Damen und Herren.

(Folie: Titel)

In meiner Dissertation habe ich den Kombinator S untersucht.
Das wichtigste Resultat lautet: Normalisierung ist entscheidbar.
Ich möchte das in der nächsten halben Stunde erläutern.

Der Kombinator S ist ein Basiselement der Kombinatorischen Logik.
Kombinatorische Logik ist ein spezielles Termersetzungssystem.
Ich zeige Ihnen zunächste ein anderes TRS
und veranschauliche daran wichtige Definitionen.

(Folie: TRS)

Ein Term ist ein markierter, geordneter, gerichteter Baum.
Die Blätter sind Konstanten- oder Variablen-Symbole,
die anderen Markierungen sind Funktionssymbole.
Jedes Symbol hat eine fixierte Zahl von Argumenten.
(Beispiel Term zeigen)

Ein Termersetzungssystem ist eine Menge von Regeln,
die beschreibt, wie bestimmte Muster in Termen
durch andere Muster ersetzt werden.
(Beispiel Regel zeigen)

Ein Redex ist ein Teilterm,
auf den die linke Seite einer Ersetzungsregel paßt.
(Beispiel Reduktionsschritt zeigen)
In einem Redex instantiiere ich die Mustervariablen
und ersetze ihn durch die ebenso instantiierte rechte Regelseite.
Auf diese Weise definiert ein TRS eine Relation -> auf Termen.
Die heißt Reduktion.

(Beispiel komplette Reduktion)
Hier ist eine Reduktionskette,
sie endet in einem Term, auf den keine Regel mehr paßt,
der also keinen Redex enthält.
Solch ein Term heißt Normalform.

Vom Startterm aus wären auch andere Reduktionsketten möglich.
In diesem Fall führen aber alle auf die gleiche Normalform.
Das ist wünschenswert, wenn wir TRS als Berechnungsmodelle betrachten:

die Rechnung soll mit einem wohlbestimmten Resultat enden.

Bei beliebigen TRS ist das allerdings nicht so.
Wir klassifizieren die Reduktionsrelation
(nach ihrem Wohlverhalten)

(Folie: Eigenschaften TRS)

Ein Term heißt schwach normalisierend,
wenn er wenigstens eine Normalform hat. (Bild)

Ein Term heißt stark normalisierend,
wenn er keine unendliche Reduktion hat. (Bild)

Wenn wir ein TRS vor uns haben,
dann sind typische Fragestellungen:

- Normalisierung: hat ein Term X eine Normalform
- Wortproblem: sind Term X und Term Y konvertierbar

Letzteres heißt Wortproblem,
weil es zuerst bei Wort-Ersetzungs-Systemen untersucht wurde,
die Halbgruppen definieren.

Ein Ersetzungssystem heißt konfluent,
wenn divergente Teilreduktionen immer zusammenführbar sind. (Bild)

Es ist klar, daß in einem konfluenten System
jeder Term höchstens eine Normalform hat. (Bild)
Konfluente Systeme sind also
in vieler Hinsicht leichter zu handhaben.

Trotzdem sind Normalisierung und Wortproblem inhärent schwer.
Das sehen wir an folgendem Beispiel:

(Folie Kombinatorische Logik)

Terme der Kombinatorische Logik sind binäre Bäume,
die Blätter sind Kombinatoren. (Bild Signatur)

Zu jedem Kombinator gibt es eine Ersetzungsregel.
Die linke Regelseite ist eine Leiter,
die rechte Seite ein Term, der nur aus Variablen besteht.
So etwas nennen wir Kombination.

Die wichtigsten Kombinatoren sehen wir hier (Bild Regeln).
Die Namen bedeuten I - Identität, K - Konstanz, S - Ver"S"chmelzung.

Das S ist das aus dem Thema meiner Dissertation.

Eine Menge von Kombinatoren heißt Basis,
wenn sie jede Kombination herstellen kann. (Beispiel)

Wir koennen sogar auf I verzichten. (Beispiel)
Es war nur aus kulturellen Gründen auf der Folie.

(Folie clsintro)

S-K-Terme können jede Kombination erzeugen.
Damit können wir Terme des Lambda-Kalküls in S-K-Terme übersetzen.
Der Lambda-Kalkül ist aber so mächtig
wie berechenbare Funktionen oder Turing-Maschinen.
Deswegen sind alle interessanten Eigenschaften von S-K-Termen unentscheidbar.
Zum Beispiel Normalisierung und Wortproblem.

Wir sollten uns Terme der Kombinatorischen Logik
also nicht als kodierte Zahlen vorstellen,
sondern als Programmtexte.
Eigenschaften von Programmen (hält es? mit welchem ergebnis?)
sind eben nicht durch andere Programme entscheidbar.

Die Basis (S,K) ist schön klein, und K allein ist trivial.
Jeder K-Term ist stark normalisierend.
Die Schwierigkeit von CL(S,K) scheint also im S zu stecken.
Deswegen fragten Barendregt und Kollegen in den 70er Jahren:
wie schwer ist CL(S)?

Offensichtlich ist CL(S) konfluent. Jedes CL(...) ist konfluent.
Außerdem ist CL(S) nicht löschend, im Gegensatz zu K.
Deswegen stimmen für CL(S) schwache und starke Normalisierung überein:
Wenn ein S-Term eine Normalform hat,
dann führen alle Reduktionspfade dorthin.

Gibt es S-Terme ohne Normalform?

(Folien normalisierendes Bäumchen)

Die erwähnten Holländer setzten
einen Preis von 25 Gulden für einen solchen Term aus.
Barendregt selbst gewann ihn.

(Folien nicht-normalisierendes Bäumchen)

Normalisierung sieht man also den Termen nicht an.

Umso überraschender: ich beweise,
daß es doch entscheidbar ist, ob ein S-Term normalisiert.

Als wesentliches Hilfsmittel dabei benutze ich
rationale Sprachen von Bäumen.

(Folie Rational)

Das sind Mengen von Bäumen,
die ich durch reguläre Grammatiken erzeugen kann.
Beispiele sehen wir hier. (alle Terme, Redexe)

Wichtig ist, daß in den Grammatikregeln
auf der linken Seite immer nur ein Nichtterminal steht.

Wenn wir diese Regeln von rechts nach links lesen,
beschreiben sie die Zustandsübergänge eines endlichen Automaten,
der im Baum von den Blättern nach oben läuft.

Wir kennen das alles von regulären Wort-Sprachen.
Neben der Charakterisierung
durch reguläre Grammatiken und endliche Automaten
haben wir die Beschreibung durch rationale Ausdrücke,
durch eine syntaktische Kongruenzrelation von endlichem Index,
und durch eine Definition im Sinne Medvedevs.

Die klassischen Resultate übertragen sich auf Baumsprachen.
Rationale Sprachen sind effektiv abgeschlossen
unter Vereinigung, Durchschnitt und Komplement.

Dazu müssen wir Automaten deterministisch machen.
Das geht mit der Potenzmengenkonstruktion.
Die ist aufwendig.
Deswegen habe ich ein Programm RX geschrieben,
das solche Rechnungen ausführt.

Für das Beispiel N = M - U gehts aber noch von Hand. (Bild)
Es ist klar, wie ein Term aussehen muß, der keinen Redex enthält.

Ich zeige jetzt, daß bestimmte rationale Mengen von S-Termen
geeignete Reduktionseigenschaften haben.

(Folie FF)

Q1 sind alle Terme außer S selbst,
also alle Bäume mit wenigstens zwei Blättern.

Q2 sind alle Terme außer S und S S,
also alle Bäume mit wenigstens drei Blättern.

Nach den Abschluß-Eigenschaften sind die Mengen Q1 und Q2
als Komplemente endlicher, also rationaler Mengen,
selbst rational. Man kann auch die Grammatik ganz leicht hinschreiben.
Brauchen wir aber nicht.

Wir definieren F. Wir wollen später zeigen,
daß jedes F F eine unendliche Reduktion hat.

Dazu brauchen wir das Lemma.
Beweis (Tafel): Q1 = S S + Q2
also Q1 F = S S F + Q2 F <= S * F ++ Q2 Q1 <= F.   QED.

Nun der Satz.
Beweis (Tafel): erstens. S F * F -> F F (* F) und S * F F -> * F (F F)

oder: Q2 Q1 F. Wenn das Q2 eine unendliche Reduktion hat,
dann sind wir fertig.

Ansonsten hat es Normalform S * oder S * *.

Wenn S *, dann ist der * kein S, also S Q1.
Dann S Q1 Q1 F -> Q1 F (Q1 F) <= F F nach Lemma.

Wenn S * *, dann S * * Q1 F -> * Q1 (* Q1) F <= Q2 Q1 F <= F F.

In jedem Fall ist es ein Schritt, und dann wieder F F.
Das geht also immer so weiter. QED.

Das Beispiel haben wir schon gesehen (frühere Folie).

(Folie N0)

Jetzt suchen wir Terme mit Normalform.
Wir bestimmen eine Klasse N0 von Termen, die ihr Argument nicht aktivieren.
Das soll folgendes bedeuten:

N0 x ist ein Term mit einem Argument (der Variablen x).
Die Argumentposition heißt passiv (da ist das x am Anfang),
die Funktionsposition heißt aktiv.

Nun soll N0 x eine Normalform haben,
die aber keine aktiven x enthält. (Beispiel.)

Wir können nachrechnen,
daß die Menge N0 rational ist,
die Grammatik steht hier.

Jetzt stellen wir uns an Stelle der Variablen x einen Term vor.
Er spielt für die Reduktion überhaupt keine Rolle,
da er nicht aktiviert wird.
Wenn er also eine Normalform war,
dann ist das Resultat auch eine Normalform.
Deswegen folgt der Satz.

(Folie N1)

Jetzt gestatten wir eine Aktivierung,
aber das x soll höchstens ein Argument bekommen,
wie hier im Beispiel.

Wieder ergibt sich, daß die Menge N01 aller solcher Terme
rational ist. Die Grammatik ist etwas komplizierter,
aber überschaubar. Sie enthält natürlich das N0 von vorhin.

Wir stellen uns nun an Stelle von x einen Term vor.
Er wird aktiviert, aber wenn er aus N0 ist,
hat der aktivierte Term (mit Argument) eine Normalform,
nach dem vorigen Satz.

Diese aktivierten Teilterme können geschachtelt sein (Beispiel),
aber das macht nichts. Es dauert nur länger.
Wie lange? Ein Beispiel war auf der allerersten Bilder-Folie.

Wir haben nun rationale Sprachen von Termen mit und ohne Normalform gesehen.

(Folie Decide)

Leider bleibt ein Rest,
für den sich die Normalisierung nicht so direkt angeben läßt.

Dieser Rest ist aber nicht groß.
Mit RX läßt sich ausrechnen, welche Terme dort sind.
Zur Not auch von Hand.

Dann kann ich zeigen, daß nach einer gewissen Anzahl
von Reduktionsschritten ein Term in einer der beiden
bekannten Klassen erreicht wird.
Das gibt ein Verfahren, das Normalisierung entscheidet.

Leider folgt daraus nicht
die Rationalität der Menge aller normalisierenden Terme.
Ich komme gleich darauf zu sprechen.

(Folie Resultate/Offen/Verwandt)

Ich fasse meine Resultate zusammen:

für S-Terme ist Normalisierung entscheidbar.
Das Entscheidungsverfahren benutzt rationale Sprachen.

CL(S) hat keine Grund-Schleifen.
Das schließt bestimmte Muster in unendlichen Reduktionen aus.

CL(S) ist top-terminierend. Ich definiere das jetzt nicht.
Anschaulich bedeutet es, daß unendliche Reduktionen konvergieren.
Der Häufungspunkt ist ein unendlicher Baum,

den kann man als Normalform betrachten.

Diese unendlichen Normalformen kann man endlich darstellen
(indem man Muster angibt),
und über Vergleich dieser Darstellungen
wird hoffentlich das Wortproblem für CL(S) entscheidbar.

Für den wesentlich einfachereren Kombinator L,
der nur zwei Argumente hat,
zeigt Statman die Entscheidbarkeit des Wortproblems.
Ebenso andere Autoren, mit anderen Ideen.
Beide Beweise sind sehr kompliziert,
so ich für S noch Schlimmes befürchte.

Ich habe gezeigt,
daß die Menge aller normalisierenden S-Terme berechenbar ist.
Aus der Konstruktion folgt nicht,
daß sie rational ist.

Ich glaube trotzdem daran, und kann auch eine Grammatik angeben.
Sie hat 38 Nichtterminale und 1002 Regeln.

Die hängt aber von Raten und maschinellem Überprüfen ab.
An dieses Verfahren glaube ich zwar,
aber erwarte das nicht vom Betrachter.

Ich formuliere das Problem um:
Ich suche die Menge der normalisierenden Terme.
Die Menge der Normalformen ist rational.
Ich möchte zusätzlich die Menge aller ihrer Vorgänger
bezüglich der Reduktion in meinem Termersetzungsystem.
Ist diese auch rational? Wie kann ich sie konstruieren?

Die Frage, wie sich TRS zu rationalen Sprachen verhalten,
wird in letzter Zeit von verschiedenen Autoren untersucht.

Es gibt einige Klassen von TRS, die Rationalität effektiv erhalten.
Es funktioniert, grob gesagt, nur dann,
wenn die Ersetzungsregeln keine Variablen kopieren
und ihr Abstand von der Wurzel immer 0 oder 1 ist.
Das ist bei CL(S) aber nicht der Fall.

Trotzdem scheint es in diesem Fall zu funktionieren.
Ich würde gern wissen, woran das "wirklich" liegt.