

Contents

| | | |
|-----------|---|-----------|
| I | Introduction and Background | 1 |
| 1 | Introduction | 3 |
| 1.1 | The Power of Types | 3 |
| 1.2 | Real World Problems | 5 |
| 1.3 | The Contributions of the Thesis | 8 |
| 1.4 | The Structure of the Thesis | 9 |
| 2 | The Formal Background of Free Theorems | 11 |
| 2.1 | Free Theorems for the Simply Typed Lambda Calculus | 11 |
| 2.1.1 | Definition of the Simply Typed Lambda Calculus | 11 |
| 2.1.2 | Relational Parametricity and Free Theorems | 21 |
| 2.2 | Adding General Recursion | 26 |
| 2.2.1 | Changes to the Calculus | 26 |
| 2.2.2 | Changes to the Parametricity Results | 32 |
| 2.3 | Adding Selective Strictness | 34 |
| 2.3.1 | Changes to the Calculus | 35 |
| 2.3.2 | Changes to the Parametricity Results | 37 |
| 2.4 | Explicit Type Abstraction and Instantiation | 40 |
| 3 | State of the Art | 43 |
| 3.1 | Theoretical Developments of Free Theorems | 43 |
| 3.1.1 | Results Building on a Standard Denotational Semantics | 44 |
| 3.1.2 | Results Building on an Operational Semantics | 47 |
| 3.1.3 | Formalization of Parametricity in Pure Type Systems | 50 |
| 3.2 | Applications of Free Theorems | 51 |
| II | New Results | 53 |
| 4 | Exemplifying the Necessity of Strictness Conditions | 57 |
| 4.1 | The Calculus | 60 |
| 4.2 | Refined Typing | 61 |
| 4.3 | An Alternative System of Typing Rules | 66 |
| 4.4 | Terms that Give Rise to Counterexamples | 69 |
| 4.4.1 | Term Generation via TermFind: Strategy and Definition | 69 |
| 4.4.2 | Detailed Explanations on the Design of TermFind | 74 |
| 4.4.3 | Properties of TermFind | 80 |
| 4.5 | Generation of Complete Counterexamples | 84 |

| | | |
|------------|---|------------|
| 4.5.1 | Choosing Type and Relation Environments | 88 |
| 4.5.2 | Requirements for Term Environments | 89 |
| 4.5.3 | Restrictions to TermFind | 94 |
| 4.5.4 | Creating Extra Information — Concrete Constructions | 95 |
| 4.5.5 | Correctness of ExFind | 105 |
| 4.5.6 | A Closer Look on Completeness | 107 |
| 4.5.7 | The Implementation of ExFind | 110 |
| 4.6 | Summary | 111 |
| 4.7 | Outlook | 113 |
| 5 | Taming Selective Strictness | 115 |
| 5.1 | Motivation for a Refined Type System | 117 |
| 5.2 | A Refined Type System | 120 |
| 5.2.1 | A Former Refinement Approach | 120 |
| 5.2.2 | The New Approach | 121 |
| 5.3 | Improvement of the Algorithmic Properties | 128 |
| 5.3.1 | Guarantee of Termination | 129 |
| 5.3.2 | Allowing Non-Refined Input | 131 |
| 5.3.3 | Finding Optimal Annotations | 136 |
| 5.4 | The Implemented Algorithm | 139 |
| 5.5 | Summary | 140 |
| 5.6 | Outlook | 142 |
| 6 | Looking at Quantitative Aspects | 145 |
| 6.1 | The Calculus | 148 |
| 6.2 | An Instrumented Semantics for Counting Costs | 150 |
| 6.3 | Parametricity Theory Involving Costs | 153 |
| 6.4 | The Parametricity Theory at Work | 159 |
| 6.4.1 | Simple Examples | 162 |
| 6.4.2 | Considering <i>foldr</i> / <i>build</i> | 167 |
| 6.5 | Summary | 175 |
| 6.6 | Outlook | 176 |
| III | Conclusion | 177 |
| 7 | Conclusion | 179 |
| 7.1 | Consideration of Programming Features | 179 |
| 7.2 | Parametricity Enables Efficiency Assertions | 181 |
| IV | Appendix | 183 |
| A | Proofs | 185 |
| A.1 | Proofs from Chapter 4 | 185 |
| A.2 | Proofs from Chapter 5 | 194 |
| A.3 | Proofs from Chapter 6 | 205 |
| | Bibliography | 215 |
| | Index | 225 |

Figures

| | | |
|------|---|-----|
| 2.1 | Type and term syntax of λ^α | 13 |
| 2.2 | Typing rules of λ^α | 16 |
| 2.3 | Type semantics of λ^α | 18 |
| 2.4 | Term semantics of λ^α | 18 |
| 2.5 | Logical relation for λ^α | 22 |
| 2.6 | Type and term syntax of $\lambda_{\text{fix}}^\alpha$ | 27 |
| 2.7 | Typing rules of $\lambda_{\text{fix}}^\alpha$ | 27 |
| 2.8 | Type semantics of $\lambda_{\text{fix}}^\alpha$ | 31 |
| 2.9 | Term semantics of $\lambda_{\text{fix}}^\alpha$ | 31 |
| 2.10 | Logical relation for $\lambda_{\text{fix}}^\alpha$ | 33 |
| 2.11 | Type and term syntax of $\lambda_{\text{seq}}^\alpha$ | 36 |
| 2.12 | Typing rules of $\lambda_{\text{seq}}^\alpha$ | 36 |
| 2.13 | Type semantics of $\lambda_{\text{seq}}^\alpha$ | 36 |
| 2.14 | Term semantics of $\lambda_{\text{seq}}^\alpha$ | 37 |
| 2.15 | Logical relation for $\lambda_{\text{seq}}^\alpha$ | 38 |
| 4.1 | Type and term syntax of $\lambda_{\text{fix}^+}^\alpha$ and $\lambda_{\text{fix}^*}^\alpha$ | 60 |
| 4.2 | Type semantics of $\lambda_{\text{fix}^+}^\alpha$ and $\lambda_{\text{fix}^*}^\alpha$ | 61 |
| 4.3 | Term semantics of $\lambda_{\text{fix}^+}^\alpha$ and $\lambda_{\text{fix}^*}^\alpha$ | 61 |
| 4.4 | Logical relation for $\lambda_{\text{fix}^+}^\alpha$ and $\lambda_{\text{fix}^*}^\alpha$ | 61 |
| 4.5 | Class membership rules for Pointed in $\lambda_{\text{fix}^*}^\alpha$ | 62 |
| 4.6 | Typing rules of $\lambda_{\text{fix}^*}^\alpha$ | 64 |
| 4.7 | Term search rule system | 67 |
| 4.8 | Phase I rules of TermFind | 72 |
| 4.9 | Phase II rules of TermFind | 73 |
| 4.10 | Phase III rules of TermFind | 73 |
| 4.11 | An example run for TermFind | 84 |
| 4.12 | ExFind's output for Example 16 as presented by the web interface | 87 |
| 4.13 | The constructions of ExFind for input $(\alpha, \beta^*, (\alpha \rightarrow \beta) \rightarrow [\beta])$ | 91 |
| 4.14 | The output of the web interface for ExFind | 112 |
| 5.1 | Visualization of the function <i>foldl</i> | 116 |
| 5.2 | Different versions of <i>foldl</i> | 117 |
| 5.3 | Annotated type syntax of $\lambda_{\text{seq}^*}^\alpha$ | 123 |
| 5.4 | Refined typing rules of $\lambda_{\text{seq}^*}^\alpha$ | 123 |

| | | |
|------|---|-----|
| 5.5 | Class membership rules for Seqable in $\lambda_{\text{seq}^*}^\alpha$ and $\lambda_{\text{seq}^+}^\alpha$ | 123 |
| 5.6 | Subtyping rules of $\lambda_{\text{seq}^*}^\alpha$ | 124 |
| 5.7 | Logical relation for $\lambda_{\text{seq}^*}^\alpha$ | 126 |
| 5.8 | Different <i>foldl</i> -versions in $\lambda_{\text{seq}}^\alpha$ -syntax | 128 |
| 5.9 | Typing rules of $\lambda_{\text{seq}^+}^\alpha$ | 131 |
| 5.10 | Schematic overview of the final algorithm for refined typing | 132 |
| 5.11 | Conditional typing rules of $\lambda_{\text{seq}^C}^\alpha$ | 134 |
| 5.12 | Conditional class membership rules for Seqable in $\lambda_{\text{seq}^C}^\alpha$ | 134 |
| 5.13 | Conditional subtyping rules of $\lambda_{\text{seq}^C}^\alpha$ | 134 |
| 5.14 | Conditional equality rules of $\lambda_{\text{seq}^C}^\alpha$ | 135 |
| 5.15 | Example derivation with the typing rules of $\lambda_{\text{seq}^C}^\alpha$ | 138 |
| 5.16 | Syntax of the implementation of the type refinement algorithm | 140 |
| 5.17 | Output of the web interface for <i>foldl''</i> | 141 |
| 6.1 | Comparison of evaluation costs under different evaluation strategies | 147 |
| 6.2 | Type and term syntax of $\lambda_{\text{fold}}^\alpha$ | 149 |
| 6.3 | Typing rules of $\lambda_{\text{fold}}^\alpha$ | 149 |
| 6.4 | Type semantics of $\lambda_{\text{fold}}^\alpha$ | 149 |
| 6.5 | Term semantics of $\lambda_{\text{fold}}^\alpha$ | 150 |
| 6.6 | Logical relation for $\lambda_{\text{fold}}^\alpha$ | 150 |
| 6.7 | Instrumented type semantics with embedded costs | 151 |
| 6.8 | Instrumented term semantics with costs | 152 |
| 6.9 | Logical relation with embedded costs | 154 |
| 6.10 | Fully cost-lifted logical relation | 159 |
| 7.1 | Investigation of program features | 181 |