Gleichungen

$$f(0) = 0 - y \qquad \text{und} \qquad f(1) = 1 - y$$

wird in λ-Notation

$$(\lambda x.x - y)(0) = 0 - y \qquad \text{bzw.} \qquad (\lambda x.x - y)(1) = 1 - y$$

Die λ-Notation ist zwar nicht einfacher als übliche Notationen, dafür aber systematisch, und somit besser geeignet als Grundlage für Programmiersprachen. Zudem können höherstufige Funktionen direkt in λ-Notation angegeben werden.
Die λ-Notation kann für Funktionen mehrerer Variablen erweitert werden. Zum Beispiel sind durch den Ausdruck "$x - y$" zwei Funktionen mit zwei Argumenten gegeben:

$$h(x, y) = x - y \qquad\qquad k(y, x) = x - y$$

Diese können in λ-Notation wie folgt angegeben werden:

$$h = \lambda xy.x - y \qquad\qquad k = \lambda yx.x - y$$

Eine solche Erweiterung der Notation für Funktionen mehrerer Variablen kann vermieden werden, indem man Funktionen zulässt, deren Werte nicht Zahlen, sondern Funktionen sind. Zum Beispiel kann man anstelle der zweistelligen Funktion h die einstellige Funktion h' verwenden, die gegeben ist durch

$$h' = \lambda x.(\lambda y.x - y)$$

Für jede Zahl n ist

$$h'(n) = \lambda y.n - y$$

und somit für jedes Paar von Zahlen n, m:

$$\begin{aligned}(h'(n))(m) &= (\lambda y.n - y)(m)\\ &= n - m\\ &= h(n, m)\end{aligned}$$

In diesem Sinne repräsentiert die einstellige Funktion h' die zweistellige Funktion h. Es genügen also letztlich einstellige Funktionen. (Diesen Übergang von mehrstelligen zu einstelligen Funktionen bezeichnet man auch als *Currying* oder *Schönfinkeln*; benannt nach H. B. Curry bzw. M. Schönfinkel.)

Zur Semantik

Wenn M als Funktion interpretiert werden kann, dann stellt (MN) das Ergebnis der Anwendung von M auf das Argument N dar, sofern das Ergebnis existiert.
Ein Term $(\lambda x.M)$ wird interpretiert als Funktion, deren Wert für ein Argument N berechnet wird, indem N für x in M substituiert wird.

Beispiel. Der Term $(\lambda x.x(xy))$ steht für die zweifache Anwendung einer Funktion auf ein Objekt y. Es gilt $(\lambda x.x(xy))N = N(Ny)$.

1.1 Syntax und operationelle Semantik

Gegeben sei eine unendliche Folge von Variablen, für die eine festgelegte Reihenfolge angenommen wird.

Die metasprachlichen Zeichen für Variablen sind: $u, v, w, x, y, z, x_1, x_2, x_3, \ldots$. (Sofern nichts anderes gesagt wird, bezeichnen verschiedene Zeichen verschiedene Variablen.)

Man unterscheidet zwischen zwei Varianten des ungetypten λ-Kalküls:

1. dem reinen λ-Kalkül, bei dem keine Konstanten gegeben sind,
2. dem angewandten λ-Kalkül, bei dem zusätzlich eine endliche oder unendliche Menge von Konstanten gegeben ist.

(In den ersten beiden Kapiteln wird nur der ungetypte λ-Kalkül behandelt. Daher wird die Bezeichnung "ungetypt" immer weggelassen.)

Definition 1.1 *λ-Terme* (kurz: *Terme*) sind wie folgt definiert: *λ-Terme*

1. Alle Variablen und Konstanten sind λ-Terme (*Atome*). *Atome*
2. Sind M und N λ-Terme, dann ist auch (MN) ein λ-Term (*Applikation*), der M und N als unmittelbare Teilterme enthält. *Applikation*
3. Ist M ein λ-Term, dann ist auch $(\lambda x.M)$ ein λ-Term (*Abstraktion*), der x und M als unmittelbare Teilterme enthält. *Abstraktion*

Bemerkung: Somit sind λ und z. B. λx *keine* λ-Terme.

Weitere Begriffe:

1. *Metasprachliche Variablen* (ggf. auch mit Indizes):
   (a) für λ-Terme: $M, N, P, Q, R, S, T, \ldots$
   (b) für Atome: $a, b, c, \ldots$
2. Die *Länge* eines Terms M ist die Anzahl der Vorkommen von Atomen in M. — *Länge*
3. *Teilterme* eines Terms sind der Term selbst sowie die Teilterme seiner unmittelbaren Teilterme. Alle Teilterme eines Terms mit Ausnahme seiner selbst sind dessen *echte Teilterme*. — *Teilterm*
4. Wir schreiben $M\left[P\right]$, falls P als Teilterm an einer bestimmten Stelle in M vorkommt.
5. Im Kontext von $M\left[P\right]$ bedeute der Ausdruck $M\left[Q\right]$, dass man das Vorkommen von P durch Q ersetzt hat.
6. Ein Vorkommen einer Variable x in einem Term M ist *gebunden*, falls es zu einem Teilterm $\lambda x.P$ von M gehört; andernfalls ist es *frei*. — *gebunden* *frei*
7. Falls x ein freies Vorkommen in M hat, dann heißt x *freie Variable* von M. Die Menge der freien Variablen eines Terms M bezeichnen wir mit $\mathrm{FV}(M)$. — *freie Variable*
8. M heißt *geschlossen*, falls $\mathrm{FV}(M) = \emptyset$, andernfalls *offen*. — *geschlossen/offen*
9. Ein geschlossener Term ohne Konstanten heißt *Kombinator*. — *Kombinator*
10. λ-Terme können gemäß folgenden Regeln zur *Klammerersparnis* abgekürzt werden, sofern dies keine Mehrdeutigkeiten zur Folge hat: — *Klammerersparnis*
    (a) Außenklammern können weggelassen werden.
    (b) Wir verwenden Linksklammerung, d. h. $MNPQ$ steht für $((MN)P)Q$.
    (c) $\lambda x.MN$ steht für $(\lambda x.(MN))$.
    (d) $\lambda x_1 x_2 \ldots x_n.M$ steht für $(\lambda x_1.(\lambda x_2.(\ldots(\lambda x_n.M))))$.
11. $M \doteqdot N$ bezeichne die *syntaktische Identität* von M und N.
    $M :\doteqdot N$ bedeutet, dass M und N per Definition syntaktisch identisch sind. M ist das Definiendum, N ist das Definiens.

Bemerkung. Die Terme des reinen λ-Kalküls können durch die folgende kontextfreie Grammatik charakterisiert werden, wobei Variablen die Form $v_{0\ldots0}$ haben: — *Grammatik für Terme*

1. Terminale: $\{\lambda,\ .,\ (,\ ),\ v_0,\ _0\}$
2. Nicht-Terminale: $\{T, V\}$
3. Startsymbol: T
4. Produktionen: $T \longrightarrow V \mid (TT) \mid (\lambda V.T)$
   $V \longrightarrow v_0 \mid V_0$

Beispiele.

1. $(\lambda v_0.(v_0 v_{00}))$ ist ein λ-Term der Länge 3.

   Unmittelbare Teilterme: v_0 und $(v_0 v_{00})$

   Der Term ist offen, da v_{00} frei vorkommt. Beide Vorkommen der Variable v_0 sind gebunden.

   Abgekürzt: $\lambda v_0.v_0 v_{00}$

2. In $\lambda xy.xy$ kommt der Term xy einmal vor. Es ist $\lambda xy.xy \doteq (\lambda x.(\lambda y.(xy)))$.

3. In $x(uv)(\lambda u.v(uv))uv$ kommt der Term uv zweimal vor.

   Es ist $x(uv)(\lambda u.v(uv))uv \doteq ((((x(uv))(\lambda u.(v(uv))))u)v)$.

4. In $\lambda u.uv$ kommt der Term $\lambda u.u$ *nicht* vor.

   Es ist $\lambda u.u \doteq (\lambda u.u)$ und $\lambda u.uv \doteq (\lambda u.(uv))$.

Beispiele. Es seien x, y, z verschiedene Variablen. Dann sind λ-Terme:

1. $(\lambda x.(xy))$ (vgl. letztes obiges Beispiel)
2. $((\lambda y.y)(\lambda x.(xy)))$
3. $(x(\lambda x.(\lambda x.x)))$

   Dieser Term hat die Form (MN), wobei N zwei Vorkommen von λx enthält. (Dies ist erlaubt, aber nicht empfehlenswert. Beim getypten λ-Kalkül werden wir Terme dieser Form allerdings von der Betrachtung ausschließen.)

4. $(\lambda x.(yz))$

   Dieser Term hat die Form $(\lambda x.M)$, wobei x nicht in M vorkommt. Dies bezeichnet man als *leere Abstraktion*. Solche Terme stehen für konstante Funktionen (d. h. für Funktionen, die für jede Eingabe dieselbe Ausgabe haben). *leere Abstraktion*

Beispiele. Zur Klammerersparnis:

1. $xyz(yx) \doteq (((xy)z)(yx))$
2. $\lambda x.uxy \doteq (\lambda x.((ux)y))$
3. $\lambda u.u(\lambda x.y) \doteq (\lambda u.(u(\lambda x.y)))$
4. $(\lambda u.vuu)zy \doteq (((\lambda u.((vu)u))z)y)$
5. $ux(yz)(\lambda v.vy) \doteq (((ux)(yz))(\lambda v.(vy)))$
6. $(\lambda xyz.xz(yz))uvw \doteq ((((\lambda x.(\lambda y.(\lambda z.((xz)(yz)))))u)v)w)$

Beispiele. Die folgenden geschlossenen λ-Terme (Kombinatoren) erhalten einen Namen:

1. $\mathbf{I} :\doteq \lambda x.x$
2. $\mathbf{K} :\doteq \lambda xy.x$
3. $\mathbf{S} :\doteq \lambda xyz.xz(yz)$

Definition 1.2 Für beliebige λ-Terme M, N, x definieren wir die *Substitution* $M[N/x]$ per Induktion bezüglich M als Ergebnis der Ersetzung jedes freien Vorkommens von x in M durch N, wobei Variablenkonflikte durch Auswechslung gebundener Variablen vermieden werden: *Substitution*

1. $x[N/x] :\doteq N$,
2. $a[N/x] :\doteq a$, falls $x \neq a$,
3. $(PQ)[N/x] :\doteq (P[N/x]Q[N/x])$,
4. $(\lambda x.P)[N/x] :\doteq \lambda x.P$,
5. $(\lambda y.P)[N/x] :\doteq \lambda y.P[N/x]$, falls $x \neq y$ und nicht: $y \in \mathrm{FV}(N)$ und $x \in \mathrm{FV}(P)$,
6. $(\lambda y.P)[N/x] :\doteq \lambda z.P[z/y][N/x]$, falls $x \neq y$ und $y \in \mathrm{FV}(N)$ und $x \in \mathrm{FV}(P)$, wobei z die erste Variable (in der Aufzählung aller Variablen) mit $z \notin \mathrm{FV}(NP)$ ist.

Beispiele.

1. $(\lambda x.zy)[(uv)/x] \doteq \lambda x.zy$
2. $(\lambda y.x)[y/x] \doteq \lambda z.y$ (sofern z die erste von x und y verschiedene Variable ist)
3. $(\lambda y.x(\lambda x.x))[(\lambda y.xy)/x] \doteq \lambda y.(\lambda y.xy)(\lambda x.x)$, da

$$\begin{aligned}
(\lambda y.x(\lambda x.x))[(\lambda y.xy)/x] &\doteq (\lambda y.(x(\lambda x.x)))[(\lambda y.xy)/x] && \text{(Klammern)}\\
&\doteq (\lambda y.(x(\lambda x.x))[(\lambda y.xy)/x]) && \text{(Def. 1.2, 5)}\\
&\doteq (\lambda y.(x[(\lambda y.xy)/x](\lambda x.x)[(\lambda y.xy)/x])) && \text{(Def. 1.2, 3)}\\
&\doteq (\lambda y.((\lambda y.xy)(\lambda x.x)[(\lambda y.xy)/x])) && \text{(Def. 1.2, 1)}\\
&\doteq (\lambda y.((\lambda y.xy)(\lambda x.x))) && \text{(Def. 1.2, 4)}\\
&\doteq \lambda y.(\lambda y.xy)(\lambda x.x) && \text{(Klammern)}
\end{aligned}$$

Definition 1.3

1. Wir definieren die *gebundene Umbenennung* von Variablen wie folgt: *gebundene Umbenennung*

   Falls $y \notin \mathrm{FV}(M)$, dann sei $P\big[\lambda x.M\big] \equiv_{1\alpha} P\big[\lambda y.M[y/x]\big]$.

2. Es ist α-*Konversion* (bzw. *Kongruenz*) dann wie folgt definiert: *α-Konversion* *Kongruenz*

   Es sei $P \equiv_\alpha Q$, falls $P \doteq P_1 \equiv_{1\alpha} P_2 \equiv_{1\alpha} \cdots \equiv_{1\alpha} P_n \doteq Q$.

   (Falls $P \equiv_\alpha Q$, sagen wir "P ist kongruent zu Q" oder "P α-konvertiert zu Q".)

Beispiel. Es ist $\lambda xy.x(xy) \equiv_\alpha \lambda uv.u(uv)$:

$$\begin{aligned}
\lambda xy.x(xy) \doteq \lambda x.(\lambda y.x(xy)) &\equiv_{1\alpha} \lambda x.(\lambda v.x(xv))\\
&\equiv_{1\alpha} \lambda u.(\lambda v.u(uv)) \doteq \lambda uv.u(uv)
\end{aligned}$$

Lemma 1.4 *Für alle λ-Terme M, N und Variablen x gilt:*

1. $M[x/x] \backsimeq M$.
2. *Wenn* $x \notin \mathrm{FV}(M)$, *dann* $M[N/x] \backsimeq M$.
3. *Wenn* $x \in \mathrm{FV}(M)$, *dann* $\mathrm{FV}(M[N/x]) = \mathrm{FV}(N) \cup (\mathrm{FV}(M) \setminus \{x\})$.

Beweis. Direkte Anwendung von Definition 1.2. QED

Lemma 1.5 *Es sei keine im λ-Term M gebundene Variable frei in den λ-Termen P, Q und z. Dann gilt:*

1. *Wenn* $z \notin \mathrm{FV}(M)$, *dann* $M[z/x][P/z] \backsimeq M[P/x]$.
2. *Wenn* $z \notin \mathrm{FV}(M)$, *dann* $M[z/x][x/z] \backsimeq M$.
3. $M[Q/x][P/x] \backsimeq M[(Q[P/x])/x]$.
4. *Wenn* $y \notin \mathrm{FV}(P)$, *dann* $M[Q/y][P/x] \backsimeq M[P/x][(Q[P/x])/y]$.
5. *Wenn* $y \notin \mathrm{FV}(P)$ *und* $x \notin \mathrm{FV}(Q)$, *dann* $M[Q/y][P/x] \backsimeq M[P/x][Q/y]$.

Beweis. Die Einschränkung bezüglich gebundener Variablen in M schließt eine Anwendung von Definition 1.2 (6) aus.
Die Beweise von (1), (3) und (4) erfolgen per Induktion über dem Aufbau von M.
(2) folgt aus (1) und Lemma 1.4 (1); (5) folgt aus (4) und Lemma 1.4 (2). QED

Bemerkung. Wird in Lemma 1.5 die Beschränkung bezüglich der in M gebundenen Variablen aufgehoben, aber $\backsimeq$ durch $\equiv_\alpha$ ersetzt, so gelten die resultierenden Aussagen (1)-(5) ebenfalls.

Lemma 1.6

1. *Wenn* $P \equiv_\alpha Q$, *dann* $\mathrm{FV}(P) = \mathrm{FV}(Q)$.
2. *Für jeden Term P und Variablen $x_1, \ldots, x_n$ existiert ein Term P' mit $P \equiv_\alpha P'$, in dem keine der Variablen $x_1, \ldots, x_n$ gebunden vorkommt.*
3. *Es ist $\equiv_\alpha$ eine Äquivalenzrelation. Das heißt, es gilt:*

   Reflexivität: $P \equiv_\alpha P$.

   Symmetrie: Wenn $P \equiv_\alpha Q$, *dann* $Q \equiv_\alpha P$.

   Transitivität: Wenn $P \equiv_\alpha Q$ *und* $Q \equiv_\alpha M$, *dann* $P \equiv_\alpha M$.

Beweis. Übung. QED

Lemma 1.7 (Kongruenz von $\equiv_\alpha$)
Wenn $M \equiv_\alpha M'$ *und* $N \equiv_\alpha N'$, *dann* $M[N/x] \equiv_\alpha M'[N'/x]$.

Beweis. Siehe Hindley & Seldin (2008), Appendix A1. QED

Bemerkungen.

1. Substitutionen verhalten sich bezüglich α-Konversion unproblematisch: Das Ergebnis der Substitution mittels eines zu N kongruenten Terms N' unterscheidet sich vom Ergebnis der Substitution mit N nur durch Kongruenz.
2. Durch α-Konversion können Variablen in einem Term stets erst separiert werden. Dadurch können später kompliziertere Substitutionen vermieden werden.
3. Kongruente Terme werden dieselbe Interpretation haben.

Bemerkung. Im Folgenden schreiben wir $P \overset{\rhd_{1\beta}}{\equiv_{1\alpha}} Q$ für "$P \rhd_{1\beta} Q$ oder $P \equiv_{1\alpha} Q$", und $P \overset{\equiv_{1\alpha}}{\rhd_{1\beta}\lhd_{1\beta}} Q$ für "$P \equiv_{1\alpha} Q$ oder $P \rhd_{1\beta} Q$ oder $Q \rhd_{1\beta} P$", etc. ($P \lhd_{1\beta} Q$ steht für $Q \rhd_{1\beta} P$.)

Definition 1.8

1. Ein Term der Form $(\lambda x.M)N$ heißt *β-Redex* (kurz: *Redex*, von *reducible expression*), und der zugehörige Term $M[N/x]$ ist dessen *Kontraktum*. — *β-Redex*, *Kontraktum*
2. Die Operation der *β-Kontraktion* ist definiert durch: — *β-Kontraktion*

$$P\big[\underbrace{(\lambda x.M)N}_{\beta\text{-Redex}}\big] \rhd_{1\beta} P\big[\underbrace{M[N/x]}_{\text{Kontraktum}}\big].$$

3. Kann ein Term P durch eine (möglicherweise leere) endliche Folge von β-Kontraktionen und gebundenen Umbenennungen in einen Term Q überführt werden, d. h. gilt

$$P \eqsim P_1 \overset{\rhd_{1\beta}}{\equiv_{1\alpha}} P_2 \overset{\rhd_{1\beta}}{\equiv_{1\alpha}} \cdots \overset{\rhd_{1\beta}}{\equiv_{1\alpha}} P_n \eqsim Q$$

   so sagt man, dass P zu Q *β-reduziert*. Notation: $P \rhd_\beta Q$. — *β-Reduktion*

4. Zwei Terme P und Q heißen *β-gleich* (oder *β-konvertibel*), falls gilt: — *β-Gleichheit*, *β-Konversion*

$$P \eqsim P_1 \overset{\equiv_{1\alpha}}{\rhd_{1\beta}\lhd_{1\beta}} P_2 \overset{\equiv_{1\alpha}}{\rhd_{1\beta}\lhd_{1\beta}} \cdots \overset{\equiv_{1\alpha}}{\rhd_{1\beta}\lhd_{1\beta}} P_n \eqsim Q$$

   Notation: $P =_\beta Q$.

5. Für eine (möglicherweise leere) endliche oder unendliche Folge von β-Kontraktionen

$$P \eqsim P_1 \rhd_{1\beta} P_2 \rhd_{1\beta} P_3 \rhd_{1\beta} \cdots$$

   heißt $(P_1, P_2, P_3, \ldots)$, bzw. die angegebene Folge selbst, *β-Reduktionsfolge* von P. — *β-Reduktionsfolge*

Bemerkung. Durch die beiden Operationen α-Konversion (bzw. gebundene Umbenennung) und β-Kontraktion ist eine *operationelle Semantik* für λ-Terme gegeben. Die Interpretation von λ-Termen ist durch deren Verhalten unter diesen beiden Operationen festgelegt. — *operationelle Semantik*

Definition 1.9

1. P ist in *β-Normalform* (kurz: *β-NF*), falls P kein β-Redex enthält. *β-Normalform*
2. Wenn $P \rhd_\beta Q$ gilt, und Q in β-Normalform ist, dann heißt Q *β-Normalform von P*. (Wir sagen dann auch, dass P die β-Normalform Q *hat*.)
3. P heißt *(schwach) normalisierbar*, falls es eine β-Normalform von P gibt. *normalisierbar*
4. P heißt *stark normalisierbar*, falls es keine unendliche β-Reduktionsfolge von P gibt. *stark normalisierbar*

Bemerkung. Man beachte den Unterschied zwischen "in β-NF *sein*" und "eine β-NF *haben*". Um festzustellen, ob ein Term in β-NF *ist*, muss nur überprüft werden, ob der Term ein β-Redex enthält oder nicht. Um jedoch zu zeigen, dass ein Term eine β-NF *hat*, muss man zeigen, dass der Term zu einer β-NF reduziert (d. h. man muss zeigen, dass es für den Term eine endliche β-Reduktionsfolge gibt, die mit einem Term in β-NF endet).

Beispiele.

1. $(\lambda x.x)N \rhd_{1\beta} N$
2. $(\lambda x.y)N \rhd_{1\beta} y$
3. $(\lambda x.x(xy))N \rhd_{1\beta} N(Ny)$
4. $(\lambda x.(\lambda y.yx)z)v \rhd_{1\beta} (\lambda x.zx)v \rhd_{1\beta} zv$

   Der Term zv ist β-Normalform von $(\lambda x.(\lambda y.yx)z)v$.
5. $\mathbf{\Omega} :\doteq (\lambda x.xx)(\lambda x.xx)$ hat keine β-Normalform; $\mathbf{\Omega}$ ist nicht in β-NF, und es gibt nur eine unendliche β-Reduktionsfolge: $(\lambda x.xx)(\lambda x.xx) \rhd_{1\beta} (\lambda x.xx)(\lambda x.xx) \rhd_{1\beta} \cdots$.

   Allerdings können λ-Terme, die den $\mathbf{\Omega}$-Kombinator enthalten, eine β-Normalform haben. Zum Beispiel hat der Term $(\lambda x.y)\mathbf{\Omega}$ die β-NF y, da $(\lambda x.y)\mathbf{\Omega} \rhd_{1\beta} y$.

   Der Term $(\lambda x.y)\mathbf{\Omega}$ ist also schwach normalisierbar; er ist aber nicht stark normalisierbar, da es eine unendliche β-Reduktionsfolge gibt: $(\lambda x.y)\mathbf{\Omega} \rhd_{1\beta} (\lambda x.y)\mathbf{\Omega} \rhd_{1\beta} \cdots$.
6. $(\lambda x.xxy)(\lambda x.xxy) \rhd_{1\beta} (\lambda x.xxy)(\lambda x.xxy)y \rhd_{1\beta} (\lambda x.xxy)(\lambda x.xxy)yy \rhd_{1\beta} \cdots$.

   Der Term $(\lambda x.xxy)(\lambda x.xxy)$ hat keine β-Normalform.

   Der Term $(\lambda u.v)((\lambda x.xxy)(\lambda x.xxy))$ ist schwach normalisierbar mit β-Normalform v, aber nicht stark normalisierbar.

Beispiel. Es ist $(\lambda xyz.xzy)(\lambda xy.x) =_\beta (\lambda xy.x)(\lambda x.x)$, da

$$\begin{aligned}(\lambda xyz.xzy)(\lambda xy.x) &\equiv_\alpha (\lambda xyz.xzy)(\lambda uv.u)\\ &\rhd_{1\beta} \lambda yz.(\lambda uv.u)zy\\ &\rhd_{1\beta} \lambda yz.(\lambda v.z)y\\ &\rhd_{1\beta} \lambda yz.z\end{aligned}$$

und

$$\begin{aligned}(\lambda xy.x)(\lambda x.x) &\equiv_\alpha (\lambda xy.x)(\lambda u.u)\\ &\rhd_{1\beta} \lambda y.(\lambda u.u)\\ &\doteq \lambda yu.u\\ &\equiv_\alpha \lambda yz.z\end{aligned}$$

Lemma 1.10

1. *Wenn* $P \rhd_\beta Q$, *dann* $\mathrm{FV}(P) \supseteq \mathrm{FV}(Q)$.
2. *Wenn* $P \equiv_\alpha P'$, $Q \equiv_\alpha Q'$ *und* $P \rhd_\beta Q$, *dann* $P' \rhd_\beta Q'$.
3. *Wenn* $P \equiv_\alpha P'$, $Q \equiv_\alpha Q'$ *und* $P =_\beta Q$, *dann* $P' =_\beta Q'$.
4. *Wenn* $M \rhd_\beta N$ *und* $P \rhd_\beta Q$, *dann* $P[M/x] \rhd_\beta Q[N/x]$.
5. *Wenn* $M =_\beta N$ *und* $P =_\beta Q$, *dann* $P[M/x] =_\beta Q[N/x]$.

Lemma 1.11 *Die Klasse aller β-Normalformen lässt sich durch folgende Regeln induktiv definieren:*

1. *Jedes Atom a ist eine β-Normalform.*
2. *Wenn $M_1, \ldots, M_n$ β-Normalformen sind, dann ist auch $aM_1 \ldots M_n$ eine β-Normalform.*
3. *Wenn M eine β-Normalform ist, dann ist auch $\lambda x.M$ eine β-Normalform.*

Das heißt, eine β-Normalform hat die Form $\lambda x_1 \ldots x_n.aM_1 \ldots M_m$, wobei die Terme M_i (für $1 \leq i \leq m$) dieselbe Form haben.

Beweis. Wir müssen zeigen, dass M eine β-Normalform ist genau dann, wenn M durch die Regeln (1)-(3) erzeugt werden kann. Die Implikation von rechts nach links ist offensichtlich.
Für die Implikation von links nach rechts zeigen wir per Induktion über M, dass M mit den Regeln (1)-(3) erzeugt werden kann, falls M eine β-Normalform ist.
Sei M eine β-Normalform.

1. Falls $M \doteq a$, dann kann M gemäß Regel (1) erzeugt werden.
2. Falls $M \doteq (PQ)$, dann können P und Q nach Induktionsvoraussetzung durch Anwendungen der Regeln (1)-(3) erzeugt werden, wobei P keine Abstraktion ist. Also ist $P \doteq a$ oder $P \doteq aM_1 \ldots M_k$. Damit ist $M \doteq aQ$ oder $M \doteq aM_1 \ldots M_k Q$, was mit Regel (2) erzeugt werden kann.
3. Falls $M \doteq \lambda x.P$, dann kann P nach Induktionsvoraussetzung durch die Regeln (1)-(3) erzeugt werden, und durch eine Anwendung von Regel (3) somit auch M. QED

Bemerkung. Man beachte, dass Lemma 1.11 lediglich eine Aussage über die Klasse aller β-Normalformen ist, aber nichts aussagt über die Klasse der λ-Terme, die eine β-Normalform haben.

Lemma 1.12 *Ein beliebiger λ-Term hat entweder die in Lemma 1.11 genannte Form, oder er enthält einen Teilterm der Form $\lambda x_1 \ldots x_n.(\lambda x.M)NM_1 \ldots M_m$, für $m, n \geq 0$.*

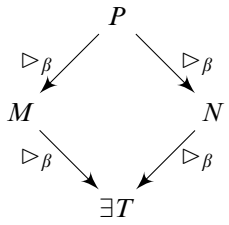
Beweis. Betrachte den ausgeschlossenen Unterfall von $M \eqsim (PQ)$ im Beweis von Lemma 1.11. QED

Theorem 1.13 (Church & Rosser, 1936)

1. *Wenn $P \rhd_\beta M$ und $P \rhd_\beta N$, dann existiert ein Term T, so dass $M \rhd_\beta T$ und $N \rhd_\beta T$.*
2. *Wenn $M =_\beta N$, dann existiert ein Term T, so dass $M \rhd_\beta T$ und $N \rhd_\beta T$.*

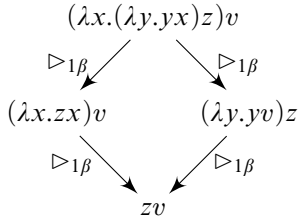
Bemerkungen.

1. Eigenschaft (1) bezeichnet man als *Konfluenz* (engl. *confluence*) der β-Reduktion oder als *Church-Rosser(-Eigenschaft)*. Diagrammatisch: *Konfluenz* *Church-Rosser*



2. Eigenschaft (2) bedeutet, dass zwei β-gleiche Terme dieselbe Interpretation haben, da es immer einen Term gibt, zu dem beide reduzieren.

Beispiel.



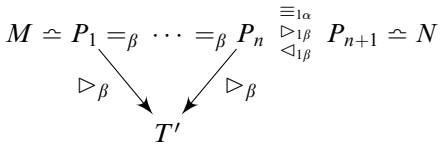
Beweis von Theorem 1.13 (2). Per Induktion über der Anzahl der Schritte von M nach N:

Anzahl der Schritte $= 0$: trivial.

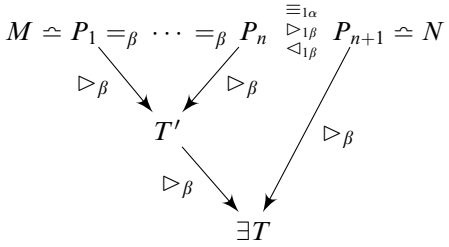
Anzahl der Schritte $= n + 1$: Nach Induktionsvoraussetzung gilt

$$P_1 \rhd_\beta T', \ldots, P_n \rhd_\beta T'$$

für einen Term T'. Für den Schritt von n nach $n+1$ besteht die folgende Situation:



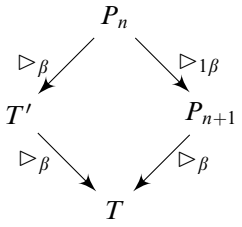
und wir müssen zeigen:



Fall 1 ($\equiv_{1\alpha}$): $T :\eqsim T'$, da $P_{n+1} \equiv_{1\alpha} P_n \rhd_\beta T'$.

Fall 2 ($\lhd_{1\beta}$): $T :\eqsim T'$, da $P_{n+1} \rhd_{1\beta} P_n \rhd_\beta T'$.

Fall 3 ($\rhd_{1\beta}$): Da $P_n \rhd_\beta T'$ und $P_n \rhd_{1\beta} P_{n+1}$, existiert nach Theorem 1.13 (1) ein Term T, so dass $T' \rhd_\beta T$ und $P_{n+1} \rhd_\beta T$:



Somit $M \rhd_\beta T$ (da $M =_\beta P_n$) und $N \rhd_\beta T$. QED

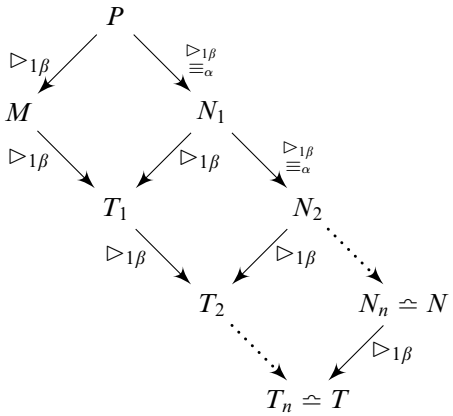
Für den Beweis von Theorem 1.13 (1) werden noch weitere Definitionen und Lemmas benötigt. Die Schwierigkeit besteht darin, dass Konfluenz von $\rhd_\beta$ nicht einfach bewiesen werden kann, indem man zunächst Konfluenz von $\rhd_{1\beta}$ beweist, d. h.:

$$\text{Wenn } P \rhd_{1\beta} M \text{ und } P \rhd_{1\beta} N, \text{ dann } \exists T\colon M \rhd_{1\beta} T \text{ und } N \rhd_{1\beta} T. \qquad (*)$$

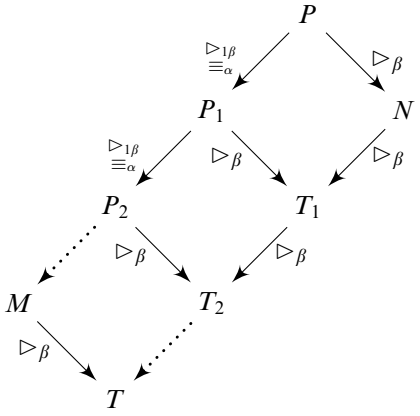
Konfluenz von $\rhd_\beta$ könnte dann mit $(*)$ bewiesen werden, indem man zunächst den Spezialfall

$$\text{wenn } P \rhd_{1\beta} M \text{ und } P \rhd_\beta N, \text{ dann } \exists T\colon M \rhd_\beta T \text{ und } N \rhd_{1\beta} T$$

per Induktion über der Länge der Reduktionsfolge (einschließlich α-Konversionen) von P nach N beweist; schematisch:



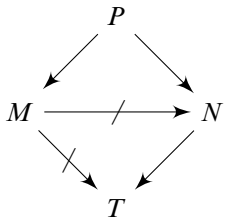
Der allgemeine Fall könnte dann durch eine zweite Induktion über der Länge der Reduktionsfolge (wieder einschließlich α-Konversionen) von P nach M bewiesen werden:



Jedoch gilt $(*)$ nicht für alle λ-Terme. Ein Gegenbeispiel ist

$$P \mathrel{\hat{=}} (\lambda y.uyy)(\mathbf{I}z) \qquad (\text{wobei } \mathbf{I} \mathrel{\hat{=}} \lambda x.x)$$

In diesem Fall gilt $P \rhd_{1\beta} u(\mathbf{I}z)(\mathbf{I}z)$ und $P \rhd_{1\beta} (\lambda y.uyy)z \rhd_{1\beta} uzz$. Aber $u(\mathbf{I}z)(\mathbf{I}z)$ kann nicht zu uzz kontrahiert werden, d. h. $u(\mathbf{I}z)(\mathbf{I}z) \not\rhd_{1\beta} uzz$.
Der Term $u(\mathbf{I}z)(\mathbf{I}z)$ könnte zu uzz in einem Schritt reduziert werden, indem man die beiden nicht-überlappenden Redexe $(\mathbf{I}z)$ und $(\mathbf{I}z)$ parallel kontrahiert. Man könnte daher versuchen, Konfluenz für eine Relation der parallelen Reduktion zu beweisen, um dann die Konfluenz von $\rhd_\beta$ durch zwei Induktionen wie bei $(*)$ zu beweisen.
Allerdings kann man eine parallele Reduktion nicht einfach als simultane nicht-überlappende Kontraktionen definieren, da Konfluenz auch für eine solche Relation nicht gilt. Ein Gegenbeispiel kann für den Term $P \mathrel{\hat{=}} (\lambda x.R_1)R_2$ angegeben werden, wobei $R_1 \mathrel{\hat{=}} (\lambda y.xyz)w$ und $R_2 \mathrel{\hat{=}} (\lambda u.u)v$ beides Redexe sind. Der Term P kontrahiert zu $(\lambda y.R_2yz)w \mathrel{\hat{=}} M$ durch eine (triviale) simultane nicht-überlappende Kontraktion. Durch eine simultane nicht-überlappende Kontraktion von R_1 und R_2 in P erhält man $(\lambda x.xwz)v \mathrel{\hat{=}} N$, was nur reduziert werden kann zu entweder N selbst (durch α-Konversion) oder zu $vwz \mathrel{\hat{=}} T$ (durch Kontraktion). Aber durch alleinige Verwendung von simultanen nicht-überlappenden Kontraktionen kann M weder auf N noch auf T reduziert werden. Das heißt, wir haben die folgende Situation (wobei die Pfeile für simultane nicht-überlappende Kontraktionen stehen):



Man kann aber eine Relation der parallelen Reduktion (das sogenannte *minimal complete development*) angeben, für die Konfluenz gilt. Dazu definieren wir zunächst bestimmte Ergebnisse von Kontraktionen.

Definition 1.14 Seien R und S mit $R \neq S$ β-Redexe in P, wobei $R \rhd_{1\beta} R'$, wodurch P zu P' abgeändert wird.
Das *Residuum* $\mathrm{Res}(S, R)$ von S bezüglich R ist ein Redex in P', definiert wie folgt: *Residuum*

1. R ist ein (echter) Teilterm von S. Dann ist $\mathrm{Res}(S, R) :\eqsim S\left[R'\right]$.

   (Das heißt, S hat die Form $(\lambda x.M)N$, und R kommt in M oder in N vor. Die Kontraktion $R \rhd_{1\beta} R'$ ändert M zu M' oder N zu N', d. h. S wird zu $(\lambda x.M')N$ oder $(\lambda x.M)N'$ in P'. Dann ist $\mathrm{Res}(S, R)$ entweder $(\lambda x.M')N$ oder $(\lambda x.M)N'$.)

2. R ist kein Teilterm von S. Dann ist $\mathrm{Res}(S, R) :\eqsim S$.

   (Das heißt, R und S überlappen sich in P nicht. Die Kontraktion von R ändert somit das Redex S nicht, und S ist das Residuum bezüglich R in P'.)

Bemerkung. Neben den betrachteten Fällen (1) und (2) können noch weitere Fälle unterschieden werden, für die $\mathrm{Res}(S, R)$ festzulegen wäre. Wir verzichten darauf, da diese zusätzlichen Fälle im Folgenden nicht auftreten.

Definition 1.15 Sei $\mathcal{R} = \{R_1, \ldots, R_n\}$ eine Menge von Redexen in P. Ein Redex R_i heißt *minimal*, falls kein R_j echter Teilterm von R_i ist. *minimal*

Es gilt $P \rhd_{\mathrm{mcd}} Q$ (*minimal complete development* von P zu Q), falls Q aus P durch das folgende (nicht-deterministische) Verfahren hervorgeht: *minimal complete development*

(1) Wähle ein minimales Element R_i in $\mathcal{R}$.

(2) β-kontrahiere R_i in P: $P \rhd_{1\beta} P'$.

(3) Sei $\mathcal{R}' = \bigcup_{j \neq i} \mathrm{Res}(R_j, R_i)$. ($\mathcal{R}'$ hat somit $n-1$ Elemente.)

(4) Falls $\mathcal{R}' \neq \emptyset$, dann gehe zu Schritt (1) mit $\mathcal{R} := \mathcal{R}'$ und $P :\eqsim P'$.

(5) Falls $\mathcal{R}' = \emptyset$, dann sei $Q \equiv_\alpha P'$.

(Falls $P \rhd_{\mathrm{mcd}} Q$ gilt, sagen wir auch, dass P *zu* Q *mcd-reduziert*.)

Bemerkungen.

1. Die Relation $\rhd_{\mathrm{mcd}}$ ist relativ zu einer gewählten Menge $\mathcal{R}$ von Redexen definiert. Für verschiedene Mengen $\mathcal{R}$ gibt es somit verschiedene Relationen $\rhd_{\mathrm{mcd}}$. Man müsste also für die jeweiligen Mengen von Redexen $\mathcal{R}$ genaugenommen von Relationen $\rhd_{\mathrm{mcd}}^{\mathcal{R}}$ sprechen. Wir nehmen stattdessen an, dass die Menge der Redexe jeweils geeignet gewählt ist, nämlich so, dass alle Redexe des jeweils betrachteten Terms in $\mathcal{R}$ enthalten sind, bzw. im Beweis der Konfluenz von $\rhd_{\mathrm{mcd}}$ (Lemma 1.18) derart, dass die dort genannten Bedingungen erfüllt sind.
2. Für $\mathcal{R} = \emptyset$ gilt $P \rhd_{\mathrm{mcd}} P$.

3. Die Relation $\rhd_{\text{mcd}}$ ist nicht transitiv, wie das folgende Beispiel zeigt:

$$(\lambda x.xy)(\lambda z.z) \rhd_{\text{mcd}} (\lambda z.z)y \text{ und } (\lambda z.z)y \rhd_{\text{mcd}} y, \text{ aber } (\lambda x.xy)(\lambda z.z) \not\rhd_{\text{mcd}} y.$$

(Das Redex $(\lambda z.z)y$ ist kein Residuum von $(\lambda x.xy)(\lambda z.z)$.)

Beispiel.
Wir betrachten wieder $P \doteq (\lambda x.R_1)R_2$, wobei $R_1 \doteq (\lambda y.xyz)w$ und $R_2 \doteq (\lambda u.u)v$.
Es ist $P \rhd_{\text{mcd}} vwz \doteq Q$.
Die Menge der Redexe in P ist $\mathcal{R} = \{(\lambda x.R_1)R_2, R_1, R_2\}$.

(1) Wähle das minimale Element R_1.

(2) $P \rhd_{1\beta} (\lambda x.xwz)R_2 \doteq P'$.

(3) $\mathcal{R}' = \{\text{Res}(P, R_1), \text{Res}(R_2, R_1)\} = \{P', R_2\}$.

(4) $\mathcal{R}' \neq \emptyset$.

(1) Wähle das minimale Element R_2.

(2) $P' \rhd_{1\beta} (\lambda x.xwz)v \doteq P''$.

(3) $\mathcal{R}'' = \{\text{Res}(P', R_2)\} = \{P''\}$.

(4) $\mathcal{R}'' \neq \emptyset$.

(1) Wähle das minimale Element P''.

(2) $P'' \rhd_{1\beta} vwz \doteq P'''$.

(3) $\mathcal{R}''' = \emptyset$.

(5) Sei $Q \equiv_\alpha P'''$.

Lemma 1.16 (Invarianz von $\rhd_{\text{mcd}}$ bezüglich $\equiv_\alpha$)
Wenn $P \rhd_{\text{mcd}} Q$ und $P \equiv_\alpha P'$, dann $P' \rhd_{\text{mcd}} Q$.

Lemma 1.17 (Invarianz von $\rhd_{\text{mcd}}$ bezüglich Substitution)
Wenn $M \rhd_{\text{mcd}} M'$ und $N \rhd_{\text{mcd}} N'$, dann $M[N/x] \rhd_{\text{mcd}} M'[N'/x]$.

Lemma 1.18 (Konfluenz von $\rhd_{\text{mcd}}$)
Wenn $P \rhd_{\text{mcd}} Q$ und $P \rhd_{\text{mcd}} R$, dann existiert ein Term T, so dass $Q \rhd_{\text{mcd}} T$ und $R \rhd_{\text{mcd}} T$.

Beweis. Mit Lemma 1.16 können wir annehmen, dass die gegebenen mcd-Reduktionen keine α-Schritte enthalten. Der Beweis erfolgt per Induktion über der Struktur von P.

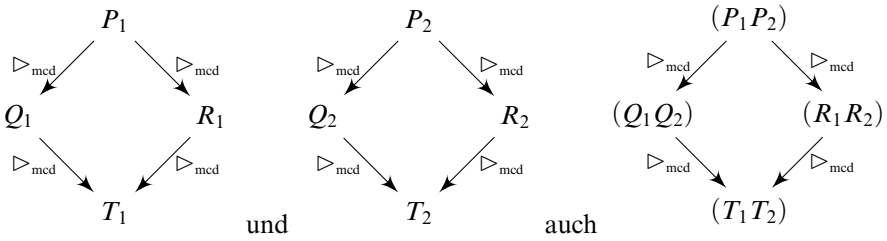
Fall 1: $P \doteq a$. Dann ist $Q \doteq R \doteq P$. Setze $T :\doteq P$.

Fall 2: $P \doteq \lambda x.P_1$. Dann sind alle Redexe von P in P_1, und da keine α-Schritte vorkommen, gilt $Q \doteq \lambda x.Q_1$ und $R \doteq \lambda x.R_1$, wobei $P_1 \rhd_{\text{mcd}} Q_1$ und $P_1 \rhd_{\text{mcd}} R_1$.

Nach Induktionsvoraussetzung existiert ein Term T_1, so dass $Q_1 \rhd_{\text{mcd}} T_1$ und $R_1 \rhd_{\text{mcd}} T_1$. Setze $T :\doteq \lambda x.T_1$.

Fall 3: $P \doteq (P_1 P_2)$, und alle Redexe von $\mathcal{R}$ sind in P_1 und P_2, d. h. P selbst wird nicht reduziert.

Nach Induktionsvoraussetzung existieren Terme T_1 und T_2, so dass mit



gilt.

Setze $T :\doteq (T_1 T_2)$.

Fall 4: $P \doteq (\lambda x.M)N$, und das Residuum von P wird in nur einer der beiden vorausgesetzten mcd-Reduktionen kontrahiert; wir nehmen an, dass es in $P \rhd_{\text{mcd}} Q$ kontrahiert wird.

Dann hat $P \rhd_{\text{mcd}} Q$ die Form

$$P \doteq (\lambda x.M)N \rhd_{\text{mcd}} (\lambda x.M')N' \rhd_{1\beta} M'[N'/x] \doteq Q$$

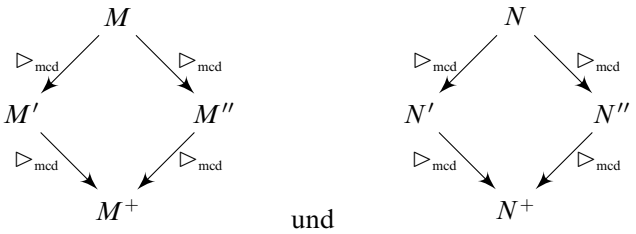
(wobei $M \rhd_{\text{mcd}} M'$ und $N \rhd_{\text{mcd}} N'$).

Die andere vorausgesetzte mcd-Reduktion $P \rhd_{\text{mcd}} R$ hat die Form

$$P \doteq (\lambda x.M)N \rhd_{\text{mcd}} (\lambda x.M'')N'' \doteq R$$

(wobei $M \rhd_{\text{mcd}} M''$ und $N \rhd_{\text{mcd}} N''$).

Nach Induktionsvoraussetzung für M und N existieren Terme M^+ und N^+, so dass



Setze $T :\doteq M^+[N^+/x]$. Dann gilt mit Lemma 1.17:

$$Q \doteq M'[N'/x] \rhd_{\text{mcd}} M^+[N^+/x]$$

Die α-Konversionen in den mcd-Reduktionen für M'' und N'' können wie folgt separiert werden:

$$M'' \rhd_{\text{mcd}} M^* \equiv_\alpha M^+ \qquad\qquad N'' \rhd_{\text{mcd}} N^* \equiv_\alpha N^+$$

wobei wir annehmen, dass in $M'' \rhd_{\text{mcd}} M^*$ und $N'' \rhd_{\text{mcd}} N^*$ keine α-Schritte vorkommen. Wir erhalten so $R \rhd_{\text{mcd}} T$:

$$R \eqdot (\lambda x.M'')N'' \rhd_{\text{mcd}} (\lambda x.M^*)N^* \rhd_{1\beta} M^*[N^*/x] \equiv_\alpha M^+[N^+/x]$$

Fall 5: $P \eqdot (\lambda x.M)N$, und beide vorausgesetzten mcd-Reduktionen kontrahieren das Residuum von P.

Dann hat $P \rhd_{\text{mcd}} Q$ die Form

$$P \eqdot (\lambda x.M)N \rhd_{\text{mcd}} (\lambda x.M')N' \rhd_{1\beta} M'[N'/x] \eqdot Q$$

und $P \rhd_{\text{mcd}} R$ hat die Form

$$P \eqdot (\lambda x.M)N \rhd_{\text{mcd}} (\lambda x.M'')N'' \rhd_{1\beta} M''[N''/x] \eqdot R$$

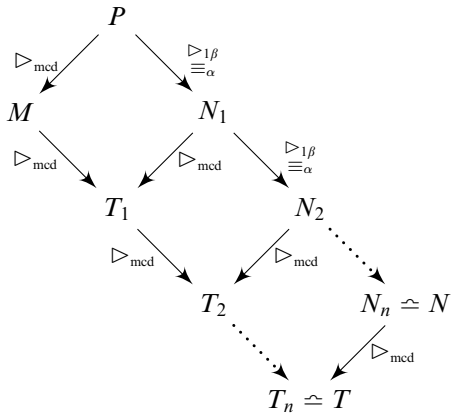
Wir argumentieren wie in Fall 4, und wählen $T :\eqdot M^+[N^+/x]$. Mit Lemma 1.17 ergibt sich dann die Behauptung. QED

Bemerkung. Der Beweis der Konfluenz von $\rhd_{\text{mcd}}$ beruht wesentlich darauf, dass alle Redexe bereits im Ausgangsterm vorliegen und dadurch kontrolliert werden können.

Beweis von Theorem 1.13 (1). Unter Verwendung der Konfluenz von $\rhd_{\text{mcd}}$ (Lemma 1.18) kann man zeigen:

Wenn $P \rhd_{\text{mcd}} M$ und $P \rhd_\beta N$, dann existiert ein Term T, so dass $M \rhd_\beta T$ und $N \rhd_{\text{mcd}} T$.

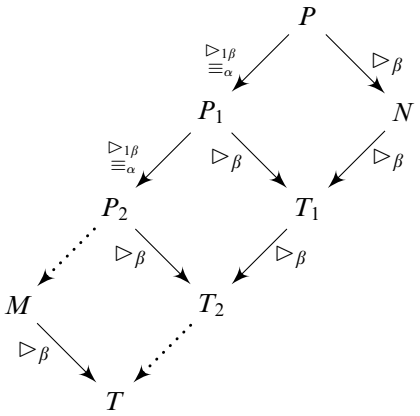
Der Nachweis erfolgt durch eine Induktion gemäß folgendem Schema:



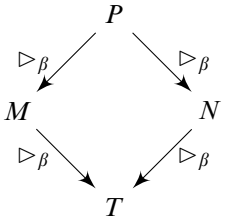
Es gilt: $R \rhd_{1\beta} S$ impliziert $R \rhd_{\text{mcd}} S$, $R \rhd_{\text{mcd}} S$ impliziert $R \rhd_\beta S$, und $\rhd_\beta$ ist transitiv.
Folglich gilt:

Wenn $P \rhd_{1\beta} M$ und $P \rhd_\beta N$, dann existiert ein Term T, so dass $M \rhd_\beta T$ und $N \rhd_\beta T$.

Hieraus folgt die Behauptung durch eine Induktion gemäß folgendem Schema:



Das heißt, wir erhalten:



QED

Korollar 1.19

1. *Wenn M und N β-Normalformen von P sind, dann gilt $M \equiv_\alpha N$.*

   (Das heißt, β-Normalformen sind eindeutig modulo Kongruenz.)

2. *Wenn $M =_\beta N$ gilt und N in β-NF ist, dann gilt $M \rhd_\beta N$.*

   (Aufgrund des Church-Rosser-Theorems reduzieren M und N zu einem Term T. Da N in β-NF ist, gilt $N \equiv_\alpha T$. Somit gilt $M \rhd_\beta N$.)

3. *Wenn $M =_\beta N$ gilt, dann haben entweder sowohl M als auch N keine β-NF, oder beide Terme haben dieselbe β-NF (modulo Kongruenz).*

4. *Wenn $M =_\beta N$ gilt und M, N in β-Normalform sind, dann gilt $M \equiv_\alpha N$.*

   (Der λ-Kalkül ist also konsistent in dem Sinne, dass nicht alle λ-Terme β-gleich sind; mit anderen Worten, die Relation $=_\beta$ ist nicht trivial. Man betrachte die beiden Terme $\lambda xy.xy$ und $\lambda xy.yx$. Beide Terme sind in β-Normalform, aber nicht kongruent. Aufgrund des Korollars können sie nicht β-gleich sein.)

Definition 1.20

1. Eine *L-Reduktionsfolge* ist eine β-Reduktionsfolge, bei der immer das am weitesten links stehende Redex im jeweiligen Term kontrahiert wird. Ein Redex $(\lambda x_1.M_1)N_1$ ist dabei *weiter links stehend* als $(\lambda x_2.M_2)N_2$ (im betrachteten Term), falls sich λx_1 links von λx_2 befindet. *L-Reduktionsfolge*

2. Eine *QL-Reduktionsfolge* ist eine β-Reduktionsfolge $(M_1, M_2, M_3, \ldots)$, so dass es zu jedem M_i, das nicht letztes Glied der Folge ist, ein M_j und ein M_{j+1} mit $j \geq i$ gibt, so dass beim Übergang von M_j zu M_{j+1} das linkeste Redex in M_j kontrahiert wird. *QL-Reduktionsfolge*

Bemerkungen.

1. L steht für *leftmost*, QL für *quasi-leftmost*. Eine QL-Reduktionsfolge ist also eine β-Reduktionsfolge, bei der "immer wieder" das linkeste Redex kontrahiert wird.
2. L-Reduktionsfolgen entspricht bei Programmiersprachen die Strategie der *lazy evaluation*.

Beispiel. Eine L-Reduktionsfolge für den Term

$$\overbrace{(\underbrace{(\lambda x.x)}_{\text{Redex 2}}\underbrace{((\lambda y.yy)(\lambda z.z))}_{\text{Redex 2}})}^{\text{Redex 1}}\underbrace{((\lambda u.u)(\lambda v.v))}_{\text{Redex 3}}$$

ist gegeben durch:

$$\begin{aligned}
\underbrace{((\lambda x.x)((\lambda y.yy)(\lambda z.z)))}_{\text{linkestes Redex}}((\lambda u.u)(\lambda v.v)) &\rhd_{1\beta} \underbrace{((\lambda y.yy)(\lambda z.z))}_{\text{linkestes Redex}}((\lambda u.u)(\lambda v.v)) \\
&\rhd_{1\beta} \underbrace{((\lambda z.z)(\lambda z.z))}_{\text{linkestes Redex}}((\lambda u.u)(\lambda v.v)) \\
&\rhd_{1\beta} \underbrace{(\lambda z.z)((\lambda u.u)(\lambda v.v))}_{\text{linkestes Redex}} \\
&\rhd_{1\beta} \underbrace{(\lambda u.u)(\lambda v.v)}_{\text{linkestes Redex}} \\
&\rhd_{1\beta} \lambda v.v
\end{aligned}$$

Theorem 1.21 *Falls ein λ-Term M eine β-Normalform hat, dann terminiert jede mit M beginnende L-Reduktionsfolge (und damit auch jede QL-Reduktionsfolge).*

Bemerkungen.

1. Für die Beweise siehe Barendregt (2012), Abschnitt 13.2.
2. Man beachte die Kontraposition des Theorems. Zum Nachweis, dass ein Term M *keine* β-Normalform hat, genügt es zu zeigen, dass es eine mit M beginnende nicht-terminierende L- (bzw. QL-) Reduktionsfolge gibt.

Theorem 1.22 *Fixpunktkombinatoren*

Es gibt Fixpunktkombinatoren Y, d. h. Kombinatoren mit der Eigenschaft

1. $Yx =_\beta x(Yx)$,

bzw. mit der stärkeren Eigenschaft

2. $Yx \rhd_\beta x(Yx)$.

Beweis.

1. Für den auf Curry zurückgehenden Kombinator (vgl. Rosenbloom, 1950)

$$\Upsilon :\eqsim \lambda x.(\lambda y.x(yy))\underbrace{(\lambda y.x(yy))}_{\eqsim :M}$$

gilt

$$\Upsilon x \rhd_\beta MM \rhd_\beta x(MM) \lhd_\beta x(\Upsilon x)$$

Υ erfüllt jedoch nicht (2).

2. Für den von Turing (1937) angegebenen Kombinator

$$\Theta :\eqsim (\lambda zx.x(zzx))\underbrace{(\lambda zx.x(zzx))}_{\eqsim :N}$$

gilt

$$\Theta x \rhd_\beta (\lambda x.x(NNx))x \rhd_\beta x(NNx) \eqsim x(\Theta x)$$

Θ erfüllt damit natürlich auch (1). QED

Korollar 1.23 *Für jedes N gibt es ein M, so dass für $n \geq 0$ gilt: $My_1 \dots y_n =_\beta N[M/x]$.*

Beweis. Setze $M :\eqsim Y(\lambda xy_1 \dots y_n.N)$ für einen Fixpunktkombinator Y. QED

Bemerkungen.

1. Wählen wir für Y den Fixpunktkombinator Θ, d. h. $M :\eqsim \Theta(\lambda xy_1 \dots y_n.N)$, dann gilt sogar $My_1 \dots y_n \rhd_\beta N[M/x]$.
2. Jede "intuitive" Gleichung der Form $xy_1 \dots y_n = N$, die x durch einen Term N definiert, in dem x selbst wieder vorkommen kann (und die insofern "selbstbezüglich" ist) besitzt als Lösung einen Term M.

Theorem 1.24 *M ist ein Fixpunktkombinator, d. h. $Mx =_\beta x(Mx)$, genau dann, wenn M Fixpunkt von* **SI** *ist, d. h. wenn $M =_\beta \mathbf{SI}M$.*

Beweis. (Siehe Barendregt, 2012, 6.5.3.)
Es ist $\mathbf{SI} =_\beta \lambda yz.z(yz)$.
Sei M Fixpunkt von **SI**, d. h. $M =_\beta \mathbf{SI}M$. Dann ist $MN =_\beta \mathbf{SI}MN =_\beta N(MN)$, d. h. M ist ein Fixpunktkombinator.
Sei $Mx =_\beta x(Mx)$. Dann ist Mx nicht in Normalform, da sonst Mx und $x(Mx)$ kongruent wären. Damit gilt $Mx \rhd_\beta xP$ und $x(Mx) \rhd_\beta xP$ für ein P. Ferner gilt $M \rhd_\beta \lambda z.N$, da M als Kombinator nicht mit einer Variable beginnen kann. Damit gilt

$$\lambda x.Mx =_\beta \lambda x.(\lambda z.N)x =_\beta \lambda x.N[x/z] =_\beta M$$

(d. h. η-Konversion (siehe folgende Definition) ist für M beweisbar).
Somit ist $M =_\beta \lambda x.Mx =_\beta \lambda x.x(Mx) =_\beta \mathbf{SI}M$. QED

Definition 1.25

1. Ein Term der Form $\lambda x.Mx$ heißt *η-Redex*, falls $x \notin \mathrm{FV}(M)$. Der Term M ist sein *Kontraktum*. *η-Redex* *Kontraktum*
2. Die Operation der *η-Kontraktion* ist definiert durch: *η-Kontraktion*

$$P\big[\lambda x.Mx\big] \rhd_{1\eta} P\big[M\big], \text{ falls } x \notin \mathrm{FV}(M).$$

3. Es ist $P \rhd_{\beta\eta} Q$ (d. h. *P $\beta\eta$-reduziert* zu Q), falls $P \mathrel{\dot{=}} P_1 \overset{\equiv_{1\alpha}\ \rhd_{1\beta}\ \rhd_{1\eta}}{} P_2 \overset{\equiv_{1\alpha}\ \rhd_{1\beta}\ \rhd_{1\eta}}{} \cdots \overset{\equiv_{1\alpha}\ \rhd_{1\beta}\ \rhd_{1\eta}}{} P_n \mathrel{\dot{=}} Q$. *$\beta\eta$-Reduktion*
4. Es ist $P =_{\beta\eta} Q$ (d. h. *P ist $\beta\eta$-gleich* zu Q), falls $P \mathrel{\dot{=}} P_1 \overset{\equiv_{1\alpha}\ \rhd_{1\beta}\ \lhd_{1\beta}\ \rhd_{1\eta}\ \lhd_{1\eta}}{} P_2 \overset{\equiv_{1\alpha}\ \rhd_{1\beta}\ \lhd_{1\beta}\ \rhd_{1\eta}\ \lhd_{1\eta}}{} \cdots \overset{\equiv_{1\alpha}\ \rhd_{1\beta}\ \lhd_{1\beta}\ \rhd_{1\eta}\ \lhd_{1\eta}}{} P_n \mathrel{\dot{=}} Q$. *$\beta\eta$-Gleichheit*

   ($\beta\eta$-gleiche Terme heißen auch *$\beta\eta$-konvertibel*. Falls keine β-Kontraktionen vorkommen, sprechen wir von *η-Konversion* $=_\eta$.) *$\beta\eta$-Konversion* *η-Konversion*

Bemerkung. $\beta\eta$-Gleichheit besagt intuitiv, dass es für die Bedeutung eines Terms nur auf sein Verhalten bei Anwendung auf einen anderen Term ankommt (also Extensionalität vorliegt; vgl. Lemma 1.38).

Lemma 1.26 *Lemma 1.10 gilt auch für* $\rhd_{\beta\eta}$.

Theorem 1.27 *$\beta\eta$-Reduktion genügt Church-Rosser.*

Beweis. Siehe Hindley & Seldin (2008), Appendix A2B.

1.2 λ-Definierbarkeit rekursiver Funktionen

Definition 1.28 Sei $M^0N :\mathrel{\dot{=}} N$ und $M^{n+1}N :\mathrel{\dot{=}} M(M^nN)$.
Dann sind die *Church-Ziffern* wie folgt definiert: $\underline{n} :\mathrel{\dot{=}} \lambda xy.x^n y$. *Church-Ziffern*

Bemerkungen.

1. Vergleiche Wittgenstein (1922), 6.021: “Die Zahl ist der Exponent einer Operation”.
2. Falls $\underline{m} =_\beta \underline{n}$, dann ist $m = n$, da Church-Ziffern in β-Normalform sind.
3. Es gilt $\underline{n}PQ \rhd_\beta P^nQ$.

Lemma 1.29 *Es gibt Kombinatoren* $\mathbf{N}, \mathbf{V}, \mathbf{D}$ *und* $\mathbf{R}$ *mit folgenden Eigenschaften:*

1. $\mathbf{N}\underline{k} =_\beta \underline{k+1}$ *(Nachfolger)*
2. $\mathbf{V}\underline{k+1} =_\beta \underline{k}$ *(Vorgänger)*
3. $\mathbf{D}PQ\underline{0} =_\beta P$ *(Paar- oder Bedingungskombinator; If-then-else)*
   $\mathbf{D}PQ\underline{k+1} =_\beta Q$

4. $\mathbf{R}PQ\underline{0} =_\beta P$ *(Rekursionskombinator)*

   $\mathbf{R}PQ\underline{k+1} =_\beta Q\underline{k}(\mathbf{R}PQ\underline{k})$

Bemerkungen.

1. $\mathbf{D}\underline{n}\,\underline{m}$ entspricht dem geordneten Paar $\langle n, m\rangle$, da gemäß (3) das erste oder zweite Element ausgewählt werden kann.
2. $\mathbf{D}PQ\underline{n}$ entspricht dem Operator *If* $n = 0$ *then* P*, else* Q.

Beweis von Lemma 1.29.

1. $\mathbf{N} :\eqsim \lambda uxy.x(uxy)$
3. $\mathbf{D} :\eqsim \lambda xyz.z(\mathbf{K}y)x$
2. $\mathbf{V} :\eqsim \lambda x.x(\lambda z.\mathbf{D}(\mathbf{N}(z\,\underline{0}))(z\,\underline{0}))(\mathbf{D}\underline{0}\underline{0})\,\underline{1}$

   Wir zeigen durch Induktion über k:

$$(\underbrace{\lambda z.\mathbf{D}(\mathbf{N}(z\,\underline{0}))(z\,\underline{0})}_{\eqsim:P})^{k+1}(\mathbf{D}\underline{0}\underline{0}) =_\beta \mathbf{D}\underline{k+1}\,\underline{k}$$

   Induktionsanfang:

$$P^1(\mathbf{D}\underline{0}\underline{0}) =_\beta \mathbf{D}(\mathbf{N}(\mathbf{D}\underline{0}\underline{0}\underline{0}))(\mathbf{D}\underline{0}\underline{0}\underline{0}) =_\beta \mathbf{D}(\mathbf{N}\underline{0})\,\underline{0} =_\beta \mathbf{D}\underline{1}\,\underline{0}$$

   Induktionsschritt: Sei $P^{k+1}(\mathbf{D}\underline{0}\underline{0}) =_\beta \mathbf{D}\underline{k+1}\,\underline{k}$. Dann ist

$$\begin{aligned}
P^{k+2}(\mathbf{D}\underline{0}\underline{0}) &=_\beta P(P^{k+1}(\mathbf{D}\underline{0}\underline{0}))\\
&=_\beta P(\mathbf{D}\underline{k+1}\,\underline{k}) \qquad \text{(Induktionsvoraussetzung)}\\
&=_\beta \mathbf{D}(\mathbf{N}(\mathbf{D}\underline{k+1}\,\underline{k}\,\underline{0}))(\mathbf{D}\underline{k+1}\,\underline{k}\,\underline{0})\\
&=_\beta \mathbf{D}(\mathbf{N}\underline{k+1})\,\underline{k+1}\\
&=_\beta \mathbf{D}\underline{k+2}\,\underline{k+1}
\end{aligned}$$

   Damit ist

$$\begin{aligned}
\mathbf{V}\underline{k+1} &=_\beta \underline{k+1}P(\mathbf{D}\underline{0}\underline{0})\,\underline{1}\\
&=_\beta P^{k+1}(\mathbf{D}\underline{0}\underline{0})\,\underline{1}\\
&=_\beta \mathbf{D}\underline{k+1}\,\underline{k}\,\underline{1}\\
&=_\beta \underline{k}
\end{aligned}$$

4. $\mathbf{R} :\eqsim \Theta(\lambda uxyz.\mathbf{D}x(y(\mathbf{V}z)(uxy(\mathbf{V}z)))z)$

   $\mathbf{R}$ ist nach Korollar 1.23 Lösung von $\mathbf{R}xyz =_\beta \mathbf{D}x(y(\mathbf{V}z)(\mathbf{R}xy(\mathbf{V}z)))z$. QED

Bemerkung. Der Rekursionskombinator $\mathbf{R}$ lässt sich auch ohne die Verwendung eines Fixpunktkombinators angeben, und zwar als stark normalisierbarer Term (siehe Hindley & Seldin, 2008, Beweis von Theorem 4.11).

Definition 1.30 Der λ-Term P *definiert* die k-stellige zahlentheoretische Funktion f, falls für alle $m_1, \ldots, m_k$ gilt, dass $P\,\underline{m_1}\ldots\underline{m_k} \simeq_\beta \underline{f(m_1,\ldots,m_k)}$. Unter Verwendung der Abkürzung $\vec{m}$ für $m_1,\ldots,m_k$ schreiben wir für Letzteres auch $P\,\underline{\vec{m}} \simeq_\beta \underline{f(\vec{m})}$. Dabei bedeutet

λ-Definierbarkeit

$$P\,\underline{\vec{m}} \simeq_\beta \underline{n},\ \text{dass} \begin{cases} P\,\underline{\vec{m}} =_\beta \underline{n} \iff f(\vec{m}) = n, & \text{falls } f(\vec{m}) \text{ definiert ist,} \\ P\,\underline{\vec{m}} \text{ hat keine } \beta\text{-Normalform,} & \text{falls } f(\vec{m}) \text{ nicht definiert ist.} \end{cases}$$

Definition 1.31 Die *primitiv-rekursiven Funktionen* sind induktiv definiert wie folgt:

primitiv-rekursive Funktion

1. Die *Zahl* $0\colon \mathbb{N}^0 \to \mathbb{N}$ ist eine 0-stellige primitiv-rekursive Funktion.
2. Die *Nachfolgerfunktion* $s\colon \mathbb{N} \to \mathbb{N}$ mit $s(n) = n+1$ ist primitiv-rekursiv.
3. Für jedes $n \geq 1$ und $i \leq n$ ist die *Projektion* $\pi_i^n\colon \mathbb{N}^n \to \mathbb{N}$ mit
$$\pi_i^n(m_1,\ldots,m_n) = m_i \qquad (m_1,\ldots,m_n \in \mathbb{N})$$
primitiv-rekursiv.
4. Ist $n \geq 1$ und $1 \leq i \leq k$, und sind $h\colon \mathbb{N}^k \to \mathbb{N}$ und alle $g_i\colon \mathbb{N}^n \to \mathbb{N}$ primitiv-rekursiv, dann ist auch die *Komposition* $f\colon \mathbb{N}^n \to \mathbb{N}$ mit
$$f(m_1,\ldots,m_n) = h(g_1(m_1,\ldots,m_n),\ldots,g_k(m_1,\ldots,m_n))$$
primitiv-rekursiv.
5. Ist $k \geq 0$, und sind $g\colon \mathbb{N}^k \to \mathbb{N}$ und $h\colon \mathbb{N}^{k+2} \to \mathbb{N}$ primitiv-rekursiv, dann ist auch die per *Rekursion* definierte Funktion $f\colon \mathbb{N}^{k+1} \to \mathbb{N}$ mit
$$\begin{aligned} f(0,m_1,\ldots,m_k) &= g(m_1,\ldots,m_k) \\ f(n+1,m_1,\ldots,m_k) &= h(n,f(n,m_1,\ldots,m_k),m_1,\ldots,m_k) \end{aligned}$$
primitiv-rekursiv.

Bemerkung. Statt $h(g_1(\vec{m}),\ldots,g_k(\vec{m}))$ schreiben wir auch $(h \circ [g_1;\ldots;g_k])(\vec{m})$.

Theorem 1.32 *Jede primitiv-rekursive Funktion ist λ-definierbar.*

Beweis. Wir geben λ-Terme an, die den jeweiligen Klauseln (1)-(5) in Definition 1.31 entsprechen.

1. $0\colon \mathbb{N}^0 \to \mathbb{N}$ ist λ-definiert durch den Term $\underline{0}$.
2. $s\colon \mathbb{N} \to \mathbb{N}$ ist λ-definiert durch den Term $\mathbf{N}$.
3. $\pi_i^n\colon \mathbb{N}^n \to \mathbb{N}$ ist λ-definiert durch den Term $\lambda x_1 \ldots x_n.x_i$.
4. Falls $h\colon \mathbb{N}^k \to \mathbb{N}$ und $g_i\colon \mathbb{N}^n \to \mathbb{N}$ durch P und Q_i λ-definiert sind $(1 \leq i \leq k)$, und $f\colon \mathbb{N}^n \to \mathbb{N}$ gegeben ist als $f(\vec{m}) := (h \circ [g_1;\ldots;g_k])(\vec{m})$ für alle $\vec{m} = (m_1,\ldots,m_n)$, dann wird die Funktion $f\colon \mathbb{N}^n \to \mathbb{N}$ durch den Term $\lambda\vec{x}.P(Q_1\vec{x})\ldots(Q_k\vec{x})$ λ-definiert, wobei $Q_i\vec{x} :\doteq (\ldots(Q_i x_1)\ldots)x_n$.

5. Falls $g\colon \mathbb{N}^k \to \mathbb{N}$ und $h\colon \mathbb{N}^{k+2} \to \mathbb{N}$ durch die λ-Terme P und Q λ-definiert sind, und $f\colon \mathbb{N}^{k+1} \to \mathbb{N}$ gegeben ist durch

$$\begin{aligned} f(0,\vec{m}) &= g(\vec{m}) \\ f(n+1,\vec{m}) &= h(n, f(n,\vec{m}), \vec{m}) \end{aligned}$$

dann wird f λ-definiert durch den Term $\lambda u\vec{x}.\mathbf{R}(P\vec{x})(\lambda uv.Quv\vec{x})u$.

Beweis durch Induktion über n:
Induktionsanfang:

$$\begin{aligned} (\lambda u\vec{x}.\mathbf{R}(P\vec{x})(\lambda uv.Quv\vec{x})u)\,\underline{0}\,\underline{\vec{m}} &=_\beta \mathbf{R}(P\,\underline{\vec{m}})(\lambda uv.Quv\,\underline{\vec{m}})\,\underline{0} \\ &=_\beta P\,\underline{\vec{m}} \\ &=_\beta \underline{g(\vec{m})} \qquad \text{(nach Voraussetzung über } g\text{)} \end{aligned}$$

Induktionsschritt:

$$\begin{aligned} (\lambda u\vec{x}.\mathbf{R}(P\vec{x})(\lambda uv.Quv\vec{x})u)\,\underline{n+1}\,\underline{\vec{m}} &=_\beta \mathbf{R}(P\,\underline{\vec{m}})(\lambda uv.Quv\,\underline{\vec{m}})\,\underline{n+1} \\ &=_\beta (\lambda uv.Quv\,\underline{\vec{m}})\,\underline{n}(\mathbf{R}(P\,\underline{\vec{m}})(\lambda uv.Quv\,\underline{\vec{m}})\,\underline{n}) \\ &=_\beta Q\,\underline{n}(\mathbf{R}(P\,\underline{\vec{m}})(\lambda uv.Quv\,\underline{\vec{m}})\,\underline{n})\,\underline{\vec{m}} \\ &=_\beta Q\,\underline{n}((\lambda u\vec{x}.\mathbf{R}(P\vec{x})(\lambda uv.Quv\vec{x})u)\,\underline{n}\,\underline{\vec{m}})\,\underline{\vec{m}} \\ &=_\beta Q\,\underline{n}\,\underline{f(n,\vec{m})}\,\underline{\vec{m}} \qquad \text{(Induktionsvoraussetzung)} \\ &=_\beta \underline{h(n, f(n,\vec{m}), \vec{m})} \qquad \text{(nach Vorauss. über } h\text{)} \end{aligned}$$

QED

Beispiel. Die Funktion $add\colon \mathbb{N}^2 \to \mathbb{N}$ mit $add(n,m) = n+m$ ist primitiv-rekursiv wie folgt definiert:

$$\begin{aligned} add(0,m) &= \pi^1_1(m) \\ add(n+1,m) &= (s \circ \pi^3_2)(n, add(n,m), m) \end{aligned}$$

Die λ-Definition der Funktion π^1_1 ist der Term $\lambda x.x \mathrel{\widehat{=}} \mathbf{I}$, und die λ-Definition der Komposition $s \circ \pi^3_2$ ist der Term $\lambda y_1y_2y_3.\mathbf{N}((\lambda x_1x_2x_3.x_2)y_1y_2y_3)$. Somit ergibt sich nach dem Schema in (5) als λ-Definition der Funktion add der Term

$$\mathbf{Add} \mathrel{\widehat{=}} \lambda ux.\mathbf{R}(\mathbf{I}x)(\lambda uv.(\lambda y_1y_2y_3.\mathbf{N}((\lambda x_1x_2x_3.x_2)y_1y_2y_3))uvx)u$$

Man kann alle durch die schematische Übersetzung erzeugten Redexe kontrahieren, und erhält diesen vereinfachten Term:

$$\mathbf{Add} \rhd_\beta \lambda ux.\mathbf{R}x(\lambda uv.\mathbf{N}v)u$$

Die Berechnung von $1+1$ (unter Verwendung von Lemma 1.29, 4 und 1) ist:

$$\begin{aligned}
\mathbf{Add}\,\underline{1}\,\underline{1} &\rhd_\beta \mathbf{R}\,\underline{1}(\lambda uv.\mathbf{N}v)\,\underline{1} \\
&=_\beta (\lambda uv.\mathbf{N}v)\,\underline{0}(\mathbf{R}\,\underline{1}(\lambda uv.\mathbf{N}v)\,\underline{0}) \\
&\rhd_\beta \mathbf{N}(\mathbf{R}\,\underline{1}(\lambda uv.\mathbf{N}v)\,\underline{0}) \\
&=_\beta \mathbf{N}\,\underline{1} \\
&=_\beta \underline{2}
\end{aligned}$$

Definition 1.33 Eine Funktion $f : \mathbb{N}^n \to \mathbb{N}$ $(n \geq 0)$ heißt *partiell-rekursiv* genau dann, wenn es primitiv-rekursive Funktionen g und h gibt, so dass für alle $\vec{m} = (m_1, \ldots, m_n)$

partiell-rekursive Funktion

$$f(\vec{m}) = h(\mu k.g(\vec{m}, k) = 0)$$

wobei der μ-Operator wie folgt definiert ist:

$$(\mu k.g(\vec{m}, k) = 0) := \begin{cases} \text{das kleinste } k, \text{ für das } g(\vec{m}, k) = 0 \text{ gilt,} \\ \qquad\qquad\qquad\qquad \text{falls es ein solches } k \text{ gibt;} \\ \text{undefiniert, falls es kein solches } k \text{ gibt.} \end{cases}$$

Falls stets ein kleinstes k existiert, heißt die Funktion $f(\vec{m}) = h(\mu k.g(\vec{m}, k) = 0)$ *rekursiv* oder *total rekursiv*.

total rekursive Funktion

Theorem 1.34 *Jede partiell-rekursive Funktion ist λ-definierbar.*

Beweis. Wir betrachten partiell-rekursive Funktionen

$$f(\vec{m}) := h(\mu k.g(\vec{m}, k) = 0)$$

wobei g und h primitiv-rekursive Funktionen seien, die durch die Terme P bzw. Q λ-definiert sind (vgl. Theorem 1.32).

Ansatz: Um $\mu k.g(\vec{m}, k) = 0$ zu berechnen, kann man eine Funktion F programmieren, so dass für $F(k)$ der Wert k ausgegeben wird, falls $g(\vec{m}, k) = 0$, und andernfalls $F(k+1)$ aufgerufen wird. Startet man das Programm für $k = 0$, wird also das kleinste k ausgegeben, für das $g(\vec{m}, k) = 0$. Gesucht ist also ein entsprechender Term M, so dass (grob gesprochen) gilt:

$$M\vec{x}y = (\mathit{If}\ P\vec{x}y = 0, \mathit{\ then\ } y, \mathit{\ else\ } M\vec{x}y + 1)$$

Die Funktion $f(\vec{m})$ könnte dann durch den Term $\lambda\vec{x}.Q(M\vec{x}\underline{0})$ repräsentiert werden.

Betrachte die Gleichung

$$M\vec{x}y =_\beta \mathbf{D}y(M\vec{x}(\mathbf{N}y))(P\vec{x}y) \qquad (\star)$$

Nach Korollar 1.23 ist $\mathbf{\Theta}\underbrace{(\lambda u\vec{x}y.\mathbf{D}y(u\vec{x}(\mathbf{N}y))(P\vec{x}y))}_{=:Z}$ eine Lösung der Gleichung.

Behauptung: f wird λ-definiert durch den Term $\lambda\vec{x}.Q(\mathbf{\Theta}Z\vec{x}\underline{0})$.
Dazu genügt es zu zeigen:

$$\mathbf{\Theta}Z\underline{\vec{m}}\underline{0} =_\beta \underline{k_1}, \text{ falls } k_1 \text{ kleinstes } k \text{ mit } g(\vec{m},k)=0$$

(da dann $Q\underline{k_1} =_\beta \underline{f(\vec{m})}$).
Wir werden zeigen:

$$\text{Wenn } g(\vec{m},k)\neq 0 \text{ für alle } k<k_1, \text{ dann } \mathbf{\Theta}Z\underline{\vec{m}}\underline{0} =_\beta \mathbf{D}\underline{k_1}(\mathbf{\Theta}Z\underline{\vec{m}}\underline{k_1+1})(P\underline{\vec{m}}\underline{k_1}). \qquad (\star\star)$$

Hieraus ergibt sich:

$$\text{Wenn } k_1 \text{ kleinstes } k \text{ mit } g(\vec{m},k)=0, \text{ dann } \mathbf{\Theta}Z\underline{\vec{m}}\underline{0} =_\beta \underline{k_1}, \text{ da } P\underline{\vec{m}}\underline{k_1} =_\beta \underline{0}.$$

Beweis von $(\star\star)$ durch Induktion über k_1:

Für $k_1=0$: $\mathbf{\Theta}Z\underline{\vec{m}}\underline{0} =_\beta \mathbf{D}\underline{0}(\mathbf{\Theta}Z\underline{\vec{m}}\underline{1})(P\underline{\vec{m}}\underline{0})$ mit $(\star)$

Für $k_1>0$:

$$\begin{aligned}\mathbf{\Theta}Z\underline{\vec{m}}\underline{0} &=_\beta \mathbf{D}\underline{k_1-1}(\mathbf{\Theta}Z\underline{\vec{m}}\underline{k_1})\underbrace{(P\underline{\vec{m}}\underline{k_1-1})}\quad \text{(Induktionsvoraussetzung)}\\ &\qquad\qquad =_\beta \underline{l+1} \text{ für ein } l\geq 0, \text{ da } g(\vec{m},k_1-1)\neq 0;\\ &\qquad\qquad \text{also } P\underline{\vec{m}}\underline{k_1-1}\neq_\beta \underline{0}\\ &=_\beta \mathbf{\Theta}Z\underline{\vec{m}}\underline{k_1}\\ &=_\beta \mathbf{D}\underline{k_1}(\mathbf{\Theta}Z\underline{\vec{m}}\underline{k_1+1})(P\underline{\vec{m}}\underline{k_1}) \quad \text{mit } (\star)\end{aligned}$$

Es bleibt zu zeigen:

> Wenn $f(\vec{m})$ undefiniert ist, d. h. wenn $g(\vec{m},k)\neq 0$ für alle k bei gegebenem $\vec{m}$, dann hat $\mathbf{\Theta}Z\underline{\vec{m}}\underline{0}$ keine β-Normalform.

Mit $(\star)$ gilt

$$\mathbf{\Theta}Z\underline{\vec{m}}\underline{k_1} =_\beta \mathbf{D}\underline{k_1}(\mathbf{\Theta}Z\underline{\vec{m}}\underline{k_1+1})(P\underline{\vec{m}}\underline{k_1})$$

und da wir $\mathbf{\Theta}$ (und nicht $\mathbf{\Upsilon}$) als Fixpunktkombinator gewählt haben, gilt sogar (vgl. Bemerkung (1) zu Korollar 1.23):

$$\mathbf{\Theta}Z\underline{\vec{m}}\underline{k_1} \rhd_\beta \mathbf{D}\underline{k_1}(\mathbf{\Theta}Z\underline{\vec{m}}\underline{k_1+1})(P\underline{\vec{m}}\underline{k_1})$$

Nach Voraussetzung ist $P\underline{\vec{m}}\underline{k}\neq_\beta \underline{0}$ für jedes $\underline{k}$. Somit gilt:

$$\begin{aligned}\mathbf{\Theta}Z\underline{\vec{m}}\underline{0} &\rhd_\beta \mathbf{D}\underline{0}(\mathbf{\Theta}Z\underline{\vec{m}}\underline{1})(P\underline{\vec{m}}\underline{0}) \rhd_\beta \mathbf{\Theta}Z\underline{\vec{m}}\underline{1}\\ &\rhd_\beta \mathbf{D}\underline{1}(\mathbf{\Theta}Z\underline{\vec{m}}\underline{2})(P\underline{\vec{m}}\underline{1}) \rhd_\beta \mathbf{\Theta}Z\underline{\vec{m}}\underline{2}\end{aligned}$$

$$\rhd_\beta \mathbf{D}\underline{2}(\mathbf{\Theta} Z\,\vec{\underline{m}}\,\underline{3})(P\,\vec{\underline{m}}\,\underline{2}) \rhd_\beta \mathbf{\Theta} Z\,\vec{\underline{m}}\,\underline{3}$$
$$\rhd_\beta \cdots$$

Diese Reduktionsfolge ist QL (*quasi-leftmost*), d. h. es kommt immer wieder vor, dass ein linkester Term kontrahiert wird, nämlich ein Term der Form $\mathbf{D}MN\,\underline{l+1}$. Wir haben also eine nicht-terminierende QL-Reduktionsfolge. Also hat $\mathbf{\Theta} Z\,\vec{\underline{m}}\,\underline{0}$ keine β-Normalform (vgl. Theorem 1.21). QED

Theorem 1.35 *Jede λ-definierbare Funktion ist partiell-rekursiv.*

Beweisskizze. Sei f eine n-stellige Funktion, die durch P λ-definiert ist. Dann gilt:

$f(k_1,\ldots,k_n) =$ dasjenige k, für das die Gleichung $P\,\underline{k_1}\ldots\underline{k_n} = \underline{k}$ die Endformel der kürzesten Ableitung in $\lambda\beta$ ist (siehe folgender Abschnitt), die mit einer Formel der Gestalt $P\,\underline{k_1}\ldots\underline{k_n} = \underline{m}$ endet, falls es eine solche Ableitung in $\lambda\beta$ gibt.

Andernfalls ist $f(k_1,\ldots,k_n)$ undefiniert.

Nach geeigneter Gödelisierung (siehe Bemerkung auf S. 33) erweist sich f als partiell-rekursive Funktion. QED

1.3 Die formalen Theorien $\lambda\beta$ und $\lambda\beta\eta$

Bisher haben wir den λ-Kalkül auf der Grundlage der durch β-Kontraktion, α-Konversion und ggf. η-Kontraktion gegebenen operationellen Semantik von λ-Termen behandelt. Nun betrachten wir formale Systeme, die Entsprechendes leisten.

Definition 1.36 Wir definieren die *formalen Theorien* $\lambda\beta$ und $\lambda\beta\eta$, deren *Formeln* alle Gleichungen der Form $M = N$ für λ-Terme M, N sind. *formale Theorien $\lambda\beta$ und $\lambda\beta\eta$*

Axiome von $\lambda\beta$ sind alle Instanzen der folgenden *Axiomenschemata*:

(α) $\lambda x.M = \lambda y.M[y/x]$, falls $y \notin \mathrm{FV}(M)$

(β) $(\lambda x.M)N = M[N/x]$

(ρ) $M = M$ (Reflexivität)

Bei $\lambda\beta\eta$ zusätzlich:

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

Die *Regeln* sind:

(σ) $\dfrac{M = N}{N = M}$ (Symmetrie)

(τ) $\dfrac{M = N \quad N = P}{M = P}$ (Transitivität)

(μ) $\dfrac{N = N'}{MN = MN'}$

$$(\nu)\ \frac{M = M'}{MN = M'N}$$

$$(\xi)\ \frac{M = M'}{\lambda x.M = \lambda x.M'} \qquad \text{(schwache Extensionalität)}$$

(Die über dem Regelstrich stehenden Formeln heißen *Prämissen*, die darunter stehende Formel heißt *Konklusion*.)

Sei Γ eine Menge von Formeln und A eine Formel. Eine *Ableitung* von A aus Γ in $\lambda\beta$ oder $\lambda\beta\eta$ ist ein (nach oben verzweigender) Baum, *Ableitung*

1. dessen Blätter Axiome von $\lambda\beta$, bzw. $\lambda\beta\eta$, oder Formeln aus Γ sind,
2. dessen übrige Knoten Formeln sind, die mit Regeln aus den jeweils unmittelbar darüber stehenden Formeln abgeleitet wurden,
3. und dessen Wurzelknoten A ist.

Die Formeln in Γ bezeichnet man auch als *Annahmen*, die Formel A als *Endformel*.

Ist $\mathcal{T}$ eine formale Theorie, dann heißt

$$\mathcal{T}, \Gamma \vdash A$$

dass A in $\mathcal{T}$ aus Annahmen Γ ableitbar ist. Ist $\Gamma = \emptyset$, so nennt man die Ableitung von A in $\mathcal{T}$ auch *Beweis* (von A in $\mathcal{T}$), d. h. *Beweis*

$\lambda\beta \vdash M = N$ heißt, dass $M = N$ in $\lambda\beta$ beweisbar ist.

$\lambda\beta\eta \vdash M = N$ heißt, dass $M = N$ in $\lambda\beta\eta$ beweisbar ist.

Die beiden *formalen Theorien* $\lambda\beta_{\triangleright}$ und $\lambda\beta\eta_{\triangleright}$ sind definiert als die Systeme $\lambda\beta$ bzw. $\lambda\beta\eta$ ohne die Regel (σ), d. h. ohne Symmetrie. *formale Theorien* $\lambda\beta_{\triangleright}$ *und* $\lambda\beta\eta_{\triangleright}$

$\lambda\beta_{\triangleright} \vdash M = N$ heißt, dass $M = N$ in $\lambda\beta_{\triangleright}$ beweisbar ist.

$\lambda\beta\eta_{\triangleright} \vdash M = N$ heißt, dass $M = N$ in $\lambda\beta\eta_{\triangleright}$ beweisbar ist.

Beispiel. Die Ableitung

$$\dfrac{\dfrac{\dfrac{(\beta)\ \overline{(\lambda y.yx)z = zx}}{(\xi)\ \overline{\lambda x.(\lambda y.yx)z = \lambda x.zx}}}{(\nu)\ \overline{(\lambda x.(\lambda y.yx)z)v = (\lambda x.zx)v}} \qquad (\beta)\ \overline{(\lambda x.zx)v = zv}}{(\tau)\ \overline{(\lambda x.(\lambda y.yx)z)v = zv}}$$

ist ein Beweis von $(\lambda x.(\lambda y.yx)z)v = zv$ in jedem der eben eingeführten Systeme.

Lemma 1.37

1. $M \rhd_{\beta} N \iff \lambda\beta_{\triangleright} \vdash M = N$.
2. $M \rhd_{\beta\eta} N \iff \lambda\beta\eta_{\triangleright} \vdash M = N$.

3. $M =_\beta N \iff \lambda\beta \vdash M = N$.

4. $M =_{\beta\eta} N \iff \lambda\beta\eta \vdash M = N$.

Beweis. Übung. QED

Bemerkung. Die Relationen auf der linken Seite beruhen auf der operationellen Semantik des λ-Kalküls. Die Beweisbarkeitsbehauptungen auf der rechten Seite beruhen auf der jeweiligen formalen Theorie. Das Lemma drückt somit aus, dass die jeweiligen formalen Theorien für die jeweilige operationelle Semantik sowohl korrekt ("$\Longleftarrow$") als auch vollständig ("$\Longrightarrow$") sind.

Lemma 1.38 *Ersetzt man in der Definition von $\lambda\beta\eta$ das Axiom (η) durch die Regel*

$$(\chi)\ \frac{MP = NP \text{ für alle } P}{M = N}$$

oder durch die Regel

$$(\zeta)\ \frac{Mx = Nx}{M = N}\ \textit{falls } x \notin \mathrm{FV}(MN)$$

so sind in den resultierenden Systemen jeweils dieselben Gleichungen wie vorher ableitbar.

Bemerkung. Bei (χ) handelt es sich um eine sogenannte ω-Regel, d. h. eine Regel mit unendlich vielen Prämissen. Wir beschränken uns hier auf die Betrachtung der Regel (ζ). Beide Regeln drücken (ebenso wie das Axiomenschema (η)) *Extensionalität* von $=$ aus. Für die Gleichheit von Funktionen f und g bedeutet Extensionalität:

$$\text{Für alle } x\text{: Wenn } f(x) = g(x), \text{ dann } f = g.$$

Für die Gleichheit von Programmen würde man Extensionalität hingegen nicht fordern, sondern Gleichheit *intensional* verstehen. Die Theorie $\lambda\beta$ (und entsprechend $\lambda\beta_\triangleright$) ist ebenfalls intensional, d. h. es gilt *nicht*:

$$\text{Für alle Terme } T\text{: Wenn } \lambda\beta \vdash MT = NT, \text{ dann } \lambda\beta \vdash M = N.$$

(Ein Gegenbeispiel liefern die Terme $M \doteq y$ und $N \doteq \lambda x.yx$.)
Die Systeme mit (η), (χ) oder (ζ) sind dagegen extensional.

Beweis von Lemma 1.38.
"$(\eta) \implies (\zeta)$":

$$\dfrac{\dfrac{(\eta)\ \dfrac{}{\lambda x.Mx = M}}{(\sigma)\ \dfrac{M = \lambda x.Mx}{(\tau)\ \dfrac{M = \lambda x.Nx}{M = N}}} \quad (\xi)\ \dfrac{Mx = Nx}{\lambda x.Mx = \lambda x.Nx} \quad (\eta)\ \dfrac{}{\lambda x.Nx = N}}{}$$

Anwendungen von (ζ) können also stets durch eine Ableitung dieser Form ersetzt werden.

"$(\zeta) \implies (\eta)$":

$$\dfrac{\dfrac{}{(\lambda x.Mx)x = Mx}\,(\beta)}{\lambda x.Mx = M}\,(\zeta)$$

Die Verwendung von (η) kann also stets durch eine Ableitung dieser Form ersetzt werden. QED

1.4 Unentscheidbarkeitsresultate

Theorem 1.39 (Church, 1936b)
Die Menge $NF_\beta := \{M \mid M \text{ hat eine } \beta\text{-Normalform}\}$ ist nicht entscheidbar.

Beweisskizze. Wir können die einstelligen partiell-rekursiven Funktionen so abzählen, $f_1, f_2, \ldots$, dass die Funktion u mit $u(m,n) :\simeq f_m(n)$ partiell-rekursiv ist. Nun werde u durch P λ-definiert. Dann gilt:

$$P\,\underline{m}\,\underline{n} \text{ hat eine } \beta\text{-Normalform} \iff u(m,n) \text{ ist definiert.}$$

Wäre NF_β entscheidbar, wäre g mit

$$g(n) := \begin{cases} u(n,n)+1 & \text{falls } u(n,n) \text{ definiert} \\ 1 & \text{sonst} \end{cases}$$

eine total-rekursive Funktion. Damit wäre $g = f_k$ für ein k, also

$$u(k,k) = f_k(k) = g(k) = u(k,k)+1$$

da f_k total. Widerspruch. QED

Bemerkung. Im Folgenden setzen wir voraus, dass λ-Termen natürliche Zahlen so zugeordnet werden können, dass verschiedenen Termen stets verschiedene Zahlen entsprechen. Eine solche Zuordnung heißt *Gödelisierung* (wobei wir uns hier nicht für eine konkrete derartige Zuordnung interessieren). Die einem Term M zugeordnete Zahl heißt *Gödelnummer* von M, die wir als $\ulcorner M \urcorner$ notieren. Es ist dann $\underline{\ulcorner M \urcorner}$ die der Gödelnummer $\ulcorner M \urcorner$ des Terms M entsprechende Church-Ziffer, d. h. ein λ-Term.

Theorem 1.40 (Church, 1936b) *Die Relation $=_\beta$ ist nicht entscheidbar.*

Beweisskizze. Die zu einem Term β-gleichen Terme lassen sich rekursiv aufzählen (z. B. mit Lemma 1.37 (3) über Ableitungen der entsprechenden Gleichungen in $\lambda\beta$). Sei

$f(m,k) :=$ Gödelnummer des k-ten Terms, der zum Term mit Gödelnummer m β-gleich ist;

$$h(m) := \begin{cases} 0 & \text{falls } m \text{ eine Gödelnummer eines Terms in } \beta\text{-Normalform ist,} \\ 1 & \text{sonst.} \end{cases}$$

Die Funktionen f und h sind primitiv-rekursiv. Sie seien durch die Terme F und H λ-definiert.
Betrachte die Gleichung

$$Mxy =_\beta \mathbf{D}\underline{1}(Mx(\mathbf{N}y))(H(Fxy))$$

Eine Lösung dieser Gleichung ist nach Korollar 1.23:

$$\Upsilon(\lambda uxy.\underbrace{\mathbf{D}\underline{1}(ux(\mathbf{N}y))(H(Fxy))}_{\doteq: V})$$

Es gilt dann:

1. $(\Upsilon(\lambda uxy.V))\underline{m}\,\underline{0} =_\beta \underline{1}$, falls m Gödelnummer eines Terms ist, der zu einem Term in β-Normalform β-gleich ist.
2. Andernfalls hat $(\Upsilon(\lambda uxy.V))\,\underline{m}\,\underline{0}$ keine β-Normalform.

Falls nun $=_\beta$ entscheidbar wäre, dann wäre $(\Upsilon(\lambda uxy.V))\ulcorner\underline{M}\urcorner\,\underline{0} =_\beta \underline{1}$ entscheidbar. Also wäre NF_β entscheidbar, im Widerspruch zu Theorem 1.39. QED

Theorem 1.41 (Church, 1936a)
Die Prädikatenlogik erster Stufe PL ist nicht entscheidbar.

Beweisskizze. Die Relation $=_\beta$ ist nicht entscheidbar. Folglich ist (aufgrund von Lemma 1.37, 3) auch die formale Theorie $\lambda\beta$ nicht entscheidbar. Nun gilt:

$$\lambda\beta \vdash M = N \text{ genau dann, wenn } PL \vdash (F_1 \wedge \ldots \wedge F_8) \to E(\overline{\overline{\ulcorner M\urcorner}}, \overline{\overline{\ulcorner N\urcorner}})$$

Hierbei sei E ein ausgezeichnetes zweistelliges Prädikat und

$$\begin{aligned}
\overline{\overline{0}} &:\doteq z\\
\overline{\overline{1}} &:\doteq f(z)\\
\overline{\overline{2}} &:\doteq f(f(z))\\
&\vdots
\end{aligned}$$

für ausgezeichnete z und f, die in der Sprache von PL vorhanden sind.
Die formale Theorie $\lambda\beta$ umfasst 8 Regeln bzw. Axiomenschemata. Die PL-Formeln $F_1, \ldots, F_8$ seien deren prädikatenlogische Übersetzungen durch Gödelisierung:

1. Die Regel (σ) wird in die PL-Formel $E(\overline{\overline{\ulcorner M\urcorner}}, \overline{\overline{\ulcorner N\urcorner}}) \to E(\overline{\overline{\ulcorner N\urcorner}}, \overline{\overline{\ulcorner M\urcorner}})$ übersetzt.
2. Die Regel (τ) wird in die PL-Formel $E(\overline{\overline{\ulcorner M\urcorner}}, \overline{\overline{\ulcorner N\urcorner}}) \wedge E(\overline{\overline{\ulcorner N\urcorner}}, \overline{\overline{\ulcorner P\urcorner}}) \to E(\overline{\overline{\ulcorner M\urcorner}}, \overline{\overline{\ulcorner P\urcorner}})$ übersetzt.

Und so weiter für die restlichen Regeln und Axiomenschemata.

Wäre die Prädikatenlogik entscheidbar, so wäre damit auch $\lambda\beta$ (bzw. $=_\beta$) entscheidbar, im Widerspruch zu Theorem 1.40. QED

2 Kombinatorische Logik

Die Kombinatorische Logik (kurz: CL) soll dasselbe leisten wie der λ-Kalkül, allerdings ohne die Verwendung von gebundenen Variablen. Dadurch werden Substitutionen vereinfacht, und α-Konversion wird nicht benötigt. Im Vergleich zur λ-Notation führt dies jedoch zu schwerer lesbaren Termen.
Zur Motivation des Verzichts auf gebundene Variablen betrachten wir das Kommutativgesetz für die Addition:

$$\text{für alle } x, y\text{: } x + y = y + x$$

in dem die Variablen x und y gebunden vorkommen. Zur Vermeidung dieser Variablenbindung führen wir zunächst einen Additionsoperator A ein:

$$A(x, y) = x + y \qquad (\text{für alle } x, y)$$

und definieren dann einen Operator **C** durch

$$(\mathbf{C}(f))(x, y) = f(y, x) \qquad (\text{für alle } f, x, y)$$

Das Kommutativgesetz kann nun so angegeben werden:

$$A = \mathbf{C}(A)$$

wobei zumindest unmittelbar keine gebundenen Variablen mehr vorkommen.
Der Operator **C** ist ein Beispiel für einen Kombinator. Weitere Beispiele sind:

B	$(\mathbf{B}(f, g))(x) = f(g(x))$	Komposition zweier Funktionen
B′	$(\mathbf{B}'(f, g))(x) = g(f(x))$	reverse Komposition zweier Funktionen
S	$(\mathbf{S}(f, g))(x) = f(x, g(x))$	(stärkere) Komposition zweier Funktionen
I	$\mathbf{I}(f) = f$	Identität
K	$(\mathbf{K}(c))(x) = c$	bildet konstante Funktionen

2.1 Syntax und operationelle Semantik

Gegeben sei eine unendliche Folge von Variablen mit fester Reihenfolge. Der Einfachheit halber seien die Variablen der Kombinatorischen Logik dieselben wie die des λ-Kalküls. Im Folgenden seien **K** und **S** vorgegebene Konstanten. Wenn außer diesen noch weitere Konstanten hinzukommen, heißt das System *angewandt* (sonst *rein*). Wir betrachten nur die reine Kombinatorische Logik.

Definition 2.1 *CL-Terme* sind wie folgt definiert: *CL-Terme*

1. Alle Variablen und Konstanten sind CL-Terme (*Atome*). *Atome*
2. Sind X und Y CL-Terme, dann ist auch (XY) ein CL-Term (*Applikation*), der X und Y als unmittelbare Teilterme enthält. *Applikation*

Weitere Begriffe:

1. Ein *geschlossener* CL-Term enthält keine Variablen. *geschlossen*
2. Ein *Kombinator* enthält nur **K** und **S** als Atome. *Kombinator*
3. *Metasprachliche Variablen* (ggf. auch mit Indizes):
   (a) für CL-Terme: $U, V, W, X, Y, Z, \ldots$
   (b) für Atome: $a, b, c, \ldots$
4. $\mathrm{FV}(X)$ sei die Menge der Variablen in X.
5. *Klammerersparnis*: *Klammerersparnis*
   (a) Außenklammern können wegfallen.
   (b) Wir verwenden wieder Linksklammerung, d. h. $UVWX$ steht für $((UV)W)X$.
6. $X \doteq Y$ bezeichne wieder die *syntaktische Identität* von X und Y.
7. Für die *Länge* eines Terms und für (*echte*) *Teilterme* gilt das für λ-Terme Gesagte.

Beispiele.
1. $\mathbf{S}xy(\mathbf{K}y)(\mathbf{KKSS})$ ist ein CL-Term.
2. $\mathbf{S}(\mathbf{KS})$ ist ein CL-Term (und ein Kombinator).

Definition 2.2 Da in CL-Termen keine gebundenen Variablen vorkommen können, ist *Substitution* $Y[U/x]$ in offensichtlicher Weise definiert wie folgt: *Substitution*

1. $x[U/x] :\doteq U$,
2. $a[U/x] :\doteq a$, falls $x \neq a$,
3. $(VW)[U/x] :\doteq (V[U/x]W[U/x])$.

Definition 2.3
1. Ein Term der Form $\mathbf{K}XY$ oder $\mathbf{S}XYZ$ heißt *schwaches Redex* (kurz: *Redex*). *schwaches Redex*
2. Die Operation der *schwachen Kontraktion* ist definiert durch: *schwache Kontraktion*

$$U\big[\mathbf{K}XY\big] \rhd_{1w} U\big[X\big]$$

$$U\big[\mathbf{S}XYZ\big] \rhd_{1w} U\big[XZ(YZ)\big]$$

   ("w" für *weak*).
3. Kann ein Term X durch eine (möglicherweise leere) endliche Folge von schwachen Kontraktionen in einen Term Y überführt werden, d. h. gilt

$$X \doteq V_1 \rhd_{1w} V_2 \rhd_{1w} \cdots \rhd_{1w} V_n \doteq Y$$

   so sagt man, dass X zu Y *schwach reduziert*. Notation: $X \rhd_w Y$. *schwache Reduktion*

4. Zwei Terme X und Y heißen *schwach gleich* (oder *schwach konvertibel*), falls gilt: *schwache Gleichheit* *schwache Konversion*

$$X \eqsim V_1 \overset{\triangleright_{1w}}{\underset{\triangleleft_{1w}}{}} V_2 \overset{\triangleright_{1w}}{\underset{\triangleleft_{1w}}{}} \cdots \overset{\triangleright_{1w}}{\underset{\triangleleft_{1w}}{}} V_n \eqsim Y$$

   (wobei $U \overset{\triangleright_{1w}}{\underset{\triangleleft_{1w}}{}} W$ für "$U \triangleright_{1w} W$ oder $W \triangleright_{1w} U$" steht). Notation: $X =_w Y$.

5. Für eine (möglicherweise leere) endliche oder unendliche Folge von schwachen Kontraktionen

$$X \eqsim X_1 \triangleright_{1w} X_2 \triangleright_{1w} X_3 \triangleright_{1w} \cdots$$

   heißt $(X_1, X_2, X_3, \ldots)$, bzw. die angegebene Folge selbst, *schwache Reduktionsfolge* von X. *schwache Reduktionsfolge*

Definition 2.4

1. X ist in *schwacher Normalform* (kurz: *schwache NF*), falls X kein schwaches Redex enthält. *schwache Normalform*
2. Wenn $X \triangleright_w Y$ gilt, und Y in schwacher Normalform ist, dann heißt Y *schwache Normalform von* X. (Wir sagen dann auch, dass X die schwache Normalform Y *hat*.)
3. X heißt *(schwach) normalisierbar*, falls es eine schwache Normalform von X gibt. *normalisierbar*
4. X heißt *stark normalisierbar*, falls es keine unendliche schwache Reduktionsfolge von X gibt. *stark normalisierbar*

Beispiele.

1. Sei $\mathbf{B} :\eqsim \mathbf{S}(\mathbf{KS})\mathbf{K}$. Dann gilt $\mathbf{B}XYZ \triangleright_w X(YZ)$:

$$\begin{aligned}
\mathbf{S}(\mathbf{KS})\mathbf{K}XYZ &\triangleright_w \mathbf{KS}X(\mathbf{K}X)YZ && \text{mit } \mathbf{S}(\mathbf{KS})\mathbf{K}X \triangleright_{1w} \mathbf{KS}X(\mathbf{K}X)\\
&\triangleright_w \mathbf{S}(\mathbf{K}X)YZ && \text{mit } \mathbf{KS}X \triangleright_{1w} \mathbf{S}\\
&\triangleright_w \mathbf{K}XZ(YZ) && \text{mit } \mathbf{S}(\mathbf{K}X)YZ \triangleright_{1w} \mathbf{K}XZ(YZ)\\
&\triangleright_w X(YZ) && \text{mit } \mathbf{K}XZ \triangleright_{1w} X
\end{aligned}$$

   (Vergleiche den eingangs erwähnten Operator **B**.)

2. Es gilt $\mathbf{SKK}X \triangleright_w X$, d. h. **SKK** verhält sich wie der eingangs erwähnte Identitätsoperator **I**.

   Wir setzen $\mathbf{I} :\eqsim \mathbf{SKK}$.

Bemerkungen.

1. Schwache Reduktion $\triangleright_w$ ist invariant unter Substitution (vgl. Lemma 1.10), d. h. es gilt: Wenn $X \triangleright_w Y$ und $U \triangleright_w V$, dann $U[X/z] \triangleright_w V[Y/z]$.
2. Es gilt Church-Rosser (vgl. Theorem 1.13):
   (a) Wenn $X \triangleright_w U$ und $X \triangleright_w V$, dann existiert ein Term T, so dass $U \triangleright_w T$ und $V \triangleright_w T$.
   (b) Wenn $X =_w Y$, dann existiert ein Term T, so dass $X \triangleright_w T$ und $Y \triangleright_w T$.

2.2 Die formale Theorie CLw

Definition 2.5 Wir definieren die *formale Theorie CLw*, deren *Formeln* alle Gleichungen der Form $X = Y$ für CL-Terme X, Y sind. *formale Theorie CLw*

Axiome von CLw sind alle Instanzen der folgenden *Axiomenschemata*:

(K) $\mathbf{K}XY = X$

(S) $\mathbf{S}XYZ = XZ(YZ)$

(ρ) $X = X$ (Reflexivität)

Die *Regeln* sind:

(σ) $\dfrac{X = Y}{Y = X}$ (Symmetrie)

(τ) $\dfrac{X = Y \quad Y = Z}{X = Z}$ (Transitivität)

(μ) $\dfrac{X = X'}{YX = YX'}$

(ν) $\dfrac{Y = Y'}{YX = Y'X}$

Ableitung und *Beweis* sind analog zu Definition 1.36 definiert, und

CL$w \vdash X = Y$ heißt, dass $X = Y$ in CLw beweisbar ist,

CL$w_{\triangleright} \vdash X = Y$ heißt, dass $X = Y$ in CLw ohne die Regel (σ) beweisbar ist.

Lemma 2.6

1. *$X =_w Y$ genau dann, wenn CL$w \vdash X = Y$.*
2. *$X \triangleright_w Y$ genau dann, wenn CL$w_{\triangleright} \vdash X = Y$.*

Beweis. Übung. QED

2.3 Verhältnis zwischen λ-Kalkül und CL

Um das Verhältnis zwischen λ-Kalkül und CL zu untersuchen, betrachten wir Übersetzungen von CL-Termen in λ-Terme und umgekehrt.

Definition 2.7 Für einen CL-Term X ist der λ-Term X_λ wie folgt definiert:

1. $x_\lambda :\mathrel{\widehat{=}} x$
2. $\mathbf{K}_\lambda :\mathrel{\widehat{=}} \lambda xy.x$
3. $\mathbf{S}_\lambda :\mathrel{\widehat{=}} \lambda xyz.xz(yz)$
4. $(XY)_\lambda :\mathrel{\widehat{=}} X_\lambda Y_\lambda$

(Wir identifizieren dabei kongruente λ-Terme; d. h. es ist $M \mathrel{\widehat{=}} N$, falls $M \equiv_\alpha N$, für λ-Terme M, N.)

Lemma 2.8

1. *Wenn* $X \rhd_w Y$, *dann* $X_\lambda \rhd_\beta Y_\lambda$.
2. *Wenn* $X =_w Y$, *dann* $X_\lambda =_\beta Y_\lambda$.

Beweis. Benutze CL$w_\rhd$ und $\lambda\beta_\rhd$, bzw. CLw und $\lambda\beta$. QED

Bemerkung. Die Umkehrung gilt *nicht*. Es ist z. B. $\mathbf{S}_\lambda\mathbf{K}_\lambda =_\beta \mathbf{K}_\lambda(\mathbf{S}_\lambda\mathbf{K}_\lambda\mathbf{K}_\lambda)$, nicht jedoch $\mathbf{SK} =_w \mathbf{K}(\mathbf{SKK})$. Keiner der CL-Terme kontrahiert, während beide λ-Terme Redexe enthalten.

Definition 2.9 Für einen λ-Term M ist der CL-Term M_{CL} wie folgt definiert:

1. $x_{\mathrm{CL}} :\doteq x$
2. $(MN)_{\mathrm{CL}} :\doteq M_{\mathrm{CL}}N_{\mathrm{CL}}$
3. $(\lambda x.M)_{\mathrm{CL}} :\doteq [x].M_{\mathrm{CL}}$

wobei $[x].Y$ ("Abstraktion x Punkt Y") für CL-Terme Y wie folgt definiert ist:

1. $[x].x :\doteq \mathbf{SKK}$ (abgekürzt: $\mathbf{I} :\doteq \mathbf{SKK}$);
2. $[x].Y :\doteq \mathbf{K}Y$, falls $x \notin \mathrm{FV}(Y)$;
3. $[x].Ux :\doteq U$, falls $x \notin \mathrm{FV}(U)$;
4. $[x].(UV) :\doteq \mathbf{S}([x].U)([x].V)$, falls die vorherigen Fälle nicht zutreffen.

Beispiel. Es ist $[x].xxz \doteq \mathbf{S}([x].xx)([x].z) \doteq \mathbf{S}(\mathbf{S}([x].x)([x].x))(\mathbf{K}z) \doteq \mathbf{S}(\mathbf{SII})(\mathbf{K}z)$.

Bemerkungen.

1. Bei $[x].Y$ handelt es sich um eine metasprachliche Operation, d. h. ein Ausdruck $[x].Y$ ist kein CL-Term, sondern *repräsentiert* nur einen CL-Term.
2. Es ist $x \notin \mathrm{FV}([x].Y)$. Insofern verhält sich $[x]$ wie ein variablenbindender Operator.

Lemma 2.10 *Es gilt* $([x].Y)Z \rhd_w Y[Z/x]$.

Beweis. Induktion über der Struktur von Y:

1. $Y \doteq x$: $([x].x)Z \doteq \mathbf{I}Z \rhd_w Z \doteq x[Z/x]$
2. Y ist ein Atom und $Y \not\doteq x$: $([x].Y)Z \doteq \mathbf{K}YZ \rhd_w Y \doteq Y[Z/x]$
3. $Y \doteq (UV)$:
   (a) $x \notin \mathrm{FV}(Y)$: $([x].Y)Z \doteq \mathbf{K}YZ \rhd_w Y \doteq Y[Z/x]$
   (b) $x \notin \mathrm{FV}(U)$ und $V \doteq x$: $([x].Y)Z \doteq UZ \doteq Ux[Z/x]$
   (c) keiner der vorherigen Fälle:

$$\begin{aligned}([x].Y)Z &\doteq \mathbf{S}([x].U)([x].V)Z \\ &\rhd_w ([x].U)Z(([x].V)Z) \\ &\rhd_w (U[Z/x])(V[Z/x]) \qquad \text{(Induktionsvoraussetzung)} \\ &\doteq Y[Z/x]\end{aligned}$$

QED

Korollar 2.11 (Kombinatorische Vollständigkeit)
Sei V ein Term mit $\{x_1, \ldots, x_n\} \subseteq \mathrm{FV}(V)$. Dann gibt es einen Term U, in dem $x_1, \ldots, x_n$ nicht vorkommen, so dass $UX_1 \ldots X_n \rhd_w V[X_1/x_1] \ldots [X_n/x_n]$.

Beweis. Setze $U :\doteq [x_1].\ldots.[x_n].V$. QED

Bemerkung. Damit kann man jeden Kombinator U, der durch eine "neue" Kontraktion $UX_1 \ldots X_n \rhd_w W$ gegeben ist, wobei W nur aus $X_1, \ldots, X_n$ zusammengesetzt ist, in CL durch einen variablenfreien Term definieren. Mit Hilfe von **S** und **K** lassen sich also "alle" Kombinatoren ausdrücken.

Lemma 2.12
1. *Für CL-Terme X gilt: $(X_\lambda)_{CL} \doteq X$.*
2. *Für λ-Terme M gilt: $(M_{CL})_\lambda =_{\beta\eta} M$.*

Beweis.
1. Induktion über der Struktur von X:
   (a) $(x_\lambda)_{\mathrm{CL}} \doteq x_{\mathrm{CL}} \doteq x$
   (b) $(\mathbf{K}_\lambda)_{\mathrm{CL}} \doteq (\lambda xy.x)_{\mathrm{CL}} \doteq [x].([y].x) \doteq [x].\mathbf{K}x \doteq \mathbf{K}$
   (c) $(\mathbf{S}_\lambda)_{\mathrm{CL}} \doteq (\lambda xyz.xz(yz))_{\mathrm{CL}}$
   $\doteq [x].([y].([z].xz(yz)))$
   $\doteq [x].([y].\mathbf{S}([z].xz)([z].yz))$
   $\doteq [x].([y].\mathbf{S}xy)$
   $\doteq [x].\mathbf{S}x$
   $\doteq \mathbf{S}$
   (d) $((UV)_\lambda)_{\mathrm{CL}} \doteq (U_\lambda V_\lambda)_{\mathrm{CL}} \doteq (U_\lambda)_{\mathrm{CL}}(V_\lambda)_{\mathrm{CL}} \doteq UV$ (Letzteres nach IV)
2. Induktion über der Struktur von M:
   (a) $M \doteq x$: $(x_{\mathrm{CL}})_\lambda \doteq x_\lambda \doteq x$
   (b) $M \doteq (PQ)$:
   $((PQ)_{\mathrm{CL}})_\lambda \doteq (P_{\mathrm{CL}}Q_{\mathrm{CL}})_\lambda \doteq (P_{\mathrm{CL}})_\lambda(Q_{\mathrm{CL}})_\lambda =_{\beta\eta} PQ$ (Letzteres nach IV)
   (c) $M \doteq (\lambda x.M')$: Zu zeigen ist $((\lambda x.M')_{\mathrm{CL}})_\lambda \doteq ([x].(M')_{\mathrm{CL}})_\lambda =_{\beta\eta} \lambda x.M'$.
   Induktion über der Struktur von M':
      i. $([x].x_{\mathrm{CL}})_\lambda \doteq \mathbf{I}_\lambda \doteq \mathbf{S}_\lambda\mathbf{K}_\lambda\mathbf{K}_\lambda =_\beta \lambda x.x$
      ii. falls $x \notin \mathrm{FV}(P_{\mathrm{CL}})$:

$$\begin{aligned}([x].P_{\mathrm{CL}})_\lambda &\doteq (\mathbf{K}P_{\mathrm{CL}})_\lambda \\ &\doteq \mathbf{K}_\lambda(P_{\mathrm{CL}})_\lambda \\ &\doteq (\lambda xy.x)(P_{\mathrm{CL}})_\lambda \qquad \text{wobei } y \text{ neu} \\ &=_\beta \lambda y.(P_{\mathrm{CL}})_\lambda \\ &=_{\beta\eta} \lambda y.P \qquad \text{(Induktionsvoraussetzung)}\end{aligned}$$

iii. falls $x \notin \mathrm{FV}(P_{\mathrm{CL}})$ und $Q_{\mathrm{CL}} \doteq x$:

$$\begin{aligned}
([x].(PQ)_{\mathrm{CL}})_\lambda &\doteq ([x].(P_{\mathrm{CL}}Q_{\mathrm{CL}}))_\lambda \\
&\doteq ([x].(P_{\mathrm{CL}}x))_\lambda \\
&\doteq (P_{\mathrm{CL}})_\lambda \\
&=_{\beta\eta} P \qquad \text{(Induktionsvoraussetzung)} \\
&=_\eta \lambda x.(Px) \\
&\doteq \lambda x.(PQ)
\end{aligned}$$

iv. sonst: $$\begin{aligned}
([x].(PQ)_{\mathrm{CL}})_\lambda &\doteq (\mathbf{S}([x].P_{\mathrm{CL}})([x].Q_{\mathrm{CL}}))_\lambda \\
&\doteq \mathbf{S}_\lambda([x].P_{\mathrm{CL}})_\lambda([x].Q_{\mathrm{CL}})_\lambda \\
&=_{\beta\eta} \mathbf{S}_\lambda(\lambda x.P)(\lambda x.Q) \qquad \text{(Induktionsvoraussetzung)} \\
&\doteq (\lambda uvy.uy(vy))(\lambda x.P)(\lambda x.Q) \\
&=_\beta \lambda y.(\lambda x.P)y((\lambda x.Q)y) \\
&=_\beta \lambda x.(PQ)
\end{aligned}$$

QED

Korollar 2.13 *Es gilt also mit Lemma 2.8: Wenn $M_{CL} =_w N_{CL}$, dann $M =_{\beta\eta} N$.*

Bemerkung. Es gilt aber *nicht*: $(M_{\mathrm{CL}})_\lambda =_\beta M$.
Zum Beispiel ist $((\lambda x.yx)_{\mathrm{CL}})_\lambda \doteq ([x].yx)_\lambda \doteq y_\lambda \doteq y \neq_\beta \lambda x.yx$.

Unterschiede zwischen λ-Kalkül und CL

Keine der folgenden Behauptungen gilt:

$$\begin{aligned}
M_{\mathrm{CL}} \rhd_w N_{\mathrm{CL}} &\implies M \rhd_\beta N & \qquad M_{\mathrm{CL}} \rhd_w N_{\mathrm{CL}} &\impliedby M \rhd_\beta N \\
M_{\mathrm{CL}} =_w N_{\mathrm{CL}} &\implies M =_\beta N & \qquad M_{\mathrm{CL}} =_w N_{\mathrm{CL}} &\impliedby M =_\beta N
\end{aligned}$$

Man darf also zur Auswertung von λ-Termen M, N, wenn es nur um β-Reduktion geht, *nicht* so verfahren: $M \rightsquigarrow M_{\mathrm{CL}} \rhd_w N_{\mathrm{CL}} \rightsquigarrow N$.
Das Problem mit "$\implies$" besteht darin, dass

$$(\eta) \quad [x].Xx = X, \text{ falls } x \notin \mathrm{FV}(X)$$

in CLw gilt. Denn es ist

$$(\lambda x.Mx)_{\mathrm{CL}} \doteq [x].M_{\mathrm{CL}}x \doteq M_{\mathrm{CL}}, \text{ falls } x \notin \mathrm{FV}(M).$$

Das Problem mit "$\impliedby$" besteht darin, dass

$$(\xi) \; \frac{X = X'}{[x].X = [x].X'}$$

in CLw *nicht* gilt. Denn es ist

$$\begin{aligned}[x].\mathbf{S}xyz &\doteq \mathbf{S}([x].\mathbf{S}xy)([x].z)\\ &\doteq \mathbf{S}(\mathbf{S}([x].\mathbf{S}x)([x].y))(\mathbf{K}z)\\ &\doteq \mathbf{S}(\mathbf{SS}(\mathbf{K}y))(\mathbf{K}z)\end{aligned}$$

und

$$\begin{aligned}[x].xz(yz) &\doteq \mathbf{S}([x].xz)([x].yz)\\ &\doteq \mathbf{S}(\mathbf{S}([x].x)([x].z))(\mathbf{K}(yz))\\ &\doteq \mathbf{S}(\mathbf{SI}(\mathbf{K}z))(\mathbf{K}(yz))\end{aligned}$$

Also ist zwar $\mathbf{S}xyz =_w xz(yz)$, aber *nicht* $[x].\mathbf{S}xyz =_w [x].xz(yz)$.

Extensionalität

Die Hinzunahme von (ξ) zu CLw bewirkt *volle* Extensionalität (ζ):

$$\begin{array}{l}(\eta)\ \dfrac{}{[x].Xx = X}\\ (\sigma)\ \dfrac{}{X = [x].Xx}\\ (\tau)\ \dfrac{X = [x].Yx}{Xx = Yx}\end{array}\quad (\xi)\ \frac{Xx = Yx}{[x].Xx = [x].Yx}\qquad \begin{array}{l}\\ \\ (\eta)\ \dfrac{}{[x].Yx = Y}\\ (\tau)\ \dfrac{}{X = Y}\end{array}$$

Während also bei $\lambda\beta$ die Hinzunahme von (η) Extensionalität zur Folge hat, wohingegen (ξ) eo ipso gilt, bewirkt bei CLw die Hinzunahme von (ξ) Extensionalität, während (η) eo ipso gilt. Das zeigt die Disparatheit beider Systeme.

Starke Reduktion

Wir definieren *starke Reduktion* $\succ$ durch Erweiterung von CL$w_{\triangleright}$ um *starke Reduktion*

$$(\xi)\ \frac{X \succ Y}{[x].X \succ [x].Y}$$

Dann gilt für λ-Terme M, N:

$$M \triangleright_{\beta\eta} N \implies M_{\mathrm{CL}} \succ N_{\mathrm{CL}}$$

Die Umkehrung $M_{\mathrm{CL}} \succ N_{\mathrm{CL}} \implies M \triangleright_{\beta\eta} N$ gilt jedoch *nicht*, da aus $X \succ Y$ nicht $X_\lambda \triangleright_{\beta\eta} Y_\lambda$ folgt.
Es gilt lediglich

$$M_{\mathrm{CL}} \succ N_{\mathrm{CL}} \implies M =_{\beta\eta} N$$

Es ist η-Konversion jetzt ein Fall von (ρ) und gilt in beliebiger Richtung. Daraus ergibt sich für λ-Terme M, N:

$$M =_{\beta\eta} N \iff M_{\mathrm{CL}} \succ\!\prec N_{\mathrm{CL}}$$

wobei $\succ\!\!\prec$ der symmetrische Abschluss von $\succ$ ist. Für CL-Terme X, Y gilt damit:

$$X \succ\!\!\prec Y \iff X_\lambda =_{\beta\eta} Y_\lambda.$$

Abschwächung von $[x].Y$

Um β-Gleichheit (statt nur $\beta\eta$-Gleichheit) in CLw repräsentieren zu können, kann man die Definition von $[x].Y$ so abschwächen, dass (η) nicht automatisch gilt, z. B. durch Weglassen der Klausel

3. $[x].Ux :\doteq U$, falls $x \notin \mathrm{FV}(U)$

in der Definition von $[x].Y$.
Jedoch gilt selbst dann (ξ) *nicht*, da nun

$$[x].\mathbf{S}xyz \doteq \mathbf{S}(\mathbf{S}(\mathbf{S}(\mathbf{KS})\mathbf{I})(\mathbf{K}y))(\mathbf{K}z)$$

und (wie zuvor)

$$[x].xz(yz) \doteq \mathbf{S}(\mathbf{SI}(\mathbf{K}z))(\mathbf{K}(yz))$$

Es gilt also wieder *nicht* $[x].\mathbf{S}xyz =_w [x].xz(yz)$, obgleich $\mathbf{S}xyz =_w xz(yz)$ gilt.
Man kann mit dieser neuen Definition von $[x].Y$ jedoch zeigen, dass

$$\lambda\beta \vdash M = N \iff (\mathrm{CL}w + \otimes) \vdash M_{\mathrm{CL}} = N_{\mathrm{CL}}$$
$$(\mathrm{CL}w + \otimes) \vdash X = Y \iff \lambda\beta \vdash X_\lambda = Y_\lambda$$

wobei $\otimes$ eine Erweiterung von CLw um ein bestimmtes Regelschema bzw. eine endliche Menge von Axiomen ist (vgl. die formale Theorie CLζ_β in Hindley & Seldin (2008), Ch. 9).

Korrektheit

Bei der modifizierten Definition von $[x].Y$ gilt $(\mathbf{S}_\lambda)_{\mathrm{CL}} =_w \mathbf{S}$ *nicht*. Eine weitergehende Modifikation ist jedoch möglich, so dass statt Lemma 2.12 (für $=_{\beta\eta}$) jetzt für $=_\beta$ gilt:

1. *Für CL-Terme X:* $(X_\lambda)_{CL} \doteq X$.

2. *Für λ-Terme M:* $(M_{CL})_\lambda =_\beta M$.

Damit ist die Auswertung von übersetzten λ-Termen in CLw korrekt, was in der funktionalen Programmierung oft ausgenutzt wird.

Vollständigkeit

Bezüglich Vollständigkeit gilt Folgendes: Wir betrachten eine erweiterte Sprache, in der zusätzliche Funktionen ausgewertet werden können (sog. δ-Regeln). Die entsprechenden Reduktionen sind $\rhd_{1\beta\delta}$, $\rhd_{\beta\delta}$ und $\rhd_{l\beta\delta}$ (l für *leftmost*), bzw. $\rhd_{1w\delta}$ und $\rhd_{w\delta}$. Dann gilt:

Wenn $M \rhd_{1l\beta\delta} N$, mit M geschlossen und nicht von der Form $[x].P$, so ist $M_{\mathrm{CL}} \rhd_{1w\delta} N_{\mathrm{CL}}$.

Falls ein getyptes System 2. Stufe mit Int, Bool, Char als Grundtypen gegeben ist, in dem ein Fixpunktoperator Υ existiert und M vom Grundtyp ist, dann gilt:

$$M \rhd_{l\beta\delta} N \implies M_{\mathrm{CL}} \rhd_{w\delta} N_{\mathrm{CL}}$$

Das bedeutet, dass man *alles*, was man in $\lambda\beta$ finden kann, durch Übersetzung in CLw finden kann. Dies wird zum Beispiel in der funktionalen Programmiersprache Miranda ausgenutzt (siehe Turner, 1979).
Es ergibt sich daraus für eine Konstante c eines Grundtyps (die per Definition in β-Normalform ist):

$$\begin{aligned} \lambda\beta\delta \vdash M = c &\implies M =_{\beta\delta} c \\ &\implies M \rhd_{l\beta\delta} c \\ &\implies M_{\mathrm{CL}} \rhd_{w\delta} c \\ &\implies \mathrm{CL}w\delta \vdash M_{\mathrm{CL}} = c \end{aligned}$$

sowie umgekehrt

$$\begin{aligned} \mathrm{CL}w\delta \vdash M_{\mathrm{CL}} = c &\implies M_{\mathrm{CL}} =_{w\delta} c \\ &\implies M_{\mathrm{CL}} \rhd_{w\delta} c \\ &\implies \underbrace{(M_{\mathrm{CL}})_{\lambda}}_{=_{\beta} M} \rhd_{\beta\delta} c \\ &\implies \lambda\beta\delta \vdash M = c \end{aligned}$$

Jede Berechnung eines Werts für M kann also in CLw durchgeführt werden.

Bemerkung. Für eine ornithologische Behandlung der Kombinatorischen Logik sei auf Smullyan (2000) verwiesen.

3 Der einfach getypte λ-Kalkül

Es gibt zwei Versionen der Typisierung des λ-Kalküls:

1. *Curry-Typisierung*: Terme sind die Terme der ungetypten Theorie. Jeder Term hat eine (möglicherweise leere) Menge möglicher Typen (*implizite* Typisierung, "type assignment"). *Curry-Typisierung*
2. *Church-Typisierung*: Terme haben assoziierte Typen, die damit in der Regel eindeutig sind (*explizite* Typisierung). *Church-Typisierung*

Wir behandeln die Curry-Typisierung, und zwar für die einfachste Form, die nur sogenannte einfache Typen enthält. Für den einfach getypten λ-Kalkül verwenden wir die Bezeichnung $\lambda\!\to$. Wir folgen dabei der Darstellung in Barendregt (1992). (Da wir ausschließlich den einfach getypten λ-Kalkül behandeln, verzichten wir im Folgenden auf den Zusatz "einfach".)

Bemerkung. Für den Kalkül $\lambda\!\to$ gilt starke Normalisierung. Daher sind nicht alle rekursiven Funktionen in $\lambda\!\to$ definierbar; die partiell-rekursiven Funktionen sowieso nicht, aber auch nicht alle total rekursiven.
Definiere dazu die Funktion F wie folgt (bei geeigneter Aufzählung der getypten Terme):

$$F(n,m) = \begin{cases} k & \text{genau dann, wenn der } n\text{-te getypte Term angewandt} \\ & \text{auf das Argument } \underline{m} \text{ die } \beta\text{-Normalform } \underline{k} \text{ hat,} \\ 0 & \text{sonst.} \end{cases}$$

Dann kann die (totale) Funktion $g(n) := F(n,n)+1$ aber nicht in $\lambda\!\to$ definierbar sein: Sei g in $\lambda\!\to$ definiert durch den p-ten getypten Term. Dann ist $g(p) = F(p,p)$; aber nach Definition ist $g(p) = F(p,p)+1$. Widerspruch.

3.1 Implizite Typisierung

Bemerkung. Um unnötige Komplikationen bei der Typisierung zu vermeiden, schließen wir im Folgenden bestimmte λ-Terme von der Betrachtung aus, nämlich

1. Terme, in denen eine Variable sowohl frei als auch gebunden vorkommt,
2. (Teil-)Terme der Form $\lambda x.M$, in denen λx auch in M vorkommt.

(Terme wie z. B. $(x(\lambda x.(\lambda x.x)))$ werden also nicht betrachtet.) Die Ausdrucksstärke wird dadurch nicht wesentlich eingeschränkt.

Definition 3.1 Die Menge der *Typen* von $\lambda\!\to$ ist wie folgt definiert: *Typen*

1. *Typvariablen* $\alpha, \beta, \gamma, \delta, \alpha_1, \alpha_2, \ldots$ sind Typen. *Typvariablen*
2. Mit σ und τ ist $(\sigma \to \tau)$ ein Typ (sog. *Funktionstyp*). *Funktionstyp*

Es steht $\sigma_1 \to \sigma_2 \to \cdots \to \sigma_{n-1} \to \sigma_n$ für $\sigma_1 \to (\sigma_2 \to (\cdots (\sigma_{n-1} \to \sigma_n) \cdots))$, d. h. wir verwenden bei Funktionstypen *Rechtsklammerung*.

1. Ein *Urteil* hat die Form $M : \sigma$ für einen λ-Term M und einen Typ σ. *Urteil*

   Dabei heißt M das *Subjekt* des Urteils. (Den Typ σ bezeichnet man in diesem Zusammenhang auch als *Prädikat*. Man sagt "M hat den Typ σ" oder dergleichen.) *Subjekt*

2. Eine *Deklaration* ist ein Urteil, dessen Subjekt eine Termvariable ist. *Deklaration*

3. Eine *Basis* Γ ist eine endliche Menge von Deklarationen, deren Subjekte paarweise verschieden sind. *Basis*

4. Eine *Sequenz* hat die Form $\Gamma \vdash M : \sigma$ für eine Basis Γ und ein Urteil $M : \sigma$. *Sequenz*

Definition 3.2 Sequenzen $\Gamma \vdash M : \sigma$, die ausdrücken, dass das Urteil $M : \sigma$ in der Basis Γ gilt, kann man im *Kalkül* $\lambda\!\to$ ableiten, der gegeben ist durch das Axiomenschema *Kalkül* $\lambda\!\to$

$$(\mathrm{Id})\quad \Gamma, x : \sigma \vdash x : \sigma$$

zusammen mit einer $\to$-Einführungsregel und einer $\to$-Beseitigungsregel:

$$(\to\mathrm{I})\ \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x.M) : \sigma \to \tau} \qquad (\to\mathrm{E})\ \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash (MN) : \tau}$$

("I" von *introduction*, "E" von *elimination*.)
Ist $\Gamma \vdash M : \sigma$ in $\lambda\!\to$ *ableitbar*, so schreiben wir $\Gamma \vdash_{\lambda\to} M : \sigma$ oder auch $\Gamma \vdash M : \sigma$. (Wir identifizieren also häufig Sequenzen mit der Behauptung ihrer Ableitbarkeit. Es ergibt sich dann aus dem Kontext, was jeweils gemeint ist.) *ableitbar*

Beispiele.

1. Es ist $\vdash_{\lambda\to} \lambda xy.x : \sigma \to \tau \to \sigma$:

$$\dfrac{\dfrac{\dfrac{}{x : \sigma, y : \tau \vdash x : \sigma}(\mathrm{Id})}{x : \sigma \vdash \lambda y.x : \tau \to \sigma}(\to\mathrm{I})}{\vdash \lambda x.\lambda y.x : \sigma \to \tau \to \sigma}(\to\mathrm{I})$$

   Der Kombinator **K** hat also den Typ $\sigma \to \tau \to \sigma$.

2. Es ist $\vdash_{\lambda\to} \lambda xyz.xz(yz) : (\sigma \to \tau \to \tau') \to (\sigma \to \tau) \to \sigma \to \tau'$:

   Sei $\Gamma = \{x : \sigma \to \tau \to \tau', y : \sigma \to \tau, z : \sigma\}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : \sigma \to \tau \to \tau'}(\mathrm{Id}) \quad \dfrac{}{\Gamma \vdash z : \sigma}(\mathrm{Id})}{\Gamma \vdash xz : \tau \to \tau'}(\to\mathrm{E}) \quad \dfrac{\dfrac{}{\Gamma \vdash y : \sigma \to \tau}(\mathrm{Id}) \quad \dfrac{}{\Gamma \vdash z : \sigma}(\mathrm{Id})}{\Gamma \vdash yz : \tau}(\to\mathrm{E})}{x : \sigma \to \tau \to \tau', y : \sigma \to \tau, z : \sigma \vdash xz(yz) : \tau'}(\to\mathrm{E})}{x : \sigma \to \tau \to \tau', y : \sigma \to \tau \vdash \lambda z.xz(yz) : \sigma \to \tau'}(\to\mathrm{I})}{x : \sigma \to \tau \to \tau' \vdash \lambda yz.xz(yz) : (\sigma \to \tau) \to \sigma \to \tau'}(\to\mathrm{I})}{\vdash \lambda xyz.xz(yz) : (\sigma \to \tau \to \tau') \to (\sigma \to \tau) \to \sigma \to \tau'}(\to\mathrm{I})$$

   Der Kombinator **S** hat also den Typ $(\sigma \to \tau \to \tau') \to (\sigma \to \tau) \to \sigma \to \tau'$.

Bemerkung. Konstanten können hinzukommen. Entsprechende Konstantendeklarationen gehören dann zu jeder Basis dazu. Ein Beispiel ist der Fixpunktkombinator $\mathbf{Y}: (\sigma \to \sigma) \to \sigma$ für alle Typen σ in der Programmiersprache ML.

Definition 3.3

1. Ein geschlossener Term M heißt *typbar*, falls $\vdash M : \sigma$ für einen Typ σ. *typbar*
2. Ein Term M mit freien Variablen $x_1, \ldots, x_n$ heißt *typbar*, falls $\Gamma \vdash M : \sigma$ für einen Typ σ, wobei $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ für gewisse Typen $\sigma_1, \ldots, \sigma_n$.
3. Sei $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ eine Basis. Dann sei $\Gamma(x_i) := \sigma_i$.
4. Sei V eine Menge von Variablen und Γ eine Basis. Dann sei die *Einschränkung* von Γ auf V: *Einschränkung*
$$\Gamma|_V := \{x : \sigma \mid x \in V \text{ und } \Gamma(x) = \sigma\}$$
5. Der *Vorbereich* von Γ sei $\mathrm{dom}(\Gamma) := \{x_1, \ldots, x_n\}$. *Vorbereich*
6. *Typsubstitution*: $\sigma[\tau/\alpha]$ bedeutet eine gleichzeitige Ersetzung aller Vorkommen der Typvariable α im Typ σ durch den Typ τ. (Siehe auch Definition 3.11.) *Typsubstitution*

   Für eine Basis $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ steht $\Gamma[\tau/\alpha]$ für das Resultat der Typsubstitutionen $\sigma_i[\tau/\alpha]$, für $1 \leq i \leq n$, in Γ.

Bemerkung. Ist $\Gamma \vdash M : \sigma$, so bezeichnet man $M : \sigma$ auch als *hypothetisches Urteil*, da es von der Basis Γ abhängt; bei $\vdash M : \sigma$ spricht man von einem *kategorischen Urteil* $M : \sigma$.

Beispiel. Der Term xx ist *nicht* typbar. Man betrachte

$$\dfrac{(\mathrm{Id})\ \dfrac{}{x : \sigma \to \tau, x : \sigma \vdash x : \sigma \to \tau}\ \lightning \qquad (\mathrm{Id})\ \dfrac{}{x : \sigma \to \tau, x : \sigma \vdash x : \sigma}\ \lightning}{(\to\mathrm{E})\ \dfrac{}{x : \sigma \to \tau, x : \sigma \vdash xx : \tau}}$$

Dies ist *keine* korrekte Ableitung, da die Subjekte der Basis $\{x : \sigma \to \tau, x : \sigma\}$ nicht paarweise verschieden sind.
Folglich ist auch $\lambda x.xx$ nicht typbar, da der Beginn einer entsprechenden Ableitung nur die obige Form haben könnte. Somit hat z. B. auch $\mathbf{\Omega}$ keinen Typ.

Lemma 3.4

1. *Wenn* $\Gamma \subseteq \Gamma'$ *für Basen* Γ *und* Γ', *dann* $(\Gamma \vdash M : \sigma \implies \Gamma' \vdash M : \sigma)$, *wobei* $x \notin \mathrm{dom}(\Gamma')$ *für in* M *gebundene Variablen* x. *(Monotonie)*
2. *Wenn* $\Gamma \vdash M : \sigma$, *dann* $\mathrm{FV}(M) \subseteq \mathrm{dom}(\Gamma)$.
3. *Wenn* $\Gamma \vdash M : \sigma$, *dann* $\Gamma|_{\mathrm{FV}(M)} \vdash M : \sigma$.
4. *Wenn* $\Gamma \vdash x : \sigma$, *dann* $(x : \sigma) \in \Gamma$.
5. *Wenn* $\Gamma \vdash (\lambda x.M) : \sigma$, *dann* $\sigma \doteq \sigma_1 \to \sigma_2$ *für gewisse* σ_1, σ_2 *und* $\Gamma, x : \sigma_1 \vdash M : \sigma_2$.
6. *Wenn* $\Gamma \vdash MN : \sigma$, *dann* $\Gamma \vdash M : \tau \to \sigma$ *und* $\Gamma \vdash N : \tau$ *für ein* τ.
7. *Wenn* $\Gamma \vdash M : \sigma$ *und* M' *Teilterm von* M *ist, dann* $\Gamma' \vdash M' : \sigma'$ *für gewisse* Γ', σ'.

8. *Wenn* $\Gamma \vdash M : \sigma$, *dann* $\Gamma[\tau/\alpha] \vdash M : \sigma[\tau/\alpha]$.
9. *Wenn* $\Gamma, x : \sigma \vdash M : \tau$ *und* $\Gamma \vdash N : \sigma$, *dann* $\Gamma \vdash M[N/x] : \tau$.
10. *Wenn* $\Gamma \vdash M : \sigma$ *und* $M \rhd_\beta M'$, *dann* $\Gamma \vdash M' : \sigma$. *(Subjektreduktion)*

Bemerkungen.
1. Letzteres besagt, dass Typen invariant unter Subjektreduktion sind.
2. Umgekehrt dazu gilt *nicht* Invarianz unter *Subjektexpansion* $\lhd_\beta$, d. h. wenn $M \lhd_\beta M'$ und $\Gamma \vdash M : \sigma$, dann gilt i. A. *nicht* $\Gamma \vdash M' : \sigma$.

   Beispiel: Es ist zwar $\mathbf{I} \lhd_\beta \mathbf{KI}(\lambda x.xx)$ und $\vdash \mathbf{I} : \sigma \to \sigma$, aber $\not\vdash \mathbf{KI}(\lambda x.xx) : \sigma \to \sigma$.

Wir zeigen im Folgenden, dass alle typbaren Terme stark normalisierbar sind. (Die schwache Normalisierbarkeit wurde schon von Turing gezeigt; die starke Normalisierbarkeit geht auf Tait (1967) zurück.) Die Umkehrung dieser Behauptung gilt nicht, wie das Beispiel des nicht typbaren Terms $\lambda x.xx$ in β-NF zeigt.

Definition 3.5
1. Sei SN die Menge der stark normalisierbaren λ-Terme. SN
2. Für Mengen A, B von λ-Termen sei $A \to B := \{M \mid \text{für alle } N \in A \text{ ist } MN \in B\}$.
3. Die *Interpretation* von Typen ist wie folgt induktiv erklärt: *Interpretation (von Typen)*

$$[\![\alpha]\!] := \text{SN, für alle Typvariablen } \alpha;$$
$$[\![\sigma \to \tau]\!] := [\![\sigma]\!] \to [\![\tau]\!].$$

Bemerkung. Die Interpretation eines Funktionstyps ist eine Menge von λ-Termen, die hinsichtlich des vorgegebenen Vor- und Nachbereichs die gewünschte Überführungseigenschaft haben.

Definition 3.6 Eine Menge A von Termen heißt *saturiert*, wenn gilt: *saturiert*

(a) $A \subseteq \text{SN}$,

(b) $xR_1 \ldots R_n \in A$, falls x Termvariable und $R_1, \ldots, R_n \in \text{SN}$, für $n \geq 0$,

(c) $(\lambda x.M)NR_1 \ldots R_n \in A$, falls $(M[N/x])R_1 \ldots R_n \in A$ für $N, R_1, \ldots, R_n \in \text{SN}$, für $n \geq 0$.

$\text{SAT} := \{A \mid A \text{ saturiert}\}$. SAT

Lemma 3.7 *Für jeden Typ* σ *von* $\lambda\to$ *gilt:* $[\![\sigma]\!]$ *ist saturiert.*

Beweis.
1. σ ist Typvariable. Zu zeigen: SN ist saturiert.
   (a) $\text{SN} \subseteq \text{SN}$.
   (b) $xR_1 \ldots R_n \in \text{SN}$, falls $R_1, \ldots, R_n \in \text{SN}$.

(c) Sei $(M[N/x])R_1 \ldots R_n \in \mathrm{SN}$ mit $N, R_1, \ldots, R_n \in \mathrm{SN}$.
Dann gilt auch: $M \in \mathrm{SN}$, sonst könnte $(M[N/x])R_1 \ldots R_n$ nicht stark normalisierbar sein. Wir betrachten $(\lambda x.M)NR_1 \ldots R_n$. Jede Reduktionsfolge sieht dann wie folgt aus, wobei $M \rhd_\beta M'$, $N \rhd_\beta N'$ und $R_i \rhd_\beta R_i'$ für $1 \leq i \leq n$:

$$\begin{aligned}(\lambda x.M)NR_1 \ldots R_n &\rhd_\beta (\lambda x.M')N'R_1' \ldots R_n' \\ &\rhd_{1\beta} M'[N'/x]R_1' \ldots R_n' \\ &\rhd_\beta \ldots\end{aligned}$$

Damit erhält man (unter Verwendung von Lemma 1.10, 4):

$$(M[N/x])R_1 \ldots R_n \rhd_\beta (M'[N'/x])R_1' \ldots R_n' \rhd_\beta \cdots$$

Da diese Folge terminiert, terminiert auch die erste, d. h. $(\lambda x.M)NR_1 \ldots R_n$ ist stark normalisierbar.

2. Mit A und B saturiert, ist $A \to B$ saturiert:
   (a) Sei $M \in A \to B$. Wegen Definition 3.6 (b) gilt $x \in A$ für alle Variablen x (Fall $n = 0$). Also $Mx \in B$. Da Mx stark normalisierbar, ist M stark normalisierbar. Somit ist $A \to B \subseteq \mathrm{SN}$.
   (b) Sei $M \in A$. Dann ist $M \in \mathrm{SN}$. Dann ist $xR_1 \ldots R_n M \in B$ für $R_i \in \mathrm{SN}$. Damit ist auch $xR_1 \ldots R_n \in A \to B$.
   (c) Sei $M \in A$. Dann ist $M \in \mathrm{SN}$.
   Dann $(\lambda x.P)NR_1 \ldots R_n M \in B$, falls $(P[N/x])R_1 \ldots R_n M \in B$.
   Dann $(\lambda x.P)NR_1 \ldots R_n \in A \to B$, falls $(P[N/x])R_1 \ldots R_n \in A \to B$. QED

Definition 3.8

1. Eine *Bewertung* ρ ist eine Abbildung ρ: Variablen $\to \lambda$-Terme. *Bewertung*
2. Sei ρ eine Bewertung. Dann ist die *Interpretation* des Terms M unter ρ: *Interpretation (von Termen)*

$$[\![M]\!]_\rho := M[\rho(x_1)/x_1, \ldots, \rho(x_n)/x_n]$$

   wobei $\{x_1, \ldots, x_n\}$ die Menge *aller* freien Variablen in M ist.
3. Eine Bewertung ρ *erfüllt* $M : \sigma$, falls $[\![M]\!]_\rho \in [\![\sigma]\!]$. Notation: $\rho \vDash M : \sigma$. *Erfüllbarkeit*
4. Eine Bewertung ρ *erfüllt* eine Basis Γ, falls $\rho \vDash x : \sigma$ für alle $(x : \sigma) \in \Gamma$.
   Notation: $\rho \vDash \Gamma$.
5. Eine Basis Γ *erfüllt* $M : \sigma$, falls gilt:

   Für alle Bewertungen ρ: Wenn $\rho \vDash \Gamma$, dann $\rho \vDash M : \sigma$.

   Notation: $\Gamma \vDash M : \sigma$.

Lemma 3.9 (Korrektheit) *Wenn* $\Gamma \vdash M : \sigma$*, dann* $\Gamma \vDash M : \sigma$.

Beweis. Per Induktion über der Struktur der Ableitung von $\Gamma \vdash M : \sigma$.

Fall (Id): Sei $\Gamma = \Gamma' \cup \{x : \sigma\}$ und $M \doteq x$. Es ist $\Gamma', x : \sigma \vdash x : \sigma$. Angenommen $\rho \vDash \Gamma' \cup \{x : \sigma\}$, dann gilt insbesondere $\rho \vDash x : \sigma$. Also $\Gamma', x : \sigma \vDash x : \sigma$.

Fall ($\to$I): Sei $M \doteq \lambda x.N$. Dann ist $\sigma \doteq \sigma_1 \to \sigma_2$ und $\Gamma, x : \sigma_1 \vdash N : \sigma_2$.

Nach Induktionsvoraussetzung gilt $\Gamma, x : \sigma_1 \vDash N : \sigma_2$.

Das heißt, falls $\rho \vDash \Gamma$ und $\rho(x) \in [\![\sigma_1]\!]$, dann $[\![N]\!]_\rho \in [\![\sigma_2]\!]$.

Dann ist $[\![(\lambda x.N)x]\!]_\rho \in [\![\sigma_2]\!]$, da $[\![\sigma_2]\!]$ saturiert.

Das heißt, falls $\rho \vDash \Gamma$ und $\rho(x) = Q \in [\![\sigma_1]\!]$, dann $[\![\lambda x.N]\!]_\rho Q = [\![(\lambda x.N)x]\!]_\rho \in [\![\sigma_2]\!]$.

Also $[\![\lambda x.N]\!]_\rho \in [\![\sigma_1 \to \sigma_2]\!]$.

Fall ($\to$E): Sei $M \doteq PQ$. Dann $\Gamma \vdash P : \tau \to \sigma$ und $\Gamma \vdash Q : \tau$.

Nach Induktionsvoraussetzung gilt $\Gamma \vDash P : \tau \to \sigma$ und $\Gamma \vDash Q : \tau$.

Das heißt, falls $\rho \vDash \Gamma$, dann $[\![P]\!]_\rho \in [\![\tau \to \sigma]\!]$ und $[\![Q]\!]_\rho \in [\![\tau]\!]$.

Dann ist $[\![PQ]\!]_\rho \in [\![\sigma]\!]$ nach Definition von $[\![\ ]\!]$, da $[\![PQ]\!]_\rho = [\![P]\!]_\rho [\![Q]\!]_\rho$. QED

Theorem 3.10 *Wenn* $\Gamma \vdash M : \sigma$*, dann ist* M *stark normalisierbar.*

Beweis. Angenommen $\Gamma \vdash M : \sigma$. Dann gilt aufgrund Korrektheit $\Gamma \vDash M : \sigma$. Das heißt, für alle Bewertungen ρ gilt: Wenn $\rho \vDash \Gamma$, dann $\rho \vDash M : \sigma$.
Sei $\rho_{id}(x) := x$ für jede freie Variable x in M. Dann gilt $\rho_{id} \vDash \Gamma$, da $x \in [\![\sigma]\!]$ für jedes x (da $[\![\sigma]\!]$ saturiert ist).
Also gilt $\rho_{id} \vDash M : \sigma$, d. h. $M \doteq [\![M]\!]_{\rho_{id}} \in [\![\sigma]\!]$. Es ist $[\![\sigma]\!]$ saturiert und M damit stark normalisierbar. QED

3.2 Der Typisierungsalgorithmus

Im Folgenden stellen wir einen Algorithmus zur Typisierung von Termen dar, der für einen gegebenen Term einen Typ liefert, falls der Term typbar ist, und andernfalls fail ausgibt. Damit lassen sich sofort die beiden wichtigsten Entscheidbarkeitsfragen der impliziten Typisierung lösen:

1. Gegeben M und σ, gilt $\vdash M : \sigma$? (*Typprüfung*, engl. *type checking*) *Typprüfung*
2. Gegeben M, gibt es σ mit $\vdash M : \sigma$? (*Typbarkeit*, engl. *typability*) *Typbarkeit*

Die dritte Entscheidbarkeitsfrage in diesem Kontext

3. Gegeben σ, gibt es M mit $\vdash M : \sigma$? (*type inhabitation*) *type inhabitation*

wird durch den Curry-Howard-Isomorphismus gelöst (siehe Abschnitt 3.3).

Zunächst sind einige Vorbereitungen über Substitution und Unifikation von Typvariablen und Typen notwendig.

Definition 3.11

1. Eine *Substitution (in Typen)* ist eine Abbildung s : Typvariablen $\to$ Typen, wobei $s(\alpha) \neq \alpha$ nur für endlich viele α. *Substitution (in Typen)*

   Wir schreiben s auch als $\{\alpha_1 = \sigma_1, \ldots, \alpha_n = \sigma_n\}$, falls $s(\alpha_i) = \sigma_i$.

2. Offenbar determiniert s eine Abbildung $\bar{s}$ von Typen in Typen:

$$\bar{s}(\alpha) :\doteq s(\alpha)$$
$$\bar{s}(\sigma \to \tau) :\doteq \bar{s}(\sigma) \to \bar{s}(\tau)$$

   Wir identifizieren s und $\bar{s}$ und schreiben $s(\sigma)$ oder $\sigma[\sigma_1/\alpha_1, \ldots, \sigma_n/\alpha_n]$.

3. Für eine Basis $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ ist $s(\Gamma) = \{x_1 : s(\sigma_1), \ldots, x_n : s(\sigma_n)\}$.

4. Für Substitutionen s_1 und s_2 ist $s_1 \circ s_2$ (kurz: $s_1 s_2$) in natürlicher Weise definiert.

   Entsprechend ist $s_1 \circ s_2(\sigma) \doteq s_1(s_2(\sigma))$.

5. Ein *Unifikator* für σ und τ ist ein s mit $s(\sigma) \doteq s(\tau)$. *Unifikator*

   Ein Unifikator für eine Menge von Gleichungen $E = \{\sigma_1 = \tau_1, \ldots, \sigma_n = \tau_n\}$ ist ein s mit $s(\sigma_i) \doteq s(\tau_i)$ für alle i mit $1 \leq i \leq n$.

6. Ein *allgemeinster Unifikator* (*mgu*, "most general unifier") von σ und τ (bzgl. E) ist ein Unifikator s, so dass für jeden anderen Unifikator s' von σ und τ (bzgl. E) gilt: $s' = s_1 \circ s$ für eine Substitution s_1. *allgemeinster Unifikator*

   Wir schreiben $s = \mathrm{mgu}(\sigma, \tau)$ bzw. $s = \mathrm{mgu}(E)$.

7. Es ist τ eine *Variante* von σ, falls es s_1 und s_2 gibt mit $s_1(\tau) \doteq \sigma$ und $s_2(\sigma) \doteq \tau$. *Variante*

Beispiele.

1. $\alpha \to (\beta \to \alpha)$ und $\alpha \to (\beta \to \beta)$ haben $\{\alpha/\beta\}$ als mgu.
2. $\beta \to (\alpha \to \beta)$ und $(\gamma \to \gamma) \to \delta$ haben $\{\gamma \to \gamma/\beta, \alpha \to (\gamma \to \gamma)/\delta\}$ als mgu.

Bemerkung. Zwei mgus bezüglich derselben Menge sind immer Varianten voneinander. In diesem Sinne sind mgus eindeutig.

Theorem 3.12 *Es gibt einen Algorithmus, der zu jedem Gleichungssystem E einen mgu von E liefert, falls E unifizierbar ist, und* fail *liefert, falls E nicht unifizierbar ist.*

Beweis. Siehe Logikprogrammierung. Ein Verfahren, das auf Herbrand zurückgeht, besteht aus Regeln zur Umformung von Gleichungssystemen $E = \{\sigma_1 = \tau_1, \ldots, \sigma_n = \tau_n\}$, wobei wir $E_1 \dot\cup E_2$ für die Vereinigung disjunkter Mengen E_1 und E_2 schreiben: *Unifikationsalgorithmus*

$$(id)\ \frac{E \dot\cup \{\sigma = \sigma\}}{E}$$

$$(sym)\ \frac{E \dot\cup \{\sigma = \alpha\}}{E \cup \{\alpha = \sigma\}}\ \text{falls } \sigma \text{ keine Typvariable ist}$$

$$(\mathit{fail})\ \frac{E \dot\cup \{\alpha = \sigma\}}{\mathsf{fail}}\ \text{falls } \alpha \text{ in } \sigma \text{ vorkommt}$$

$$(\mathit{subst})\ \frac{E \dot\cup \{\alpha = \sigma\}}{E[\sigma/\alpha] \cup \{\alpha = \sigma\}}\ \text{falls } \alpha \text{ nicht in } \sigma \text{ vorkommt und } \alpha \text{ in } E \text{ vorkommt}$$

$$(\mathit{func})\ \frac{E \dot\cup \{\tau_1 \to \tau_2 = \sigma_1 \to \sigma_2\}}{E \cup \{\tau_1 = \sigma_1, \tau_2 = \sigma_2\}}$$

Man kann zeigen, dass die Anwendung dieser Reduktionsregeln auf ein Gleichungssystem terminiert. Das Resultat ist entweder fail oder $\{\alpha_1 = \sigma_1, \ldots, \alpha_n = \sigma_n\}$, wobei $\{\alpha_1 = \sigma_1, \ldots, \alpha_n = \sigma_n\}$ der mgu von E ist. QED

Sequenzen $\Gamma \vdash M : \sigma$ ordnen wir nun solche Gleichungssysteme E zu.

Definition 3.13

Das mit der Sequenz $\Gamma \vdash M : \sigma$ *assoziierte Gleichungssystem* $E(\Gamma \vdash M : \sigma)$ ist wie folgt definiert: *assoziiertes Gleichungssystem*

1. $E(\Gamma \vdash x : \sigma) := \{\sigma = \Gamma(x)\}$,
2. $E(\Gamma \vdash \lambda x.M : \sigma) := \{\sigma = \alpha \to \beta\} \cup E(\Gamma, x : \alpha \vdash M : \beta)$ für neue Typvariablen α, β,
3. $E(\Gamma \vdash MN : \sigma) := E(\Gamma \vdash M : \alpha \to \sigma) \cup E(\Gamma \vdash N : \alpha)$ für eine neue Typvariable α.

Bemerkung. Der *Typisierungsalgorithmus* besteht in der Aufstellung eines assoziierten Gleichungssystems $E(\Gamma \vdash M : \sigma)$ mit anschließender Anwendung des Unifikationsalgorithmus. *Typisierungsalgorithmus*

Beispiele.

1. $E(x : \alpha \vdash x : \beta) = \{\beta = \alpha\}$
2. $$\begin{aligned} E(\vdash \lambda xy.x : \alpha \to \beta) &= \{\alpha \to \beta = \alpha_1 \to \alpha_2\} \cup E(x : \alpha_1 \vdash \lambda y.x : \alpha_2) \\ &= \{\alpha \to \beta = \alpha_1 \to \alpha_2, \alpha_2 = \alpha_3 \to \alpha_4\} \cup E(x : \alpha_1, y : \alpha_3 \vdash x : \alpha_4) \\ &= \{\alpha \to \beta = \alpha_1 \to \alpha_2, \alpha_2 = \alpha_3 \to \alpha_4, \alpha_4 = \alpha_1\} \end{aligned}$$

   Lösung:

$$\begin{array}{r} (\mathit{func})\ \dfrac{\{\alpha \to \beta = \alpha_1 \to \alpha_2, \alpha_2 = \alpha_3 \to \alpha_4, \alpha_4 = \alpha_1\}}{\{\alpha = \alpha_1, \beta = \alpha_2, \alpha_2 = \alpha_3 \to \alpha_4, \alpha_4 = \alpha_1\}} \\ (\mathit{subst})\ \dfrac{\{\alpha = \alpha_1, \beta = \alpha_3 \to \alpha_4, \alpha_2 = \alpha_3 \to \alpha_4, \alpha_4 = \alpha_1\}}{\{\alpha = \alpha_1, \beta = \alpha_3 \to \alpha_4, \alpha_2 = \alpha_3 \to \alpha_4, \alpha_4 = \alpha_1\}}\ [\alpha_3 \to \alpha_4/\alpha_2] \\ (\mathit{subst})\ \dfrac{\{\alpha = \alpha_1, \beta = \alpha_3 \to \alpha_4, \alpha_2 = \alpha_3 \to \alpha_4, \alpha_4 = \alpha_1\}}{\{\alpha = \alpha_1, \beta = \alpha_3 \to \alpha_1, \alpha_2 = \alpha_3 \to \alpha_1, \alpha_4 = \alpha_1\}}\ [\alpha_1/\alpha_4] \end{array}$$

   Es gilt also $\vdash \lambda xy.x : \alpha_1 \to (\alpha_3 \to \alpha_1)$.

3. $$\begin{aligned} E(\vdash \lambda x.xx : \sigma) &= \{\sigma = \alpha_1 \to \alpha_2\} \cup E(x : \alpha_1 \vdash xx : \alpha_2) \\ &= \{\sigma = \alpha_1 \to \alpha_2\} \cup E(x : \alpha_1 \vdash x : \alpha_3 \to \alpha_2) \cup E(x : \alpha_1 \vdash x : \alpha_3) \\ &= \{\sigma = \alpha_1 \to \alpha_2, \alpha_3 \to \alpha_2 = \alpha_1, \alpha_3 = \alpha_1\} \end{aligned}$$

Lösungsversuch:

$$\dfrac{\dfrac{\dfrac{\{\sigma = \alpha_1 \to \alpha_2, \alpha_3 \to \alpha_2 = \alpha_1, \alpha_3 = \alpha_1\}}{\{\sigma = \alpha_1 \to \alpha_2, \alpha_1 = \alpha_3 \to \alpha_2, \alpha_3 = \alpha_1\}}\,(sym)}{\{\sigma = (\alpha_3 \to \alpha_2) \to \alpha_2, \alpha_1 = \alpha_3 \to \alpha_2, \alpha_3 = \alpha_3 \to \alpha_2, \alpha_3 = \alpha_1\}}\,[\alpha_3 \to \alpha_2/\alpha_1]\,(subst)}{\mathsf{fail}}\,(fail)$$

Das Gleichungssystem lässt sich nicht lösen, da $\alpha_3 = \alpha_3 \to \alpha_2$ nicht terminiert. Der Term $\lambda x.xx$ ist also nicht typbar.

Lemma 3.14 (Korrektheit und Vollständigkeit)

1. *Sei s Lösung von $E(\Gamma \vdash M : \sigma)$. Dann gilt: $s(\Gamma) \vdash M : s(\sigma)$.*
2. *Wenn $s(\Gamma) \vdash M : s(\sigma)$, dann gilt: Es gibt ein s', das die Typvariablen in Γ und σ wie s interpretiert, und s' ist Lösung von $E(\Gamma \vdash M : \sigma)$.*

   Typvariablen, die von s und s' verschieden interpretiert werden, können dabei stets so gewählt werden, dass sie in $\Gamma \cup \{\sigma\}$ nicht vorkommen.

Beweis.

1. Induktion über der Struktur von M:

   Fall $M \doteq x$: Die Substitution s ist Lösung von $E(\Gamma \vdash x : \sigma)$, d. h. $s(\sigma) = s(\Gamma(x))$.
   Also kommt $x : s(\sigma)$ in Γ vor. Somit $s(\Gamma) \vdash x : s(\sigma)$.

   Fall $M \doteq \lambda x.P$: Wenn s Lösung von $E(\Gamma \vdash \lambda x.P : \sigma)$ ist, dann ist s Lösung von

   $$\{\sigma = \alpha \to \beta\} \cup E(\Gamma, x : \alpha \vdash P : \beta).$$

   Nach Induktionsvoraussetzung gilt dann $s(\Gamma), x : s(\alpha) \vdash P : s(\beta)$. Damit gilt

   $$s(\Gamma) \vdash \lambda x.P : s(\alpha) \to s(\beta)$$

   und da $s(\sigma) \doteq s(\alpha) \to s(\beta)$ auch $s(\Gamma) \vdash \lambda x.P : s(\sigma)$.

   Fall $M \doteq PQ$: Wenn s Lösung von $E(\Gamma \vdash PQ : \sigma)$ ist, dann ist s Lösung von

   $$E(\Gamma \vdash P : \alpha \to \sigma) \quad \text{und} \quad E(\Gamma \vdash Q : \alpha).$$

   Nach Induktionsvoraussetzung gilt dann

   $$s(\Gamma) \vdash P : s(\alpha \to \sigma) \quad \text{und} \quad s(\Gamma) \vdash Q : s(\alpha)$$

   wobei wegen $s(\alpha \to \sigma) \doteq s(\alpha) \to s(\sigma)$ auch $s(\Gamma) \vdash P : s(\alpha) \to s(\sigma)$ gilt.
   Damit gilt $s(\Gamma) \vdash PQ : s(\sigma)$.

2. Induktion über der Struktur von M:

   Fall $M \doteq x$: Es ist $s(\Gamma) \vdash x : s(\sigma)$, d. h. $(x : s(\sigma)) \in s(\Gamma)$. Daher gilt $s(\sigma) = s(\Gamma(x))$.
   Somit ist s selbst Lösung von $E(\Gamma \vdash x : \sigma)$.

Fall $M \doteq \lambda x.P$: Es ist $s(\Gamma) \vdash \lambda x.P : s(\sigma)$, d. h. $s(\sigma) = \sigma_1 \to \sigma_2$ für gewisse σ_1, σ_2, und es ist $s(\Gamma), x : \sigma_1 \vdash P : \sigma_2$.

Sei s' wie s, jedoch erweitert um $\alpha_1 \mapsto \sigma_1$ und $\alpha_2 \mapsto \sigma_2$ mit α_1, α_2 neu. Dann ist $s'(\Gamma), x : s'(\alpha_1) \vdash P : s'(\alpha_2)$.

Damit stimmen s und s' auf den in Γ und σ vorkommenden Typvariablen überein, und s' ist Lösung von $\{\sigma = \alpha_1 \to \alpha_2\}$.

Nach Induktionsvoraussetzung gibt es ein s'' als Lösung von $E(\Gamma, x : \alpha_1 \vdash P : \alpha_2)$ derart, dass s'' mit s' auf den Typvariablen in $\Gamma, \alpha_1, \alpha_2$ übereinstimmt.

Ferner kann angenommen werden, dass die Typvariablen, die von s' und s'' verschieden interpretiert werden, nicht in σ vorkommen, also $s'(\sigma) = s''(\sigma)$.

Damit ist s'' also Lösung von $\{\sigma = \alpha_1 \to \alpha_2\} \cup E(\Gamma, x : \alpha_1 \vdash P : \alpha_2)$, d. h. von $E(\Gamma \vdash \lambda x.P : \sigma)$, und stimmt auf den Typvariablen in Γ und σ mit s überein.

Fall $M \doteq PQ$: Es ist $s(\Gamma) \vdash PQ : s(\sigma)$, d. h. $s(\Gamma) \vdash P : \tau \to s(\sigma)$ und $s(\Gamma) \vdash Q : \tau$ für ein τ.

Wir definieren s' als s erweitert um $\alpha \mapsto \tau$, wobei α neu. Dann stimmt s' mit s auf den Typvariablen in Γ und σ überein. Wir haben also $s'(\Gamma) \vdash P : s'(\alpha) \to s'(\sigma)$ und $s'(\Gamma) \vdash Q : s'(\alpha)$.

Nach Induktionsvoraussetzung gibt es Lösungen s_1'' und s_2'' von $E(\Gamma \vdash P : \alpha \to \sigma)$ bzw. $E(\Gamma \vdash Q : \alpha)$, die mit s' auf den Typvariablen in Γ, σ, α übereinstimmen.

Ferner können wir annehmen, dass die bei der Konstruktion von $E(\Gamma \vdash P : \alpha \to \sigma)$ und $E(\Gamma \vdash Q : \alpha)$ neu eingeführten Typvariablen voneinander verschieden sind.

Dann ist $s_1'' \cup s_2''$ Lösung von $E(\Gamma \vdash P : \alpha \to \sigma) \cup E(\Gamma \vdash Q : \alpha)$, das heißt von $E(\Gamma \vdash PQ : \sigma)$, und stimmt mit s auf den Typvariablen in Γ und σ überein. QED

In $\lambda\to$ können einem typbaren Term verschiedene Typen zugeordnet werden. Alle diese Typen sind jedoch Substitutionsinstanzen eines sogenannten Haupttyps.

Definition 3.15

1. Es heißt σ *Haupttyp* für einen geschlossenen Term M, falls gilt:
   (a) $\vdash M : \sigma$, und
   (b) wenn $\vdash M : \sigma'$, dann gibt es eine Substitution s, so dass $\sigma' = s(\sigma)$.
2. Es heißt $\langle \Gamma, \sigma \rangle$ *Hauptpaar* für M, falls gilt:
   (a) $\Gamma \vdash M : \sigma$, und
   (b) wenn $\Gamma' \vdash M : \sigma'$, dann gibt es eine Substitution s, so dass $\sigma' = s(\sigma)$ und $\Gamma' \supseteq s(\Gamma)$.

Beispiele.

1. $\alpha \to \alpha$ ist ein Haupttyp von $\mathbf{I}$.
2. $\langle \{x : (\alpha \to \alpha) \to \beta\}, \beta \rangle$ ist ein Hauptpaar für $x\mathbf{I}$.

Theorem 3.16 *Es gibt einen Algorithmus, der zu jedem geschlossenen Term M einen Haupttyp und zu jedem Term M ein Hauptpaar liefert, falls M überhaupt einen Typ hat, und andernfalls* fail *ausgibt.*

Beweis. Wir betrachten beliebige Terme M. Für geschlossene Terme folgt das Resultat dann als Spezialfall für die leere Basis.
Sei $\Gamma_0 := \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}$ und $\sigma_0 := \beta$, wobei $x_1, \ldots, x_n$ die freien Variablen in M sind. Jede mgu-Lösung von $E(\Gamma_0 \vdash M : \sigma_0)$ ist ein Hauptpaar für M, falls es eine Lösung gibt; andernfalls wird fail ausgegeben.

1. M hat Typ $\iff$ $\Gamma \vdash M : \sigma$ für gewisse Γ, σ
   $\iff$ $s(\Gamma_0) \vdash M : s(\sigma_0)$ für gewisses s
   $\iff$ $E(\Gamma_0 \vdash M : \sigma_0)$ ist lösbar (Lemma 3.14)
2. Sei s mgu-Lösung von $E(\Gamma_0 \vdash M : \sigma_0)$. Dann ist $s(\Gamma_0) \vdash M : s(\sigma_0)$.

   Sei nun $\Gamma' \vdash M : \sigma'$ und $\widetilde{\Gamma} := \Gamma'|_{\mathrm{FV}(M)}$. Dann ist $\widetilde{\Gamma} \vdash M : \sigma'$.

   Sei s' so gewählt, dass $s'(\Gamma_0) = \widetilde{\Gamma}$, sowie $s'(\sigma_0) = \sigma'$. Dann ist $s'(\Gamma_0) \vdash M : s'(\sigma_0)$.

   Dann gilt aufgrund Lemma 3.14 (2) für ein s'', das die Typvariablen in Γ_0 und σ_0 wie s' interpretiert: s'' ist Lösung von $E(\Gamma_0 \vdash M : \sigma_0)$.

   Da s mgu ist, gilt $s'' = s_1 \circ s$ für ein s_1, d. h. $\sigma' = s''(\sigma_0) = s_1(s(\sigma_0))$.

   Ferner haben wir, dass $\Gamma' \supseteq \widetilde{\Gamma}$ mit $\widetilde{\Gamma} = s''(\Gamma_0) = s_1(s(\Gamma_0))$.

   Es ist $\langle s(\Gamma_0), s(\sigma_0)\rangle$ also ein Hauptpaar für M. QED

Theorem 3.17

1. *Typbarkeit ist entscheidbar.*
2. *Typprüfung ist entscheidbar.*

Beweis.

1. Sei ein geschlossener Term M gegeben. Der Algorithmus aus Theorem 3.16 liefert einen Haupttyp σ, falls M typbar ist, andernfalls fail.
2. Zur Prüfung von $\vdash M : \sigma'$ für gegebenes σ' wenden wir zunächst den Algorithmus aus Theorem 3.16 an. Falls M einen Typ hat, ist das Ergebnis der Typisierung $\vdash M : \sigma$ für einen Haupttyp σ. Da σ ein Haupttyp ist, muss es dann eine Substitution s geben, so dass $\sigma' = s(\sigma)$, sofern σ' ein Typ von M ist. Ob es eine solche Substitution s gibt, kann leicht algorithmisch überprüft werden. QED

Theorem 3.18 *Das Problem, ob es zu einem gegebenen Typ σ einen Term gibt ("type inhabitation"), ist entscheidbar.*

Beweis. Es gibt M mit $\vdash M : \sigma$ genau dann, wenn es einen Beweis für σ (als Formel) in der positiven Implikationslogik gibt. Dies ergibt sich aus dem im folgenden Abschnitt behandelten *Curry-Howard-Isomorphismus* (Theorem 3.23). Die positive Implikationslogik ist entscheidbar. QED

3.3 Der Curry-Howard-Isomorphismus

Zwischen getyptem λ-Kalkül und Logik bestehen gewisse Entsprechungen, die grob wie folgt gegeben sind:

Getypter λ-Kalkül	*Logik*
Typ	Formel, Aussage
typbarer offener Term	Ableitung mit Annahmen
typbarer geschlossener Term	Beweis (Ableitung ohne Annahmen)
β-Kontraktion	Kontraktion von Ableitung
typbarer Term in β-Normalform	Ableitung in Normalform
β-Gleichheit	Gleichheit von Ableitungen

Definition 3.19

1. Typvariablen heißen auch *Aussagevariablen*, Typen auch *(implikationslogische) Formeln*. *Formeln*
2. Eine endliche Menge von Formeln heißt *Kontext*. *Kontext*

   Metasprachliche Variablen für Kontexte sind $\Delta, \Delta', \ldots$
3. Die *positive Implikationslogik* $P{\to}$ ist gegeben durch das Axiomenschema *positive Implikationslogik*

   $$(\mathrm{Id})\ \ \Delta, \sigma \vdash \sigma$$

   und die beiden Regeln:

   $$(\to\mathrm{I})\ \frac{\Delta, \sigma \vdash \tau}{\Delta \vdash \sigma \to \tau} \qquad (\to\mathrm{E})\ \frac{\Delta \vdash \sigma \to \tau \quad \Delta \vdash \sigma}{\Delta \vdash \tau}$$

   ($P{\to}$ heißt *positive* Implikationslogik, da keine Negation vorkommt.)
4. $\Delta \vdash_{P\to} \sigma$ bedeutet, dass $\Delta \vdash \sigma$ in $P{\to}$ *ableitbar* ist. *ableitbar*
5. Für ein Urteil $M : \sigma$ sei $(M : \sigma)^\circ \doteq \sigma$.
6. Für eine Basis $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ sei Γ° der Kontext $\{\sigma_1, \ldots, \sigma_n\}$.

Lemma 3.20 *Wenn $\Gamma \vdash_{\lambda\to} M : \sigma$, dann $\Gamma^\circ \vdash_{P\to} \sigma$.*

Beweis. Durch Anwendung von $^\circ$ auf jedes Urteil in der $\lambda{\to}$-Ableitung von $\Gamma \vdash M : \sigma$ erhält man eine $P{\to}$-Ableitung von $\Gamma^\circ \vdash \sigma$. QED

Lemma 3.21 *Es gibt einen Algorithmus, der zu jedem typbaren Term M eine Ableitung von $\Delta \vdash \sigma$ in $P{\to}$ liefert derart, dass $\Delta = \Gamma^\circ$ und $\Gamma \vdash_{\lambda\to} M : \sigma$.*

Beweis. Mit dem in Theorem 3.16 erwähnten Algorithmus lässt sich zu jedem typbaren Term M dessen Hauptpaar $\langle \Gamma, \sigma \rangle$ erzeugen; dieses kann jeweils direkt in eine $P{\to}$-Sequenz $\Delta \vdash \sigma$ mit $\Delta = \Gamma^\circ$ umgeformt werden. Die zu deren Ableitung nötige $P{\to}$-Regelanwendung ist jeweils durch die Form von M festgelegt. QED

Das heißt: Jeder typbare Term M kodiert eine Ableitung in $P\!\to$. Aus dieser Ableitung lassen sich durch Substitution alle Ableitungen von $\Gamma^\circ \vdash \sigma$ in $P\!\to$ gewinnen, die Ableitungen von $\Gamma \vdash M : \sigma$ in $\lambda\!\to$ entsprechen.

Lemma 3.22 *Zu jeder Ableitung von $\Delta \vdash \sigma$ in $P\!\to$ lässt sich ein Term M und eine Ableitung von $\Gamma \vdash M : \sigma$ in $\lambda\!\to$ konstruieren mit $\Gamma^\circ = \Delta$.*

Beweis. Induktion über dem Aufbau der Ableitung von $\Delta \vdash \sigma$ in $P\!\to$:

1. Alle in Anfangssequenzen (Id) von $P\!\to$ vorkommenden Formeln σ werden durch Typdeklarationen $x : \sigma$ ersetzt, wobei die Variable x so gewählt ist, dass allen Vorkommen einer Formel σ *dieselbe* Deklaration $x : \sigma$ und *verschiedenen* Formeln σ und τ Deklarationen $x : \sigma$ und $y : \tau$ mit *verschiedenen* Variablen x und y entsprechen.

2. Sei in $P\!\to$ der Schritt
$$(\to\!\mathrm{I})\ \frac{\sigma_1, \ldots, \sigma_n, \sigma \vdash \tau}{\sigma_1, \ldots, \sigma_n \vdash \sigma \to \tau}$$
angewendet, wobei für $\sigma_1, \ldots, \sigma_n, \sigma \vdash \tau$ schon eine Ableitung in $\lambda\!\to$ von
$$x_1 : \sigma_1, \ldots, x_n : \sigma_n, x : \sigma \vdash M : \tau$$
vorliegt. Dann verlängern wir diese unter Verwendung von $(\to\!\mathrm{I})$ in $\lambda\!\to$ zu
$$x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash \lambda x.M : \sigma \to \tau.$$

3. Sei in $P\!\to$ der Schritt
$$(\to\!\mathrm{E})\ \frac{\sigma_1, \ldots, \sigma_n \vdash \sigma \to \tau \qquad \sigma_1, \ldots, \sigma_n \vdash \sigma}{\sigma_1, \ldots, \sigma_n \vdash \tau}$$
angewendet, wobei in $\lambda\!\to$ für $\sigma_1, \ldots, \sigma_n \vdash \sigma \to \tau$ und $\sigma_1, \ldots, \sigma_n \vdash \sigma$ schon Ableitungen von
$$x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \sigma \to \tau \qquad \text{und} \qquad x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash N : \sigma$$
vorliegen. Man beachte, dass einem Typ σ_i genau eine Variable x_i zugeordnet ist. Durch eine Anwendung von $(\to\!\mathrm{E})$ erhalten wir dann eine Ableitung in $\lambda\!\to$ von $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash MN : \tau$. QED

Theorem 3.23 (Curry-Howard-Isomorphismus)
Sei M_P die gemäß Lemma 3.21 zu einem in $\lambda\!\to$ typbaren Term M gehörende Ableitung Π in $P\!\to$. Sei Π_λ der gemäß Lemma 3.22 zu einer Ableitung in $P\!\to$ gehörende Term von $\lambda\!\to$. Dann gilt:

1. *$(\Pi_\lambda)_P$ ist eine Ableitung in $P\!\to$, aus der sich Π durch Substitution von Formeln für Aussagevariablen gewinnen lässt.*

2. *$(M_P)_\lambda$ ist (bis auf Umbenennung von freien und/oder gebundenen Variablen) ein Term, der aus M durch Identifizierung von freien oder gebundenen Variablen entsteht.*

Beweis. Aus den Lemmas 3.21 und 3.22. QED

Ein Beispiel für (2) ist der λ-Term

$$u(zx)(zy)$$

der nach dem Typisierungsalgorithmus durch

$$u:\alpha\to\alpha\to\beta, z:\gamma\to\alpha, x:\gamma, y:\gamma \vdash_{\lambda\to} u(zx)(zy):\beta$$

getypt wird. Für $\Gamma = \{u:\alpha\to\alpha\to\beta, z:\gamma\to\alpha, x:\gamma, y:\gamma\}$ ist $\Gamma^\circ = \{\alpha\to\alpha\to\beta, \gamma\to\alpha, \gamma\}$. Die korrespondierende Ableitung $(u(zx)(zy))_P$ in $P\to$

$$\dfrac{\dfrac{\dfrac{}{\Gamma^\circ\vdash\alpha\to(\alpha\to\beta)}(\mathrm{Id})\quad \dfrac{\dfrac{}{\Gamma^\circ\vdash\gamma\to\alpha}(\mathrm{Id})\quad \dfrac{}{\Gamma^\circ\vdash\gamma}(\mathrm{Id})}{\Gamma^\circ\vdash\alpha}(\to\mathrm{E})}{\Gamma^\circ\vdash\alpha\to\beta}(\to\mathrm{E})\quad \dfrac{\dfrac{}{\Gamma^\circ\vdash\gamma\to\alpha}(\mathrm{Id})\quad \dfrac{}{\Gamma^\circ\vdash\gamma}(\mathrm{Id})}{\Gamma^\circ\vdash\alpha}(\to\mathrm{E})}{\Gamma^\circ\vdash\beta}(\to\mathrm{E})$$

liefert

$$\alpha\to\alpha\to\beta, \gamma\to\alpha, \gamma \vdash_{P\to} \beta.$$

Dieser Ableitung entspricht gemäß Lemma 3.22 ein λ-Term der Form $u(zx)(zx)$, in dem x und y identifiziert sind.
Zur Identifizierung der Variablen kommt es dadurch, dass beim Übergang von $\lambda\to$ zu $P\to$ Information verlorengeht, die beim Übergang von $P\to$ zu $\lambda\to$ nicht mehr rekonstruiert werden kann. Das hängt damit zusammen, dass bei der Zuordnung von Variablen zu Formeln (= Typen) in Anfangssequenzen (Id) von $\lambda\to$ gemäß Lemma 3.22 keine zwei Variablen denselben Typ haben können, d. h. beim Übergang von $\lambda\to$ zu $P\to$ keine Sequenz der Form $\Gamma, x:\sigma, y:\sigma \vdash M:\tau$ auftreten kann.
Die Entsprechung von Formeln und Typen sowie von typbaren Termen und Ableitungen bezeichnet man daher auch etwas schwächer als *Curry-Howard-Korrespondenz*. Eine andere Fassung des aussagenlogischen Kalküls erlaubt es jedoch, ohne diesen Informationsverlust auszukommen. Dazu wird auf die Beweistheorie verwiesen (siehe Troelstra & Schwichtenberg, 2001, Ch. 6; vgl. auch Sørensen & Urzyczyn, 2006, Ch. 4).

Der Curry-Howard-Isomorphismus induziert Reduktions- und Gleichheitsrelationen für Ableitungen, die $\rhd_\beta$ und $=_\beta$ entsprechen. Diese behandelt man ebenfalls in der Beweistheorie, genauer in der Theorie des natürlichen Schließens (siehe Prawitz, 2006). Betrachte das β-Redex $(\lambda x.M)N$ mit Typ τ:

$$\dfrac{\dfrac{\dfrac{\mathscr{D}_1}{\Gamma, x:\sigma\vdash M:\tau}(\to\mathrm{I})}{\Gamma\vdash\lambda x.M:\sigma\to\tau}\quad \dfrac{\mathscr{D}_2}{\Gamma\vdash N:\sigma}}{\Gamma\vdash(\lambda x.M)N:\tau}(\to\mathrm{E})$$

Es ist

$$(\lambda x.M)N \rhd_{1\beta} M[N/x]$$

Dem entspricht eine Kontraktion $\rhd^{\circ}_{1\beta}$ für Ableitungen im Kalkül des natürlichen Schließens:

$$\dfrac{\dfrac{\begin{array}{c}[\sigma]^n\\ \mathscr{D}_1^{\circ}\\ \tau\end{array}}{\sigma\to\tau}(\to\mathrm{I})^n \quad \begin{array}{c}\mathscr{D}_2^{\circ}\\ \sigma\end{array}}{\tau}(\to\mathrm{E}) \qquad \rhd^{\circ}_{1\beta} \qquad \begin{array}{c}\mathscr{D}_2^{\circ}\\ \sigma\\ \mathscr{D}_1^{\circ}\\ \tau\end{array}$$

wobei alle bei $(\to\mathrm{I})$ gelöschten Vorkommen der Annahme σ durch Kopien der mit σ endenden Ableitung

$$\begin{array}{c}\mathscr{D}_2^{\circ}\\ \sigma\end{array}$$

ersetzt werden; falls es keine solchen Vorkommen gibt, wird die Ableitung zu

$$\begin{array}{c}\mathscr{D}_1^{\circ}\\ \tau\end{array}$$

umgeformt.
Die Ersetzung aller Vorkommen der Annahme σ entspricht der Ersetzung aller Vorkommen von x in M durch N, d. h. der Substitution $M[N/x]$.
Der Normalisierbarkeit von λ-Termen entspricht die Normalisierbarkeit von Ableitungen im natürlichen Schließen und umgekehrt. Haben zwei Ableitungen dieselbe Normalform, so sind diese gleich im Sinne von β-Gleichheit.
Inhaltlich repräsentiert die linke Ableitung eine Argumentation, in der ein Lemma der Form $\sigma\to\tau$ verwendet wird. In der rechten, kontrahierten Ableitung kommt dieses Lemma nicht mehr vor. Normalisierbarkeit von Ableitungen bedeutet, dass durch die Verwendung von Lemmas nichts gezeigt werden kann, was nicht auch direkt gezeigt werden könnte. Dies rechtfertigt die Verwendung von Lemmas, was i. A. kürzere Ableitungen ermöglicht.

4 Der polymorph getypte λ-Kalkül

Der polymorph getypte λ-Kalkül heißt auch *System F* oder *getypter λ-Kalkül 2. Stufe*, kurz: $\lambda 2$.

Motivation: Es soll zum Beispiel die Identitätsfunktion $id \doteq \lambda x.x : \alpha \to \alpha$ unabhängig von einem speziellen Typ α sein. Man will also, dass $id \doteq \lambda x.x : \forall\alpha.(\alpha \to \alpha)$.

Definition 4.1

1. Die Menge der *Typen* von $\lambda 2$ ist wie folgt definiert: *Typen*
   (a) *Typvariablen* $\alpha, \beta, \gamma, \delta, \alpha_1, \alpha_2, \ldots$ sind Typen. *Typvariablen*
   (b) Mit σ und τ ist $(\sigma \to \tau)$ ein Typ (sog. *Funktionstyp*). *Funktionstyp*
   (c) Mit α und σ ist auch $\forall\alpha.\sigma$ ein Typ (sog. *universeller (polymorpher) Typ*). *universeller Typ*
2. Der Allquantor $\forall$ bindet stärker als $\to$.
   Es steht $\forall\alpha_1\alpha_2\ldots\alpha_n.\sigma$ für $(\forall\alpha_1.(\forall\alpha_2.(\ldots(\forall\alpha_n.\sigma))))$.
3. Vorkommen der Typvariable α in $\forall\alpha.\sigma$ heißen *gebunden*. *gebunden*
   Die *Menge der freien Typvariablen* $\mathrm{FV}(\sigma)$ für einen Typ σ ist analog zu $\mathrm{FV}(M)$ für λ-Terme M definiert. *freie Typvariable*
4. Eine Substitution $\sigma[\tau/\alpha]$ ist nur dann erlaubt, wenn τ in σ frei einsetzbar für α ist (d. h. falls τ keine Variablen enthält, die in $\sigma[\tau/\alpha]$ durch $\forall$ gebunden würden).

Definition 4.2 Die *Typisierung* in $\lambda 2$ ist definiert durch das Axiomenschema *Typisierung*

$$(\mathrm{Id})\ \ \Gamma, x:\sigma \vdash x:\sigma$$

und die Regeln:

$$(\to\mathrm{I})\ \frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash (\lambda x.M):\sigma\to\tau} \qquad (\to\mathrm{E})\ \frac{\Gamma \vdash M:\sigma\to\tau \quad \Gamma \vdash N:\sigma}{\Gamma \vdash MN:\tau}$$

$$(\forall\mathrm{I})\ \frac{\Gamma \vdash M:\sigma}{\Gamma \vdash M:\forall\alpha.\sigma}\ \text{falls } \alpha \notin \mathrm{FV}(\Gamma) \qquad (\forall\mathrm{E})\ \frac{\Gamma \vdash M:\forall\alpha.\sigma}{\Gamma \vdash M:\sigma[\tau/\alpha]}$$

Ist $\Gamma \vdash M:\sigma$ in $\lambda 2$ *ableitbar*, so schreiben wir $\Gamma \vdash_{\lambda 2} M:\sigma$. *ableitbar*

Beispiele.

1. Es ist $\vdash_{\lambda 2} \lambda x.x : \forall\alpha.(\alpha\to\alpha)$:

$$\dfrac{\dfrac{\dfrac{}{x:\alpha \vdash x:\alpha}(\mathrm{Id})}{\vdash \lambda x.x:\alpha\to\alpha}(\to\mathrm{I})}{\vdash \lambda x.x:\forall\alpha.(\alpha\to\alpha)}(\forall\mathrm{I})$$

   Es ist auch $\vdash_{\lambda 2} \lambda x.x : \forall\alpha.\alpha\to\forall\beta.\beta$ und $\vdash_{\lambda 2} \lambda x.x : \forall\beta.(\forall\alpha.\alpha\to\beta)$.

2. Es ist $\vdash_{\lambda 2} \lambda xy.y : \forall\alpha.\forall\beta.(\alpha \to (\beta \to \beta))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x:\alpha, y:\beta \vdash y:\beta}(\mathrm{Id})}{x:\alpha \vdash \lambda y.y : \beta \to \beta}(\to\mathrm{I})}{\vdash \lambda xy.y : \alpha \to (\beta \to \beta)}(\to\mathrm{I})}{\vdash \lambda xy.y : \forall\beta.(\alpha \to (\beta \to \beta))}(\forall\mathrm{I})}{\vdash \lambda xy.y : \forall\alpha.\forall\beta.(\alpha \to (\beta \to \beta))}(\forall\mathrm{I})$$

3. Es ist $\vdash_{\lambda 2} \lambda x.xx : \forall\beta.(\forall\alpha.\alpha \to \beta)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x:\forall\alpha.\alpha \vdash x:\forall\alpha.\alpha}(\mathrm{Id})}{x:\forall\alpha.\alpha \vdash x:\alpha\to\beta}(\forall\mathrm{E})[\alpha\to\beta/\alpha] \quad \dfrac{\dfrac{}{x:\forall\alpha.\alpha \vdash x:\forall\alpha.\alpha}(\mathrm{Id})}{x:\forall\alpha.\alpha \vdash x:\alpha}(\forall\mathrm{E})[\alpha/\alpha]}{x:\forall\alpha.\alpha \vdash xx:\beta}(\to\mathrm{E})}{\vdash \lambda x.xx : \forall\alpha.\alpha\to\beta}(\to\mathrm{I})}{\vdash \lambda x.xx : \forall\beta.(\forall\alpha.\alpha\to\beta)}(\forall\mathrm{I})$$

Es ist auch $\vdash_{\lambda 2} \lambda x.xx : \forall\beta.(\forall\alpha.\alpha \to (\beta \to \beta))$ und $\vdash_{\lambda 2} \lambda x.xx : \forall\alpha.\alpha \to \forall\beta.\beta$.

Definition 4.3

1. Es sei $\sigma \sqsupset \tau$, falls

   $\tau \doteq \forall\alpha.\sigma$ für eine Typvariable α (τ ist *Generalisierung* von σ),

   oder

   $\sigma \doteq \forall\alpha.\sigma_1$ und $\tau \doteq \sigma_1[\sigma_2/\alpha]$ für einen Typ σ_2 (τ ist *Spezialisierung* von σ).

2. Es sei $\sigma \sqsupseteq \tau$ genau dann, wenn es $n \geq 0$ gibt, so dass $\sigma \doteq \sigma_1 \sqsupset \cdots \sqsupset \sigma_n \doteq \tau$.

3. Es sei σ^0 der Typ σ ohne Quantorenpräfix (d. h. führende Quantoren sind weggelassen).

Beispiele.

1. $(\forall\alpha_1 \ldots.\forall\alpha_n.\sigma)^0 \doteq \sigma$
2. $(\forall\alpha.(\forall\beta.\beta \to \alpha))^0 \doteq \forall\beta.\beta \to \alpha$

Bemerkungen.

1. Die Relation $\sqsupset$ ist nicht symmetrisch:

   Zwar gilt, dass wenn $\sigma \sqsupset \forall\alpha.\sigma$, dann $\forall\alpha.\sigma \sqsupset \sigma$. Aber wenn $\forall\alpha.\sigma_1 \sqsupset \sigma_1[\sigma_2/\alpha]$, dann gilt i. A. nicht $\sigma_1[\sigma_2/\alpha] \sqsupset \forall\alpha.\sigma_1$.

2. Intuitiv bedeutet $\sigma \sqsupseteq \tau$, dass $\Gamma \vdash M : \tau$ aus $\Gamma \vdash M : \sigma$ alleine durch Anwendung von $\forall$-Regeln ableitbar ist.

Lemma 4.4 *Sei Γ gegeben. Dann ist $\sigma \sqsupseteq \tau$ mit $\sigma \doteq \sigma_1 \sqsupset \cdots \sqsupset \sigma_n \doteq \tau$, wobei keine in Γ frei vorkommende Typvariable bei einem Übergang von σ_i nach σ_{i+1} generalisiert wird,*

genau dann, wenn es eine (Teil-)Ableitung folgender Form gibt:

$$\left.\begin{array}{c}\Gamma \vdash M : \sigma \\ \vdots \\ \Gamma \vdash M : \tau\end{array}\right\} \textit{nur } \forall\textit{-Regeln.}$$

Beweis. Definition von $\sqsupseteq$. QED

Lemma 4.5

1. *Wenn* $\Gamma \vdash x : \sigma$*, dann gibt es* σ' *mit* $\sigma' \sqsupseteq \sigma$ *derart, dass* $(x : \sigma') \in \Gamma$.
2. *Wenn* $\Gamma \vdash \lambda x.M : \xi$*, dann gibt es* σ *und* τ *derart, dass* $\Gamma, x : \sigma \vdash M : \tau$ *und* $(\sigma \to \tau) \sqsupseteq \xi$.
3. *Wenn* $\Gamma \vdash MN : \tau$*, dann gibt es* σ *und* τ' *mit* $\tau' \sqsupseteq \tau$ *derart, dass* $\Gamma \vdash M : \sigma \to \tau'$ *und* $\Gamma \vdash N : \sigma$.
4. *Wenn* $\Gamma, x : \sigma \vdash M : \tau$ *und* $\Gamma \vdash N : \sigma$*, dann* $\Gamma \vdash M[N/x] : \tau$.

Bemerkung. Vergleiche Lemma 3.4 (4)-(6) und (9). Auch die übrigen Aussagen von Lemma 3.4 gelten für $\lambda 2$. Subjektreduktion werden wir beweisen; dazu benötigen wir noch das folgende Lemma.

Lemma 4.6 *Wenn* $(\sigma \to \tau) \sqsupseteq (\sigma' \to \tau')$*, dann ist* $(\sigma' \to \tau') \doteq s(\sigma \to \tau)$ *für eine Substitution* s*, d. h.* $\sigma' \to \tau'$ *ist spezieller als* $\sigma \to \tau$.

Beweis. Sei $(\sigma \to \tau) \doteq \sigma_1 \sqsupset \cdots \sqsupset \sigma_n \doteq (\sigma' \to \tau')$.
Wir zeigen: $\sigma_i^0 \doteq s_i(\sigma \to \tau)$ für jedes s_i $(1 \leq i \leq n)$.
Damit folgt die Behauptung, da $(\sigma' \to \tau')^0 \doteq (\sigma' \to \tau')$.

Beweis durch Induktion über n:

Für $n = 1$: s_1 ist die leere Substitution.

Für $n = m + 1$: Sei $\sigma_m^0 \doteq s_m(\sigma \to \tau)$.

(a) Sei $\sigma_{m+1} \doteq \forall\alpha.\sigma_m$, dann ist $\sigma_{m+1}^0 \doteq \sigma_m^0$, also $s_{m+1} := s_m$.

(b) Sei $\sigma_m \doteq \forall\alpha.\rho$ und $\sigma_{m+1} \doteq \rho[\rho_1/\alpha]$. Dann setze $s_{m+1} := s_m[\rho_1/\alpha]$, wobei $s_m[\rho_1/\alpha]$ sich von s_m nur durch $\alpha \mapsto \rho_1$ unterscheidet. QED

Theorem 4.7 (Subjektreduktion) *Wenn* $\Gamma \vdash M : \sigma$ *und* $M \rhd_\beta M'$*, dann* $\Gamma \vdash M' : \sigma$.

Beweis. Wir betrachten den Fall $M \doteq (\lambda x.P)Q \rhd_{1\beta} P[Q/x] \doteq M'$. Daraus ergibt sich der Rest.

Angenommen, es gilt $\Gamma \vdash (\lambda x.P)Q : \sigma$.

Nach Lemma 4.5 (3) gibt es dann τ und σ' mit $\sigma' \sqsupseteq \sigma$ derart, dass

$$\Gamma \vdash \lambda x.P : \tau \to \sigma' \quad \text{und} \quad \Gamma \vdash Q : \tau.$$

Nach Lemma 4.5 (2) gibt es dann τ und $\sigma' \sqsupseteq \sigma$ sowie τ' und σ'' derart, dass

$$\Gamma, x:\tau' \vdash P:\sigma'' \quad \text{und} \quad \Gamma \vdash Q:\tau$$

mit $(\tau' \to \sigma'') \sqsupseteq (\tau \to \sigma')$.

Nach Lemma 4.6 gibt es dann τ und $\sigma' \sqsupseteq \sigma$, so dass

$$\Gamma, x:\tau \vdash P:\sigma' \quad \text{und} \quad \Gamma \vdash Q:\tau$$

wobei τ und σ' spezieller als τ' bzw. σ'' sind.

Mit Lemma 4.5 (4) gibt es somit $\sigma' \sqsupseteq \sigma$, so dass $\Gamma \vdash P[Q/x]:\sigma'$.

Nach Lemma 4.4 gilt dann $\Gamma \vdash P[Q/x]:\sigma$. QED

Definition 4.8

1. Eine *Bewertung* in SAT ist eine Abbildung $v:$ Typvariablen $\to$ SAT. *Bewertung*
2. Wir definieren eine *Semantik* von Typen relativ zu Bewertungen v; A ist eine Menge von Termen: *Semantik*
   (a) $[\![\alpha]\!]_v := v(\alpha)$,
   (b) $[\![\sigma \to \tau]\!]_v := [\![\sigma]\!]_v \to [\![\tau]\!]_v$,
   (c) $[\![\forall\alpha.\sigma]\!]_v := \bigcap_{A \in \mathrm{SAT}} [\![\sigma]\!]_{v[\alpha \mapsto A]}$.

Lemma 4.9 *Für jeden Typ σ und jede Bewertung v gilt: $[\![\sigma]\!]_v$ ist saturiert (vgl. Lemma 3.7).*

Beweis. Analog zum Beweis von Lemma 3.7. Es wäre noch zu zeigen, dass SAT unter Durchschnitt abgeschlossen ist. Das ist aber trivial. QED

Definition 4.10 Wir definieren *Erfüllbarkeit* wie folgt, wobei für Bewertungen ρ und Interpretationen $[\![M]\!]_\rho$ von Termen das in Definition 3.8 Gesagte gilt: *Erfüllbarkeit*

1. $\rho, v \vDash M:\sigma \;:\Longleftrightarrow\; [\![M]\!]_\rho \in [\![\sigma]\!]_v$;
2. $\rho, v \vDash \Gamma \;:\Longleftrightarrow\; \rho, v \vDash x:\sigma$ für alle $(x:\sigma) \in \Gamma$;
3. $\Gamma \vDash M:\sigma \;:\Longleftrightarrow$ Für alle Bewertungen ρ, v gilt: Wenn $\rho, v \vDash \Gamma$, dann $\rho, v \vDash M:\sigma$.

Lemma 4.11 (Korrektheit) *Wenn $\Gamma \vdash M:\sigma$, dann $\Gamma \vDash M:\sigma$.*

Theorem 4.12 *Wenn $\Gamma \vdash M:\sigma$, dann ist M stark normalisierbar.*

Beweis. Angenommen $\Gamma \vdash M:\sigma$. Dann gilt aufgrund Korrektheit $\Gamma \vDash M:\sigma$. Das heißt, für alle Bewertungen ρ, v gilt: Wenn $\rho, v \vDash \Gamma$, dann $\rho, v \vDash M:\sigma$.
Da $[\![\sigma]\!]_v$ für jede Bewertung v saturiert ist, gilt $\rho_{id}, v \vDash \Gamma$ für jede Bewertung v, wobei $\rho_{id}(x) := x$ für jede Variable x. Also gilt $\rho_{id}, v \vDash M:\sigma$, d. h. $M \in [\![\sigma]\!]_v$. Somit ist M stark normalisierbar. QED

Bemerkungen.

1. Für $\lambda 2$ sind Typbarkeit, Typprüfung und *type inhabitation* unentscheidbar.
2. $\lambda 2$ hat mehr typbare Terme als $\lambda\!\to$.

   Es ist $\lambda x.xx$ ein Beispiel für einen stark normalisierbaren Term, der nicht in $\lambda\!\to$, aber in $\lambda 2$ typbar ist.
3. Ist Typbarkeit in $\lambda 2 =$ Normalisierbarkeit?

   Nein. Jedoch kann jeder Term in Normalform in $\lambda 2$ getypt werden, d. h. es gilt für jeden Term M in Normalform:

   $x_1 : \forall\alpha.\alpha, \ldots, x_n : \forall\alpha.\alpha \vdash M : \sigma$ für ein σ, wobei $x_1, \ldots, x_n$ freie Variablen in M sind.
4. Es ist aber starke Normalisierbarkeit = Typbarkeit in $\lambda\cap$ (System D).

   In $\lambda\cap$ gibt es keine universellen Typen, aber sog. *intersection types* $\sigma \cap \tau$, für die folgende Typisierungsregeln gelten:

   $$(\cap\mathrm{I})\ \frac{\Gamma \vdash M : \sigma \qquad \Gamma \vdash M : \tau}{\Gamma \vdash M : \sigma \cap \tau} \qquad (\cap\mathrm{E})\ \frac{\Gamma \vdash M : \sigma \cap \tau}{\Gamma \vdash M : \sigma} \qquad (\cap\mathrm{E})\ \frac{\Gamma \vdash M : \sigma \cap \tau}{\Gamma \vdash M : \tau}$$

   $M : \sigma \cap \tau$ drückt somit aus, dass M sowohl den Typ σ als auch den Typ τ hat.

   Es ist $\vdash_{\lambda\cap} \lambda x.xx : ((\sigma\to\tau)\cap\sigma)\to\tau$:

   $$\dfrac{(\to\mathrm{E})\ \dfrac{(\cap\mathrm{E})\ \dfrac{(\mathrm{Id})\ \dfrac{}{x : (\sigma\to\tau)\cap\sigma \vdash x : (\sigma\to\tau)\cap\sigma}}{x : (\sigma\to\tau)\cap\sigma \vdash x : \sigma\to\tau} \qquad (\cap\mathrm{E})\ \dfrac{(\mathrm{Id})\ \dfrac{}{x : (\sigma\to\tau)\cap\sigma \vdash x : (\sigma\to\tau)\cap\sigma}}{x : (\sigma\to\tau)\cap\sigma \vdash x : \sigma}}{(\to\mathrm{I})\ \dfrac{x : (\sigma\to\tau)\cap\sigma \vdash xx : \tau}{\vdash \lambda x.xx : ((\sigma\to\tau)\cap\sigma)\to\tau}}$$

Literaturverzeichnis

Henk P. Barendregt: *Lambda calculi with types*, in: S. Abramsky, D. M. Gabbay und T. S. Maibaum (Hrsg.), *Handbook of Logic in Computer Science (vol. 2)*, S. 117–309, Oxford University Press, 1992.

Henk P. Barendregt: *The Lambda Calculus. Its Syntax and Semantics*, College Publications, 2012. (Neuauflage der 2. Auflage von 1984.)

Alonzo Church: *A set of postulates for the foundation of logic*, Annals of Mathematics **33** (1932), 346–366.

Alonzo Church: *A Note on the Entscheidungsproblem*, Journal of Symbolic Logic **1** (1936a), 40–41.

Alonzo Church: *An unsolvable problem of elementary number theory*, American Journal of Mathematics **58** (1936b), 345–363.

Alonzo Church: *A formulation of the simple theory of types*, Journal of Symbolic Logic **5** (1940), 56–68.

Alonzo Church und J. Barkley Rosser: *Some properties of conversion*, Transactions of the American Mathematical Society **39** (1936), 472–482.

Haskell B. Curry: *Grundlagen der Kombinatorischen Logik*, American Journal of Mathematics **52** (1930), 509–536, 789–834.

J. Roger Hindley und Jonathan P. Seldin: *Lambda-Calculus and Combinators, an Introduction*, Cambridge University Press, 2008 (reprinted 2010).

Dag Prawitz: *Natural Deduction. A Proof-Theoretical Study*, Almqvist & Wiksell, 1965. Neuauflage 2006, Dover Publications.

Paul C. Rosenbloom: *The Elements of Mathematical Logic*, Dover Publications, 1950 (reprinted 2005).

Moses Schönfinkel: *Über die Bausteine der mathematischen Logik*, Mathematische Annalen **92** (1924), 305–316.

Moses Schönfinkel und Paul Bernays: *Zum Entscheidungsproblem der mathematischen Logik*, Mathematische Annalen **99** (1929), 342–372.

Raymond Smullyan: *To Mock a Mockingbird. . . . Including an Amazing Adventure in Combinatory Logic*, Oxford University Press, 2000.

Morten H. Sørensen und Paweł Urzyczyn: *Lectures on the Curry-Howard Isomorphism*, Elsevier, 2006.

William W. Tait: *Intensional interpretations of functionals of finite type I*, Journal of Symbolic Logic **32** (1967), 198–212.

Anne S. Troelstra und Helmut Schwichtenberg: *Basic Proof Theory*, 2nd edition, Cambridge University Press, 2001.

Alan M. Turing: *The* $\mathfrak{p}$*-function in* λ*-K-conversion*, Journal of Symbolic Logic **2** (1937), 164.

David A. Turner: *A new implementation technique for applicative languages*, Software: Practice and Experience **9** (1979), 31–49.

Ludwig Wittgenstein: *Tractatus Logico-Philosophicus*, Kegan Paul, 1922.

Sachverzeichnis

α-Konversion, 9
ableitbar
    in $\lambda\!\!\rightarrow$, 46
    in $\lambda 2$, 60
    in $P\!\!\rightarrow$, 56
Ableitung, 31
Abstraktion, 6
allgemeinster Unifikator, 51
Applikation, 6, 35
assoziiertes Gleichungssystem, 52
Atome, 6, 35
Aussagevariablen, 56

β-Gleichheit, 11
β-Kontraktion, 11
β-konvertibel, 11
β-Normalform, 12
β-Redex, 11
β-Reduktion, 11
β-Reduktionsfolge, 11
$\beta\eta$-Gleichheit, 24
$\beta\eta$-Konversion, 24
$\beta\eta$-Reduktion, 24
Basis, 46
Beweis, 31
Bewertung, 49, 63

Church-Typisierung, 45
Church-Ziffern, 24
CL-Term, 35
CLw, 38
confluence, 14
Curry-Howard-Isomorphismus, 55, 57
Curry-Howard-Korrespondenz, 58
Curry-Typisierung, 45

Deklaration, 46

η-Kontraktion, 24
η-Konversion, 23, 24, 42
η-Redex, 24

echter Teilterm, 7, 36
Einschränkung, 47
Erfüllbarkeit, 49, 63

Fixpunktkombinatoren, 22
formale Theorie
    CLw, 38
    $\lambda\beta$, 30
    $\lambda\beta\eta$, 30
    $\lambda\beta\eta_{\triangleright}$, 31
    $\lambda\beta_{\triangleright}$, 31
Formel
    $\lambda\beta$-, 30
    CLw, 38
    implikationslogische, 56
    prädikatenlogische, 34
frei, 7
freie Typvariable, 60
freie Variable, 7
Funktion
    partiell-rekursiv, 28
    primitiv-rekursiv, 26
    rekursiv, 28
    total rekursiv, 28
Funktionstyp, 45, 60

gebunden, 7, 60
gebundene Umbenennung, 9
Generalisierung, 61
geschlossen, 7
Gleichungssystem, 52
Grammatik für Terme, 7

hypothetisches Urteil, 47

Interpretation
    von Termen, 49
    von Typen, 48
intersection types, 64

kategorisches Urteil, 47

Klammerersparnis, 7, 36
Kombinator, 7, 36
    I, 8
    K, 8
    S, 8
    Θ, 23
    ϒ, 23
Konfluenz, 14
Kongruenz, 9
Kontext, 56
Kontraktion
    β-, 11
    η-, 24
    schwache, 36
Kontraktum, 11, 24
Konversion
    α-, 9
    β-, 11
    $\beta\eta$-, 24
    η-, 23, 24, 42
    schwache, 37

$\lambda\beta$, 30
$\lambda\beta\eta$, 30
$\lambda\beta\eta_{\triangleright}$, 31
$\lambda\beta_{\triangleright}$, 31
$\lambda\!\rightarrow$, 46
$\lambda 2$, 60
$\lambda\cap$, 64
λ-Definierbarkeit, 26
λ-Kalkül, 6
    angewandt, 6
    einfach getypt, 45
    polymorph getypt, 60
    rein, 6
    ungetypt, 5
λ-Term, 6
L-Reduktionsfolge, 21
leere Abstraktion, 8
Länge, 7, 36

metasprachliche Variablen, 7, 36
mgu, 51
minimal, 17
minimal complete development, 17

offen, 7
operationelle Semantik, 11, 35

$P\!\rightarrow$, 56
partiell-rekursive Funktion, 28
PL, 34
polymorph getypter λ-Kalkül, 60
polymorpher Typ, 60
positive Implikationslogik, 56
primitiv-rekursive Funktion, 26
Prädikat, 46
Prädikatenlogik erster Stufe, 34

QL-Reduktionsfolge, 22

Redex, 11, 24, 36
    β-, 11
    η-, 24
    schwaches, 36
reducible expression, 11
Reduktion
    β-, 11
    $\beta\eta$-, 24
    schwache, 36
    starke, 42
Reduktionsfolge
    β-, 11
    L- (leftmost), 21
    QL- (quasi-leftmost), 22
    schwache, 37
rekursive Funktion, 28
Residuum, 17

SAT, 48
saturiert, 48
schwache Gleichheit, 37
schwache Konversion, 37
schwache Reduktion, 36
schwache Reduktionsfolge, 37

schwaches Redex, 36
Semantik
    operationelle, 11, 35
    von Typen, 63
Sequenz, 46
SN, 48
Spezialisierung, 61
stark normalisierbar, 12, 37, 48
starke Reduktion, 42
Subjekt, 46
Subjektexpansion, 48
Subjektreduktion, 48, 62
Substitution, 9, 36, 47, 51
syntaktische Identität, 36
System D, 64
System F, 60

Teilterm, 7, 36
    echter, 7, 36
    unmittelbarer, 6, 35
Term
    λ-, 6
    CL-, 35
    einfach getypt, 46
    polymorph getypt, 60
total rekursive Funktion, 28
Typ
    einfacher, 45
    Funktions-, 45
    intersection, 64
    polymorpher, 60
    universeller, 60
typability, 50
typbar, 47
Typbarkeit, 50, 55, 64
type checking, 50
type inhabitation, 50, 64
Typen, 45, 60
Typisierung, 60
Typisierungsalgorithmus, 52
Typprüfung, 50, 55, 64
Typsubstitution, 47
Typvariablen, 45, 60

Unifikationsalgorithmus, 51
Unifikator, 51
universeller Typ, 60
unmittelbarer Teilterm, 6, 35
Urteil, 46
    hypothetisches, 47
    kategorisches, 47

Variante, 51
Vorbereich, 47