RIJKSUNIVERSITEIT GRONINGEN

Inductive Types in Constructive Languages

Proefschrift

ter verkrijging van het doctoraat in de Wiskunde en Natuurwetenschappen aan de Rijksuniversiteit Groningen op gezag van de Rector Magnificus Dr. F. van der Woude in het openbaar te verdedigen op

vrijdag 24 maart 1995 des namiddags te 4.00 uur

door

Peter Johan de Bruin

geboren op 20 september 1964 te Harderwijk

Promotor: Prof. dr. G. R. Renardel de Lavalette

Preface

The fascination for mathematical truth has been the driving force for my research as it had been for my master's thesis written in Nijmegen. Being amazed at the lack of a general language for the irrefutable expression of mathematical argument, I looked for it in the direction of what is called Type Theory. I started my research in Groningen in 1988 under supervision of Roland Backhouse, and enjoyed his discussion club with Paul Chisholm, joyous Grant Malcolm, Albert Thijs, and broadly interested Ed Voermans. When Backhouse moved to Eindhoven in 1990, I and Thijs remained in Groningen and our roads parted.

The advent in 1991 of Gerard Renardel who accepted to take up my supervision with fresh interest, started a new period of seeking to assemble all available pieces. I owe him much for his constant trust and support in keeping up courage, his help in formulating ideas, his effort to curtail outgrowing branches, and for leaving the choice to carry this work through entirely to me. I also thank Jan Terlouw for his modest cooperation, and my fellow Ph.D. students for their good company and friendship.

Now my heart has turned from abstract truth to living Truth, and since 1993 I'm living in community *Agapè* of the Blessed Sacrament Fathers in Amsterdam. I am glad to thank its members, Aad, Eugène, Gerard, Herman, Jan, Paul, Pieter, and Theo, for their support while finishing this thesis. I thank the members of the Ph.D. committee, Roland C. Backhouse (Eindhoven), Wim H. Hesselink (Groningen), Gerard R. Renardel de Lavalette (Groningen), and Michel Sintzoff (Louvain-la-Neuve), for accepting this duty and providing kind comments on the manuscript, especially Wim who gave detailed comments which helped me to prepare the final text. Though it is tough material, my presentation sometimes being terse and not all ideas given sufficient scientific support, I hope you will get a catch of its beauty.

Peter de Bruin

Contents

Summary 5

1 Introduction 7
- 1.1 Mathematical language . . . 7
- 1.2 Our approach to mathematical language . . . 8
- 1.3 Type Theory and Set Theory . . . 10
- 1.4 Related efforts . . . 11
- 1.5 Relational calculus . . . 12
- 1.6 ADAM's Type Theory . . . 12
- 1.7 Aspects of induction and recursion . . . 13
- 1.8 Frameworks for studying induction . . . 14
- 1.9 Our treatment of induction and recursion . . . 15
- 1.10 Other kinds of inductive types . . . 16
- 1.11 Original contributions . . . 16

2 The language ADAM 18
- 2.1 Language definition mechanism . . . 18
- 2.2 Basic grammar of ADAM . . . 20
- 2.3 Production rules . . . 23
  - 2.3.1 Terms . . . 23
  - 2.3.2 Patterns . . . 23
  - 2.3.3 Definitions . . . 24
  - 2.3.4 Declarations . . . 25
  - 2.3.5 Coercion . . . 25
- 2.4 Types and universes . . . 26
- 2.5 Products and function spaces . . . 27
- 2.6 Sums and declaration types . . . 29
- 2.7 Families and quantifiers . . . 31
- 2.8 Finite types . . . 31
- 2.9 Infinite types . . . 33
- 2.10 Equality predicate . . . 33
- 2.11 The type of propositions . . . 34
- 2.12 More derived notions . . . 36
  - 2.12.1 Predicates . . . 36
  - 2.12.2 Subtypes . . . 36

2.12.3 Subsets . . . 37
2.12.4 Relational notations . . . 37
2.12.5 Currying . . . 38
2.12.6 Pattern matching . . . 39
2.12.7 Linear proof notation . . . 39
2.13 Conclusion . . . 39

3 Common induction and recursion principles **40**
3.1 Examples of inductive types . . . 40
3.2 More on natural numbers . . . 43
3.3 Inductive subset definitions . . . 44
3.3.1 Sets inductively defined by rules . . . 44
3.3.2 The well-founded part of a relation . . . 45
3.3.3 Inductive definitions as operators . . . 47
3.3.4 Fixed points in a lattice . . . 47
3.4 From induction to recursion . . . 48
3.5 Conclusion . . . 49

4 Categories and algebra **50**
4.1 Categorical notions . . . 50
4.2 Algebras and signatures . . . 53
4.3 Initial algebras, catamorphisms . . . 54
4.4 Algebras with equations . . . 57
4.5 Initial algebras related to well-founded relations . . . 60
4.6 An aside: monads . . . 61
4.7 Algebraic Specification . . . 63
4.8 Concluding remarks . . . 64

5 Specifying inductive types **65**
5.1 Single inductive types . . . 65
5.1.1 Operator domains . . . 65
5.1.2 Operators with arity . . . 67
5.1.3 The wellordering of a single inductive type . . . 68
5.2 Mutually inductive types . . . 68
5.2.1 Using an exponential category . . . 69
5.2.2 Plain algebra signatures . . . 70
5.3 Production rules for polynomial functors . . . 71
5.3.1 Positive type expressions . . . 72
5.3.2 A type of polynomial functors . . . 72
5.4 Adding equations . . . 72
5.5 Conclusion . . . 73

6 Recursors in constructive type theories **74**
6.1 Algebraic recursion, or paramorphisms . . . 74
6.2 Recursive dependent functions . . . 76
6.3 Mendler's approach . . . 78

6.4 Recursors for mutual induction and recursion . . . 81
6.5 Summary . . . 83

7 Co-inductive types 85
7.1 Dualizing F-algebras . . . 85
7.2 Anamorphism schemes . . . 87
7.3 Dual recursion . . . 89
7.4 Dual equations . . . 90
7.5 Terminal interpretation of equations . . . 90
7.6 Conclusion . . . 91

8 Existence of inductively defined sets 92
8.1 Using transfinite ordinal induction . . . 92
8.2 Kerkhoff's proof . . . 94
8.3 Algebras with equations . . . 96

9 Partiality 98
9.1 Domain theory . . . 98
9.2 Optional objects . . . 101
9.3 Building recursive cpo's by co-induction . . . 103
9.4 Recursive object definitions . . . 104
9.5 Conclusion . . . 106

10 Related subjects 107
10.1 Impredicative type theories . . . 107
10.1.1 Weak initial algebras . . . 108
10.1.2 Weak final algebras . . . 109
10.2 Using type-free values . . . 109
10.2.1 Henson's calculus TK . . . 109
10.3 Inductive universe formation . . . 110
10.4 Bar recursion . . . 112

11 Reflections and conclusion 113
11.1 Mathematical language . . . 113
11.2 Constructive Type Theory . . . 114
11.3 Language definition mechanism . . . 115
11.4 Proofs and proof notation . . . 116
11.5 Inductive types . . . 117
11.6 Directions for further research . . . 118

A Set theory 120
A.1 ZFC axioms . . . 120
A.2 Set encodings . . . 121
A.3 Ordinals . . . 122
A.4 Cardinals . . . 122
A.5 A model of ZFC . . . 123

A.6 An inductive model of ZFC . . . 124
A.7 Anti-foundation . . . 125

B ADAM's Type Theory **126**
B.1 Abstract syntax . . . 126
B.2 Meta-predicates . . . 127
B.3 Universes . . . 129
B.4 Products . . . 129
B.5 Sums . . . 130
B.6 Finite types . . . 130
B.7 Naturals . . . 130
B.8 Equality . . . 131
B.9 Existential propositions . . . 131
B.10 Semantics . . . 132
B.11 More derived notations . . . 134

C Proof elimination in Type Theory **135**
C.1 Introduction . . . 135
C.2 The basic system . . . 137
C.3 Strong existence . . . 137
C.3.1 New rules . . . 137
C.3.2 Difficulties with reduction to canonical form . . . 138
C.4 Applications . . . 139
C.4.1 Iota . . . 139
C.4.2 Quotient types . . . 139
C.4.3 Inductive types . . . 142
C.5 Conclusion . . . 143

D Naturality of Polymorphism **144**
D.1 Introduction . . . 144
D.2 Polymorphic typed lambda calculus . . . 146
D.3 Turning type constructors into relation constructors . . . 147
D.4 Naturality of expressions . . . 148
D.5 Applications . . . 151
D.6 Dinatural transformations . . . 153
D.7 Second-order languages . . . 154
D.8 Overloaded operators . . . 155

Index **156**

Bibliography **160**

Samenvatting (Dutch summary) **165**

Inductieve Typen in Constructieve Talen

Samenvatting

Deze dissertatie gaat over constructieve talen: talen om wiskundige constructies formeel in uit te drukken. Het begrip *constructie* omvat niet alleen berekeningen, zoals die in een programmeertaal kunnen worden uitgedrukt, maar ook beweringen en bewijzen, zoals die in een wiskundige logica kunnen worden uitgedrukt, en in het bijzonder de constructie van gestructureerde wiskundige objecten zoals rijtjes en bomen. *Typen* kan men zich voorstellen als klassen van zulke objecten, en *inductieve typen* zijn typen waarvan de objecten gegenereerd worden door productieregels.

Het doel van deze dissertatie is tweeledig. Ten eerste ben ik op zoek naar talen waarin de wiskundige zijn inspiraties goed gestructureerd, correct, en toch zo vrij mogelijk kan uitdrukken. Ten tweede wil ik de uiteenlopende benaderingen van inductieve typen in één kader samenbrengen, zodat men kan zien hoe de diverse constructie- en afleidingsregels uit een enkel basis-idee voortvloeien en ook hoe deze regels eventueel gegeneraliseerd kunnen worden. Als basis-idee gebruik ik het begrip *initiële algebra* uit de categorieëntheorie.

Mijn onderzoek naar wiskundige talen heeft niet tot een afgerond voorstel geleid. De huidige presentatie beperkt zich tot algemene overwegingen en een deels formele, deels informele beschrijving van een taal, ADAM. Deze dient vervolgens als medium voor de studie van inductieve typen, die het hoofdbestanddeel van de dissertatie vormt.

De opzet van ADAM is als volgt. Om de geldigheid van de in de taal geformuleerde argumenten te garanderen, behoeft deze een degelijke grondslag. Hiervoor stel ik een constructieve type-theorie ATT samen, een combinatie van de "Intuitionistic Theory of Types" van P. Martin-Löf en de "Calculus of Constructions" van Th. Coquand. Teneinde alle wiskundige redeneervormen te kunnen omvatten, voeg ik de iota- of descriptie-operator van Frege toe. Het is niet noodzakelijk om inductieve typen als basisprincipe op te nemen; natuurlijke getallen volstaan om deze te construeren.

Op deze grondslag bouw ik vervolgens de taal ADAM door te bezien hoe we constructies en bewijzen die ik tegenkwam of zelf opstelde zo natuurlijk mogelijk maar wel volgens de regels van typetheorie kon opschrijven. De formele definitie van ADAM, voor zover beschikbaar, en haar semantiek in termen van de onderliggende type-theorie worden gelijktijdig gegeven door een twee-niveau-grammatica. Dit maakt het in beginsel mogelijk de taal naar behoefte met behoud van geldigheid uit te breiden met notaties of deeltalen voor speciale toepassingen, zoals programmacorrectheid. De voorgestelde notaties dienen dan ook niet als onaantastbaar te worden beschouwd. Het enige kenmerkende taalelement is wellicht de notatie voor (en het consistente gebruik van) *families* van objecten.

Als voorbereiding op inductieve typen geef ik eerst de klassieke benaderingen van inductieve definities weer, waarna ik de benodigde machinerie in ADAM introduceer – de beginselen van categorieëntheorie en algebra.

De kern van de verhandeling wordt gevormd door de beschrijving en rechtvaardiging van inductieve typen als initiële algebra's. Eerst beschouw ik op abstract niveau de

diverse manieren waarop inductieve typen gespecificeerd kunnen worden en hoe deze specificaties (in de vorm van een polynomiale functor) een algebra-signatuur bepalen, eventueel met gelijkheden. Vervolgens analyseer en generaliseer ik de manieren waarop recursieve functies op een inductief type gedefinieerd kunnen worden. Dan bezie ik in hoeverre deze constructieprincipes gedualiseerd kunnen worden tot co-inductieve typen, ofwel finale co-algebra's. Ten slotte construeer ik, uitgaande van hetzij elementaire verzamelingenleer of typetheorie, daadwerkelijk initiële algebra's en finale co-algebra's voor een willekeurige polynomiale functor, en bewijs daarmee de relatieve consistentie van alle beschreven constructieprincipes ten opzichte van ADAM's typetheorie ATT.

Het voorgaande wordt aangevuld met de behandeling van enkele aan inductieve typen verwante onderwerpen. Ten eerste zijn dat recursieve datatypen met partiële objecten, zoals die in programmeertalen voorkomen waarbij men rekening moet houden met mogelijk niet-terminerende programmadelen. Ik vat de benodigde domeintheorie samen, en construeer zulke domeinen in ADAM uitgaande van finale co-algebra's. Verder bespreek ik kort inductieve typen in impredicatieve talen, typen als verzameling van type-vrije objecten, en het principe van bar-recursie, en doe ik een suggestie voor de inductieve definitie van nieuwe type-universa binnen een typetheorie. Ten slotte geef ik enkele verdere overwegingen over wiskundige taal en bewijsnotatie, en vat de benaderingen van inductieve typen samen.

De appendices bevatten de basisprincipes van de verzamelingenleer en van ATT, de benodigde toevoeging van de iota-operator ofwel bewijs-eliminatie aan typetheorie, en een studie naar uniformiteits-eigenschappen (*natuurlijkheid*) van polymorfe objecten, die ik in enkele gevallen nodig heb.

Omslag-diagram

Aanschouw het wiskundig universum,
zich ontwikkelend van oorspronkelijke eenheid
tot categorische dualiteit.

De centrale straal bevat
het initiële en het finale type,
samen met de overige platte eindige typen.

Zij worden geflankeerd door de duale principes
van gegeneraliseerde som en product
en van initiële en finale dekpunt-constructie.

STELLINGEN

behorende bij het proefschrift

Inductive Types in Constructive Languages

1. Vers 18:62 van de Bhagavad Gita geeft een zeer goede verklaring van het woord *Islam* (overgave), wat verwant is met *Salam* (vrede):

   > *Zoek dan uw toevlucht in Hem en geef uzelf met geheel uw hart aan Hem over, dan zult ge door Zijn genade tot de Opperste vrede komen en het Eeuwig Tehuis bereiken.* [Vertaling: Stichting school voor filosofie / Amsterdam]

2. Het wrede schijn-oordeel van koning Salomo [1 Koningen 3:25],

   > *"Snijdt het levende kind in tweeën en geeft de helft aan de ene en de helft aan de andere vrouw",* [Vertaling: NBG 1951]

   geeft de sleutel tot een rechtvaardig printer-toewijzingsalgoritme: versnipper de beschikbare afdrukregels in schijn gelijkmatig over alle afdrukopdrachten, rekening houdend met hun tijdstip van indienen, maar wijs de printer dan toe aan de opdracht die het eerst voltooid zou zijn.

3. De afbeelding op de omslag van dit proefschrift symboliseert het wiskundig universum.

4. De uiteindelijke betekenis van de informatietechnologie ligt niet in de producten die zij levert maar in de denkwijze die zij ons leert.

5. Poëtische zowel als mathematische inspiratie vindt het best uitdrukking in een taal die weinig beperkingen oplegt.

6. Als men in een constructieve taal wederzijds inductieve typen opneemt, dient men te letten op het onderscheid tussen de verschillende vormen van toegestane algebra-specificatie [dit proefschrift, sectie 5.2]. Evenzo dient men bij het definiëren van een recursieprincipe over wederzijds inductieve typen te letten op het onderscheid tussen "standaard recursie" [regel (6.2)] en "liberale recursie" [regel (6.11)].

7. De jaarlijks toenemende watervloed vervult deze functies: hij leert ons saamhorigheid, offervaardigheid en ongehechtheid, en hij bereidt ons zachtjes voor op de mogelijkheid van ingrijpende veranderingen in onze wereldordening.

8. Muziek draagt de essentie van het leven over aan de ziel die luistert.

9. De wereld beweegt zich onweerstaanbaar naar de heelheid.

Peter J. de Bruin