Algebraische Automaten

(verallgemeinerte Mealy-Automaten)

Jens Doll

Überblick

i) algebraische Grundlagen

ii) von der Maschinensprache zu Funktionen

iii) von Funktionen zu Automaten

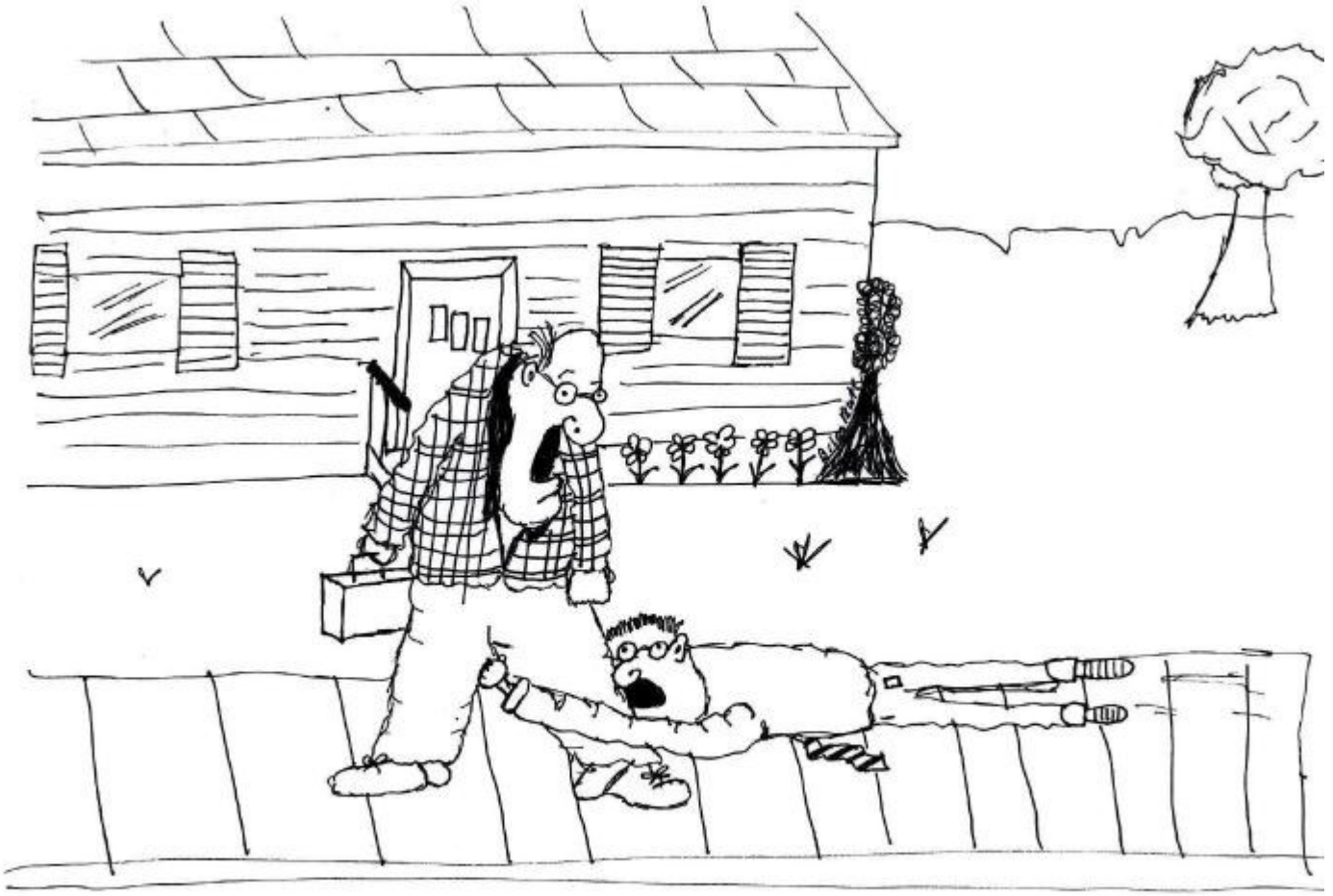
iv) vom Automaten zum regulären Ausdruck

v) vom regulären Ausdruck zum normierten Ausdruck

vI) Einschränkungen

viI) Schlußfolgerungen

Humor bei der NASA



Please, Mr. Owre, add my THEORY to the PVS prelude.

Einordnung und Elemente

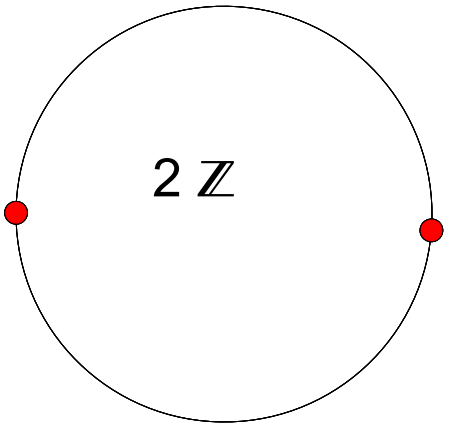
Automatentheorie

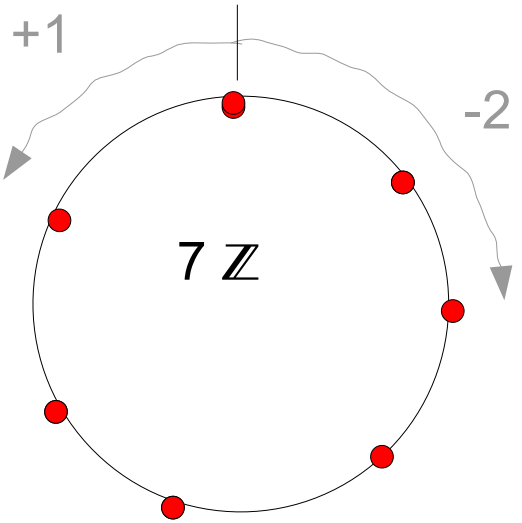
Spezifikationssprachen (Z, CASL, Prolog)

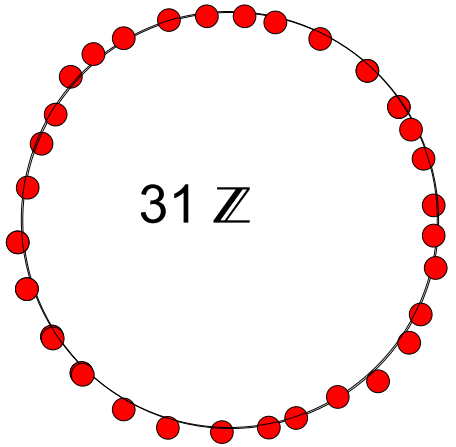
diskrete Mathematik (Galois, Algebra, Zahlentheorie)

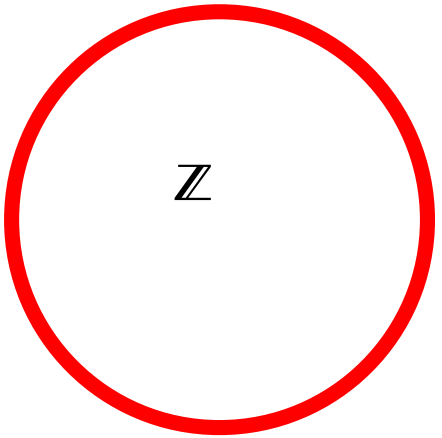
formale Sprachen Ch(0)

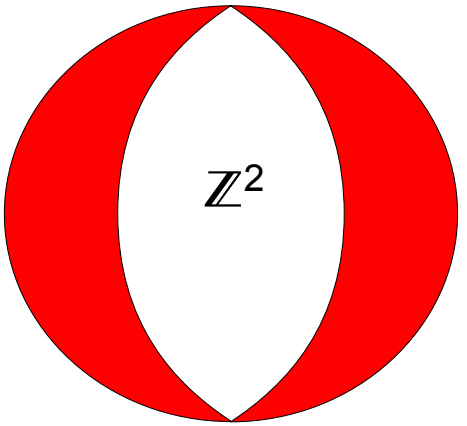
Verschiedene Körper $\mathbb{K}$ für die Trägermenge











ℚ
Quotientenkörper

Körper für T(Σ,x)

boolesche Algebra

Bijektion auf den Restklassenkörper 2 $\mathbb{Z}$

A and B	<=>	A and B
A or B	<=>	(A xor B) xor (A and B)
not A	<=>	A xor A

=> es gibt maximal 2 algebraische Körper in dieser Theorie:

2 $\mathbb{Z}$und $\mathbb{K}$

Kodierung von Zahlen

A = {0,I,-}

-16 = -IIIIIIIIIIIIIII

A = {0,1,2, -, +, *, **}

-16 = -1 - 2*3 - 1*3**2

=>
Analogie zwischen
p-adischen Zahlen
und Polynomen

Abstraktion,
von der Zahl
zur Menge

A = $\mathbb{Q}[x]$

-16 = -1 - 2 * x - 3 * x**2
die Lösbarkeit hängt von der Basis ab

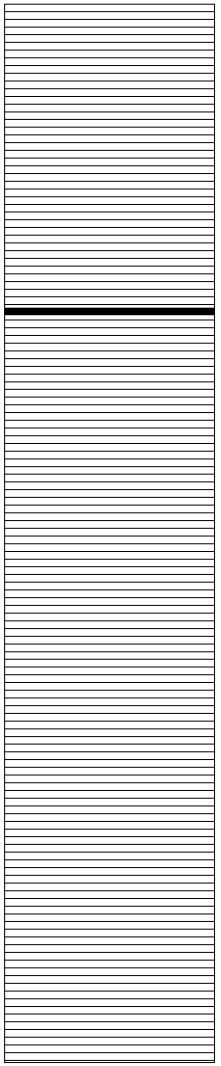
Grundtypen der Programmiersprachen

skalar		
bool	=	$2\ \mathbb{Z}$
int	=	$p\ \mathbb{Z}$
float	=	$p\ \mathbb{Z} \times q\ \mathbb{Z}$
double	=	$p\ \mathbb{Z} \times q\ \mathbb{Z}$
string	=	$\mathbb{N}^0$ (beschränkt)
enum	=	$n\ \mathbb{Z}$

*p,q prim, $n \in \mathbb{N}$

Strukturen
Relationen, Mengen, Listen, Bäume
werden nicht behandelt

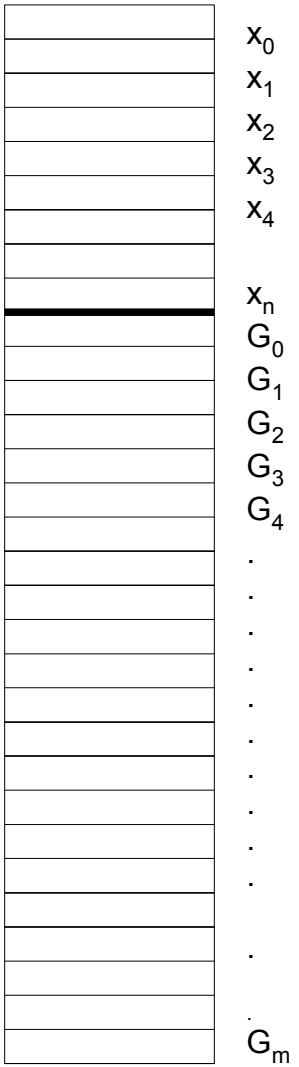
Von Neumann-Maschine*



Daten

Prozeduren

Grundblöcke



$x_i \in \mathbb{K}$

$G_i \in \mathbb{K}^n \longrightarrow \mathbb{K}^n$

*Keine Pointer, keine Stacks, keine Schnittstellen, keine Zustandsbits, keine berechneten Sprünge

Bildung von Vektoren

$$X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ \dots \\ x_{n-1} \\ x_n \end{pmatrix} \qquad G_i(x) = \begin{pmatrix} g_0(x) \\ g_1(x) \\ g_2(x) \\ \dots \\ \dots \\ g_{n-1}(x) \\ g_n(x) \end{pmatrix}$$

$x_i \in \mathbb{K}$

g_i univariate Polynome in x_i , $\mathbb{K}[x_i]$,
x_j für $j \neq i$ sind Konstanten

Grundblöcke normalisieren

Def-Use-Chains

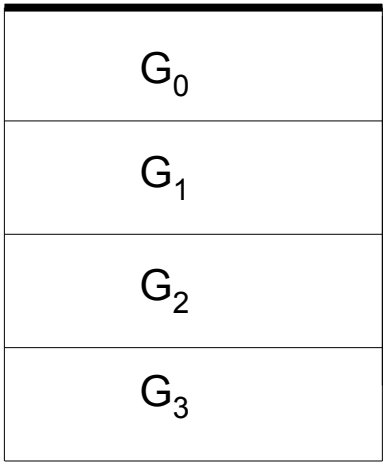
$x_1 = x_2$
$x_3 = 3 * x_1$

=> $x_1 = x_2$
$x_3 = 3 * x_2$

Fazit: dadurch werden alle Zuweisungen

innerhalb eines Grundblockes vertauschbar!

Detail der Grundblöcke



z_0 = lokale Adresse 0

z_1= bedingter Sprung*

z_2= dito

z_3= dito

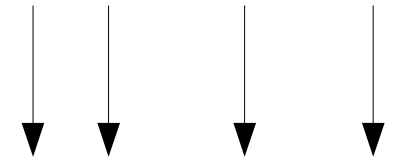
z_4= dito

*maximal 2 Nachfolgezustände

Funktionenraum als Vektorraum

Funktionen und Prädikate bilden sich aus multivariaten Monomen wie

x_1, x_1^2, $x_1 * x_2^5$, x_1^3, ...



v1 v2 v3 v4

Abstraktion auf Vektoren

Definitionsbereich I

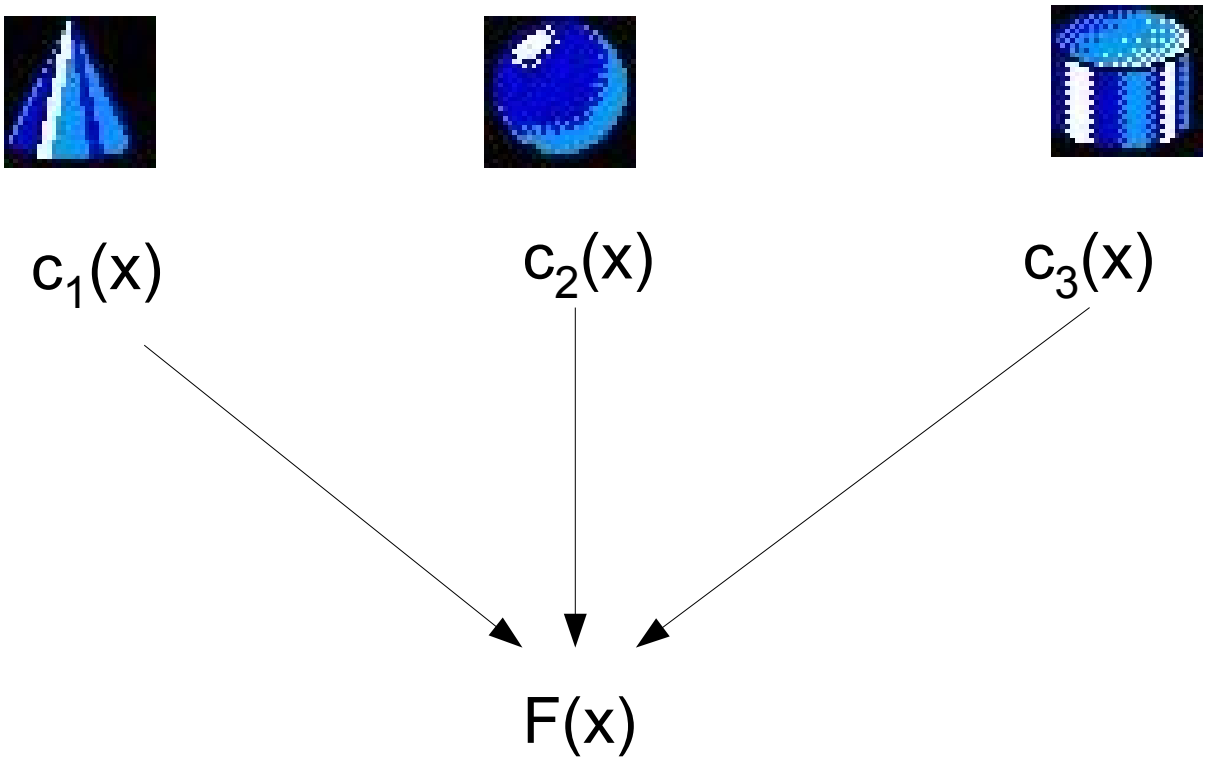
$c_i(x)$ definiert Polyeder im $\mathbb{F}^n$

$$F_i(x) := (c_i(x), G_i(x)) := \begin{pmatrix} c_i(x) \\ \\ g_0(x) \\ g_1(x) \\ g_2(x) \\ \ldots \\ \ldots \\ g_{n-1}(x) \\ g_n(x) \end{pmatrix}$$

bedingte Funktion (≈ guarded command)

Definitionsbereich II

$c_i(x)$ definiert Polytope im Σ^n



Funktionsstrukturen I

Algebra

Operationen	f + g
	f * g
Einselement	0, 1
inverses Element	-f
	1/f

Quotientenkörper

Inverse	f+(-f) = 0
	f/f = 1

Funktionsstrukturen II

Operatoren

Einselement	id
Komposition	f o g
Disposition	f | g
Iteration	f^*
inverses Element	f^{-1}

schiefe Halbgruppe

e.g. Polynome vom Grad < ∞

schiefe Gruppe

e.g. Polynome mit nur ungeraden Exponenten

Funktionenraum I

Alle Funktionen bilden sich aus
multivariaten Monomen wie
x_1, x_1^2, $x_1 * x_2^5$, x_1^3, ...

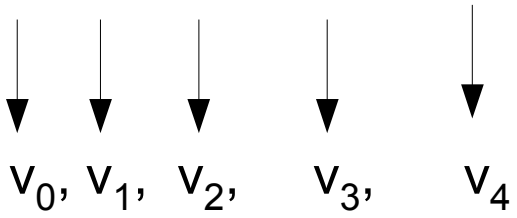
Der daraus entstehende lineare Funktionenraum
wird mit $\mathbb{F}$ bezeichnet
(≈ linearer Raum über $\mathbb{Q}$)

Die Potenzmenge P($\mathbb{F}$) bildet einen
topologischen, meßbaren Raum $\mathbb{P}$
(e.g. Maximumbetragsnorm,
offene Mengen, keine Cauchy-Folgen)

Funktionenraum II

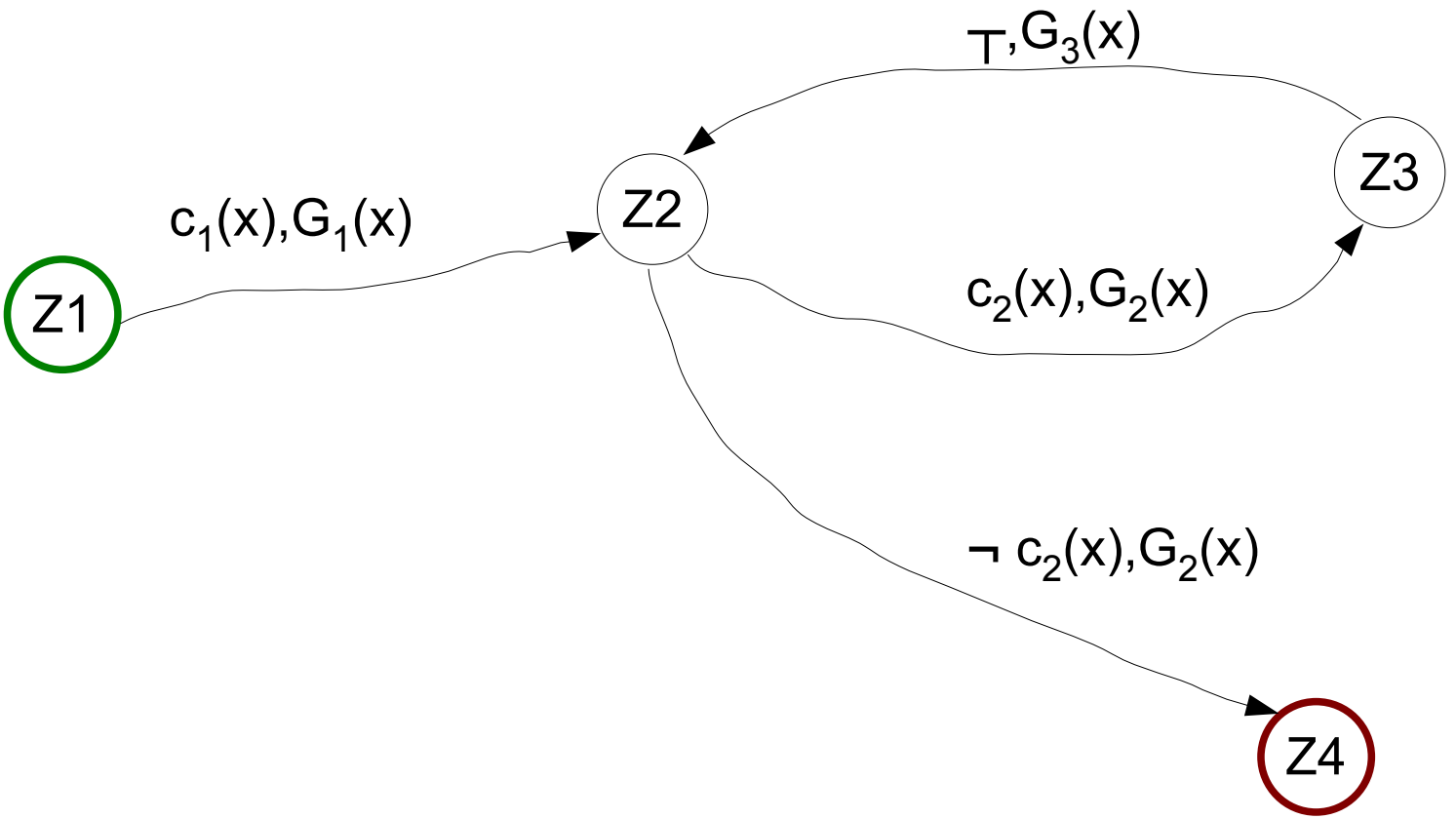
Folgen von Funktionen bilden Polygonzüge

$1, x_1, 2*x_1, 3*x_1^3, 4*x_1, \ldots$ im $\mathbb{F}^n$ (hier $\mathbb{F}^2$)



(Abstraktion auf Vektoren)

Automat mit unendlichem Alphabet



Transitive Hülle der Relation

1) Automat definiert eine Relation

2) Bildung der transitiven Hülle
(e.g. Warshalls Algorithmus)

$$A = F_4 \circ (F_3 \circ F_2)^* \circ F_1 \mid F_4 \circ F_1$$

Reduktion des Ergebnisses

Automat läßt sich eindeutig reduzieren

$A = F_4 \circ F_5 \circ F_1 \mid F_6$

mit reduzierten Funktionen

$F_5 = (F_3 \circ F_2)^* \quad r=x_1^{x2},\ x_2 \in \mathbb{N}$

$F_6 = F_4 \circ F_1 \quad r=0$

Frage zum Automatentyp

Wann ist es

- ein endlicher Automat ?
- eine Kellermaschine ?
- Eine Turingmaschine ?

Formale Definition des Automaten

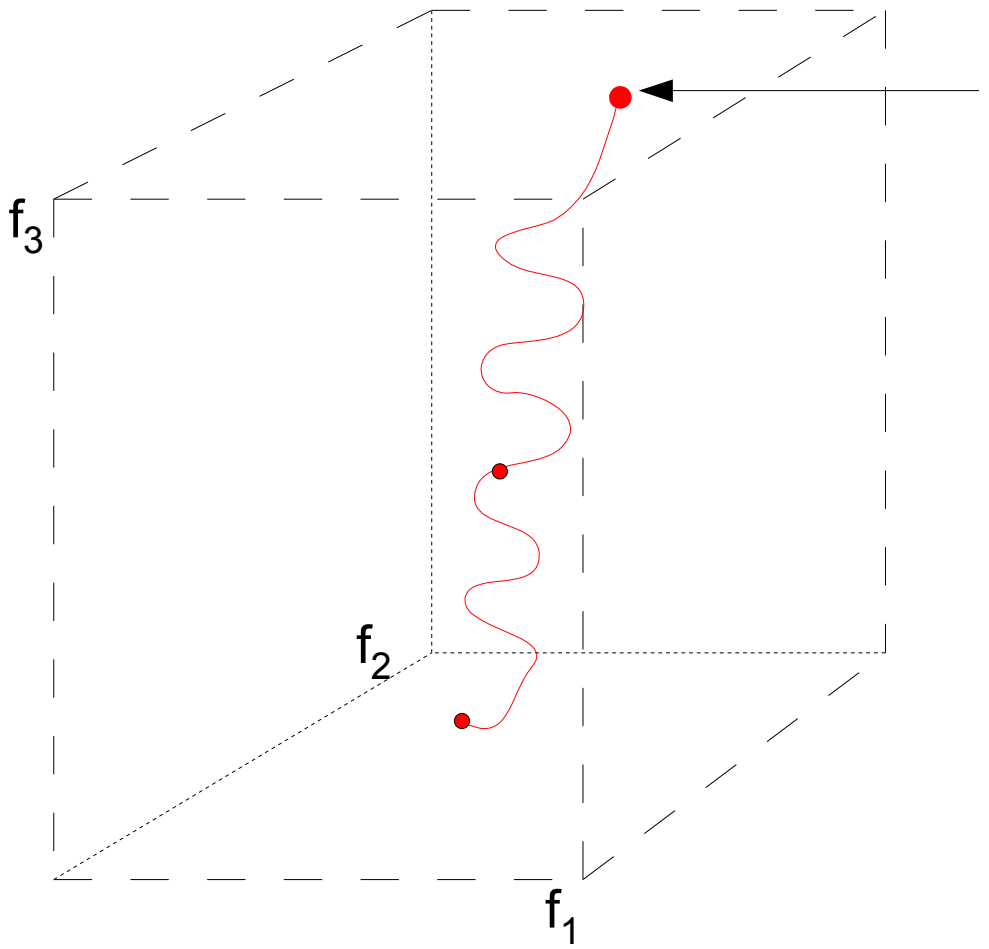
Der algebraische Automat ist ein Tupel ($\mathbb{T}$,$\mathbb{E}$,Z,S,E,U)

$\mathbb{T}$	Trägermenge T(∑,x) mit Funktionen und Prädikaten
$\mathbb{E}$	Körperaxiome und Gleichungen
Z	Zustände
S, E	Startzustände, Endzustände
U	Zustandsübergänge ϵ Z x C(x) x F(x) x Z

mit C(x) T(∑,x) $\longrightarrow$ 2$\mathbb{Z}$

mit F(x) T(∑,x) $\longrightarrow$ T(∑,x)

Polygonzüge im Funktionenraum



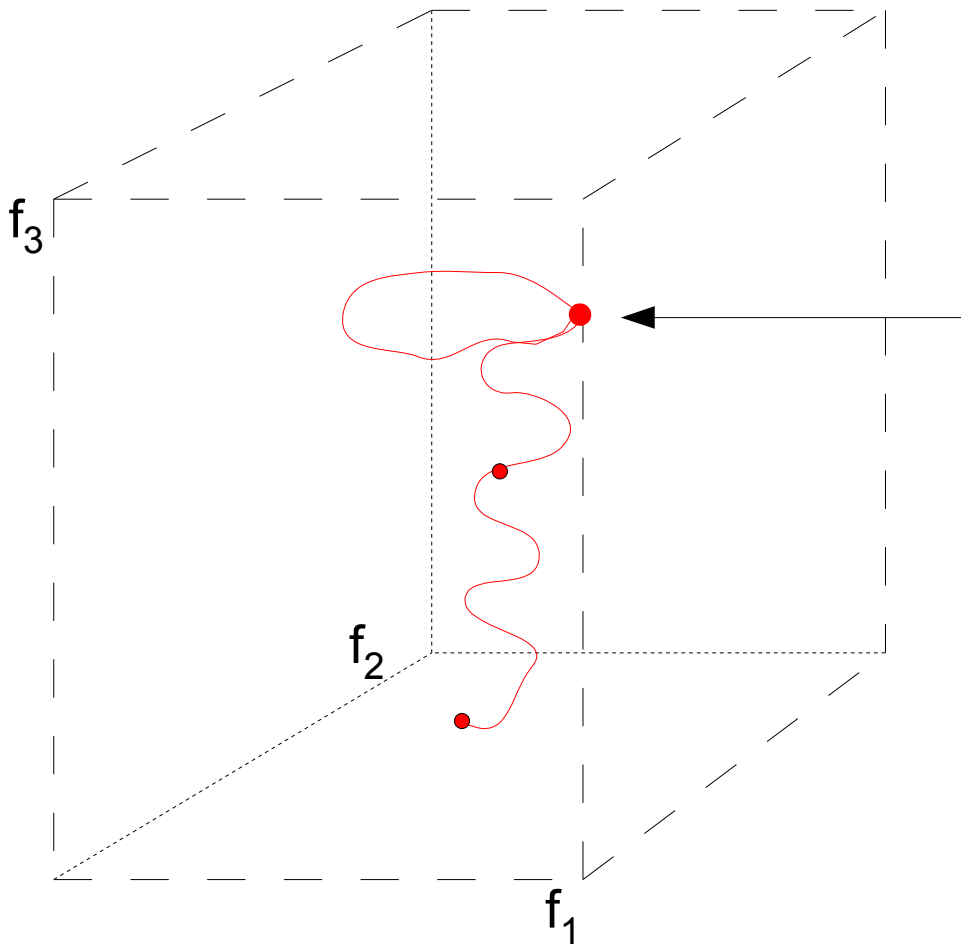
Zielfunktion erreicht

e.g.

$f(x) = x_1^{x2}, x_2 \in \mathbb{N}$

Beispiel eines Polygonzuges im $\mathbb{F}^3$

Polygonzüge im Funktionenraum



Endlose Schleife

Beispiel einer Schleife im $\mathbb{F}^3$

Komposition und Dekomposition I

$$F(x) = \begin{pmatrix} c(x) \\ f_0(x)+g_0(x) \\ f_1(x)+g_1(x) \\ f_2(x)+g_2(x) \\ \dots \\ \dots \\ f_{n-1}(x)+g_{n-1}(x) \\ f_n(x)+g_n(x) \end{pmatrix} = \begin{pmatrix} c(x) \\ f_0(x) \\ f_1(x) \\ f_2(x) \\ \dots \\ \dots \\ f_{n-1}(x) \\ f_n(x) \end{pmatrix} + \begin{pmatrix} c(x) \\ g_0(x) \\ g_1(x) \\ g_2(x) \\ \dots \\ \dots \\ g_{n-1}(x) \\ g_n(x) \end{pmatrix}$$

Komposition und Dekomposition II

$$F(x) = \begin{pmatrix} c(x) \\ f_0(x) \\ f_1(x) \\ f_2(x) \\ \dots \\ \dots \\ f_{n-1}(x) \\ f_n(x) \end{pmatrix} = \begin{pmatrix} c(x) \\ f_0(x) \\ id \\ f_2(x) \\ \dots \\ \dots \\ f_{n-1}(x) \\ f_n(x) \end{pmatrix} \circ \begin{pmatrix} c(x) \\ id \\ f_1(x) \\ id \\ \dots \\ \dots \\ id \\ id \end{pmatrix}$$

konstant in x_i

rekursiv

Komposition und Dekomposition III

$$F(x) = \begin{pmatrix} c(x) \\ f_0(x) \\ f_1(x) \\ f_2(x) \\ \dots \\ \dots \\ f_{n-1}(x) \\ f_n(x) \end{pmatrix} = \begin{pmatrix} c(x) \\ f_0(x) \\ id \\ id \\ \dots \\ \dots \\ f_{n-1}(x) \\ f_n(x) \end{pmatrix} \circ \begin{pmatrix} c(x) \\ id \\ f_2(x) \\ f_1(x) \\ \dots \\ \dots \\ id \\ id \end{pmatrix}$$

konstant in x_i

permutierend

Diskrete Integration - Komponenten

Primitiv rekursive Funktionen können
durch Erzeugende ersetzt werden

e.g.

$f_i(x) = a * f_i(x) \quad \Rightarrow \quad \oint f_i(x)\, dj := F(x,j) = a^j * f_i(x) \qquad j \in N$

$f_i(x) = f_i(x) + g_i(x) \quad \Rightarrow \quad \oint f_i(x)\, dj := F(x,j) = \sum_j g_i(x) + f_i(x) \qquad j \in N$

Anmerkung: $\oint$ ... ist Erzeugende der Folge $f_i(x)$

Diskrete Integration - Randbedingung

Danach können Bedingungen c(x) mit den Erzeugenden der f(x) komponiert werden

aus $n + j < x_1 \wedge x_1 > 0 \wedge j \in \mathbb{N}$

folgt $x_1 > n$ nach Ausführung der Schleife

Statische Berechnung

Nun ist der lokale Wertebereich von x bekannt und es können Ausdrücke untersucht werden

$$\frac{1}{g_i(x)}$$

wenn die Nullstellen von $g_i(x)$ im lokalen Wertebereich liegen

=> 

Termersetzung I

arithmetische Termersetzung

$1 + 0 = 1$
$1 * 1 = 1$

$1 + 1 = 2$

$(2 * 1) \bmod 2 = 0$

$(1 + 1) * (1 - 1) = 0$

ist gleichbedeutend mit der

operationalen Ausführung!

Termersetzung II

Algebraische Termersetzung

und Normalisierung

$a + 0 = a$
$a * 1 = a$

$a + a = 2 * a$

$(2 * a) \bmod 2 = 0$

$(a + 1) * (a - 1) = a^2 - 1$

ist gleichbedeutend mit der

algebraischen Ausführung,

technisch auch Optimierung!

Funktionsklassen

i) reduzierbare

sequentielles Lesen ∀r gilt ...

for-Schleifen ∀i gilt ...

...

ii) nicht reduzierbare

Collatz, Fermat, Ackermann

Quicksort

...

Einschränkungen und Ausblick

Einschränkungen:

- geschlossene Lösung für Polynome mit deg(P(x))>2
- mehr als 2 Bedingungen mit unterschiedlichen $f_i(x)$
- komplexe, μ-Rekursion

...

Zielvorstellung:

Elegantes Beweissystem für

- prozedurale Sprachen
- Hardwaresprachen
- ...

Anwendungen

i) Umwandlung von Goto- in While-Programme
While- in funktionale Programme
ii) Statische Berechnung von Goto-Programmen
While-Programmen
iii) Extraktion von Metadaten (Komplexität, ...)
iv) Vergleich von Programmen
v) Beweissysteme
vI) Grundlage für analytische Berechnungen
p-adische Körper
Intervallarithmetik

und nun ....


Mathematik
Vielen Dank
für Ihre
Aufmerksamkeit
Informatik