

Automaten ber Termen

Jens-Detlev Doll

FACHBEREICH INFORMATIK, UNIVERSITT HAMBURG

*Diese Arbeit ist der Lehrerin Ingrid Doll, *7.7.1922 - †1.12.2009, gewidmet.*

2000 *Mathematics Subject Classification*. Primary: 3D05, Secondary: 68Q05

Mein Dank gilt Prof. Dr. Matthias Jantzen für die fordernde und anspornende
Betreuung.

3D05=Automata and formal grammars in connection with logical questions.

68Q05=Models of computation.

ZUSAMMENFASSUNG. Todo

primitiv rekursive Funktionen neu definieren
Summe+Integral schön gestalten

Inhaltsverzeichnis

Kapitel 1. Einleitung	5
1.1. Die QUID ¹ -Methode	5
1.2. Notation	7
Kapitel 2. Das generative System	9
2.1. Die kontextfreie Syntax	9
2.2. Die Signatur	10
2.3. Gelenke zwischen Programmen	10
2.4. Bauteile von Programmen	10
2.5. Hochsprachliche Entsprechung	11
2.6. Aussagen zu QUID	11
Kapitel 3. Der Folgenraum	13
3.1. Quotientenkörper \mathbb{K}	13
3.2. Logik L über \mathbb{K}	14
3.3. Disjunktive Normalform	14
3.4. Folgenraum über dem Quotientenkörper	14
3.5. Beschränkte Operatoren	14
3.6. undefinierte Funktionen	15
Kapitel 4. Das Automatenmodell	17
4.1. Zerlegung der QUID-Grammatik	17
4.2. äquivalente Automaten zu Calc und Flow	18
4.3. Definition der Automaten	18
4.4. Aussagen zu den Automaten	19
4.5. Implementierung der Automaten	19
Kapitel 5. Das reduktive System	21
5.1. Die Trägermenge	21
5.2. Die Semantik	21
5.3. Die Theorie des Ringes \mathbb{Z}	22
5.4. Normalisierung	22
5.5. Reduktionsordnung	22
5.6. Extension von \mathbb{Z} nach \mathbb{Q}	22
5.7. Arithmetische Termersetzung	23
5.8. Algebraische Termersetzung	23
Kapitel 6. Die funktionale Repräsentation	25
6.1. Einführung von Operatoren	25
6.2. Die Komposition \circ	25
6.3. Die Partition $ $	25
6.4. Die Integration \int	26
6.5. Die QUID-Algebra	27

¹lat.quid=irgend etwas, von qui, quae, quod = welcher, welche, welches

6.6. Inneres Produkt	27
6.7. Dekomposition von Operationen	27
6.8. Ergebnis der Reduktion	28
Kapitel 7. Diskrete Funktionsanalyse	29
7.1. Anwendung der Operatoren	29
7.2. Integration der Komponenten	29
7.3. Geschlossene Lsungen der Integration	30
7.4. Funktionsalternierende Folgen	31
7.5. Substitution von Fehlerintervallen	31
7.6. Eine Beispieltransformation	32
Kapitel 8. Anwendung	33
Kapitel 9. Zusammenfassung	35
9.1. Grenzen der Berechenbarkeit	35
9.2. Grundfragen	35
9.3. Einschrnkungen	36
9.4. Ausblick und Plan	36
Kapitel 10. Anhang	37
10.1. reale Sprachen	37
10.2. Entstehungsgeschichte	38
Index	39
Literaturverzeichnis	41

Einleitung

1.1. Die QUID¹-Methode

In diesem Papier wird ein terminierendes Verfahren beschrieben, welches es ermöglicht, eine definierbare Klasse von Goto-Programmen hnlich wie in [34, p145], konfluent in eine Normalform zu verwandeln. Diese ist eine funktionale Normalform, also unabhngig von Programm- oder Variablenzustnden. Die Normalform ist derart beschaffen, dass man Aussagen ber die vom Programm berechnete Funktion machen und die semantische quivalenz von Programmen beweisen kann. Auch sind damit fr eine Teilmenge der Goto-Programme Terminierungsaussagen mglich. Da die Goto-Programme der Klasse der berechenbaren Funktionen entsprechen, lsst sich das [auf die bestimmte Klasse eingeschrnkte] Verfahren auf andere konstruktive Varianten bertragen. Gegenber der abstrakten Interpretation [10] grenzt sich das Verfahren dadurch ab, dass eine direkte Transformation und keine Iteration oder Approximation durchgefhrht wird. Zu den blichen Verifikationsverfahren [15] besteht der Unterschied, dass kein Vergleich mit einer Spezifikation erfolgt, sondern eine Denotation des Probanden erstellt wird. Zur operationalen Welt, realen Maschinen, besteht die Abgrenzung, dass die gesamten rationalen Zahlen statt der Restklassenarithmetik (bzw. n-Bit-Arithmetik) betrachtet werden. Das Thema Fehlerrechnung wird hier nur rudimentr behandelt. Gegenber den Verfahren der Softwaremessung besteht der Unterschied, dass Schwachstellen im Quellcode deterministisch und nicht statistisch berechnet werden knnen.

Die zu betrachtende Transformation findet im Raum algebraischer Funktionen und nicht im Zustandsraum konkreter Zahlen statt. Es ist eine algebraische Ausfhrung von Goto-Programmen. Sie ist auf Maschinensprachen, HDL oder JAVA bertragbar und ihr Zeit- und Speicherbedarf steht in direktem Zusammenhang mit dem analysierten Quelltext. Im weiteren Verlauf erfolgt eine Einschrnkung des allgemeinen Verfahrens auf eine einzelne algebraische Struktur, einen Ring R , um das Gesamtkonzept besser darstellen zu knnen.

Diagrammatisch ergibt sich folgende bersicht

¹lat. quid=irgend etwas, von qui, quae, quod = welcher, welche, welches



wobei $PL(1)$ die Sprache erster Stufe bedeuten soll.

1.2. Notation

$l \in \{i, o, j\}$	Verweis auf Quell-, Ziel- oder Mustersprache
T_l	das terminale Alphabet
N_l	das nonterminale Alphabet
M_l^*	das freie Monoid von $M_l = T_l \cup N_l$
Σ	die Signatur aus Funktionen und Relationen
con, rel, uni, op	Signaturelemente
x	Variablenvektor $(x_1 \dots x_n)$
x_i	skalare Komponente von x
$T(\Sigma, x)$	die Trgermenge
\rightarrow	Abbildung zwischen zwei Mengen
\Rightarrow	Reduktionsrelation
\Rightarrow^*	transitive Hlle von
$\llbracket \dots \rrbracket_l$	syntaktischer Ausdruck, Satzform
\equiv	syntaktische quivalenz
$=$	semantische quivalenz
$\mathbb{N}, \mathbb{Z}, \mathbb{Q}$	mgliche Trgermengen
$\mathbb{Q}[x]$	multivariater Polynomring ber \mathbb{Q}
id	die Identittsfunktion
$c_j(x)$	skalare Bedingung, Hyperffche
$f_i(x)$	vektorwertige Funktion
$f_{ij}(x)$	skalare Komponentenfunktion von $f_i(x)$
$FV(A), BV(A)$	freie und gebundene Variablen aus A
$\text{Reg}(F)$	quasiregulrer Ausdruck ber F
\exists	Existenzquantor
\forall	Allquantor
$G_i(x)$	Grundblock mit der Markierung i
$F = (c_i(x), G_j(x))$	Transition des Automaten, Kombinator, guarded command
$\oint f_{ij}(x) dx_j$	das skalare diskrete Integral
$\oint f_i(x) dx$	das vektorwertige diskrete Integral
$\oint_{e(x)} f_{ij}(x) dx_j$	das bestimmte diskrete Integral

... incomplete, to be continued ...