LFCS

Laboratory for Foundations of Computer Science
Department of Computer Science - The University of Edinburgh

Errata and Remarks

by

David J. Pym

LFCS Report Series **ECS-LFCS-93-265**

LFCS **May, 1993**

Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

ERRATA AND REMARKS

David J. Pym
University of Edinburgh
Scotland, U.K.*

Abstract

We give a list of errata, as known at the date of this report, for our University of Edinburgh LFCS Reports, published in the period March 1990 – May 1993, including those authored jointly with Lincoln Wallen (University of Oxford) and those authored jointly with James Harland (University of Melbourne). We include also such clarifying or illuminating remarks as have occurred to us. In chronological order, the reports are: ECS-LFCS-90-111; ECS-LFCS-90-124; CST-69-90 (ECS-LFCS-90-125); ECS-LFCS-91-146; ECS-LFCS-91-168; ECS-LFCS-92-212; ECS-LFCS-92-229; ECS-LFCS-92-246; ECS-LFCS-92-248. This report should be regarded as a companion to the aforesaid publications. Most of these papers have been published elsewhere; where appropriate, full publication details are given. Whilst all of the given (and known) errata are either typographical or minor omissions or oversights, we believe it is our scholarly responsibility to record all such available information.

1 Investigations into Proof-search in a System of First-order Dependent Function Types: ECS-LFCS-90-111, March 1990

David Pym and Lincoln Wallen, Investigations into Proof-search in a System of First-order Dependent Function Types. Proc. 10th International Conference on Automated Deduction, M. Stickel (editor), Kaiserslautern, FRG, July 1990. Lecture Notes in Artificial Intelligence 449, 236–250, Springer-Verlag, 1990. Also University of Edinburgh LFCS Report ECS-LFCS-90-111.

1. In the second paragraph after the proof of Lemma 5.7, "$f : B(a) \to B(a)$" should be "$f : B(a) \to A$".

2. In the third line of the second paragraph of Definition 5.10, "of those variables declared" should be "of those universal variables in $v\sigma$ declared".

*Author's address from 1 June, 1993: School of Computer Science, The University of Birmingham, Edgbaston, Birmingham B15 2TT, England, U.K.

3. In the first sentence after the proof of Lemma 5.9, "ground" should inserted immediately before "instantiation".

4. In Definition 5.10, the sentence beginning with "Let $w_1, w_2, \ldots, w_n, \ldots$ " should be "Let $w_1, w_2, \ldots, w_n$ be an enumeration of those universal variables declared in $D_0(v) = I_\epsilon(v), \biguplus_{\mathrm{FV}(v\sigma)}(I_\sigma(u), u : T_\sigma(u))$ such that if $w_i \lhd_{\psi,\sigma} w_j$ then $i < j$."

5. In Definition 5.10, "$D_0(v) = I_\epsilon(v) \vdash_\Sigma T_\epsilon(v) : \mathrm{Type}$" should be "$D_0(v) = I_\epsilon(v), \biguplus_{\mathrm{FV}(v\sigma)}(I_\sigma(u), u : T_\sigma(u)) \vdash_\Sigma T_\epsilon(v) : \mathrm{Type}$".

6. In the sixth and seventh lines of the second paragraph of § 7, "any ground extension σ' of" should be deleted and "$\Gamma' \Rightarrow_\Sigma A'$" should be "$\Gamma \Rightarrow_\Sigma A$".

2 The Uniform Proof-theoretic Foundation of Linear Logic Programming: ECS-LFCS-90-124, November 1990

1. On p. 2, in the line immediately after the displayed text, "$D ::= \ldots$", 'constants of linear logic and" should be deleted.

2. In Definition 2.5, on p. 8, "$\mathbf{0} \mid \mathbf{1} \mid \top \mid \bot \mid$", in the case for L, should be deleted.

3. On p. 21, two lines above Proposition 3.16, "dericted" should be "directed".

4. On p. 21, in Proposition 3.16, the first sentence should be "Let Ξ be a proof of $\Gamma \vdash \Delta$ in which Γ is a multiset of definite formulae and Δ is a multiset of goal formulae."

5. On p. 26, in the penultimate line, "$q, q\text{–}\circ p \vdash p$" should be "$\{q, q\text{–}\circ p\} \vdash_R p$'.

6. In Definition 6.1, on p. 32, "$\mathbf{0} \mid \mathbf{1} \mid \top \mid \bot \mid$" in the case for L, should be deleted.

7. On p. 38, in the 4th line, "$\{q(a), !(q(a)\text{–}\circ p(a))$" should be "$\{q(a), !(q(a)\text{–}\circ p(a))\}$".

8. On p. 38, in the 6th line, "$\{q(a), !(q(a)\text{–}\circ p(a))$" should be "$\{q(a), !(q(a)\text{–}\circ p(a))\}$" and "$\vdash_R p(a)$" should be "$\vdash_R q(a)$".

9. In the proofs of Lemmas 6.11, 6.12, 6.13 and 6.15 and Propositions 6.16 and 6.17, some of the brackets { and } and some of the subscripts R are missing from the antecedents of sequents of the form $\mathcal{P} \vdash_R \mathcal{G}$. All such omissions are obvious.

10. In Appendix A, on pp. 57 and 58, the side-condition on the quantifier rules should be "where y is not free in Γ, Δ."

3 Proofs, Search and Computation in General Logic: CST-69-90 (ECS-LFCS-90-125), November 1990

1. On p. 3, "encouragment" should be "encouragement".

2. The reference [Sm68], on pp. 14 and 76, is missing from the bibliography. It is: Smullyan, R.M. *First-order logic*. Ergebnisse der Mathematik, Volume 43, Springer-Verlag, 1968.

3. On p. 29, in the 7th line, "Types and type families" should be "Types and Families of Types".

4. On p. 32, in the bottom line, each of the the Bs, As and Ms should be decorated with $'$.

5. On p. 47, in the third line of the Proof sketch, "$M : A_{n+1}$ true" should be "$M : A$ true'.

6. On p. 69, in each of the last two lines, "N_0/x_0" should be "N_1/x_1".

7. On p. 70, in each of the fourth and fifth lines, "N_0/x_0" should be "N_1/x_1" and "x_{m-1}" should be "x_{m-2}".

8. On p. 70, in the seventh line, "Δ_0" should be "Δ_1", "x_0" should be "x_1" and "N_1" should be "N_2". On the ninth line, "N_0" should be "N_1".

9. On p. 72, in the second and third last lines, "x_m" should be "x_{m-1}", "N_0/x'_0" should be "N_1/x_1" and "N_{m-1}/x'_{m-1}" should be "N_{m-1}/x_{m-1}".

10. On p. 73, in the third and fourth lines, "x_{m-1}" should be "x_{m-2}", "N_0/x'_0" should be "N_1/x_1" and "N_{m-2}/x'_{m-2}" should be "N_{m-2}/x_{m-2}".

11. On p. 73, in the sixth line, "Δ_0" should be "Δ_1", "x_0" should be "x_1" and "N_1" should be "N_2". On the eighth line, "N_0" should be "N_1".

12. On p. 90, in the tenth line "$\rightarrow l$" should be "$(\rightarrow l)$".

13. On p. 91, in the seventh line "Πl" should be "(Πl)".

14. On p. 107, in the third line, "Spring" should be "spring".

15. In Definition 5.3.2 on p. 116, in the first indented paragraph, "a pair" should be "pairs", the second "either" should be deleted and "or **succeed** of **fail**" should be moved to the end of the paragraph.

16. In the bottom line of p. 124 and in the first and third lines of p. 125, each of the As and Bs should be followed immediately by σ.

17. On p. 126, the seventh and eighth lines should be

$$\begin{aligned}\tau_r(f) = F \;\; = \;\; & \Pi u'_1 : A'_1 \ldots u'_{r_1} : A'_{r_1} \\ & \Pi u'_{r_1+1} : A'_{r_1+1} \ldots u'_{s_1} : A'_{s_1}.cP_1 \ldots P_r.\end{aligned}$$

Similarly, the tenth and eleventh lines should be

$$\begin{aligned}\tau_r(g) = G \;\; = \;\; & \Pi v'_1 : A''_1 \ldots v'_{r_2} : A''_{r_2} \\ & \Pi v'_{r_2+1} : A''_{r_2+1} \ldots v'_{s_2} : A''_{s_2}.cP_1 \ldots P_r.\end{aligned}$$

Similarly, the nineteenth line should be

$$M_f \equiv_{\text{def}} \lambda w_1 : A'_1 \ldots \lambda w_{s_1} : A'_{s_1}[w_1/u'_1, \ldots, w_{s_1-1}/u'_{s_1-1}].w_f$$

and the twenty-second line should be

$$M_g \equiv_{\text{def}} \lambda w'_1 : A''_1 \ldots \lambda w'_{s_2} : A''_{s_2}[w'_1/v'_1, \ldots, w'_{s_2-1}/v'_{s_2-1}].w_g.$$

18. In the last line of p. 126, "A_{r_1+1}" should be "A'_{r_1+1}" and "A_{s_1}" should be "A'_{s_1}". Similarly, in the first line of p. 127, "$w_{r_2+1} : A_{r_2+1}$" should be "$w'_{r_2+1} : A''_{r_2+1}$".

19. On p. 153, in the twelfth line, "below" should be "above".

20. On p. 156, in the twenty-first line, "$f : B(a) \rightarrow B(a)$" should be "$f : B(a) \rightarrow A$".

21. On p. 157, in the second line, "Px_3" should be "px_3".

22. On p. 159, on the twentieth line, the first sentence should be "Let ψ be compatible with the ground instantiation σ."

23. On p. 160, in the fourth line, the sentence beginning with "Let $w_1, w_2, \ldots, w_n, \ldots$" should be "Let $w_1, w_2, \ldots, w_n$ be an enumeration of those universal variables declared in $D_0(v) = I_\epsilon(v), \biguplus_{FV(v\sigma)}(I_\sigma(u), u : T_\sigma(u))$ such that if $w_i \lhd_{\psi,\sigma} w_j$ then $i < j$." Similarly, "$D_0(v) = I_\epsilon(v) \vdash_\Sigma T_\epsilon(v)$: Type" should be "$D_0(v) = I_\epsilon(v), \biguplus_{FV(v\sigma)}(I_\sigma(u), u : T_\sigma(u)) \vdash_\Sigma T_\epsilon(v)$: Type".

24. On p. 166, the fourteenth line should end with a full stop rather than with a semicolon.

25. On p. 169, in the twenty-first line, "$f : B(a) \rightarrow B(a)$" should be "$f : B(a) \rightarrow A$".

26. On p. 172, in the sixteenth line, "$\mathcal{S}^{C(\Sigma)}$" should be "$\mathcal{S}^{C(\Sigma)^{op}}$".

27. On p. 182, in the statement of Lemma 7.3.13, "$I_\perp$" should be "$\perp$".

28. On p. 186, in the twelfth line, "$\hom_D(s,-)$" should be "$\hom_D(-,r)$" and "$\hom_D(-,r)$" should be "$\hom_D(s,-)$".

29. (p. 187) We remark that $Y/=_{\beta\eta} = Y$, since $C(\Sigma)$ is constructed from normal forms. Of course, in the contruction of other models, similar quotients are not necessarily trivial.

30. On p. 192, the reference [GB84] is missing from the Bibliography. It is: Goguen, J.A., Burstall, R.M. *Institutions: abstract model theory for computer science*. CSLI Report CSLI-85-30. Stanford University, 1985.

31. On p. 194, the reference [Go79] is missing from the Bibliography. It is: Goldblatt, R. *Topoi: the categorical analysis of logic*. North-Holland, 1984.

32. On p. 207, "f^*B" (fourth line) should be "$f^*\Delta$", "$q(f,B)$" (sixth line) should be "$q(f,\Delta)$" and the range context of "eval_Γ" should be "Γ" rather than "Δ".

4 Proof-search in the $\lambda\Pi$-calculus: ECS-LFCS-91-146, April 1991

David Pym and Lincoln Wallen, Proof-search in the $\lambda\Pi$-calculus. In: Logical Frameworks, G. Huet and G. Plotkin (editors), Cambridge University Press, 1991, 309-340. Also University of Edinburgh LFCS Report ECS-LFCS-91-146.

1. In Example 6.3, p. 22,"$T(\beta) = A$" should be "$T(\beta) = B(a)$".

2. We remark that the definition of D_0 (Definition 6.4, p. 22) taken in this report is equivalent to that of ECS-LFCS-90-111 (this document, § 1, ¶¶ 4, 5) because if $v \in \mathrm{Eig}(\psi)$ then $v\sigma$ is empty.

5 The Uniform Proof-theoretic Foundation of Linear Logic Programming (Extended Abstract): ECS-LFCS-91-168, June 1991

James Harland and David Pym, The Uniform Proof-theoretic Foundation of Linear Logic Programming (Extended Abstract). In: Proc. International Symposium on Logic Programming, V. Saraswat and K. Ueda (editors), MIT Press, 1991, 304-318. Also University of Edinburgh LFCS Report ECS-LFCS-91-168.

1. In Appendix A, the side-condition on the quantifier rules should be "where y is not free in Γ, Δ."

6 On Resolution in Fragments of Classical Linear Logic: ECS-LFCS-212, May 1992

J.A. Harland and D.J. Pym, On Resolution in Fragments of Classical Linear Logic. Proc. LPAR '92, A. Voronkov (editor), Lecture Notes in Artificial Intelligence 624, 30-41, Springer-Verlag, 1992. Also University of Edinburgh LFCS Report ECS-LFCS-92-212.

1. In Definition 3.10, the following, in the second sentence, should be deleted:

   > , and is such that the resolution rule is applied only when no other rule is applicable

   As printed, this phrase can be interpreted to require that all of the elements of the goal be reduced to atoms before a resolution step is permitted. Such an interpretation, a strong condition, is permissible in the absence of par (multiplicative disjunction) in definite formulae, as found in earlier work of the authors on a similar topic (ECS-LFCS-90-124, ECS-LFCS-91-168). Unfortunately, we failed to delete the condition in the LPAR '92 presentation. In fact, this phenomenon is just a manifestation of the failure of full permutability for the universal and existential quantifiers — full permutability holds just in certain fragments of the logic.

2. In Appendix A, the side-conditions on the quantifier rules should be "(y not free in Γ, Δ)".

3. On p. 13, (1), and p. 15, (4), each occurrence of ", $(q_1 \otimes q_2) \multimap s$" on the right hand branches above the conclusion should be deleted.

4. On p. 15, in the proof figure numbered (4), ", $p \multimap q$" should be deleted from the leftmost leaf. (Its presence is harmless, but unintended.)

7 A Unification Algorithm for the λΠ-calculus: ECS-LFCS-92-229, August 1992

David J. Pym, A Unification Algorithm for the λΠ-calculus. Int. J. of Foundations of Computer Science, Vol. 3, No. 3 (1992), 333-378. Also University of Edinburgh LFCS Report ECS-LFCS-92-229.

1. In the footnote on p. 1, "Spring" should be "spring".

2. The third paragraph from the bottom of p. 2 should end with a semicolon rather than with a full stop.

3. In Theorem 2.11 on p. 11, the range context of "eval_Γ" should be "Γ" rather than "Δ".

4. In Definition 3.2 on p. 18, in the first indented paragraph, "a pair" should be "pairs", the second "either" should be deleted and "or succeed of fail" should be moved to the end of the paragraph.

5. In reference [28] on p. 40, "Semantics" should be "Categorical Semantics".

8 Logic Programming via Proof-valued Computations: ECS-LFCS-92-246, November 1992

David J. Pym and Lincoln A. Wallen, Logic Programming via Proof-valued Computations. Proc. 4th UK Conference on Logic Programming, London, 1992 (K. Broda, editor), 253-262, WICS, Springer, 1992. Also University of Edinburgh LFCS Report ECS-LFCS-92-246.

1. § 6, p. 8: in clause T4, the subscript on "HYP" should be ψ, not ϕ.

2. We remark that the results of § 6 apply to Horn clauses only. Although it is stated in § 3 that we are concerned with clauses and SLD-resolution, this should have been emphasized at the beginning of § 6. We remark also that in our claim, in the last sentence of § 4, that the two derivations of Figure 2 are permutation variants of each other, we have elided some minor subtleties. We further remark that in the first paragraph of § 6, we use the terms "embed" and "embedding" informally.

9 A Synopsis on the Identification of Linear Logic Programming Languages: ECS-LFCS-92-248, November 1992

1. On p. 4, (2), ", $(q_1 \otimes q_2) \multimap s$" should be deleted from the right hand premiss. Similarly, ", $(q_1 \otimes q_2) \multimap s$" should be deleted from the right hand branch above the conclusion of the figure at the bottom of p. 4.

Acknowledgments

We are grateful to our colleagues and to several referees for their remarks.

Insert

In this insert we give a summary of the proof systems presented in the body of the thesis. The number and/or label of each rule in this insert corresponds exactly to the first number given to that rule in the body of the thesis.

The System N

We first give the rules for the system without an explicit $\rightarrow$ and then give the rules for an explicit $\rightarrow$.

Valid Signature

$$\frac{}{\vdash \langle\rangle \text{ sig}} \tag{2.1}$$

$$\frac{\vdash \Sigma \text{ sig} \quad \vdash_\Sigma K \text{ kind} \quad c \notin \text{Dom}(\Sigma)}{\vdash \Sigma, c : K \text{ sig}} \tag{2.2}$$

$$\frac{\vdash \Sigma \text{ sig} \quad \vdash_\Sigma A\text{: Type} \quad c \notin \text{Dom}(\Sigma)}{\vdash \Sigma, c : A \text{ sig}} \tag{2.3}$$

Valid Context

$$\frac{\vdash \Sigma \text{ sig}}{\vdash_\Sigma \langle\rangle \text{ context}} \tag{2.4}$$

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad \Gamma \vdash_\Sigma A\text{: Type} \quad x \notin \text{Dom}(\Gamma)}{\vdash_\Sigma \Gamma, x : A \text{ context}} \tag{2.5}$$

Valid Kinds

$$\frac{\vdash_\Sigma \Gamma \text{ context}}{\Gamma \vdash_\Sigma \text{Type kind}} \tag{2.6}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma K \text{ kind}}{\Gamma \vdash_\Sigma \Pi x : A.K \text{ kind}} \tag{2.7}$$

Valid Elements of a Kind

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad c : K \in \Sigma}{\Gamma \vdash_\Sigma c\text{:} K} \tag{2.8}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma B\text{: Type}}{\Gamma \vdash_\Sigma \Pi x : A.B\text{: Type}} \tag{2.9}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma B\text{:} K}{\Gamma \vdash_\Sigma \lambda x : A.B\text{:} \Pi x : A.K} \tag{2.10}$$

$$\frac{\Gamma \vdash_\Sigma B\text{:} \Pi x : A.K \quad \Gamma \vdash_\Sigma N\text{:} A}{\Gamma \vdash_\Sigma BN\text{:} K[N/x]} \tag{2.11}$$

$$\frac{\Gamma \vdash_\Sigma A\text{:} K \quad \Gamma \vdash_\Sigma K' \text{ kind} \quad K =_{\beta\eta} K'}{\Gamma \vdash_\Sigma A\text{:} K'} \tag{2.12}$$

Valid Elements of a Type

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad c : A \in \Sigma}{\Gamma \vdash_\Sigma c : A} \tag{2.13}$$

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad x : A \in \Gamma}{\Gamma \vdash_\Sigma x : A} \tag{2.14}$$

$$(\Pi\text{I}) \quad \frac{\Gamma \vdash_\Sigma A : \text{Type} \quad \Gamma, x : A \vdash_\Sigma M : B}{\Gamma \vdash_\Sigma \lambda x : A.M : \Pi x : A.B} \tag{2.15}$$

$$(\Pi\text{E}) \quad \frac{\Gamma \vdash_\Sigma M : \Pi x : A.B \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma MN : B[N/x]} \tag{2.16}$$

$$\frac{\Gamma \vdash_\Sigma M : A \quad \Gamma \vdash_\Sigma A' : \text{Type} \quad A =_{\beta\eta} A'}{\Gamma \vdash_\Sigma M : A'} \tag{2.17}$$

Additional Rules for Explicit $\rightarrow$s

Valid Kinds

$$\frac{\Gamma \vdash_\Sigma A : \text{Type} \quad \Gamma, x : A \vdash_\Sigma K \text{ kind} \quad x \notin \text{FV}(K)}{\Gamma \vdash_\Sigma A \rightarrow K \text{ kind}} \tag{2.18}$$

Valid Elements of a Kind

$$\frac{\Gamma \vdash_\Sigma A : \text{Type} \quad \Gamma, x : A \vdash_\Sigma B : K \quad x \notin \text{FV}(K)}{\Gamma \vdash_\Sigma \lambda x : A \,.\, B : A \rightarrow K} \tag{2.19}$$

$$\frac{\Gamma \vdash_\Sigma B : A \rightarrow K \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma BN : K} \tag{2.20}$$

$$\frac{\Gamma \vdash_\Sigma A : \text{Type} \quad \Gamma, x : A \vdash_\Sigma B : \text{Type} \quad x \notin \text{FV}(B)}{\Gamma \vdash_\Sigma A \rightarrow B : \text{Type}} \tag{2.21}$$

Valid Elements of a Type

$$(\rightarrow\text{I}) \quad \frac{\Gamma \vdash_\Sigma A : \text{Type} \quad \Gamma, x : A \vdash_\Sigma M : B \quad x \notin \text{FV}(B)}{\Gamma \vdash_\Sigma \lambda x : A \,.\, M : A \rightarrow B} \tag{2.22}$$

$$(\rightarrow\text{E}) \quad \frac{\Gamma \vdash_\Sigma M : A \rightarrow B \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma MN : B} \tag{2.23}$$

The System G

Basic Rules

$$\frac{}{\vdash \langle\rangle \ \text{sig}} \tag{3.7}$$

$$\frac{\vdash \Sigma \ \text{sig} \quad \vdash_\Sigma K \ \text{kind} \quad c \notin \text{Dom}(\Sigma)}{\vdash \Sigma, c : K \ \text{sig}} \tag{3.8}$$

$$\frac{\vdash \Sigma \ \text{sig} \quad \vdash_\Sigma A\text{: Type} \quad c \notin \text{Dom}(\Sigma)}{\vdash \Sigma, c : A \ \text{sig}} \tag{3.9}$$

$$\frac{\vdash \Sigma \ \text{sig}}{\vdash_\Sigma \langle\rangle \ \text{context}} \tag{3.10}$$

$$\frac{\vdash_\Sigma \Gamma \ \text{context} \quad \Gamma \vdash_\Sigma A\text{: Type} \quad x \notin \text{Dom}(\Gamma)}{\vdash_\Sigma \Gamma, x : A \ \text{context}} \tag{3.11}$$

$$\frac{\vdash_\Sigma \Gamma \ \text{context}}{\Gamma \vdash_\Sigma \text{Type} \ \text{kind}} \tag{3.12}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma K \ \text{kind} \quad x \notin \text{FV}(K)}{\Gamma \vdash_\Sigma A \to K \ \text{kind}} \tag{3.13}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma K \ \text{kind}}{\Gamma \vdash_\Sigma \Pi x\text{:}A\,.\,K \ \text{kind}} \tag{3.14}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma B \ \text{Type} \quad x \notin \text{FV}(B)}{\Gamma \vdash_\Sigma A \to B \ \text{Type}} \tag{3.15}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma B \ \text{Type}}{\Gamma \vdash_\Sigma \Pi x\text{:}A\,.\,B \ \text{Type}} \tag{3.16}$$

$$\frac{\vdash_\Sigma \Gamma \ \text{context} \quad c\text{:}K \in \Sigma}{\Gamma \vdash_\Sigma c\text{:}K} \tag{3.17}$$

$$\frac{\vdash_\Sigma \Gamma \ \text{context} \quad c\text{:}A \in \Sigma}{\Gamma \vdash_\Sigma c\text{:}A} \tag{3.18}$$

$$\frac{\vdash_\Sigma \Gamma \ \text{context} \quad x\text{:}A \in \Gamma}{\Gamma \vdash_\Sigma x\text{:}A} \tag{3.19}$$

Operational Rules

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x\text{:}A \vdash_\Sigma B\text{:}K \quad x \notin \text{FV}(K)}{\Gamma \vdash_\Sigma \lambda x\text{:}A.B\text{:}A \to K} \tag{3.20}$$

$$(\to r)\quad \frac{\Gamma \vdash_\Sigma A{:}\mathrm{Type} \quad \Gamma, x{:}A \vdash_\Sigma M{:}B \quad x \notin \mathrm{FV}(B)}{\Gamma \vdash_\Sigma \lambda x{:}A.M{:}A \to B} \tag{3.21}$$

$$\frac{\Gamma \vdash_\Sigma A{:}\ \mathrm{Type} \quad \Gamma, x{:}A \vdash_\Sigma B{:}K}{\Gamma \vdash_\Sigma \lambda x{:}A.B{:}\Pi x{:}A.K} \tag{3.22}$$

$$(\Pi r)\quad \frac{\Gamma \vdash_\Sigma A{:}\mathrm{Type} \quad \Gamma, x{:}A \vdash_\Sigma M{:}B}{\Gamma \vdash_\Sigma \lambda x{:}A.M{:}\Pi x{:}A.B} \tag{3.23}$$

$$(\to l)\,\frac{@{:}A \to B \in \Sigma \cup \Gamma \quad \Gamma \vdash_\Sigma N{:}A \quad \Gamma, y{:}B \vdash_\Sigma M{:}D \quad y \notin \mathrm{FV}(D)}{\Gamma \vdash_\Sigma M[@N/y]{:}D} \tag{3.24}$$

$$(\Pi l)\,\frac{@{:}\Pi x{:}A.B \in \Sigma \cup \Gamma \;\; \Gamma \vdash_\Sigma N{:}A \;\; B[N/x] =_{\beta\eta} C \;\; \Gamma, y{:}C \vdash_\Sigma M{:}D \;\; y \notin \mathrm{FV}(D)}{\Gamma \vdash_\Sigma M[@N/y]{:}D} \tag{3.25}$$

$$\frac{\Gamma \vdash_\Sigma N{:}A \quad \Gamma \vdash_\Sigma B{:}A \to K}{\Gamma \vdash_\Sigma BN{:}K} \tag{3.26}$$

$$\frac{\Gamma \vdash_\Sigma N{:}A \quad \Gamma \vdash_\Sigma B{:}\Pi x{:}A.K}{\Gamma \vdash_\Sigma BN{:}K[N/x]} \tag{3.27}$$

Structural Rules

$$\frac{\Gamma \vdash_\Sigma M{:}A \quad \Gamma, x{:}A, \Delta \vdash_\Sigma \alpha}{\Gamma, \Delta[M/x] \vdash_\Sigma \alpha[M/x]} \tag{3.28}$$

Equality Rules

$$\frac{\Gamma \vdash_\Sigma M{:}A \quad \Gamma \vdash_\Sigma A'{:}\ \mathrm{Type} \quad A =_{\beta\eta} A'}{\Gamma \vdash_\Sigma M{:}A'} \tag{3.29}$$

$$\frac{\Gamma \vdash_\Sigma A{:}K \quad \Gamma \vdash_\Sigma K'\ \mathrm{kind} \quad K =_{\beta\eta} K'}{\Gamma \vdash_\Sigma A{:}K'} \tag{3.30}$$

The System $\mathbf{N}^{\heartsuit}$

The system $\mathbf{N}^{\heartsuit}$ is obtained from the system $\mathbf{N}$ by replacing the rule (2.16) by the rule

$$(\Pi\mathrm{E}^{\heartsuit})\ \frac{\Gamma \vdash_{\Sigma} N : A \qquad \Gamma \vdash_{\Sigma} M : \Pi x : A \,.\, B \qquad \Gamma \vdash_{\Sigma} B[N/x] : \mathrm{Type}}{\Gamma \vdash_{\Sigma} MN : B[N/x]} \tag{3.31}$$

The System L

$(Ax1)$ $\dfrac{}{\Gamma, x{:}A, \Gamma' \Rightarrow_\Sigma A}$ — G\cut proves $\vdash_\Sigma \Gamma, x:A, \Gamma'$ context

$(Ax2)$ $\dfrac{}{\Gamma \Rightarrow_{\Sigma, c:A, \Sigma'} A}$ — G\cut proves $\vdash_{\Sigma, c:A, \Sigma'} \Gamma$ context

$(\to r)$ $\dfrac{\Gamma, x{:}A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma A \to B}$ — (a) $x \notin \mathrm{FV}(B)$

(Πr) $\dfrac{\Gamma, x{:}A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma \Pi x{:}A.B}$

$(\to l)$ $\dfrac{\Gamma \Rightarrow_\Sigma A \qquad \Gamma, z{:}B \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C}$ — (a) $@{:}\, A \to B \in \Sigma \cup \Gamma$; (b) $z \notin \mathrm{FV}(C)$

(Πl) $\dfrac{\Gamma, x{:}D \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C}$ — (a) $@{:}\, \Pi y{:}A.B \in \Sigma \cup \Gamma$; (b) $x \notin \mathrm{FV}(C)$; (c) G\cut proves $\Gamma \vdash_\Sigma N{:}A$; (d) $B[N/y] \to_{\beta\eta} D$.

The Systems NR1 and NR2

The System NR1

The system **NR1** is obtained from the system **N** by replacing the 2.23 and 2.16 rules by the rule

$$\frac{\Gamma \vdash_\Sigma M_1 : A_1 \ldots \Gamma \vdash_\Sigma M_m : A_m \quad \Gamma \vdash_\Sigma N_1 : D_1 \ldots \Gamma \vdash_\Sigma N_n : D_n}{\Gamma \vdash_\Sigma @M_1 \ldots M_m N_1 \ldots N_n : E}, \tag{4.4}$$

where $@ : \Pi x_1 : E_1 \ldots \Pi x_m : E_m \,.\, B_1 \to (B_2 \to (\ldots \to (B_n \to C)\ldots)) \in \Sigma \cup \Gamma$, $A_i[M_1/x_1, \ldots, M_{i-1}/x_{i-1}] =_{\beta\eta} E_i$ for $1 \leq i \leq m$, $B_i[M_1/x_1, \ldots, M_m/x_m] =_{\beta\eta} D_i$ for $1 \leq i \leq n$ and $C[M_1/x_1, \ldots, M_m/x_m] =_{\beta\eta} E$.

The rule 4.4 has the additional trivial premiss

$$\Gamma, x : E \vdash_\Sigma x : E\,,$$

where $x \notin \mathrm{Dom}(\Gamma)$, but for simplicity of presentation we omit to write this premiss.

The System NR2

The system **NR2** is obtained from the system **N** by replacing the 2.23 and 2.16 rules by the rule

$$\frac{\Gamma \vdash_\Sigma M_1 : E_1 \ldots \Gamma \vdash_\Sigma M_m : E_m \quad \Gamma \vdash_\Sigma N_1 : D_1 \ldots \Gamma \vdash_\Sigma N_n : D_n \quad \Gamma, x : E \vdash_\Sigma P : F \quad x \notin \mathrm{FV}(F)}{\Gamma \vdash_\Sigma P[@M_1 \ldots M_m N_1 \ldots N_n / x] : F}, \tag{4.6}$$

where $@ : \Pi x_1 : A_1 \ldots \Pi x_m : A_m \,.\, B_1 \to (B_2 \to (\ldots \to (B_n \to C)\ldots)) \in \Sigma \cup \Gamma$, $A_i[M_1/x_1, \ldots, M_{i-1}/x_{i-1}] =_{\beta\eta} E_i$ for $1 \leq i \leq m$ $B_i[M_1/x_1, \ldots, M_m/x_m] =_{\beta\eta} D_i$ for $1 \leq i \leq n$ and $C[M_1/x_1, \ldots, M_m/x_m] =_{\beta\eta} E$.

The Systems LR1 and LR2

The System LR1

The calculus **LR1** is obtained from the calculus **L** by replacing the $(\to l)$ and (Πl) rules by the rule 4.5.

$$\frac{\Gamma \Rightarrow_\Sigma D_1 \ldots \Gamma \Rightarrow_\Sigma D_n}{\Gamma \Rightarrow_\Sigma E}, \tag{4.5}$$

where @ : $\Pi x_1 : A_1 \ldots \Pi x_m : A_m \,.\, B_1 \to (B_2 \to (\ldots \to (B_n \to C)\ldots)) \in \Sigma \cup \Gamma$, **N** proves $\Gamma \vdash_\Sigma M_i : E_i$ where $A_i[M_1/x_1, \ldots, M_{i-1}/x_{i-1}] \to_{\beta\eta} E_i$ for $1 \leq i \leq m$, $B_i[M_1/x_1, \ldots, M_m/x_m] \to_{\beta\eta} D_i$ for $1 \leq i \leq n$ and $C[M_1/x_1, \ldots, M_m/x_m] \to_{\beta\eta} E$.

The rule 4.5 has the additional trivial premiss

$$\Gamma, x{:}E \Rightarrow_\Sigma E\,,$$

where $x \notin \mathrm{Dom}(\Gamma)$, but for simplicity of presentation we omit to write this premiss. The importance of this premiss is that if $n = 0$ then it is the *only* premiss that is obtained.

The System LR2

The calculus **LR2** is obtained from the calculus **L** by replacing the $(\to l)$ and (Πl) rules by the rule

$$\frac{\Gamma \Rightarrow_\Sigma D_1 \ldots \Gamma \Rightarrow_\Sigma D_n \quad \Gamma, x : E \Rightarrow_\Sigma F}{\Gamma \Rightarrow_\Sigma F}, \tag{4.7}$$

where @ : $\Pi x_1 : A_1 \ldots \Pi x_m : A_m \,.\, B_1 \to (B_2 \to (\ldots \to (B_n \to C)\ldots)) \in \Sigma \cup \Gamma$, **G**\cut proves $\Gamma \vdash_\Sigma M_i{:}E_i$ where $A_i[M_1/x_1, \ldots, M_{i-1}/x_{i-1}] \to_{\beta\eta} E_i$ for $1 \leq i \leq m$, $B_i[M_1/x_1, \ldots, M_m/x_m] \to_{\beta\eta} D_i$ for $1 \leq i \leq n$ and $C[M_1/x_1, \ldots, M_m/x_m] \to_{\beta\eta} E$.

The Systems U and R

The rules of **U** and **R** consist of the $(\rightarrow r)$, $(\rightarrow l)$ and (Πr) rules of **L**, with the side conditions modified so that contexts are only extended with new variables,together with the new form of (Πl) rule:

$$(\rightarrow r) \qquad \frac{\Gamma, x{:}A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma A \rightarrow B} \qquad \text{(a) } x \notin \mathrm{Dom}(\Gamma),\ x \notin \mathrm{FV}(B)$$

$$(\Pi r) \qquad \frac{\Gamma, x{:}A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma \Pi x{:}A.B} \qquad \text{(a) } x \notin \mathrm{Dom}(\Gamma)$$

$$(\rightarrow l) \qquad \frac{\Gamma \Rightarrow_\Sigma A \qquad \Gamma, z{:}B \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C} \qquad \begin{array}{l}\text{(a) } @{:}\, A \rightarrow B \in \Sigma \cup \Gamma \\ \text{(b) } z \notin \mathrm{Dom}(\Gamma)\end{array}$$

$$(\Pi l) \qquad \frac{\Gamma, x{:}B[\alpha/y] \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C} \qquad \begin{array}{l}\text{(a) } @{:}\, \Pi y{:}A.B \in \Sigma \cup \Gamma \\ \text{(b) } x \notin \mathrm{Dom}(\Gamma)\end{array}$$

where α is not introduced elsewhere in the derivation.

Proofs, Search and Computation in General Logic

David Pym

Doctor of Philosophy

University of Edinburgh

1990

Abstract

The Logical Framework (LF) is a formal system for the representation of logics and formal systems as theories within it, which was developed by Harper, Honsell and Plotkin. The LF consists of two components: the $\lambda\Pi$-calculus, the λ-calculus with dependent function space (or product) types; and the principle of *judgements as types* in which judgements, the units of inference in logics and other formal systems, are identified with the types of the $\lambda\Pi$-calculus. This work is strongly influenced by the work of Martin-Löf on judgements in logic. The LF is suitable for mechanical implementation because the $\lambda\Pi$-calculus is decidable.

We present a theory of search and logic programming for the $\lambda\Pi$-calculus and consequently for logics which are adequately represented in the LF.

The presentation of the $\lambda\Pi$-calculus of Harper, Honsell and Plotkin is as a (linearized) natural deduction system. The search space induced by such a system is highly non-deterministic and so the first step is to define a Gentzenized system, in which the natural deduction style Π-elimination rule is replaced by a sequent calculus style Π-elimination rule, which is sound and complete with respect to the system of Harper, Honsell and Plotkin.

The Gentzenized system thus provides the foundation for a metacalculus, possessing an important subformula property, for the inhabitation assertions of the $\lambda\Pi$-calculus which forms a suitable basis for proof-search in the $\lambda\Pi$-calculus. By exploiting the structure of a form of *hypothetico-general* judgement, we are able to obtain (complete) calculi based on certain forms of resolution rule, the search spaces of which are properly contained within that of our basic metacalculus. Furthermore, we are able to further constrain proof-search in these calculi by considering a certain uniform form for proofs.

We develop a unification algorithm for the $\lambda\Pi$-calculus by extending the work of Huet for the simply-typed λ-calculus. We use this unification algorithm to allow us to eliminate non-deterministic choices of terms, when performing proof-search with the (Πl) or resolution rules. This is done by the introduction of a class of universal variables which are later instantiated via the unification algorithm.

When unification is used to instantiate universal variables well-formedness may not be preserved. However, by extending the work of Bibel and Wallen (for first-order classical, modal and intuitionistic logics) to the $\lambda\Pi$-calculus we are able to obtain a theory which allows to accept as many such instantiations as possible by detecting when the derivation can be reordered in order to yield a well-formed proof.

We extend the theory described above to provide a notion of logic programming by admitting universal variables in endsequents: these are analogous to the *logical variables* of PROLOG. Finally, we provide a denotational semantics for our logic programs by performing a least fixed point construction in a collection of Herbrand interpretations — maps from $|C(\Sigma)|$ to (families of) sets of terms in $|\mathcal{F}|$, where $C(\Sigma)$ is the syntactic category constructed out of the $\lambda\Pi$-calculus with signature Σ, and $\mathcal{F}$ is the category of families of sets. This construction is similar to one of Miller, and exploits a Kripke-like satisfaction predicate. We characterize this construction in terms of the model theory of the $\lambda\Pi$-calculus using the Yoneda functor.

Acknowledgements

I thank my supervisors, Gordon Plotkin and Furio Honsell and also my colleague Lincoln Wallen, from whom I have learned a great deal and whose help and encouragment have been of great importance.

I thank Martín Abadi, Simon Ambler, Stuart Anderson, Arnon Avron, Rod Burstall, George Cleland, Bob Harper, Ian Mason, Dale Miller, Eugenio Moggi, Randy Pollack, John Power, Anne Salvesen, Paul Taylor (Imperial College), Paul Taylor (Edinburgh), Chris Tofts and especially James Harland for their advice, suggestions and encouragement.

This work was supported by a CASE award from the U.K. Science and Engineering Research Council in cooperation with the B.P. Research Centre. The Laboratory for Foundations of Computer Science, University of Edinburgh and the Center for the Study of Language and Information, Stanford University provided enjoyable and supportive working environments.

Declaration

This thesis was composed by myself. The work reported herein is my own unless stated otherwise, and has not previously been submitted for a degree.

David Pym

Table of Contents

Chapter 1

Introduction

1.1 Introduction

In this chapter we introduce the ideas of general logic, of search, unification and effective search, and of computation in the form of logic programming by introducing each of the subsequent chapters of this thesis.

There are two principal motivations for the construction of a general theory of logics or formal systems:

1. The desire to learn more of the nature and properties of formal systems, in particular to investigate their common "core";

2. To provide a single formal system, capable of expressing in a uniform manner a wide class of formal systems, which can be implemented on a machine. Thus, by implementing one formal system we obtain implementations of many formal systems.

Note that we have been deliberately vague about the notion of implementation — there are many possible meanings. In this work, however, we have in mind the implementation of a theorem prover or a logic programming system.

We refer to related work throughout the text of the thesis as appropriate.

1.2 The $\lambda\Pi$-calculus and the LF

The Logical Framework (LF) of [HHP87], [AHM87], [HHP89], [Sa90] is a theory of *general logic* in which a wide class of mathematical logics can be represented. This theory is suitable for mechanical implementation.

The syntactic part of the LF is the $\lambda\Pi$-*calculus* — the λ-calculus with dependent function space (or product) types. Logics are encoded as theories of the $\lambda\Pi$-calculus via the *judgements as types* principle — a development, inspired by the work of Martin-Löf [ML85] and de Bruijn [deB80], of the Curry-Howard principle of propositions as types. The most general form of judgement considered is the *hypothetico-general* judgement, which provides the inspiration for the identification of a certain class of *clausal* types.

In Chapter 2 we introduce the LF as presented in [HHP87]. Following [HHP87] we summarize the language theory of the $\lambda\Pi$-calculus and introduce the conservative extension of the type system by the purely implicational $\rightarrow$ connective. We call the resulting system **N**. The main judgement of **N** (with some notational simplification) is the typing assertion: $\Gamma \vdash M{:}A$, meaning that the term M has type A, given the type assignments for free variables and constants recorded in Γ. This relation is decidable [HHP87].

We discuss the paradigm for the representation of formal systems by considering the examples of first-order logic (Peano arithmetic) and higher-order logic as presented by Church [Ch40].

In Appendix A we present the theory of *contextual categories*, introduced by Cartmell [Ca86] and developed by Streicher [St88], and its use as the algebraic framework in which to provide the basic model theory for the $\lambda\Pi$-calculus. The syntax of the $\lambda\Pi$-calculus over a given valid signature Σ can be presented algebraically as a (syntactic) category $C(\Sigma)$ in which the objects are valid contexts and the morphisms are substitutions (tuples of terms which satisfy certain well-typing constraints). This category inherits the structure of a *contextual category with*

products from the syntactic presentation of the $\lambda\Pi$-calculus. If $\mathcal{V}$ is a contextual category with products, $\mathcal{V}$-valued models of the $\lambda\Pi$-calculus are functors from this category to the $\mathcal{V}$ which *preserve exactly* the *contextual* and *product* structure of the syntactic category. Set-theoretic models can be obtained by taking $\mathcal{V}$ to be the category $\mathcal{F}$ of sets, families of sets, *etc.*. We present the (closed) term model construction in the value category $\mathcal{F}$.

1.3 The Gentzenized $\lambda\Pi$-calculus and Cut Elimination

The presentation of the $\lambda\Pi$-calculus in Chapter 2 is that of [HHP87] which is a system of (linearized) natural deduction, **N**. In Chapter 3 we present a *Gentzentized* formulation of the $\lambda\Pi$-calculus, called **G**, in which the $\rightarrow$-elimination and Π-elimination rules of the natural deduction system are replaced by $(\rightarrow l)$ and (Πl) (read respectively as arrow-left and pi-left) rules in the manner of the elimination rules of Gentzen sequents [Ge34], [Pr65], [How80]. The $\rightarrow$-elimination rule of the system **N** (with some notational simplification) is of the form:

$$(\rightarrow \mathrm{E})\ \frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B},$$

and this is replaced by the $(\rightarrow l)$ rule which (with some notational simplification) is of the form:

$$(\rightarrow l)\ \frac{x : A \rightarrow B \in \Gamma \quad \Gamma \vdash N : A \quad \Gamma, y : B \vdash M : C}{\Gamma \vdash M[xN/y] : C},$$

where y is not free in C. The latter rule should be understood to say that if x is a map from A to B, if we can construct the term N of type A in the context Γ and if we can construct the term M of type C in the context $\Gamma, y{:}B$, then we can construct the term $M[xN/y]$ of type C in the context Γ. The Π-elimination rule of the system **N** (with some notational simplification) is of the form:

$$(\Pi \mathrm{E})\ \frac{\Gamma \vdash M{:}\,(\Pi x{:}A.B) \qquad \Gamma \vdash N{:}\,A}{\Gamma \vdash MN{:}\,B[N/x]}$$

where $B[N/x]$ denotes capture-avoiding substitution of N for free occurrences of x in B, and this is replaced by the (Πl) which (with some notational simplification) is of the form:

$$(\Pi l)\ \frac{x : \Pi z{:}A\,.\,B \in \Gamma \quad \Gamma \vdash N : A \quad \Gamma, y : D \vdash M : C}{\Gamma \vdash M[xN/y] : C},$$

where D is the normal form of $B[N/z]$ and where y is not free in C. This rule is read in manner similar to that for the $(\rightarrow l)$ rule.

The cut rule is not admissible in the system obtained from **N** by making just these replacements and so must be taken explicitly in **G** in order to retain completeness. However, by working with β-normal forms we obtain a *cut elimination* theorem: this is a difficult theorem. This is related to the work of [ML71a], [ML71b] and [How80] in the setting of simple types.

1.4 Proof-search in the $\lambda\Pi$-calculus

The systems **N** and **G** which we present in Chapters 2 and 3 are both systems for deriving assertions of the form $\Gamma \vdash M : A$ which is read as "the object M has type A in the context Γ", and the provability of these assertions is decidable. **N** is a (linearized) natural deduction style system and **G** is a sequent style system of equivalent strength. From the point of view of proof-search we are interested in assertions or *inhabitation* sequents of the form $\Gamma \Rightarrow_\Sigma A$ which is read as "Γ inhabits A" with the meaning $(\exists M)(\Gamma \vdash M{:}A)$: the provability of such assertions is semi-decidable. The cut-free system **G**\cut of Chapter 3 provides a satisfactory basis for a calculus for proof-search in the $\lambda\Pi$-calculus, and in Chapter 4 we present a calculus of inhabitation sequents which may be seen as a metacalculus for the calculi of typing assertions of Chapters 2 and 3.

This system, called **L**, is sound and complete relative to **G**\cut for the semi-decidable relation of inhabitation: $\Gamma \Rightarrow A$. The judgements of **L** assert the *existence* of proofs of the judgements of **G**\cut. **L** is the starting point for our

investigation into proof-search. A typical rule of **L** is the Π-left rule:

$$(\Pi l)\quad \frac{\Gamma, z{:}D \Rightarrow C}{\Gamma \Rightarrow C}$$

(a) $w : (\Pi x{:}A.B) \in \Gamma$

(b) $z \notin \mathrm{Dom}(\Gamma)$, the variables occurring in Γ

(c) $\Gamma \vdash N{:}A$

(d) $B[N/x] \rightarrow_{\beta\eta} D$, $\rightarrow_{\beta\eta}$ the usual $\beta\eta$-reduction.

A similar rule arises when the type of w is of the form $A \rightarrow B$. **L** is almost a *logicistic* system, in the sense of Gentzen, meaning that there is only one localized appeal to an external notion, that notion being an appeal to the system **G**\cut in side condition (c) of the rule above. Indeed, with respect to the Π-type structure of terms in the language, **L** has a *subformula property* [Ge34]. As a consequence, if the inference rules are used as *reduction operators* from conclusion to premisses then **L** induces a search space of derivations of a given sequent.[1] Notice that if the (Πl) rule is used as a reduction, the choice of term N to use in the premiss is unconstrained by the conclusion of the rule. The sub*formula* property of the Π-types does not extend to a full sub*term* property (*cf.* the quantifier rules of the predicate calculus). The *axiom* sequent (or *closure* condition for the reduction system) is:

$$\Gamma, x{:}A, \Gamma' \Rightarrow A$$

i.e., the conclusion occurs as the type of a declaration in the context.

We present also two calculi which exploit *clausal form* types in order to utilize forms of *resolution* rule which encode several applications of the →- and Π-left rules in a single inference step, *cf.* the resolution rule of first-order Horn clause logic programming [Ll84]. We consider also a class of *uniform proofs* [MNPS89]. Some of the ideas of Chapter 4 were first introduced in [Pl87].

---

[1] Kleene [Kl68] explains this in the case of the predicate calculus. Sequent systems used in this way are systems of *block tableaux* [Sm68].

1.5 Unification for the $\lambda\Pi$-calculus

In Chapter 5 we develop a unification algorithm for the $\lambda\Pi$-calculus . This algorithm is a development of the algorithm of Huet [Hu75] so that we unify not only objects, but their types as well.

For example consider the pair of terms

$$\lambda x{:}o\,.\,y(x,u) : \Pi x{:}o\,.\,f(x,u) \tag{1.1}$$

where the constant symbol $f : o \rightarrow o \rightarrow \mathrm{Type}$, and

$$\lambda x{:}o\,.\,z(x,v) : \Pi x{:}o\,.\,g(x,w)\,, \tag{1.2}$$

where the constant symbol $g : o \rightarrow o \rightarrow \mathrm{Type}$. The types in 5.1 and 5.2 are unified if f is substituted for g and u is substituted for w. In general we shall have to unify the types of the Π-abstracted variables as well, but in this case they are both just the constant o. Applying this substitution to the objects, we are left with the problem of unifying

$$\lambda x{:}o\,.\,y(x,u)$$

and

$$\lambda x{:}o\,.\,z(x,v)\,,$$

which now have the same type. These are unified if u is substituted for v and y is substituted for z.

The problem of unifying a pair of well-typed objects and their types is reduced to that of unifying a set of pairs of objects, provided the original pair satisifies a certain *similarity* condition.

1.6 Effective Search in the $\lambda\Pi$-calculus

In Chapter 6 we introduce two metacalculi for **L**. The system **U** is formed from **L** by removing the appeal to **G**\cut in the (Πl) rule. The (Πl) rule of **U** is thus:

$$(\Pi l)\quad \frac{\Gamma, z{:}B[\alpha/x] \Rightarrow C}{\Gamma \Rightarrow C} \qquad \begin{array}{l}\text{(a) } w : (\Pi x{:}A.B) \in \Gamma \\ \text{(b) } z \notin \mathrm{Dom}(\Gamma).\end{array}$$

This rule introduces a free, or *universal* variable into the proof, denoted here by α. Universal variables are distinct from the usual *eigenvariables* that are bound by the derivation (and explicitly declared in contexts). The axiom sequent of **U** is used to compensate for the omission of term information in the (Πl) rule as follows:

$$\Gamma, x{:}B, \Gamma' \Rightarrow A \qquad \text{(a) } (\exists\sigma)\, B\sigma \equiv A\sigma$$

where σ is an *instantiation* of the universal variables of the sequent, and $B\sigma$ denotes the normal form of the term resulting from the application of σ (as a substitution) to B. The calculation of instantiations can be performed by the unification algorithm of Chapter 5.

A **U**-proof is a pair $\langle \psi, \sigma \rangle$ consisting of a **U**-derivation ψ and an instantiation σ such that $\psi\sigma$ (the application of σ as a substitution to ψ) is an **L**-proof. Not every instantiation that closes the leaves of a given **U**-derivation will yield an **L**-proof when applied. It is sufficient to check that the instantiation can be *well-typed* in the derivation to ensure that the result is an **L**-proof. If Γ is the context and A the type of the universal variable α when introduced into the **U**-derivation, the well-typing condition for α amounts to:

$$\Gamma\sigma \vdash \alpha\sigma{:}\, A\sigma$$

ensuring that side condition (c) of the (Πl) rule of **L** — the condition omitted from the (Πl) rule of **U** — is nevertheless satisfied in $\psi\sigma$. The unconstrained choice of term in the (Πl) rule of **L** is replaced by a highly constrained choice in **U**. This wholesale reduction in the search space is analogous, of course, to that obtained by Robinson in the context of the predicate calculus [Ro65].

The well-typing of an instantiation depends on the structure of the derivation from which it is calculated. However, even if it fails to be well-typed in that derivation it may be well-typed in some permutation of the derivation, since rule applications (or reductions) can sometimes be permuted whilst leaving the endsequent (*i.e.*, root) of the derivation and its leaves unchanged. The degree to which this can be done is summarized in the form of a *Permutation Theorem,* in the sense of Kleene [Kl52] and Curry [Cu52], and underlies the second of our metacalculi, a system called **R**.

The rules of **R** are just the rules of **U** (so the derivations are the same) but the condition for instantiations to yield a proof is weakened. An **R**-proof is again a pair consisting of a derivation ψ and an instantiation σ under which ψ is closed, but now we require only that there exist perhaps another (closed) derivation ψ^* in which the instantiation is well-typed; *i.e.*, $\psi^*\sigma$ is an **L**-proof. Of course the crucial computational question is whether the existence of at least one suitable ψ^*, given ψ and σ, can be determined as the search progresses. We show that this is indeed the case using a *reduction ordering*: a notion which was introduced by Bibel [Bi81] for classical connectives and extended by Wallen in [Wa89] to various non-classical connectives. An **R**-proof therefore corresponds to an equivalence class of **U**-proofs of the same endsequent consisting of all permutation variants of the original derivation in which the calculated instantiation is well-typed.

Some of the ideas of Chapter 6 were first introduced in [Pl87].

1.7 Computation and its Semantics

There is an important sense in which hypothetico-general judgements

$$\bigwedge_{x_1:A_1} \cdots \bigwedge_{x_n:A_n} J_1 \vdash \cdots (J_m \vdash J)$$

are an appropriate notion of (higher-order) Horn clause. In first-order logic, a Horn clause is an expression of the form

$$\forall x_1 \ldots \forall x_r (A \subset B_1 \wedge \ldots \wedge B_k)$$

where A is a positive literal and $B_1, \ldots, B_k$ are all negative literals and all of the variables occurring in these literals are among $x_1, \ldots, x_r$ [Ll84]. If we ignore for the time being the fact that not all of the variables in the hypothetico-general judgement are quantified (later in the thesis we shall see that this unimportant), and consider that the conjunctions of the first-order Horn clause may be viewed as implications, via "Currying", the correspondence is clear. We shall consider these ideas further in later chapters where they will provide us with a good deal of intuition to support our notion of logic programming. For the purposes of this chapter, however, we shall content ourselves with a brief consideration of the work of Amy Felty [Fe87]. Felty considers the encoding of LF signatures in Miller and Nadathur's higher-order logic programming language λPROLOG [MN86], [NM88].

Felty translates the constant declarations of LF signatures into λPROLOG clauses. As a simple example consider a possible LF encoding of the ∧-introduction rule of first-order logic (this is the formulation taken by Felty):

$$\wedge\mathrm{I} : \Pi A{:}\mathit{bool}.\, \Pi B{:}\mathit{bool}.\, T(A) \rightarrow T(B) \rightarrow T(A \wedge B),$$

in the obvious notation. This is encoded as the λPROLOG clause

```
proof ( A and B ) :- proof A , proof B.
```

Thus the λPROLOG predicate `proof` is the counterpart in this setting of the judgement $T : \mathit{bool} \rightarrow \mathrm{Type}$, and the program variables `A` and `B` are the counterparts of the universally quantified objects in the LF type.

The reader is referred to [Fe87] for details of the general translation of LF expressions into λPROLOG clauses.

In Chapter 7 we describe how λΠ-calculus sequents, when extended with a notion of *program variable* or *logical variable* which is suggested by the universal variables of Chapter 6, yield a notion of *logic programming* in which the components of a λΠ-calculus context are considered to be the program clauses. We are able to describe an analogy with (pure) PROLOG programs which provides us with valuable intuition. We provide an *operational semantics* for this notion of logic

programming by extending the procedures of Chapter 6 to apply to non-ground endsequents, which correspond to a program (the context) together with a query (the succedent type of the sequent). This operational semantics is both sound and complete in an appropriate sense.

We provide a *denotational semantics* by performing a least fixed point construction in the manner of Miller [Mi89] over a collection of *Herbrand interpretations*, with respect to a Kripke-like satisfaction predicate [Sm73], [Mi89]. Unfortunately, the least fixed point construction fails to yield a model in the sense of Appendix A; it can only be defined on the objects of $C(\Sigma)$.

However we observe that the Yoneda functor yields a model which, in an appropriate sense, contains the least fixed point construction and for which a suitable completeness theorem holds.

1.8 Encoded Logics

Appendix B contains the LF signatures as presented in [AHM87]. A wide class of logics is included:

- Kleene's three-valued logic;
- First-order logic with a choice operator;
- Hilbert style modal logics;
- Natural deduction style S4;
- Classical lambda calculus;
- Call-by-value lambda calculus;
- Lambda I calculus;
- Linear lambda calculus;

- Hoare logic.

Representation theorems (in the LF) for each logic are included.

1.9 Insert

We include an insert (placed in a pocket attached to the inside of the back cover) which contains a list of (the rules of) the proof systems introduced in the body of the thesis. This insert is provided for the convenience of the reader.

Chapter 2

The $\lambda\Pi$-calculus and the Logical Framework

2.1 Introduction

In Chapter 1 we motivated the need for a *general theory of logic(s)* and in particular the need for such a theory to satisfy the following important criteria:

- The theory should provide a uniform paradigm or mechanism for the representation of a wide class of formal or mathematical logics;

- The theory should, in an appropriate sense, be suitable for implementation on a machine.

The Logical Framework (LF) of [HHP87], [AHM87] and [HHP89] is an attempt to provide a general theory of logic(s) which satisfies these criteria. In this respect the LF is substantially successful in that a wide class of logics has been represented within it [AHM87] and in that the language of the LF is decidable. There is a machine-implemented proof-editing system due to Pollack [Po89], which implements the type theory of the LF; actually, Pollack's system implements *Generalized Type Systems* [Ba89].

The LF consists of two components, one formal and the other not:

- A language;

- A paradigm for representing (by encoding) logics in that language.

The language of the LF is the $\lambda\Pi$-calculus. The $\lambda\Pi$-calculus is a weak type theory that is closely related to AUT-PI and AUT-QE [vD80], to Martin-Löf's early type theory [ML73], to Huet and Coquand's *Calculus of Constructions* [CH85] and to Meyer and Reinhold's λ^{π} [MR86]. The $\lambda\Pi$-calculus is called $\lambda\mathbf{P}$ by Barendregt in [Ba89], where he systematically places it in the "λ-cube" of type theories.

As used in the LF, the $\lambda\Pi$-calculus is able to specify both the language of a logic — its formulae and its axiom and rule schemes — and its proofs. The language of a logic is defined in a general theory of expressions which exploits the λ-calculus structure to provide binding operators, substitution, capture, α-conversion, and schematic abstraction and instantiation.

The treatment of rules and proofs adopted by the LF is based on the notion of a *judgement* [ML85], the unit of assertion in a logical system.[1] A logic is a system for asserting basic judgements and, according to Martin-Löf's analysis, the set of judgements is then closed under two higher-order judgement forms that are used to specify inference rule schemes and to model discharge and variable occurrence conditions such as arise in Hilbert systems or systems of natural deduction. Rules are viewed as the proofs of (possibly higher-order) judgements that specify them. There is no distinction between primitive and derivable rules, although admissible rules (to be defined later) are rather different and are not susceptible to the same analysis. The extension to higher-order judgements allows for a natural presentation of many logical systems that avoids side conditions on rules.

Judgements, rules and proofs are represented in the LF type theory by applying the *judgements as types* principle of [HHP87] in which each judgement is identified with the type of its proofs. Each basic judgement is represented by a base (or non-functional) type of the LF type theory, and the higher-order judgements are represented in a logic-independent way by higher (or functional) types. Proofs, and hence (derived) rules, are represented as terms of the LF type theory, with

[1] See also Schroeder-Heister [SH84] for a related point of view.

the computationally important consequence that proof checking is reduced to type checking.

In this chapter we present the basic syntactic theory of the $\lambda\Pi$-calculus and the LF. The basic model theory of the $\lambda\Pi$-calculus is presented in Appendix A. In Section 2.2 we present the $\lambda\Pi$-calculus, along with some of its important proof-theoretic properties. In Section 2.3, we introduce the LF's theory of expressions and formulae. In Section 2.4 we consider the treatment of judgements, rules (including derived rules), and proofs in the LF. The method is illustrated for first-order logic (over the language of Peano arithmetic) and higher-order logic (over the language of Peano arithmetic). The presentation of Sections 2.2, 2.3 and 2.4 is closely based on that of [HHP87].

2.2 The λΠ-calculus

The type theory of the LF is closely related to the Π-fragment of AUT-PI, a language belonging to the so-called AUTOMATH family. The LF type theory is a language with entities of three levels: *objects*, *types and families of types*, and *kinds*. Objects are classified by types, types and families of types by kinds. The kind Type classifies the types; the other kinds classify functions f which yield a type $f(x_1)\ldots(x_n)$ when applied to objects $x_1,\ldots,x_n$ of certain types determined by the kind of f. Any function definable in the system has a type as domain, while its range can either be a type, if it is an object, or a kind, if it is a family of types. The λΠ-calculus is therefore predicative.

The theory we shall deal with is a formal system for deriving assertions of one of the following shapes:

$\vdash \Sigma$ sig	Σ is a signature
$\vdash_\Sigma \Gamma$ context	Γ is a context
$\Gamma \vdash_\Sigma K$ kind	K is a kind
$\Gamma \vdash_\Sigma A{:}K$	A has kind K
$\Gamma \vdash_\Sigma M{:}A$	M has type A

where the syntax is specified by the following grammar:

Signatures	Σ	::=	$\langle\rangle \mid \Sigma, c:K \mid \Sigma, c:A$
Contexts	Γ	::=	$\langle\rangle \mid \Gamma, x:A$
Kinds	K	::=	$\text{Type} \mid \Pi x{:}A.K$
Types and Families of Types	A	::=	$c \mid \Pi x{:}A.B \mid \lambda x{:}A.B \mid AM$
Objects	M	::=	$c \mid x \mid \lambda x{:}A.M \mid MN$.

We let M, N *etc.* range over expressions for objects, A, B *etc.* range over expressions for types and families of types, K, L *etc.* range over expressions for kinds, x, y *etc.* range over expressions for variables, and c, d *etc.* range over expressions for constants. We let @ range over both constants and variables and U, V *etc.* range over expressions for objects, types and families of types and kinds.

λ and Π are binding constructs, and we write $A \to B$ for $\Pi x{:}A\,.\,B$ when x does not occur free in B and $A \to K$ for $\Pi x{:}A\,.\,K$ when x does not occur free in K. We write $U(M)$ when we wish to emphasize that the expression U depends on the object M and write $U[M/x]$ for the usual notion of capture-avoiding substitution in which the variable x in the expression U is replaced by the object M.

A term is said to be *well-typed in a signature and context* if it can be shown to either be a kind, have a kind, or have a type in that signature and context. A term is *well-typed* if it is well-typed in some signature and context.

The notions of β- and $\beta\eta$-reduction, written $\to_\beta$ and $\to_{\beta\eta}$, can be defined both at the level of objects and at the level of types and families of types in the obvious way, for details see [HHP89] and [Sa90]. We define $U =_\beta V$ if and only if $U \twoheadrightarrow_\beta W$ and $V \twoheadrightarrow_\beta W$ for some term W and $U =_{\beta\eta} V$ if and only if $U \twoheadrightarrow_{\beta\eta} W'$ and $V \twoheadrightarrow_{\beta\eta} W'$ for some term W', where * denotes transitive closure. For simplicity we shall write $\to_\beta$ for $\to_\beta^*$ and $\to_{\beta\eta}$ for $\to_{\beta\eta}^*$. We assume α-conversion throughout. We shall write $\equiv$ to denote syntactic identity and $\mathrm{NF}(U)$ to denote the normal form (usually the $\beta\eta$-normal, but the β-normal form where stated) of the term U. We write $\mathrm{FV}(U)$ to denote the collection of free variables of the expression U. We write $\mathrm{Dom}(\Sigma)$ and $\mathrm{Dom}(\Gamma)$ to denote the collections of constants and variables that occur as labels in the signature Σ and the context Γ respectively.

We shall need to distinguish between *derivable rules* and *admissible rules* for a given proof system. A rule is derivable in a given proof system if there is a derivation of its conclusion from its premisses in that proof system. A rule is admissible in a given proof system if the provability of its premisses in that system implies the provability of its conclusion in that system. The reader is referred to [HS86] and [Tr73] for further discussion of these definitions.

The inference rules which we take to define the $\lambda\Pi$-calculus appear in Table 2.1.

Valid Signature

$$\frac{}{\vdash \langle\rangle \text{ sig}} \tag{2.1}$$

$$\frac{\vdash \Sigma \text{ sig} \quad \vdash_\Sigma K \text{ kind} \quad c \notin \text{Dom}(\Sigma)}{\vdash \Sigma, c : K \text{ sig}} \tag{2.2}$$

$$\frac{\vdash \Sigma \text{ sig} \quad \vdash_\Sigma A\text{: Type} \quad c \notin \text{Dom}(\Sigma)}{\vdash \Sigma, c : A \text{ sig}} \tag{2.3}$$

Valid Context

$$\frac{\vdash \Sigma \text{ sig}}{\vdash_\Sigma \langle\rangle \text{ context}} \tag{2.4}$$

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad \Gamma \vdash_\Sigma A\text{: Type} \quad x \notin \text{Dom}(\Gamma)}{\vdash_\Sigma \Gamma, x : A \text{ context}} \tag{2.5}$$

Valid Kinds

$$\frac{\vdash_\Sigma \Gamma \text{ context}}{\Gamma \vdash_\Sigma \text{Type kind}} \tag{2.6}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma K \text{ kind}}{\Gamma \vdash_\Sigma \Pi x : A.K \text{ kind}} \tag{2.7}$$

Valid Elements of a Kind

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad c : K \in \Sigma}{\Gamma \vdash_\Sigma c\text{:} K} \tag{2.8}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma B\text{: Type}}{\Gamma \vdash_\Sigma \Pi x : A.B\text{: Type}} \tag{2.9}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma B\text{:} K}{\Gamma \vdash_\Sigma \lambda x : A.B\text{:} \Pi x : A.K} \tag{2.10}$$

$$\frac{\Gamma \vdash_\Sigma B\text{:} \Pi x : A.K \quad \Gamma \vdash_\Sigma N\text{:} A}{\Gamma \vdash_\Sigma BN\text{:} K[N/x]} \tag{2.11}$$

$$\frac{\Gamma \vdash_\Sigma A\text{:} K \quad \Gamma \vdash_\Sigma K' \text{ kind} \quad K =_{\beta\eta} K'}{\Gamma \vdash_\Sigma A\text{:} K'} \tag{2.12}$$

Valid Elements of a Type

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad c : A \in \Sigma}{\Gamma \vdash_\Sigma c\text{:} A} \tag{2.13}$$

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad x : A \in \Gamma}{\Gamma \vdash_\Sigma x\text{:} A} \tag{2.14}$$

$$\frac{\Gamma \vdash_\Sigma A\text{: Type} \quad \Gamma, x : A \vdash_\Sigma M\text{:} B}{\Gamma \vdash_\Sigma \lambda x : A.M\text{:} \Pi x : A.B} \tag{2.15}$$

$$\frac{\Gamma \vdash_\Sigma M\text{:} \Pi x : A.B \quad \Gamma \vdash_\Sigma N\text{:} A}{\Gamma \vdash_\Sigma MN\text{:} B[N/x]} \tag{2.16}$$

$$\frac{\Gamma \vdash_\Sigma M\text{:} A \quad \Gamma \vdash_\Sigma A'\text{: Type} \quad A =_{\beta\eta} A'}{\Gamma \vdash_\Sigma M\text{:} A'} \tag{2.17}$$

Table 2.1

$\beta\eta$-conversion over $K \cup A \cup M$ is not Church-Rosser, so the order of technical priority in which the basic metatheoretical results are proved is crucial. The next theorem summarizes these results in a convenient order (here α ranges over the basic assertions of the type theory).

THEOREM 2.2.1 (THE BASIC METATHEORY OF THE $\lambda\Pi$-CALCULUS)

1. Thinning is an admissible rule: if $\Gamma \vdash_\Sigma \alpha$ and $\vdash_{\Sigma,\Sigma'} \Gamma, \Gamma'$ context, then $\Gamma, \Gamma' \vdash_{\Sigma,\Sigma'} \alpha$.

2. Transitivity is an admissible rule: if $\Gamma \vdash_\Sigma M : A$ and $\Gamma, x : A, \Delta \vdash_\Sigma \alpha$, then $\Gamma, \Delta[M/x] \vdash_\Sigma \alpha[M/x]$.

3. Uniqueness of types and kinds: if $\Gamma \vdash_\Sigma M : A$ and $\Gamma \vdash_\Sigma M : A'$, then $A =_{\beta\eta} A'$, and similarly for kinds.

4. Subject reduction (or closure): if $\Gamma \vdash_\Sigma M : A$ and $M \twoheadrightarrow_{\beta\eta} M'$, then $\Gamma \vdash_\Sigma M' : A$, and similarly for types.

5. All well-typed terms are strongly normalizing.

6. All well-typed terms are Church-Rosser.

7. Each of the five relations defined by the inference system of Table 2.1 is decidable, as is the property of being well-typed.

8. Strengthening is an admissible rule: if $\Gamma, x : A, \Gamma' \vdash_\Sigma \alpha$ and if $x \notin \mathrm{FV}(\Gamma') \cup \mathrm{FV}(\alpha)$ then $\Gamma, \Gamma' \vdash_\Sigma \alpha$. □

The proof of this theorem, due to Salvesen [Sa90], is rather complicated. The methods developed by van Daalen in his thesis [vD80] can be adapted to this type theory. The main difficulty lies in obtaining the Church-Rosser property and the essential step in obtaining the proof of this property is to first reformulate the $\lambda\Pi$-calculus as a system with equality judgements in which type labels are explicit, *i.e.*, the assertions of equality have shape $\Gamma \vdash_\Sigma M = N : A$, *etc.*. This

move is sufficient to allow the methods of van Daalen to go through. The reader is referred to [Sa89] for the details of the proof.[2] Similar properties are proved for the system with just β-reduction in [HHP89]. We note that for the proof of Part 8 we must first prove decidability. We note that we must prove the Church-Rosser property, strong normalization and have presence of type labels in order to prove the decidability of the type theory.

Following [HHP87] we outline only the proof of strong normalization (Part 5) since it can be proved independently of the other metatheoretic results. Moreover it yields a corollary that is useful for characterizing the terms that are definable in the $\lambda\Pi$-calculus.

We define a translation τ of the $\lambda\Pi$-calculus types and kinds into S, the set of simple types over a base type ω, and a translation $|-|$ of $\lambda\Pi$-calculus type families and objects into $\Lambda(K)$, the set of untyped λ-terms with a set of constants $K = \{\pi_\sigma | \sigma \in S\}$. These translations are extended to signatures and contexts in the obvious way.

[2]Harper [Ha88] also considers an equational formulation of the $\lambda\Pi$-calculus as a basis for the construction of *environment models* [Me82] of the $\lambda\Pi$-calculus.

DEFINITION 2.2.2 (TRANSLATION TO SIMPLE TYPES)

The definition of the translation divides conveniently into two clauses:

- The τ-clause:
  - Kinds

$$\begin{aligned}\tau(\text{ Type}) &= \omega\\ \tau(\Pi x{:}A.K) &= \tau(A) \to \tau(K)\end{aligned}$$

  - Types and type families

$$\begin{aligned}\tau(c) &= \omega\\ \tau(\Pi x{:}A.B) &= \tau(A) \to \tau(B)\\ \tau(\lambda x{:}A.B) &= \tau(B)\\ \tau(AM) &= \tau(A)\end{aligned}$$

- The $|-|$-clause:
  - Types and Families of Types

$$\begin{aligned}|c| &= c\\ |\Pi x{:}A.B| &= \pi_{\tau(A)}|A|(\lambda x.|B|)\\ |\lambda x{:}A.B| &= (\lambda y.\lambda x.|B|)|A| \quad (y \notin \mathrm{FV}(\lambda x.|B|))\\ |AM| &= |A|\,|M|\end{aligned}$$

    where we assume that x and y are distinct.

  - Objects

$$\begin{aligned}|c| &= c\\ |x| &= x\\ |\lambda x{:}A.M| &= (\lambda y.\lambda x.|M|)|A| \quad (y \notin \mathrm{FV}(\lambda x.|M|))\\ |MN| &= |M|\,|N|\end{aligned}$$

    where we assume that x and y are distinct. □

The precise sense in which this definition is consistent is stated in the following lemma:

LEMMA 2.2.3 (TRANSLATION CONSISTENCY)

1. If $\Gamma \vdash_\Sigma A:K$, then $\tau(\Gamma) \vdash_{\tau(\Sigma),\{\pi_\sigma \mid \pi_\sigma:\omega\to(\sigma\to\omega)\to\omega\}} |A|:\tau(K)$ in Curry's type assignment system, *i.e.*, simply-typed λ-calculus [HS86].

2. If $\Gamma \vdash_\Sigma M:A$, then $\tau(\Gamma) \vdash_{\tau(\Sigma),\{\pi_\sigma \mid \pi_\sigma:\omega\to(\sigma\to\omega)\to\omega\}} |M|:\tau(A)$ in Curry's type assignment system.

PROOF

By induction on the structure of the proof of $\Gamma \vdash_\Sigma A:K$ and $\Gamma \vdash_\Sigma M:A$. We have only to notice that any derivation of $\vdash_\Sigma \Gamma, x:A$ context contains as a subderivation a derivation of $\Gamma \vdash_\Sigma A$: Type. □

The translation has been carried out in such a way that the extra combinatorial complexity in the $\lambda\Pi$-calculus terms due to the presence of type labels is not lost. We then have the following result:

LEMMA 2.2.4 (REDUCTION UNDER TRANSLATION)

1. If $A \to_{\beta\eta} A'$, then $|A| \twoheadrightarrow_{\beta\eta} |A'|$;

2. If $M \to_{\beta\eta} M'$, then $|M| \twoheadrightarrow_{\beta\eta} |M'|$.

PROOF

By induction on the derivation of $A \to A'$ and $M \to M'$. □

Now since the Curry typable terms are strongly normalizing, so too are the terms of the $\lambda\Pi$-calculus. For more information concerning these arguments the reader is referred to [HHP89].

Moreover, it can be easily seen that $|A| \twoheadrightarrow_{\beta\eta} erase(A)$, and $|M| \twoheadrightarrow_{\beta\eta} erase(M)$, where $erase(M)$ denotes the term obtained from M by removing the type labels from the λ-abstractions, so that we obtain the following corollary, which characterizes the terms that are definable in the $\lambda\Pi$-calculus:

COROLLARY 2.2.5 (DEFINABILITY CHARACTERIZATION (PREDICATIVITY))

If $\Gamma \vdash_\Sigma M{:}A$, then $erase(M)$ can be typed in Curry's type assignment system. □

From the point of view of the LF it is important that the type theory be decidable so that the checking of object-logic proofs, which correspond to objects in the $\lambda\Pi$-calculus, is reduced to type checking at the level of the framework. This reduction is very important for the construction of machine implementations of the LF. In particular, our work on effective search and logic programming in Chapters 6 and 7 depends on decidability.

Before proceeding to discuss the LF we consider a conservative extension of the $\lambda\Pi$-calculus as defined above. We add an explicit $\rightarrow$ connective. This extension requires the following extension to the list of inference rules in Table 2.1 :

Valid Kinds

$$\frac{\Gamma \vdash_\Sigma A{:}\,\text{Type} \quad \Gamma, x{:}A \vdash_\Sigma K\,\text{kind} \quad x \notin \text{FV}(K)}{\Gamma \vdash_\Sigma A \rightarrow K\,\text{kind}} \tag{2.18}$$

Valid Elements of a Kind

$$\frac{\Gamma \vdash_\Sigma A{:}\,\text{Type} \quad \Gamma, x{:}A \vdash_\Sigma B{:}K \quad x \notin \text{FV}(K)}{\Gamma \vdash_\Sigma \lambda x{:}A\,.\,B : A \rightarrow K} \tag{2.19}$$

$$\frac{\Gamma \vdash_\Sigma B{:}A \rightarrow K \quad \Gamma \vdash_\Sigma N{:}A}{\Gamma \vdash_\Sigma BN : K} \tag{2.20}$$

$$\frac{\Gamma \vdash_\Sigma A{:}\,\text{Type} \quad \Gamma, x{:}A \vdash_\Sigma B : \text{Type} \quad x \notin \text{FV}(B)}{\Gamma \vdash_\Sigma A \rightarrow B : \text{Type}} \tag{2.21}$$

Valid Elements of a Type

$$\frac{\Gamma \vdash_\Sigma A{:}\,\text{Type} \quad \Gamma, x{:}A \vdash_\Sigma M{:}B \quad x \notin \text{FV}(B)}{\Gamma \vdash_\Sigma \lambda x{:}A\,.\,M : A \rightarrow B} \tag{2.22}$$

$$\frac{\Gamma \vdash_\Sigma M{:}A \rightarrow B \quad \Gamma \vdash_\Sigma N{:}A}{\Gamma \vdash_\Sigma MN : B} \tag{2.23}$$

We call this extended system **N**, and remark that all previously established properties continue to hold.

We write

$$\mathbf{N}\ \text{proves}\ \Gamma \vdash_\Sigma \alpha$$

to denote that the assertion $\Gamma \vdash_\Sigma \alpha$ is provable in the system **N**, *etc.*.

Finally, we note the shapes of the normal forms of the $\lambda\Pi$-calculus.

LEMMA 2.2.6 (CHARACTERIZATION OF NORMAL FORMS)

Terms in β- and $\beta\eta$-normal form have the following shapes:

1. A normal form kind has shape $\Pi x_1 : A_1 \ldots \Pi x_m : A_m \,.\, \mathrm{Type}$;
2. A normal form family of types has shape

$$\lambda x_1 : A_1 \ldots \lambda x_m : A_m \,.\, \Pi y_1 : B_1 \ldots \Pi y_n : B_n \,.\, @M_1 \ldots M_p$$

   for some m, n, $p > 0$;
3. A normal form object has shape $\lambda x_1 : A_1 \ldots \lambda x_m : A_m \,.\, @M_1 \ldots M_p$ for some m, $p > 0$.

PROOF

By induction on the structure of terms, bearing in mind that a normal form is a term with no subterms of the form $(\lambda x : A \,.\, U)M$ (for the β case) or of the form $\lambda x : A \,.\, Mx$ (for the η case). □

We note that it is possible to define a notion of *long $\beta\eta$-normal form.* Suppose that the object

$$\lambda x_1 : A_1 \ldots \lambda x_m : A_m \,.\, @M_1 \ldots M_p \tag{2.24}$$

is in $\beta\eta$-normal form and that

$$@ : \Pi y_1 : B_1 \ldots \Pi y_q : B_q \,.\, B,$$

where $q > p$. The *long $\beta\eta$-normal form* of the object 2.24 is the object

$$\lambda x_q : B_q \ldots \lambda x_{p+1} : B_{p+1} \,.\, \lambda x_1 : A_1 \ldots \lambda x_m : A_m \,.\, @M_1 \ldots M_p x_{p+1} \ldots x_q \,.$$

2.3 The Logical Framework

2.3.1 Introduction

In Sections 2.3.2 and 2.3.3 we describe and illustrate the LF's paradigm for encoding expressions and formulae by considering first-order logic and higher-order logic. (In Section 2.4.3 and 2.4.4 we describe and illustrate the LF's paradigm for encoding rules and proofs by considering the encoding of the rules and proofs of these logics.)

The LF does not deal with proof equality in the sense of Prawitz [Pr65]: the only counterpart to definitional equality is $\beta\eta$-conversion.

We use $\beta\eta$-conversion, rather that just β-conversion, in order to make the encoding of syntax more transparent. In [HHP87] the LF is presented using $\beta\eta$-conversion, but in [HHP89] it is presented using just β-conversion, so that there it is necessary to consider a notion of *canonical form* in order to replace the notion of long $\beta\eta$-normal form.

2.3.2 Theory of Expressions and Formulae

Closely following [HHP87], [HHP89] we describe the representation of the expressions and formulae of a given logic in the $\lambda\Pi$-calculus.

The paradigm for the formalization of the syntax of a language in the LF is inspired by Church's presentation of higher-order logic [Ch40] and Martin-Löf's system of arities [NPS86]. A syntactic category of a logic is represented by a type and the expressions of each category are built up using expression-forming constructors, which are represented by suitable constants of the $\lambda\Pi$-calculus. Thus expressions of a given syntactic category are represented by objects which inhabit the type which represents the given syntactic category. In this way an expression

is encoded as an object of the type associated with its category, given a context which provides suitable declarations for the free variables of the expression.[3]

Variable-binding operators are represented by constants whose domain is a higher type, so that binding is handled by the λ-calculus of the $\lambda\Pi$-calculus. An immediate consequence of this is that in a machine implementation of the $\lambda\Pi$-calculus it is necessary to implement α-conversion and variable-capture-avoiding substitution just once, at the level of the $\lambda\Pi$-calculus.

2.3.3 First-order Logic and Higher-order Logic

We illustrate the representation of syntax within the $\lambda\Pi$-calculus by considering two examples:

1. First-order logic with the language of individuals that of Peano arithmetic (as defined in Schoenfield [Sh67]).

2. Higher-order logic with the language of individuals that of Peano arithmetic, as defined in [Ch40].

The presentation of the syntax of Peano arithmetic will form a part of the signatures which correspond to both of these, Σ_{PA} and Σ_{HOL} respectively.

In a first-order language there are two syntactic categories: the *individual expressions*, which stand for individuals — the objects in the domain of quantification — and the *formulae*, which stand for propositions. These syntactic categories are represented in the $\lambda\Pi$-calculus by the type ι of individuals, and the type o of propositions. Thus Σ_{PA} begins with

$$\iota : \mathrm{Type}$$
$$o : \mathrm{Type}$$

---

[3]In precisely this sense there are no free variables in a $\lambda\Pi$-calculus assertion $\Gamma \vdash_{\Sigma} M : A$ which is well-formed: all of the free variables in $M : A$ must be declared in Γ.

The individual expression constructors of Peano arithmetic are formalized in Σ_{PA} by the following $\lambda\Pi$-calculus constants:

$$\begin{aligned} 0 &: \iota \\ \mathrm{succ} &: \iota \to \iota \\ + &: \iota \to \iota \to \iota \\ \times &: \iota \to \iota \to \iota \end{aligned}$$

Objects of type ι in Σ_{PA} represent the individual expressions of Peano arithmetic. There are no declarations for the variables of first-order logic: the variables of the object language are identified with the variables of the $\lambda\Pi$-calculus, so that an open term of type ι in Σ_{PA}, all of whose free variables are of type ι, represents an open individual expression. For example, in a context containing $x : \iota$, the term $\mathrm{succ}(x)$ has type ι as well, *i.e.*, the assertion $\Gamma_1, x{:}\iota, \Gamma_2 \vdash_\Sigma \mathrm{succ}(x) : \iota$ is provable (for suitable Γ_1 and Γ_2).

This representation is compositional and is defined by: $x^\circ = x$, $0^\circ = 0$, $\mathrm{succ}(t)^\circ = \mathrm{succ}(t^\circ)$, $t + u^\circ = +(t^\circ)(u^\circ)$, and $t \times u^\circ = \times(t^\circ)(u^\circ)$,

THEOREM 2.3.1 (ADEQUACY FOR SYNTAX, I)

The correspondence $^\circ$ is a bijection between the expressions of Peano arithmetic and the normal forms of type ι in Σ_{PA} with all free variables of type ι.

PROOF SKETCH

The translation is evidently well-defined and one-one. Surjectivity is proved by induction on the structure of the normal forms. The reader is referred to [HHP89] for the details of a similar proof for canonical forms rather than long $\beta\eta$-normal forms. □

The following are the other constant declarations for the formulae of Peano arithmetic:

$$\begin{aligned} = &: \iota \to \iota \to o & < &: \iota \to \iota \to o \\ \neg &: o \to o & \wedge &: o \to o \to o \\ \vee &: o \to o \to o & \supset &: o \to o \to o \\ \forall &: (\iota \to o) \to o & \exists &: (\iota \to o) \to o \end{aligned}$$

The encoding of formulae in the $\lambda\Pi$-calculus may be summarized by the mapping $^\circ$. This mapping is defined by induction on the structure of formulae as follows:

$$\begin{aligned}
(t = u)^\circ &= =(t^\circ)(u^\circ) \\
(t < u)^\circ &= <(t^\circ)(u^\circ) \\
(\neg\phi) &= \neg(\phi^\circ) \\
(\phi \wedge \psi)^\circ &= \wedge(\phi^\circ)(\psi^\circ) \\
(\phi \vee \psi)^\circ &= \vee(\phi^\circ)(\psi^\circ) \\
(\phi \supset \psi)^\circ &= \supset(\phi^\circ)(\psi^\circ) \\
(\forall x.\phi)^\circ &= \forall(\lambda x{:}\iota.(\phi^\circ)) \\
(\exists x.\phi)^\circ &= \exists(\lambda x{:}\iota.(\phi^\circ))
\end{aligned}$$

Note that the formula $\psi = \forall x.\phi[x]$ is represented by the term $\psi^\circ = \forall(\lambda x : \iota.\phi^\circ)$. As remarked earlier, this approach allows α-conversion and capture-avoiding-substitution to be factored out of the definition of each individual logic, leaving it to be implemented just once at the level of the framework. This treatment of binding operators relies on the variables of the first-order language being identified with the variables of the $\lambda\Pi$-calculus type theory. For example, if x and y are variables of type ι, then $x < y$ is a term of type o.[4] We can bind x by λ-abstraction, thereby obtaining $\lambda x : \iota.x < y$, and existentially quantify it by applying to it the constant $\exists$, thereby obtaining $\exists(\lambda x : \iota.x < y)$, which represents the first-order formula $\exists x.x < y$.

In this manner, each formula ϕ of Peano arithmetic is represented by a term ϕ° of type o in Σ_{PA}, all of whose free variables are of type ι: sentences are represented by the closed terms of type o. An open term M of type o is called an *incomplete formula*, and the λ-abstraction of an incomplete formula is a called *formula scheme*. For example,

$$M = \lambda\phi : o.\lambda\Phi : \iota \to o.\phi \supset \exists(\Phi)$$

---

[4]We use infix and postfix application as appropriate.

is a formula scheme which can be instantiated by application to a formula ϕ and a matrix Φ, so that the $\beta\eta$-normal form of the application

$$M(\forall(\lambda x : \iota.x = x))(\lambda x : \iota.x = x)$$

represents the first-order formula

$$\forall x.x = x \supset \exists x.x = x.$$

The next theorem makes precise the relationship between the formulae of Peano arithmetic and $\lambda\Pi$-calculus terms in the signature Σ_{PA}.

THEOREM 2.3.2 (ADEQUACY FOR SYNTAX, II)

The compositional translation $^\circ$ is a bijection between the formulae of Peano arithmetic and the long $\beta\eta$-normal forms of type o in Σ_{PA} with all free variables of type ι.

PROOF SKETCH

The proof is similar to that of Theorem 2.3.1. □

The rôle of η-conversion in the above proof is mainly to ensure that the well-typed terms of type o in Σ_{PA} are, up to the notion of definitional equality built in to the representation of the system, exactly the formulae of Peano arithmetic: there is no intrinsic difference between $\forall(<(0))$ and $\forall(\lambda x : \iota. < (0)(x))$, and indeed $\forall(\lambda x : \iota. < (0)(x)) \rightarrow_{\beta\eta} \forall(<(0))$.

The representation of the syntax of higher-order logic is given below. Quantification in higher-order logic is over a type drawn from the hierarchy of simple types with two base types ι (of individuals) and o (of propositions).

In order to avoid confusion with the types of the $\lambda\Pi$-calculus, we call the types of higher-order logic "sorts", and we shall write $\sigma \Rightarrow \tau$ for the sort of functions from sort σ to sort τ. In the formalization of higher-order logic the collection of sorts is represented as a type with members ι and o, closed under $\Rightarrow$. The

signature Σ_{HOL} thus begins as follows:

$$\begin{aligned} \mathrm{sorts} &: \mathrm{Type} \\ \iota &: \mathrm{sorts} \\ o &: \mathrm{sorts} \\ \Rightarrow &: \mathrm{sorts} \to \mathrm{sorts} \to \mathrm{sorts} \end{aligned}$$

To each sort is associated the type of objects of that sort. For pedagogic purposes we take the objects of sort ι to be the natural numbers. The objects of sort o are the propositions of higher-order logic. The quantifiers range over an arbitrary sort, rather than the fixed sort of individuals as in first-order logic. The objects of higher sort are given by typed λ-terms, and there is a form for expressing application.

$$
\begin{aligned}
\mathrm{obj} &: \mathrm{sorts} \to \mathrm{Type} \\
0 &: \mathrm{obj}(\iota) \\
\mathrm{succ} &: \mathrm{obj}(\iota \Rightarrow \iota) \\
= &: \Pi\sigma : \mathrm{sorts}.\, \mathrm{obj}(\sigma \Rightarrow \sigma \Rightarrow o) \\
\neg &: \mathrm{obj}(o \Rightarrow o) \\
\wedge &: \mathrm{obj}(o \Rightarrow o \Rightarrow o) \\
\vee &: \mathrm{obj}(o \Rightarrow o \Rightarrow o) \\
\supset &: \mathrm{obj}(o \Rightarrow o \Rightarrow o) \\
\forall &: \Pi\sigma : \mathrm{sorts}.\, \mathrm{obj}((\sigma \Rightarrow o) \Rightarrow o) \\
\exists &: \Pi\sigma : \mathrm{sorts}.\, \mathrm{obj}((\sigma \Rightarrow o) \Rightarrow o) \\
\Lambda &: \Pi\sigma : \mathrm{sorts}.\Pi\tau : \mathrm{sorts}. \\
&\qquad (\mathrm{obj}(\sigma) \to \mathrm{obj}(\tau)) \to \mathrm{obj}(\sigma \Rightarrow \tau) \\
\mathrm{ap} &: \Pi\sigma : \mathrm{sorts}.\Pi\tau : \mathrm{sorts}. \\
&\qquad \mathrm{obj}(\sigma \Rightarrow \tau) \to \mathrm{obj}(\sigma) \to \mathrm{obj}(\tau)
\end{aligned}
$$

The representation of equality and the quantifiers makes use of the dependent function types of the $\lambda\Pi$-calculus. For each sort σ, the equality relation for objects of sort σ is written $=_\sigma$; it is an object of sort $\sigma \Rightarrow \sigma \Rightarrow o$. Similarly, the quantifiers ranging over sort σ are written $\forall_\sigma$ and $\exists_\sigma$; they are objects of sort $(\sigma \Rightarrow o) \Rightarrow o$, just as in Church's formulation. The Λ and ap forms must similarly be tagged with types, which we write as subscripts. The Λ form must be tagged with both the domain and range types, unlike in Church's definition.

Adequacy theorems similar to Theorems 2.3.1 and 2.3.2 can be proved.

2.4 Theory of Rules and Proofs

The representation of inference rules and proofs is the central idea of the LF, and is inspired by the basic philosophical theory of logic presented by Martin-Löf in [ML85]. Martin-Löf isolates the essential logical notion as being that of the *judgement*, the unit of inference in a logic.

We present a brief survey of the ideas and following closely the development offered by [HHP87] and [AHM89], we provide an analysis of each of the notions pertinent to rules and proofs.

2.4.1 Basic Judgements

Each logic comes with a set of *basic* judgements: the reader is referred to appendix A or [AHM87] for a list of encoded logics. In first-order logic there is just one form of basic judgement — the assertion ϕ true that a formula ϕ is (logically) true (usually written as $\vdash \phi$ or just ϕ); in the modal logic S4 there are two judgements — the assertion that a formula is true (logically) and the assertion that a formula is valid (logically). Sequent calculi also have one basic judgement, written $\Gamma \Rightarrow \Delta$, the assertion that some formula in Δ is a logical consequence of all the formulae in Γ. In Martin-Löf's system of type theory, there are four basic judgements, A type, $A = B$, $a \in A$, and $a = b \in A$.

2.4.2 The Notion of Proof

Having identified the basic notion of judgement we must now consider the notion of proof. There are several approaches to the definition of a proof in a formal system [Sh67], [Pr65]. One view of proofs is as sequences of formulae that satisfy the condition that each formula is obtained from previous formulae by application of a rule: another view is that proofs are trees satisfying certain (formation) conditions. In traditional presentations of logical systems the inference rules determine the

set of proofs of basic judgements, and thereby also determine the set of "correct" or "evident" [ML85] basic judgements, namely those that have proofs; and such presentations are consistent with the definitions of proofs described above. The essential point is that the *notion* of a proof is independent of the particular rules of inference.

2.4.3 The Notion of Proof in the LF

In the LF the notion of proof is extended to include the view that rules are proofs of *higher-order judgements.* There are two forms: the *hypothetical* and the *schematic.* The hypothetical judgement

$$J_1 \vdash J_2$$

is the assertion that J_2 is a logical consequence of J_1, according to the rules of the logic. It is proved by a function mapping proofs of J_1 to proofs of J_2. The schematic judgement

$$\bigwedge_{x:A} J(x)$$

represents the idea of generality: the judgement $J(x)$ is *evident* [ML85] for any object x of type A: it is proved by a function mapping objects x of type A to proofs of $J(x)$. Thus rules and proofs are represented in the LF as terms of the $\lambda\Pi$-calculus: rule schemes are represented as proofs of schematic judgements and instantiation is represented by application.

Since a proof is viewed as evidence for a judgement, it is natural to identify judgements with the type of their proofs: a judgement is evident if and only if it has a proof if and only if there is a term of that type (in the signature of the logic). The type of proofs of a basic judgement is determined by the inference rules of the logic. The types of proofs of the higher-order judgement forms are defined by the type theory of the $\lambda\Pi$-calculus. Thus we define

$$J_1 \vdash J_2 \quad \text{to be} \quad J_1 \to J_2\,,$$

the type of functions mapping J_1 to J_2. Similarly, we define

$$\bigwedge_{x:A} J(x) \quad \text{to be} \quad \Pi x{:}A.J(x)\,,$$

the type of functions mapping objects x of type A to $J(x)$. This is called the *judgements as types* principle, by analogy with the propositions as types principle of Curry, deBruijn, and Howard.

It is convenient to write

$$J_1, \ldots, J_m \vdash_{x_1:A_1,\ldots,x_n:A_n} J$$

for

$$\bigwedge_{x_1:A_1} \cdots \bigwedge_{x_n:A_n} J_1 \vdash \cdots (J_m \vdash J).$$

This latter form generalizes Martin-Löf's *hypothetico-general* judgements [ML85] in that Martin-Löf's form is defined only for the basic judgements prop and true.

Note that this mode of representation makes no commitment to the semantics of a logic. It merely formalizes the idea that to make an assertion in a logical system, one must have a proof of it.

A $\lambda\Pi$-calculus type encodes an *open concept i.e.,* weakening is admissible (see Theorem 2.2.1). A judgement J, which is used in the representation of a formal system, corresponds to a *consequence relation* which is definable in that system. An object of type $J(\phi) \to J(\psi)$ encodes not only a function from proofs of $J(\phi)$ to proofs of $J(\psi)$, but also the information that ψ *follows from* ϕ in the given system. Consequence relations are discussed very fully in [Av87a] and [Av87c].

The consequence relations that are encoded by the judgements of the $\lambda\Pi$-calculus are the *ordinary, single-conclusioned* consequence relations described in [Av87a]. We note that it is a corollary of Theorem 2.2.1 that judgements in the LF encode consequence relations that satisfy weak forms of thinning, transitivity, and contraction.

As noted in Section 2.3 and earlier in this section, we attempt to achieve as much uniformity in the representation of formal systems in the LF as possible; the attempt to identify the variables of the system to be represented and the variables of the $\lambda\Pi$-calculus is one manifestation of this policy; encodings in which this identification is made are called *natural encodings.* Another is the general paradigm for the encoding of rules in which all the purely deductive aspects of

the form of a rule are encoded by the $\rightarrow$ connective of the $\lambda\Pi$-calculus, and in which all schematic aspects of the rule are encoded by the Π connective of the $\lambda\Pi$-calculus. For example, consider the formulation of the disjunction elimination rule for classical logic taken in [AHM87]

$$\frac{\Gamma_1, \phi \vdash \vartheta \quad \Gamma_2, \psi \vdash \vartheta \quad \Gamma_3 \vdash \phi \vee \psi}{\Gamma_1, \Gamma_2, \Gamma_3 \vdash \vartheta}$$

in which $\vdash$ is the consequence relation between sentences and in which the horizontal bar of the rule is the consequence relation on sequents. Both of these are represented using $\rightarrow$, and the fact that this is a rule for all suitable ϕ, ψ and ϑ is represented by quantifying (with Π) over each of these. Thus this rule is encoded by introducing a constant $\vee$E of type

$$\Pi\phi{:}o.\Pi\psi{:}o.\Pi\vartheta{:}o.(\mathrm{true}(\phi) \rightarrow \mathrm{true}(\vartheta)) \rightarrow (\mathrm{true}(\psi) \rightarrow \mathrm{true}(\vartheta))$$

$$\rightarrow \mathrm{true}(\phi \vee \psi) \rightarrow \mathrm{true}(\vartheta).$$

We take *incomplete proofs* to be open terms of judgement type J. They can be completed by substitution or by λ-abstraction, yielding proofs of schematic judgements. Abstraction on judgement type variables not occurring in J yields proofs of hypothetical judgements.

An important consequence of the judgements as types principle is that proof checking is reduced to type checking. A term M is a proof of a judgement J if and only if M has type J in the signature of the logic. This reduction is the most important reason for insisting that the type theory of the $\lambda\Pi$-calculus be decidable, for otherwise one could not construct a mechanical proof checker for a logic. Indeed many of the techniques that we develop in this thesis, especially those of Chapters 6 and 7, depend on decidability.

An encoding of a logic $\mathcal{L}$ is said to be *adequate* if there is a compositional bijection[5] between the formulae and theorems of $\mathcal{L}$ and a class of $\lambda\Pi$-calculus terms in the signature of the encoding.

---

[5]Here "compositional" means that substitution commutes with the bijective map.

2.4.4 First-order Logic and Higher-order Logic

To illustrate these ideas we continue with the representation of first-order and higher-order logic and include the representations of an illustrative selection of rules from first-order and higher-order logic formalized as systems of natural deduction, as presented in [HHP87].

First-order logic Returning to Σ_{PA}, the single judgement form ϕ true of first-order logic is represented by introducing a family of types indexed by the propositions:

$$\mathrm{true} : o \rightarrow \mathrm{Type}$$

So we see that for any formula ϕ (*i.e.*, any term of type o in Σ_{PA}), the type ϕ true (where appropriate we write "ϕ true" for "true(ϕ)") is the type of proofs of ϕ. Table 2.2 contains some rules of Peano arithmetic.

$$(\neg\neg\text{E})\ \frac{\neg\neg\phi}{\phi}$$

$$(\forall\text{I})\ \frac{\phi(x)}{\forall x.\phi(x)}$$

where x is not free in any assumption on which ϕ depends.

$$(\supset\text{I})\ \frac{\begin{matrix}(\phi)\\ \psi\end{matrix}}{\phi \supset \psi}$$

$$(\forall\text{E})\ \frac{\forall x.\phi(x)}{\phi[a/x]}$$

$$(\exists\text{E})\ \frac{\exists x.\phi(x)\quad \begin{matrix}(\phi)\\ \psi\end{matrix}}{\psi}$$

where x is not free in any assumption on which ψ depends.

$$(\text{IND})\ \frac{\phi(0)\quad \begin{matrix}(\phi(x))\\ \phi(\text{succ}(x))\end{matrix}}{\phi(x)}$$

where x is not free in any assumption, other than $\phi(x)$, on which $\phi(\text{succ}(x))$ depends.

Table 2.2

(Some rules of Peano Arithmetic)

Each rule in Table 2.2 is represented by a constant whose type is the specification of the rule, a higher-order judgement. For instance, the double negation elimination rule is given by:

$$\neg\neg\mathrm{E} : \neg\neg\phi \text{ true} \vdash_{\phi:o} \phi \text{ true}$$

The judgement is schematic in propositions ϕ, reflecting the fact that there is an instance of the $\neg\neg$E rule for each proposition ϕ, and hypothetical in proofs of $\neg\neg\phi$, so if ϕ is a formula and M is a proof of $\neg\neg\phi$ true, then $\neg\neg\mathrm{E}(\phi)(M)$ is a proof of ϕ true.

The implication introduction rule makes use of the hypothetical judgement form to model discharge. The formulation of $\supset$I in Table 2.2 takes a *hypothetical proof of* ϕ as premiss, and discharges the hypothesis. In the LF representation $\supset$I is treated as taking a *proof of a hypothetical judgement* so that the general intention is that a sufficient condition for establishing the truth of $\phi \supset \psi$ is to establish that ψ is a logical consequence of ϕ. The formalization of $\supset$I, which is schematic in ϕ and ψ, is

$$\supset\mathrm{I} : (\phi \text{ true} \vdash \psi \text{ true}) \vdash_{\phi:o,\psi:o} \phi \supset \psi \text{ true}$$

So, for example, $\supset\mathrm{I}(\phi)(\phi)(\lambda x : \phi \text{ true}.x)$ is a proof of $\phi \supset \phi$ true.

Universal elimination is given by

$$\forall\mathrm{E} : \forall(\Phi) \text{ true} \vdash_{\Phi:\iota\to o,a:\iota} \Phi(a) \text{ true}$$

The rule is schematic in Φ, the matrix of the universally quantified formula, and a, the instance. Given such and a proof M of $\forall(\Phi)$ true, $\forall\mathrm{E}(\Phi)(a)(M)$ is a proof of $\Phi(a)$ true. The action of substitution is modelled by applying the matrix to the instance.

Universal introduction is formalized like implication introduction. A condition for the truth of $\forall(\Phi)$ is that $\Phi(x)$ is true for arbitrary x. In Table 2.2 variable occurrence conditions are used for a *schematic proof of the judgement* $\Phi(x)$.

In the LF representation a *proof of the schematic judgement*, $\bigwedge_{x:\iota} \Phi(x)$, is used. The rule is given by

$$\forall\mathrm{I} : (\textstyle\bigwedge_{x:\iota} \Phi(x) \text{ true}) \vdash_{\Phi:\iota\to o} \forall(\Phi) \text{ true}$$

Existential elimination uses both hypothetical and schematic judgements, and makes use of scoping to avoid side conditions:

$$\exists\mathrm{E}: \ \exists(\Phi)\ \mathrm{true}, (\Phi(x)\ \mathrm{true} \vdash_{x:\iota} \psi\ \mathrm{true}) \vdash_{\Phi:\iota\to o,\psi:o} \psi\ \mathrm{true}$$

Since ψ is bound outermost, there is no possibility that the x of the schematic judgement form occur free in an instance.

Induction makes use of scoping and higher-order judgements:

$$\mathrm{IND}: \ \Phi(0)\ \mathrm{true}, (\Phi(x)\ \mathrm{true} \vdash_{x:\iota} \Phi(\mathrm{succ}(x))\ \mathrm{true}) \vdash_{\Phi:\iota\to o,x:\iota} \Phi(x)\ \mathrm{true}$$

The correctness of the formalization is expressed by the next theorem.

THEOREM 2.4.1 (ADEQUACY FOR THEOREMS)

There is a compositional bijection between first-order natural deduction proofs of a formula ϕ of Peano arithmetic from assumptions $\phi_1, \ldots, \phi_n$ and long $\beta\eta$-normal forms M of type ϕ° true in Σ_{PA}, all of whose free variables are of type ι or ϕ_i° true $(1 \leq i \leq n)$.

PROOF SKETCH

It is straightforward to prove by induction on the length of derivations, that if $A_1, \ldots, A_n \vdash_{\mathrm{PA}} A$ is derivable, then

$$\Gamma, x_1 : A_1\ \mathrm{true}, \ldots, x_n : A_n\ \mathrm{true} \vdash_{\Sigma_{\mathrm{PA}}} M\colon A_{n+1}\ \mathrm{true}$$

is derivable, where Γ contains assignments of the form $x : \iota$ for the free object variables x occurring in the A_is and in M, and where M faithfully encodes instantiation and application of rules. Surjectivity can be proved by induction on the structure of the normal forms of type ϕ true (for $\phi : o$), keeping in mind the uniqueness of types and the Church–Rosser property. The reader is referred to [HHP89] for the details of the proof of a similar proof for canonical forms rather than long $\beta\eta$-normal forms. □

Note that it is possible for M in the above proof to have free variables of type ι, even when $n = 0$ (*i.e.*, when there are no assumptions) and when ϕ has no free variables. This is true, for example, in a proof of

$$\forall x.\phi(x) \supset \exists x.\phi(x).$$

It is a peculiarity of informal presentations of first-order logic that the assumption that the domain of quantification is non-empty is not made explicit in proofs.[6]

We include two examples of proofs as $\lambda\Pi$-calculus terms which are taken from [HHP87]. The first is a proof of $\phi \supset (\psi \supset \phi)$ true as a well-typed term in the signature Σ_{PA}. Let x have type ϕ true, and let y have type ψ true. Abstracting the incomplete proof x with respect to y, we obtain a proof $\lambda y : \psi \text{ true}.x$ of $\psi \text{ true} \vdash \phi$ true. Applying $\supset$I to this proof, we obtain the (incomplete) proof $\supset\mathrm{I}(\psi)(\phi)(\lambda y : \psi \text{ true}.x)$ of $\psi \supset \phi$ true. Abstracting with respect to x, and applying $\supset$I again, we obtain the complete proof

$$\supset\mathrm{I}(\phi)(\psi \supset \phi)(\lambda x : \phi \text{ true}.\supset\mathrm{I}(\psi)(\phi)(\lambda y : \psi \text{ true}.x))$$

of $\phi \supset (\psi \supset \phi)$ true.

The second example illustrates — and indeed as claimed in [HHP87], provides evidence for — the claim that the traditional notion of a derivable rule has a formal counterpart in the $\lambda\Pi$-calculus.[7] We show that in the signature Σ_{PA} the

---

[6]Without this assumption the transitivity of entailment is lost, see for example [Hod77].

[7]A full study of the encoding of derivable (and admissible) rules in type theories is beyond the scope of this chapter — indeed, it is a current research problem. However, we recall that since thinning is an admissible rule in the $\lambda\Pi$-calculus, judgements are "open" concepts, which implies that an induction principle on proofs is not available. Consequently, typical admissible rules for a given logic $\mathcal{L}$, or metarules such as the deduction theorem for a Hilbert-style presentation of first-order logic, are not directly derivable in certain adequate signatures for $\mathcal{L}$.

elimination rule for the universal quantifier in Schroeder-Heister's style can be derived. The Schroeder-Heister rule is specified by the following type:

$$\forall(\Phi)\ \text{true}, ((\bigwedge_{x:\iota} \Phi(x)\ \text{true}) \vdash \psi\ \text{true}) \vdash_{\Phi:\iota\to o,\psi:o} \psi\ \text{true}.$$

It can be verified easily that the term:

$$\lambda\Phi : \iota \to o.\lambda\psi : o.\lambda p : \forall(\Phi)\ \text{true}.$$
$$\lambda q : ((\bigwedge_{x:\iota} \Phi(x)\ \text{true}) \vdash \psi\ \text{true}).$$
$$q(\lambda x : \iota.\forall\text{E}(\Phi)(x)(p))$$

has the above type.

Higher-order logic The inference rules of higher-order logic are represented in the $\lambda\Pi$-calculus in a manner which is quite similar to the representation of first-order logic. There is one judgement form, the assertion that ϕ is true, for ϕ an object of sort o.

$$\text{true} : \text{obj}(o) \to \text{Type}$$

The rules of β- and η-conversion for the λ-calculus appear as axioms about equality: they are schematic in the domain and range sorts of the functions, and in the terms themselves:

$$\beta : \bigwedge_{\sigma:\text{sorts},\tau:\text{sorts},f:\text{obj}(\sigma)\to\text{obj}(\tau),a:\text{obj}(\sigma)} \text{ap}_{\sigma,\tau}(\Lambda_{\sigma,\tau}(f), a) =_\tau f(a)\ \text{true}$$

$$\eta : \bigwedge_{\sigma:\text{sorts},\tau:\text{sorts},f:\text{obj}(\sigma\Rightarrow\tau)} \Lambda_{\sigma,\tau}(\lambda x : \text{obj}(\sigma).\, \text{ap}_{\sigma,\tau}(f, x)) =_{\sigma\Rightarrow\tau} f\ \text{true}$$

Strictly speaking, the equations in the above axioms should be written using ap, for the type of $=_\tau$ is $\text{obj}(\tau \Rightarrow \tau \Rightarrow o)$.

The formalization of the logical rules is similar to that of first-order logic. The universal introduction and elimination rules are formalized as follows:

$$\forall\text{I} : (\bigwedge_{x:\text{obj}(\sigma)} \text{ap}_{\sigma,o}(f, x)\ \text{true}) \vdash_{\sigma:\text{sorts},f:\text{obj}(\sigma\Rightarrow o)} \forall_\sigma(f)\ \text{true}$$

$$\forall\text{E} : \forall_\sigma(f)\ \text{true} \vdash_{\sigma:\text{sorts},f:\text{obj}(\sigma\Rightarrow o),a:\text{obj}(\sigma)} \text{ap}_{\sigma,o}(f, a)\ \text{true}$$

The adequacy of this representation of higher-order logic can be established by means similar to that for Peano arithmetic.

2.5 Summary

In this chapter we have presented the basic syntactic theory of the $\lambda\Pi$-calculus — including the major syntactic metatheorems — and the LF — including two important examples of the representation paradigm. The basic model theory of the $\lambda\Pi$-calculus, including the definition of the (closed) term model is presented in Appendix A. We shall need this model theory in order to provide a model-theoretic characterization of the semantics for the notion of logic programming which we develop in Chapter 7,

We note that the problem of providing a theory of the embedding of models of object-logics in models of the type theory is open.

Chapter 3

The Gentzenized $\lambda\Pi$-calculus

3.1 Introduction

The inference rules of the system **N** represent a linearized natural deduction presentation of the $\lambda\Pi$-calculus. The system is said to be:

- *Linearized* because the hypotheses are recorded in the context;
- *Natural deduction* because of the form of the application rules — they resemble the implication elimination rule of natural deduction formulations of intuitionistic logic [Ge34], [Pr65], [How80].

In this chapter we present a system **G** that is a formulation of the inference rules of the $\lambda\Pi$-calculus that resembles a sequent calculus in the style of Gentzen's LJ [Ge34], [Av87a]. This resemblance is elucidated in the light of the formulae-as-types isomorphism between proofs in intuitionistic propositional logic and the terms of the simply-typed lambda calculus [How80].

In this setting, the ($\rightarrow$E) rule of the system **N**:

$$(\rightarrow \mathrm{E})\ \frac{\Gamma \vdash_\Sigma M : A \rightarrow B \qquad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma MN : B} \tag{3.1}$$

corresponds to the sequent calculus rule:

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \phi \supset \psi}{\Gamma \vdash \psi} \tag{3.2}$$

of intuitionistic propositional logic (in the obvious notation, see [How80]). In the system **G** the ($\rightarrow$E) rule is replaced by the ($\rightarrow l$) rule:

$$(\rightarrow l)\ \frac{@{:}A \rightarrow B \in \Sigma \cup \Gamma \quad \Gamma \vdash_\Sigma N{:}A \quad \Gamma, y{:}B \vdash_\Sigma M{:}D \quad y \notin \mathrm{FV}(D)}{\Gamma \vdash_\Sigma M[@N/y]{:}D}, \tag{3.3}$$

which says that given some constant or variable $@ : A \rightarrow B \in \Sigma \cup \Gamma$, some object N of type A in the context Γ, then given an object M which inhabits the type D in the context $\Gamma, y{:}B$, ($\rightarrow l$) constructs an object $M[@N/y]$ which inhabits the type D in the context Γ. Under the formulae-as-types isomorphism, the corresponding sequent calculus rule of intuitionistic propositional logic is

$$\frac{\Gamma \vdash \phi \quad \psi, \Delta \vdash \gamma}{\phi \supset \psi, \Gamma, \Delta \vdash \gamma}, \tag{3.4}$$

see [How80], §5.

In a similar way, the (ΠE) rule of the system **N**:

$$(\Pi\mathrm{E})\ \frac{\Gamma \vdash_\Sigma M : \Pi x{:}A\,.\,B \quad \Gamma \vdash_\Sigma N{:}A}{\Gamma \vdash_\Sigma MN{:}B[N/x]}, \tag{3.5}$$

is replaced in the system **G** by the (Πl) rule:

$$(\Pi l)\ \frac{@{:}\Pi x{:}A.B \in \Sigma \cup \Gamma \quad \Gamma \vdash_\Sigma N{:}A \quad B[N/x] =_{\beta\eta} C \quad \Gamma, y{:}C \vdash_\Sigma M{:}D \quad y \notin \mathrm{FV}(D)}{\Gamma \vdash_\Sigma M[@N/y]{:}D}, \tag{3.6}$$

which is understood in a manner similar to the ($\rightarrow l$) rule.

The extension of the formulae-as-types correspondence to universal quantifiers and dependent types in the setting of *Generalized Type Systems* is reported in the work of Barendregt [Ba89]. This work is due variously to Berardi (1988), Terlouw (1988), Geuvers (1988) and Barendsen (1989). The extension of this correspondence to Gentzen-style Generalized Type Systems and sequent calculi is work in progress.

The system **G** includes the cut rule:

$$\frac{\Gamma, x{:}A \vdash_\Sigma \alpha \quad \Gamma \vdash_\Sigma N{:}A}{\Gamma \vdash_\Sigma \alpha[N/x]}.$$

We prove the soundness and completeness of the system **G** with respect to the system **N**, and prove a cut elimination theorem for the $\lambda\Pi$-calculus by proving

that the system G\cut is complete with respect to the system N provided we work with inhabiting objects in β-normal form. While this might seem surprising at first sight, as we might expect completeness for *all* inhabiting objects, by analogy with usual completeness of systems without cut, it actually corresponds to the fact that in such systems we do not get completeness for proofs, but rather for every proof in the system with cut *which is in normal form* we get a proof in the system without cut. Indeed, our proof of the completeness of the system without cut may be considered to be analogous to that given by Prawitz [Pr65] for proofs in normal form.

3.2 The Gentzenized $\lambda\Pi$-calculus

3.2.1 The System G

We present the inference rules of the Gentzenized $\lambda\Pi$-calculus. Just as in sequent calculus presentations of intuitionistic first-order logic, the rules divide in several distinct groups.

1. **Basic Rules**

$$\frac{}{\vdash \langle\rangle \ \text{sig}} \tag{3.7}$$

$$\frac{\vdash \Sigma \ \text{sig} \quad \vdash_\Sigma K \ \text{kind} \quad c \notin \text{Dom}(\Sigma)}{\vdash \Sigma, c : K \ \text{sig}} \tag{3.8}$$

$$\frac{\vdash \Sigma \ \text{sig} \quad \vdash_\Sigma A\colon \text{Type} \quad c \notin \text{Dom}(\Sigma)}{\vdash \Sigma, c : A \ \text{sig}} \tag{3.9}$$

$$\frac{\vdash \Sigma \ \text{sig}}{\vdash_\Sigma \langle\rangle \ \text{context}} \tag{3.10}$$

$$\frac{\vdash_\Sigma \Gamma \ \text{context} \quad \Gamma \vdash_\Sigma A\colon \text{Type} \quad x \notin \text{Dom}(\Gamma)}{\vdash_\Sigma \Gamma, x : A \ \text{context}} \tag{3.11}$$

$$\frac{\vdash_\Sigma \Gamma \ \text{context}}{\Gamma \vdash_\Sigma \ \text{Type} \ \text{kind}} \tag{3.12}$$

$$\frac{\Gamma \vdash_\Sigma A\colon \text{Type} \quad \Gamma, x : A \vdash_\Sigma K \ \text{kind} \quad x \notin \text{FV}(K)}{\Gamma \vdash_\Sigma A \to K \ \text{kind}} \tag{3.13}$$

$$\frac{\Gamma \vdash_\Sigma A\colon \text{Type} \quad \Gamma, x : A \vdash_\Sigma K \ \text{kind}}{\Gamma \vdash_\Sigma \Pi x{:}A\,.\,K \ \text{kind}} \tag{3.14}$$

$$\frac{\Gamma \vdash_\Sigma A\colon \text{Type} \quad \Gamma, x : A \vdash_\Sigma B : \ \text{Type} \quad x \notin \text{FV}(B)}{\Gamma \vdash_\Sigma A \to B : \ \text{Type}} \tag{3.15}$$

$$\frac{\Gamma \vdash_\Sigma A\colon \text{Type} \quad \Gamma, x : A \vdash_\Sigma B : \ \text{Type}}{\Gamma \vdash_\Sigma \Pi x{:}A\,.\,B : \ \text{Type}} \tag{3.16}$$

$$\frac{\vdash_\Sigma \Gamma \ \text{context} \quad c{:}K \in \Sigma}{\Gamma \vdash_\Sigma c{:}K} \tag{3.17}$$

$$\frac{\vdash_\Sigma \Gamma \ \text{context} \quad c{:}A \in \Sigma}{\Gamma \vdash_\Sigma c{:}A} \tag{3.18}$$

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad x{:}A \in \Gamma}{\Gamma \vdash_\Sigma x{:}A} \tag{3.19}$$

The basic rules are precisely the rules for valid signatures and valid contexts of the system **N**. Several other equivalent formulations of these rules are possible.

2. **Operational Rules**

$$\frac{\Gamma \vdash_\Sigma A{:}\,\text{Type} \quad \Gamma, x{:}A \vdash_\Sigma B{:}K \quad x \notin \text{FV}(K)}{\Gamma \vdash_\Sigma \lambda x{:}A.B{:}A \to K} \tag{3.20}$$

$$\frac{\Gamma \vdash_\Sigma A{:}\text{Type} \quad \Gamma, x{:}A \vdash_\Sigma M{:}B \quad x \notin \text{FV}(B)}{\Gamma \vdash_\Sigma \lambda x{:}A.M{:}A \to B} \tag{3.21}$$

$$\frac{\Gamma \vdash_\Sigma A{:}\,\text{Type} \quad \Gamma, x{:}A \vdash_\Sigma B{:}K}{\Gamma \vdash_\Sigma \lambda x{:}A.B{:}\Pi x{:}A.K} \tag{3.22}$$

$$\frac{\Gamma \vdash_\Sigma A{:}\text{Type} \quad \Gamma, x{:}A \vdash_\Sigma M{:}B}{\Gamma \vdash_\Sigma \lambda x{:}A.M{:}\Pi x{:}A.B} \tag{3.23}$$

$$\frac{@{:}A \to B \in \Sigma \cup \Gamma \quad \Gamma \vdash_\Sigma N{:}A \quad \Gamma, y{:}B \vdash_\Sigma M{:}D \quad y \notin \text{FV}(D)}{\Gamma \vdash_\Sigma M[@N/y]{:}D} \tag{3.24}$$

$$\frac{@{:}\Pi x{:}A.B \in \Sigma \cup \Gamma \quad \Gamma \vdash_\Sigma N{:}A \quad B[N/x] =_{\beta\eta} C \quad \Gamma, y{:}C \vdash_\Sigma M{:}D \quad y \notin \text{FV}(D)}{\Gamma \vdash_\Sigma M[@N/y]{:}D} \tag{3.25}$$

$$\frac{\Gamma \vdash_\Sigma N{:}A \quad \Gamma \vdash_\Sigma B{:}A \to K}{\Gamma \vdash_\Sigma BN{:}K} \tag{3.26}$$

$$\frac{\Gamma \vdash_\Sigma N{:}A \quad \Gamma \vdash_\Sigma B{:}\Pi x{:}A.K}{\Gamma \vdash_\Sigma BN{:}K[N/x]} \tag{3.27}$$

All of these rules are the same as those in **N**, except for the ($\to l$) 3.24 and (Πl) 3.25 rules which replace the ($\to$E) 3.1 (and 2.23) and (ΠE) 3.5 (and 2.16) rules.

3. **Structural Rules**

$$\frac{\Gamma \vdash_\Sigma M{:}A \quad \Gamma, x{:}A, \Delta \vdash_\Sigma \alpha}{\Gamma, \Delta[M/x] \vdash_\Sigma \alpha[M/x]} \tag{3.28}$$

We include the cut 3.28 rule explicitly in the system **G**. It is not admissible in the system **G**\cut. To see this, note that the rules of **G**\cut allow the derivation of β-normal forms only, whereas the cut rule allows the derivation of β-redexes, since the term M (in 3.28) can be of the form $\lambda y{:}E\,.\,P$.

4. **Equality Rules**

$$\frac{\Gamma \vdash_\Sigma M:A \quad \Gamma \vdash_\Sigma A': \text{Type} \quad A =_{\beta\eta} A'}{\Gamma \vdash_\Sigma M:A'} \tag{3.29}$$

$$\frac{\Gamma \vdash_\Sigma A:K \quad \Gamma \vdash_\Sigma K' \text{ kind} \quad K =_{\beta\eta} K'}{\Gamma \vdash_\Sigma A:K'} \tag{3.30}$$

The equality rules are the same as those taken in the system **N**.

We remark that we should like to present Gentzen-style replacements for the application rules for kinds, 3.26 and 3.27. However, for the given syntax of the $\lambda\Pi$-calculus this is not possible because kind declarations are not permitted in contexts. In particular this means that we are not able to have a premiss of the form $\Gamma, x:K \vdash_\Sigma \alpha$, where K is a kind expression, which is essential for such a rule. However, if we reformulate the $\lambda\Pi$-calculus without signatures but with kind declarations in contexts, such rules are available. In particular, the (Πl) rule for both kinds and types may then take the form:

$$\frac{@:\Pi x:A.R \in \Sigma \cup \Gamma \quad \Gamma \vdash_\Sigma N:A \quad R[N/x] =_{\beta\eta} S \quad \Gamma, y:S \vdash_\Sigma U:V \quad y \notin \mathrm{FV}(D)}{\Gamma \vdash_\Sigma U[@N/y]:V}$$

where R and S are either kind expressions or type expressions and where $U:V$ asserts either that the type U has kind V or that the object U has type V. The corresponding $(\rightarrow l)$ rule is similar.

A natural deduction presentation of such a system without signatures but with kind declarations permitted in contexts may be found in [HHP89]; and for this system it is possible to prove all of the results of this chapter for a Gentzen-style system with rules of the form described above.

We note that *weakening*, *contraction* and *permutation* rules are admissible in **G**.

PROPOSITION 3.2.1 (ADMISSIBILITY OF WEAKENING)

The weakening rule:

$$\frac{\Gamma \vdash_\Sigma \alpha \quad \vdash_{\Sigma,\Sigma'} \Gamma, \Gamma' \text{ context}}{\Gamma, \Gamma' \vdash_{\Sigma,\Sigma'} \alpha},$$

is admissible in **G**.

PROOF

The proof is by induction on the structure of proofs in **G**, and we omit the details. □

PROPOSITION 3.2.2 (ADMISSIBILITY OF CONTRACTION)

The contraction rule:

$$\frac{\Gamma, x{:}A, \Delta, y{:}A, \Theta \vdash_\Sigma \alpha}{\Gamma, x{:}A, \Delta, \Theta[x/y] \vdash_\Sigma \alpha[x/y]},$$

is admissible in **G**.

PROOF

We note that the admissibility of the contraction rule is inevitable in the presence of the cut rule: the contraction rule is merely the cut rule for the special case in which the substituting object is a variable. □

PROPOSITION 3.2.3 (ADMISSIBILITY OF PERMUTATION)

The permutation rule:

$$\frac{\Gamma, x{:}A, y{:}B, \Delta \vdash_\Sigma \alpha \qquad x \notin \mathrm{FV}(B)}{\Gamma, y{:}B, x{:}A, \Delta \vdash_\Sigma \alpha},$$

is admissible in **G**.

PROOF

The proof is by induction on the structure of proofs in **G**, and we omit the details. □

We remark that this permutation rule is a very weak one compared to that which obtains in first-order logic. This is, of course, due to the partial ordering of our contexts which is due to the presence of Π-types in the language. The weakness of this rule adumbrates much of the work which we shall do in Chapter 6, where we consider very precisely the order of construction of contexts.

3.2.2 Soundness and Completeness of G

In this section we prove the soundness and completeness of **G** with respect to **N**. These results crucially depend upon the cut rule.

THEOREM 3.2.4 (SOUNDNESS OF G)

If **G** proves $\Gamma \vdash_\Sigma U:V$ then **N** proves $\Gamma \vdash_\Sigma U:V$.

PROOF

The proof proceeds by induction on the structure of proofs in the system **N**.

Since by Theorem 2.2.1 the cut rule is admissible in **N**, the only remaining differences between **N** and **G** are that the ($\to$E) 3.1 and (ΠE) 3.5 rules of **N** are replaced in **G** by the ($\to l$) 3.24 and (Πl) 3.25 rules. Thus there are several simple cases in the induction, which correspond to the basic rules and to the ($\to r$) and (Πr) rules and which we omit; and two more difficult cases, which correspond to the ($\to l$) and (Πl) rules and which we include. Since the systems are only different for assertions that an object inhabits a type, we assume that $U:V$ is of the form $M:A$.

Suppose the last rule of **G** applied is ($\to l$):

$$\frac{\Gamma \vdash_\Sigma N:A \quad @:A \to B \in \Sigma \cup \Gamma \quad \Gamma, y:B \vdash_\Sigma M:D \quad y \notin \mathrm{FV}(D)}{\Gamma \vdash_\Sigma M[@N/y]:D}.$$

By the induction hypothesis we have that

$$\mathbf{N} \text{ proves } \Gamma \vdash_\Sigma N:A,$$

that

$$\mathbf{N} \text{ proves } \Gamma, y:B \vdash_\Sigma M:D$$

and that

$$\mathbf{N} \text{ proves } \Gamma \vdash_\Sigma @ : A \to B.$$

From **N** proves $\Gamma \vdash_\Sigma @ : A \to B$ and **N** proves $\Gamma \vdash_\Sigma N : A$ we obtain

$$\mathbf{N} \text{ proves } \Gamma \vdash_\Sigma @N : B$$

by 3.1. From this and **N** proves $\Gamma \vdash_\Sigma M : D$ we obtain

$$\mathbf{N} \text{ proves } \Gamma \vdash_\Sigma M[@N/y] : D$$

by the cut rule (which is admissible in **N** by Theorem 2.2.1). This argument is summarized conveniently by the following proof figure:

$$\dfrac{\dfrac{\Gamma, y:B \vdash_\Sigma M : D \qquad \dfrac{\dfrac{@ : A \to B \in \Sigma \cup \Gamma}{\Gamma \vdash_\Sigma @ : A \to B} \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma @N : B}}{\Gamma \vdash_\Sigma M[@N/y] : D}}{} \quad (3.1)$$

Suppose the last rule of **G** applied is (Πl) 3.25:

$$\dfrac{\Gamma \vdash_\Sigma N : A \quad @ : \Pi x : A \,.\, B \in \Sigma \cup \Gamma \quad B[N/x] =_{\beta\eta} C \quad \Gamma, y : C \vdash_\Sigma M : D \quad y \notin \mathrm{FV}(D)}{\Gamma \vdash_\Sigma M[@N/y] : D}.$$

By the induction hypothesis we have that

$$\mathbf{N} \text{ proves } \Gamma \vdash_\Sigma N : A,$$

$$\mathbf{N} \text{ proves } \Gamma, y : C \vdash_\Sigma M : D,$$

$$\mathbf{N} \text{ proves } \Gamma \vdash_\Sigma @ : \Pi x : A \,.\, B(x)$$

and that $B[N/x] =_{\beta\eta} C$. From **N** proves $\Gamma \vdash_\Sigma @ : \Pi x : A \,.\, B(x)$ and **N** proves $\Gamma \vdash_\Sigma N : A$ we obtain

$$\mathbf{N} \text{ proves } \Gamma \vdash_\Sigma @N : B[N/x],$$

and so obtain

$$\mathbf{N} \text{ proves } \Gamma \vdash_\Sigma @N : C$$

by the equality rule for types. From this and **N** proves $\Gamma \vdash_\Sigma M : D$ we obtain

$$\mathbf{N} \text{ proves } \Gamma \vdash_\Sigma M[@N/y] : D$$

by the cut rule (which is admissible in **N** by Theorem 2.2.1). This argument is summarized conveniently by the following proof figure:

$$\dfrac{\Gamma, y{:}C \vdash_\Sigma M{:}D \qquad \dfrac{B[N/x] =_{\beta\eta} C \qquad \dfrac{\dfrac{@{:}\Pi x{:}A\,.\,B(x) \in \Sigma \cup \Gamma}{\Gamma \vdash_\Sigma @{:}\Pi x{:}A\,.\,B(x)} \quad \Gamma \vdash_\Sigma N{:}A}{\Gamma \vdash_\Sigma @N{:}B[N/x]}\,(3.5)}{\Gamma \vdash_\Sigma @N{:}C}\,(3.29)}{\Gamma \vdash_\Sigma M[@N/y] : D}\,(\text{cut})$$

Note that the equality rule for types requires the premiss $\Gamma \vdash_\Sigma C{:}\text{Type}$. This is readily obtained by induction on the structure of the proof in **N** of the assertion $\Gamma, y{:}C \vdash_\Sigma M{:}D$, and for clarity we omit this from the figure. This completes the proof. □

THEOREM 3.2.5 (COMPLETENESS OF **G**)

If **N** proves $\Gamma \vdash_\Sigma U{:}V$ then **G** proves $\Gamma \vdash_\Sigma U{:}V$.

PROOF

The proof proceeds by induction on the structure of proofs in the system **N**. The only differences between **N** and **G** are that the (→E) 3.1 and (ΠE) 3.5 rules of **N** are replaced in **G** by the (→ *l*) 3.24 and (Π*l*) 3.25 rules. Thus there are several simple cases in the induction, which correspond to the basic rules and to the (→I) and (ΠI) rules and which we omit; and two more difficult cases, which correspond to the (→E) and (ΠE) rules and which we include. Again, since the systems are only different for assertions that an object inhabits a type, we assume that $U{:}V$ is of the form $M{:}A$.

Suppose the last rule of **N** applied is the (→E) rule 3.25:

$$\dfrac{\Gamma \vdash_\Sigma M : A \to B \qquad \Gamma \vdash_\Sigma N{:}A}{\Gamma \vdash_\Sigma MN{:}B}\,.$$

By the induction hypothesis we have that

$$\mathbf{G} \text{ proves } \Gamma \vdash_\Sigma M : A \to B$$

and

$$\mathbf{G} \text{ proves } \Gamma \vdash_\Sigma N : A,$$

and from these two assertions we obtain, by induction on the structure of their proofs in **G**, that **G** proves $\Gamma \vdash_\Sigma A$: Type and **G** proves $\Gamma \vdash_\Sigma B$: Type, so that we obtain

$$\mathbf{G} \text{ proves } \Gamma, y : A \to B, x : A, z : B \vdash_\Sigma z : B$$

and

$$\mathbf{G} \text{ proves } \Gamma, y : A \to B, x : A \vdash_\Sigma x : A,$$

where without loss of generality we assume that y, $x \notin \mathrm{Dom}(\Gamma)$. Therefore by the (Πl) rule 3.25 we obtain

$$\mathbf{G} \text{ proves } \Gamma, y : A \to B, x : A \vdash_\Sigma yx : B.$$

From this assertion and the assertion that

$$\mathbf{G} \text{ proves } \Gamma \vdash_\Sigma N : A$$

we obtain

$$\mathbf{G} \text{ proves } \Gamma, y : A \to B \vdash_\Sigma yN : B$$

by the cut rule 3.28. From this assertion and the assertion that **G** proves $\Gamma \vdash_\Sigma M : A \to B$ we obtain

$$\mathbf{G} \text{ proves } \Gamma \vdash_\Sigma MN : B$$

by the cut rule 3.28.

Suppose the last rule of **N** applied is the (ΠE) rule 3.5:

$$\frac{\Gamma \vdash_\Sigma M : \Pi x{:}A\,.\,B \qquad \Gamma \vdash_\Sigma N{:}A}{\Gamma \vdash_\Sigma N{:}A}\,.$$

By the induction hypothesis we have that

$$\mathbf{G} \text{ proves } \Gamma \vdash_\Sigma M : \Pi x{:}A\,.\,B$$

and

$$\mathbf{G} \text{ proves } \Gamma \vdash_\Sigma N{:}A\,,$$

and from these two assertions we obtain, by induction on the structure of their proofs in **G**, that **G** proves $\Gamma \vdash_\Sigma A :$ Type and **G** proves $\Gamma \vdash_\Sigma B :$ Type, so that we obtain

$$\mathbf{G} \text{ proves } \Gamma, y : \Pi x{:}A\,.\,B, x{:}A, z{:}B(x) \vdash_\Sigma z{:}B(x)$$

and

$$\mathbf{G} \text{ proves } \Gamma, y : \Pi x{:}A\,.\,B, x{:}A \vdash_\Sigma x{:}A\,.$$

Therefore by the (Πl) rule 3.25 we obtain

$$\mathbf{G} \text{ proves } \Gamma, y : \Pi x{:}A\,.\,B, x{:}A \vdash_\Sigma yx{:}B(x)\,.$$

From this assertion and the assertion that **G** proves $\Gamma \vdash_\Sigma N{:}A$ we obtain

$$\mathbf{G} \text{ proves } \Gamma, y : \Pi x{:}A\,.\,B \vdash_\Sigma yN{:}B[N/x]$$

by the cut rule 3.28. From this assertion and the assertion that **G** proves $\Gamma \vdash_\Sigma M : \Pi x{:}A\,.\,B$ we obtain

$$\mathbf{G} \text{ proves } \Gamma \vdash_\Sigma MN{:}B[N/x]$$

by the cut rule 3.28. This completes the proof. □

3.2.3 Cut Elimination

In this section we prove the cut elimination theorem for the $\lambda\Pi$-calculus. Just as in other λ-calculi, this result obtains only for β-normal forms, see for example [ML71a], [ML71b] and [ML73] for the relevant normalization theorem. We write **G\cut** for the system obtained by removing the cut rule from the system **G**.

First we prove the soundness and completeness of **G\cut** with respect to **N**. The completeness part of this is the core of the cut elimination result.

THEOREM 3.2.6 (SOUNDNESS OF G\CUT)

If **G\cut** proves $\Gamma \vdash_\Sigma U:V$ then **N** proves $\Gamma \vdash_\Sigma U:V$.
PROOF

This is an immediate consequence of the soundness of **G** with respect to **N**. □

The next theorem is the completeness of the system **G\cut** with respect to the system **N**. The proof of this theorem depends crucially upon a simple technical device. This device consists in the replacement of the (ΠE) 3.5 rule of the system **N** by a weaker version of this rule which we shall call ($\Pi E^\heartsuit$):

$$(\Pi E^\heartsuit)\ \frac{\Gamma \vdash_\Sigma N:A \quad \Gamma \vdash_\Sigma M:\Pi x:A.B \quad \Gamma \vdash_\Sigma B[N/x]:\mathrm{Type}}{\Gamma \vdash_\Sigma MN:B[N/x]} \tag{3.31}$$

This rule is weaker than (ΠE) because it requires the extra premiss, however it is a relatively straightforward matter to prove that system $\mathbf{N}^\heartsuit$ which is obtained from **N** by the replacement of (ΠE) by ($\Pi E^\heartsuit$) is equivalent to **N**.

LEMMA 3.2.7 (EQUIVALENCE OF ($\Pi E^\heartsuit$) AND (ΠE))

$\mathbf{N}^\heartsuit$ proves $\Gamma \vdash_\Sigma M:A$ if and only if **N** proves $\Gamma \vdash_\Sigma M:A$.
PROOF

It follows immediately that if $\mathbf{N}^\heartsuit$ proves $\Gamma \vdash_\Sigma M:A$ then **N** proves $\Gamma \vdash_\Sigma M:A$.

Conversely, suppose that $\mathbf{N}$ proves $\Gamma \vdash_\Sigma M : A$: we must prove that $\mathbf{N}^\heartsuit$ proves $\Gamma \vdash_\Sigma M : A$. The proof is by induction on the structure of proofs in $\mathbf{N}$ and the only case of interest is when the last rule applied is (ΠE) 3.5:

$$(\Pi\mathrm{E})\frac{\Gamma \vdash_\Sigma N : A \quad \Gamma \vdash_\Sigma M : \Pi x{:}A\,.\,B}{\Gamma \vdash_\Sigma MN : B[N/x]}.$$

By the induction hypothesis we have that

$$\mathbf{N}^\heartsuit \text{ proves } \Gamma \vdash_\Sigma M : \Pi x{:}A\,.\,B, \tag{3.32}$$

and that

$$\mathbf{N}^\heartsuit \text{ proves } \Gamma \vdash_\Sigma N : A. \tag{3.33}$$

From 3.32 we obtain, by induction on the structure of proofs, that

$$\mathbf{N}^\heartsuit \text{ proves } \Gamma, x{:}A \vdash_\Sigma B : \mathrm{Type}. \tag{3.34}$$

It is easy to see that the cut rule is admissible in $\mathbf{N}^\heartsuit$[1], and so from 3.34 and 3.33 we obtain by the cut rule that

$$\mathbf{N}^\heartsuit \text{ proves } \Gamma \vdash_\Sigma B[N/x] : \mathrm{Type}. \tag{3.35}$$

We then obtain

$$\mathbf{N}^\heartsuit \text{ proves } \Gamma \vdash_\Sigma MN : B[N/x] \tag{3.36}$$

by the ($\Pi\mathrm{E}^\heartsuit$) rule. □

The proof of the completeness theorem is facilitated by the next lemma.

LEMMA 3.2.8 (APPLICATION IN β-NORMAL FORM)

If M is a β-normal form and if the last rule applied in the proof in the system $\mathbf{N}$ of $\Gamma \vdash_\Sigma M : A$ is either ($\to$E) or (ΠE), then M is of the form $@M_1 \ldots M_n$ for some $@ \in \mathrm{Dom}(\Sigma) \cup \mathrm{Dom}(\Gamma)$.

PROOF

---

[1]The proof is essentially the same as that for $\mathbf{N}$.

We illustrate a simple case of the proof. Suppose the last rule applied is the ($\rightarrow$E) rule:

$$\frac{\Gamma \vdash_\Sigma N : A \rightarrow B \quad \Gamma \vdash_\Sigma P : A}{\Gamma \vdash_\Sigma NP : B}\,.$$

If NP is a β-normal form, then N can not be of the form $\lambda x{:}A\,.\,Q$. Therefore N is of the form $@M_1 \ldots M_m$ for some @ of appropriate type. □

We are now ready to prove the completeness of the system **G\cut** with respect to the system **N**. However, the proof is rather complex, so first we give an example of the essential part of the argument. The proof proceeds by induction on the structure of proofs in **N**, although in fact we shall have to consider these to be proofs in $\mathbf{N}^\heartsuit$.

Suppose that the last inference (in **N**) is of the form

$$\frac{\Gamma \vdash_\Sigma @N_1N_2{:}B \rightarrow C \qquad \Gamma \vdash_\Sigma N{:}B}{\Gamma \vdash_\Sigma @N_1N_2N{:}C}\,,$$

where $@ : \Pi x_1{:}B_1\,.\,B_2 \rightarrow B^* \rightarrow C^* \in \Sigma \cup \Gamma$ and $(B^* \rightarrow C^*)[N_1/x_1] =_{\beta\eta} B \rightarrow C$.

By induction on the structure of proofs we have that

$$\textbf{G\cut} \text{ proves } \Gamma \vdash_\Sigma N_1 : B_1\,,$$

and, also by induction on the structure of proofs, we have that

$$\textbf{G\cut} \text{ proves } \Gamma \vdash_\Sigma N_2 : B_2[N_1/x_1]\,.$$

Again, by induction on the structure of proofs, we have that

$$\textbf{G\cut} \text{ proves } \Gamma \vdash_\Sigma B \rightarrow C : \text{Type}$$

and that

$$\textbf{G\cut} \text{ proves } \Gamma \vdash_\Sigma C : \text{Type}\,,$$

and by Lemma 3.2.7 we have that

$$\mathbf{G}\backslash\text{cut proves } \Gamma \vdash_\Sigma (B_2 \to (B^* \to C^*))[N_1/x_1] : \text{Type}.$$

Therefore we have that

$$\mathbf{G}\backslash\text{cut proves } \Gamma, x_1 : (B_2 \to (B^* \to C^*))[N_1/x_1], y : B \to C, z : C \vdash_\Sigma z : C\,,$$

where $y, z \notin \text{Dom}(\Gamma)$. By the induction hypothesis we have that

$$\mathbf{G}\backslash\text{cut proves } \Gamma \vdash_\Sigma N : B\,,$$

and so by the $(\to l)$ rule we obtain

$$\mathbf{G}\backslash\text{cut proves } \Gamma, x_1 : (B_2 \to (B^* \to C^*))[N_1/x_1], y : B \to C \vdash_\Sigma yN : C\,.$$

By the $(\to l)$ rule again, we obtain

$$\mathbf{G}\backslash\text{cut proves } \Gamma, x_1 : (B_2 \to (B^* \to C^*))[N_1/x_1], \vdash_\Sigma x_1 N_2 N : C\,,$$

and so by the (Πl) rule we obtain

$$\mathbf{G}\backslash\text{cut proves } \Gamma \vdash_\Sigma @N_1 N_2 N : C\,,$$

as required. This completes the example.

THEOREM 3.2.9 (COMPLETENESS OF G\CUT)

If **N** proves $\Gamma \vdash_\Sigma U : V$ and if U is a β-normal form then **G\cut** proves $\Gamma \vdash_\Sigma U : V$.

PROOF

The proof follows a pattern that is analogous to that employed by Prawitz [Pr65] in his proof of the completeness of a sequent calculus formulation of first-order logic (without the cut rule) with respect to a natural deduction formulation. The idea is to proceed by induction on the structure of proofs in **N** and to exploit the

structure of the β-normal form U.[2] The only differences between **N** and **G\cut** are that the ($\rightarrow$E) 3.1 and (ΠE) 3.5 rules of **N** are replaced in **G\cut** by the ($\rightarrow l$) 3.24 and (Πl) 3.25 rules. Thus there are several trivial cases in the induction, which we omit, and two rather difficult cases, which we include. Again, since the systems are different only for assertions that an object inhabits a type, we assume that $U:V$ is of the form $M:A$.

Before proceeding with the cases for the ($\rightarrow$E) and (ΠE) rules, it is convenient to introduce some notation for the type of an atom @. We denote the type of an atom @ by a string of the form

$$A_1 r_1 A_2 r_2 \ldots r_{n-1} A_n r_n B(x_1, \ldots, x_n),$$

in which each r_i, $1 \leq i \leq n$, denotes either a Π-connective, in which case the variable associated with it is x_i, or a $\rightarrow$-connective, with association to the right: *e.g.,* if r_1 denotes a Π, r_2 denotes an $\rightarrow$ and r_3 denotes an $\rightarrow$ then $A_1 r_1 A_2 r_2 A_3 r_3 B(x_1, x_2, x_3)$ denotes the type $\Pi x_1 : A_1 . (A_2 \rightarrow (A_3 \rightarrow B(x_1)))$.

Suppose the last rule of **N** applied is the ($\rightarrow$E) rule 3.1 (this is the first difficult case):

$$\frac{\Gamma \vdash_\Sigma M:B \rightarrow C \qquad \Gamma \vdash_\Sigma N:B}{\Gamma \vdash_\Sigma MN:C}.$$

We have by hypothesis that MN is a β-normal form, so that by Lemma 3.2.8 M must be of the form $@N_1 \ldots N_m$, where @ is in $\Sigma \cup \Gamma$. Thus we must prove that

$$\textbf{G\cut} \text{ proves } \Gamma \vdash_\Sigma @N_1 \ldots N_m N : C,$$

given that

$$\textbf{G\cut} \text{ proves } \Gamma \vdash_\Sigma @N_1 \ldots N_m : B \rightarrow C$$

and

---

[2]See also the remarks in §§1 - 5 of [How80].

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Gamma \vdash_\Sigma N{:}B\,.$$

This is proved by exploiting the structure of the type of @.

Using the notation introduced above, we represent the type of @ by the string:

$$B_1\, r_1\, B_2 r_2\, B_3\, r_3 \ldots r_{m-1}\, B_m\, r_m\, B^* \to C^*\,,$$

where $(B^* \to C^*)[N_1/x_1, \ldots, N_m/x_m] =_{\beta\eta} B \to C$. We assume that the variable associated with r_i, if it occurs, is x_i.

By Lemma 3.2.7 we may replace the system $\mathbf{N}$ by the system $\mathbf{N}^\heartsuit$, so given that

$$\mathbf{N}^\heartsuit \text{ proves } \Gamma \vdash_\Sigma @N_1 \ldots N_m : B \to C\,,$$

then by induction on the structure of proofs we have that, for $1 \leq i \leq n$,

$$\mathbf{N}^\heartsuit \text{ proves } \Gamma \vdash_\Sigma N_i : B_i[N_1/x_1, \ldots, N_{i-1}/x_{i-1}]\,,$$

by shorter proofs: therefore by the induction hypothesis we have that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Gamma \vdash_\Sigma N_i{:}B_i[N_1/x_1, \ldots, N_{i-1}/x_{i-1}],$$

and that the N_i are β-normal forms, for $1 \leq i \leq m$. From this we obtain that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Gamma \vdash_\Sigma B_i[N_1/x_1, \ldots, N_{i-1}/x_{i-1}] : \text{Type}\ .$$

From this it follows that we are able to construct a finite set of contexts which progressively extend Γ:

$$\begin{aligned}
\Delta_1 &\equiv_{\text{def}} \Gamma, x_1 : B_2[N_1/x_1]\, r_2 \ldots r_{m-1}\, B_m[N_1/x_1]\, r_m\, (B^* \to C^*)[N_1/x_1] \\
\Delta_2 &\equiv_{\text{def}} \Delta_1, x_2 : B_3[N_1/x_1, N_2/x_2]\, r_3 \ldots r_{m-1}\, B_m[N_1/x_1, N_2/x_2]\, r_m \\
&\qquad (B^* \to C^*)[N_1/x_1, N_2/x_2] \\
&\ \ \vdots \\
\Delta_m &\equiv_{\text{def}} \Delta_{m-1}, x_{m-1} : B_m[N_1/x_1, \ldots, N_{m-1}/x_{m-1}] r_m \\
&\qquad (B^* \to C^*)[N_1/x_1, \ldots, N_{m-1}/x_{m-1}].
\end{aligned}$$

It is easy to verify, for $1 \leq i \leq m$, that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \vdash_\Sigma \Delta_i \text{ context},$$

and we remark that no confusion can arise between the x_is used to construct the Δ_is and those that occur *bound* in the type of @.

By the induction hypothesis and the admissibility of weakening we have that $\mathbf{G}\backslash\text{cut}$ proves $\Delta_m \vdash_\Sigma N{:}B$. Since

$$\mathbf{N}^\heartsuit \text{ proves } \Gamma \vdash_\Sigma @N_1 \ldots N_m : B \to C,$$

we obtain, by the induction hypothesis, that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_m \vdash_\Sigma B \to C : \text{Type}$$

and that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_m \vdash_\Sigma C : \text{Type}.$$

Therefore we have that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_m, y : B \to C, z : C \vdash_\Sigma z : C,$$

where $y, z \notin \text{Dom}(\Gamma)$.

Therefore, by the $(\to l)$ rule we obtain

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_m, y : B \to C \vdash_\Sigma yN{:}C.$$

By the induction hypothesis we have that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_m \vdash_\Sigma N_m{:}B_m[N_1/x_1, \ldots, N_{m-1}/x_{m-1}]$$

and so by either the $(\to l)$ or the (Πl) rule we obtain

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_{m-1}, x_{m-1} : B_m[N_0/x_0, \ldots, N_{m-1}/x_{m-1}]\ r_m$$

$$(B^* \to C^*)[N_0/x_0, \ldots, N_{m-1}/x_{m-1}] \vdash_\Sigma x_{m-1}N_mN{:}C.$$

By the induction hypothesis we have that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_{m-1} \vdash_\Sigma N_{m-1} : B_{m-1}$$

and so by either the $(\to l)$ rule or by the (Πl) rule we obtain

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_{m-2}, x_{m-1} : B^*_{m-1}[N_0/x_0, \ldots, N_{m-2}/x_{m-2}]\, r_{m-1}$$

$$(B^* \to C^*)[N_0/x_0, \ldots, N_{m-2}/x_{m-2}] \vdash_\Sigma x_{m-1} N_{m-1} N_m N : C\,.$$

We proceed similarly until we obtain

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_0 \vdash_\Sigma x_0 N_1 \ldots N_m N : C\,.$$

By either the $(\to l)$ rule or by the (Πl) rule we obtain

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Gamma \vdash_\Sigma @N_0 \ldots N_m N : C\,,$$

as required.

Suppose the last rule of **N** applied is the $(\Pi\mathrm{E})$ rule 3.5 (this is the second difficult case):

$$\frac{\Gamma \vdash_\Sigma M : \Pi x : B\,.\, C \qquad \Gamma \vdash_\Sigma N : B}{\Gamma \vdash_\Sigma MN : C[N/x]}\,.$$

We have by hypothesis that MN is a β-normal form, so that by Lemma 3.2.8 M must be of the form $@N_1 \ldots N_m$, where $@$ is in $\Sigma \cup \Gamma$. Thus we must prove that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Gamma \vdash_\Sigma @N_1 \ldots N_m N : C[N/x]\,,$$

given that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Gamma \vdash_\Sigma @N_1 \ldots N_m : \Pi x : B\,.\, C$$

and

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Gamma \vdash_\Sigma N : B .$$

Again, this is proved by exploiting the structure of the type of @.

Using the notation introduced above, we represent the type of @ by the string:

$$B_1\, r_1\, B_2 r_2\, B_3\, r_3 \ldots r_{m-1}\, B_m\, r_m\, B^* \to C^* ,$$

where $(\Pi x : B^* . C^*)[N_1/x_1, \ldots, N_m/x_m] =_{\beta\eta} \Pi x : B . C$. We assume that the variable associated with r_i, if it occurs, is x_i.

By Lemma 3.2.7 we may replace the system $\mathbf{N}$ by the system $\mathbf{N}^\heartsuit$, so given that

$$\mathbf{N}^\heartsuit \text{ proves } \Gamma \vdash_\Sigma @N_1 \ldots N_m : \Pi x : B . C ,$$

then we have that, for $1 \leq i \leq n$,

$$\mathbf{N}^\heartsuit \text{ proves } \Gamma \vdash_\Sigma N_i : B_i[N_1/x_1, \ldots, N_{i-1}/x_{i-1}] ,$$

by shorter proofs: therefore by the induction hypothesis we have that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Gamma \vdash_\Sigma N_i : B_i[N_1/x_1, \ldots, N_{i-1}/x_{i-1}],$$

and that the N_i are β-normal forms, for $1 \leq i \leq m$. From this we obtain that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Gamma \vdash_\Sigma B_i[N_1/x_1, \ldots, N_{i-1}/x_{i-1}] : \text{Type} .$$

From this it follows that we are able to construct a finite set of contexts which progressively extend Γ:

$$\begin{aligned}
\Delta_1 &\equiv_{\text{def}} \Gamma, x_1 : B_2[N_1/x_1]\, r_2 \ldots r_{m-1}\, B_m[N_1/x_1]\, r_m\, (\Pi x : B^* . C^*)[N_1/x_1] \\
\Delta_2 &\equiv_{\text{def}} \Delta_1, x_2 : B_3[N_1/x_1, N_2/x_2]\, r_3 \ldots r_{m-1}\, B_m[N_1/x_1, N_2/x_2]\, r_m \\
&\qquad (\Pi x : B^* . C^*)[N_1/x_1, N_2/x_2] \\
&\;\;\vdots \\
\Delta_m &\equiv_{\text{def}} \Delta_{m-1}, x_{m-1} : B_m[N_1/x_1, \ldots, N_{m-1}/x_{m-1}] r_m \\
&\qquad (\Pi x : B^* . C^*)[N_1/x_1, \ldots, N_{m-1}/x_{m-1}].
\end{aligned}$$

It is easy to verify, for $1 \leq i \leq m$, that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \vdash_\Sigma \Delta_i \text{ context},$$

and we remark that no confusion can arise between the x_is used to construct the Δ_is and those that occur *bound* in the type of @.

By the induction hypothesis and the admissibility of weakening we have that $\mathbf{G}\backslash\text{cut}$ proves $\Delta_m \vdash_\Sigma N:B$. Since

$$\mathbf{N}^\heartsuit \text{ proves } \Gamma \vdash_\Sigma @N_1 \ldots N_m : \Pi x:B.C,$$

we obtain, by the induction hypothesis, that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_m \vdash_\Sigma \Pi x:B.C : \text{Type}$$

and, relying once again on the ($\Pi\text{E}^\heartsuit$) rule, that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_m \vdash_\Sigma C[N/x] : \text{Type}.$$

Therefore we have that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_m, y : \Pi x:B.C, z : C[N/x] \vdash_\Sigma z : C[N/x],$$

where $y, z \notin \text{Dom}(\Gamma)$. Therefore, by the ($\Pi l$) rule we obtain

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_m, y : \Pi x:B.C \vdash_\Sigma yN:C[N/x].$$

By the induction hypothesis we have that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_m \vdash_\Sigma N_m:B_m$$

and so by either the ($\rightarrow l$) or the (Πl) rule we obtain

$$\begin{array}{l}\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_{m-1}, x_m : B^*_m[N_0/x'_0, \ldots, N_{m-1}/x'_{m-1}]\, r_m \\ (\Pi x:B^*.C^*)[N_0/x'_0, \ldots, N_{m-1}/x'_{m-1}] \vdash_\Sigma x_m N_m N:C[N/x].\end{array}$$

By the induction hypothesis we have that

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_{m-1} \vdash_\Sigma N_{m-1} : B_{m-1}$$

and so by the either the $(\rightarrow l)$ rule or by the (Πl) rule we obtain

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_{m-2}, x_{m-1} : B^*_{m-1}[N_0/x'_0, \ldots, N_{m-2}/x'_{m-2}]r_{m-1}$$

$$(\Pi x{:}B^* \,.\, C^*)[N_0/x'_0, \ldots, N_{m-2}/x'_{m-2}] \vdash_\Sigma x_{m-1} N_{m-1} N_m N : C[N/x] \ .$$

We proceed similarly until we obtain

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Delta_0 \vdash_\Sigma x_0 N_1 \ldots N_m N : C[N/x] .$$

By either the $(\rightarrow l)$ rule or by the (Πl) rule we obtain

$$\mathbf{G}\backslash\text{cut} \text{ proves } \Gamma \vdash_\Sigma @N_0 \ldots N_m N : C[N/x] ,$$

as required. This completes the proof. □

Cut elimination is a simple corollary of this theorem.

COROLLARY 3.2.10 (CUT ELIMINATION)

Suppose that $\mathbf{G}$ proves $\Gamma \vdash_\Sigma M : A$. If M is a β-normal form then $\mathbf{G}\backslash\text{cut}$ proves $\Gamma \vdash_\Sigma M : A$.

PROOF

By Theorem 3.2.4, if $\mathbf{G}$ proves $\Gamma \vdash_\Sigma M : A$ then $\mathbf{N}^\heartsuit$ proves $\Gamma \vdash_\Sigma M : A$. By Theorem 3.2.9, if M is a β-normal form then $\mathbf{G}\backslash\text{cut}$ proves $\Gamma \vdash_\Sigma M : A$. □

An alternative way to obtain the result of 3.2.10 is to prove the completeness of $\mathbf{G}\backslash\text{cut}$ with respect to $\mathbf{G}$. The proof then centres on the need to eliminate instances of the cut rule 3.28, and is very similar in character to the proof given above, *cf.* [Ge34].

3.3 Summary

In this chapter we have reformulated the $\lambda\Pi$-calculus as a Gentzen-style system, **G**. We have proved the soundness and completeness **G** with respect to the natural deduction system **N** of Chapter 2. We have proved the cut elimination theorem for the $\lambda\Pi$-calculus by proving the completeness of the system **G**\cut with respect to the system **N** (for β-normal forms). This cut-free system will be of some considerable utility in subsequent chapters.

Chapter 4

Proof-search in the $\lambda\Pi$-calculus

4.1 Introduction

In this chapter we present a sequent calculus **L** of $\lambda\Pi$-calculus types. Sequents are assertions of the form $\Gamma \Rightarrow_\Sigma A$. This calculus is sound and complete (relative to the system **G**\cut) for the semidecidable relation of inhabitation: $\Gamma \Rightarrow_\Sigma A$, with the intended reading $(\exists M)(\Gamma \vdash_\Sigma M:A)$.[1] Note that we require that **G**\cut proves $\Gamma \vdash_\Sigma A$: Type. This restriction is motivated by the principal of judgements as types. For example, we might be interested in proving that $\Gamma \Rightarrow_\Sigma \text{true}(\phi)$ but not that "$\Gamma \Rightarrow_\Sigma o \rightarrow$ Type".

L is a metacalculus for the system **G**\cut in the sense that inhabitation judgements of **L** assert the *existence* of proofs of the type assignment judgements of **G**\cut.

L is the starting point for our investigation into proof-search for the $\lambda\Pi$-calculus.

A characteristic rule of **L** is the (Πl) rule which corresponds to the (Πl) rule of **G**, the counterpart of the Π-elimination rule of **N**:

$$(\Pi l)\quad \frac{\Gamma, x{:}D \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C}$$

(a) $@ : \Pi y{:}A.B \in \Sigma \cup \Gamma$

(b) $x \notin \text{FV(C)}$

(c) **G**\cut proves $\Gamma \vdash_\Sigma N{:}A$

(d) $B[N/y] \rightarrow_{\beta\eta} D$.

---

[1] In fact, we prove that M is the *extract-object* of the sequent $\Gamma \Rightarrow_\Sigma A$.

This rule should be read to assert that if there is some variable or some constant $@ : \Pi y : A . B \in \Sigma \cup \Gamma$ and if the inhabitation of the type C can be proved in the context $\Gamma, x : D$ and there is some object N which can be proved to have type A such that $B[N/y] \rightarrow_{\beta\eta} D$, then the inhabitation of the type C can be proved in the context Γ: that is, the assumption of the type D can be dispensed with provided it is possible to "calculate" D from the components of $\Sigma \cup \Gamma$.

L is almost a *logicistic* system, in the sense of Gentzen, meaning that there is only one localized appeal to an external notion, that notion being an appeal to **G**\cut in side condition (c) of the rule above. Indeed, with respect to the Π-type structure of terms in the language, **L** has a *subformula property* [Ge34]. As a consequence, if the inference rules are used as *reduction operators* from conclusion to premisses then **L** induces a search space of derivations of a given sequent.[2] Notice that if the (Πl) rule is used as a reduction, the choice of term N to use in the premiss is unconstrained by the conclusion of the rule. The sub*formula* property of the Π-types does not extend to a full sub*term* property (*cf.* the quantifier rules of the predicate calculus). The *axiom* sequent (or *closure* conditions for the reduction system) is:

$$\Gamma \Rightarrow_{\Sigma} A \qquad @ : A \in \Sigma \cup \Gamma$$

i.e., the conclusion occurs as the type of a declaration in the signature or the context.

L determines our basic search space, but we are able to constrain this space by two further developments:

1. We consider a notion of *uniform proof* in which the top-level connective of the succedent of a *goal* (a sequent for which we search for a proof) is always reduced as soon as possible. This notion of uniform proof corresponds to that introduced for hereditary Harrop formulae by Miller *et al.* in [MNPS89].

---

[2]Kleene [Kl68] explains this in the case of the predicate calculus. Sequent systems used in this way are systems of *block tableaux* [Sm68].

2. We consider two calculi, of strength equivalent to **L**, in which the left rules are replaced by forms of *resolution* rule.

Finally we demonstrate, by considering the example of first-order logic, how one of the resolution rules can be used to simulate the proofs of a logic which is encoded in the LF.

4.2 A Metacalculus for G\cut

4.2.1 Sequents and the Sequent Calculus L

We define a notion of *sequent* for the $\lambda\Pi$-calculus. The antecedent of a sequent is a $\lambda\Pi$-calculus context and the succedent is a $\lambda\Pi$-calculus type.

DEFINITION 4.2.1 (SEQUENT)

A *sequent* is a triple $\langle \Sigma, \Gamma, A \rangle$, written $\Gamma \Rightarrow_\Sigma A$, where Σ is a signature, Γ a context and A a type (family). □

The intended reading of such a sequent is "Γ inhabits A in the signature Σ".

DEFINITION 4.2.2 (WELL-FORMED SEQUENT)

A sequent $\Gamma \Rightarrow_\Sigma A$ (or $\langle \Sigma, \Gamma, A \rangle$) is said to be *well-formed* just in case
G\cut proves $\Gamma \vdash_\Sigma A$: Type. □

By Theorem 2.2.1 we have that the well-formedness problem for sequents is decidable.

We define a semi-logicistic calculus, **L**, for deriving well-formed sequents. The system is comprised of two *axiom schemata* and four *operational rules*, one left and one right for the each of the $\rightarrow$-types and Π-types.

We include no structural rules and indeed, provided we work with β-normal forms, the calculus is sound and complete with respect to **G**\cut in their absense. Appropriate forms of the structural rules of *weakening*, *permutation*, *contraction* and *cut* are admissible in **L**. We write $\mathrm{NF}(U)$ to denote the $\beta\eta$-normal form of the expression U. Note that if the rules are read as reduction operators then we must also require that any variables that extend a given context do not already occur in that context, *e.g.* in $(\rightarrow r)$ we must require that $x \notin \mathrm{Dom}(\Gamma)$.

DEFINITION 4.2.3 (THE CALCULUS **L**)

The rules of the sequent calculus **L** are given below:

$(Ax1)$ $\dfrac{}{\Gamma, x{:}A, \Gamma' \Rightarrow_\Sigma A}$ (**G**\cut proves $\vdash_\Sigma \Gamma, x{:}A, \Gamma'$ context)

$(Ax2)$ $\dfrac{}{\Gamma \Rightarrow_{\Sigma, c:A, \Sigma'} A}$ (**G**\cut proves $\vdash_{\Sigma, c:A, \Sigma'} \Gamma$ context)

$(\rightarrow r)$ $\dfrac{\Gamma, x{:}A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma A \rightarrow B}$ (a) $x \notin \mathrm{FV}(B)$

(Πr) $\dfrac{\Gamma, x{:}A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma \Pi x{:}A.B}$

$(\rightarrow l)$ $\dfrac{\Gamma \Rightarrow_\Sigma A \qquad \Gamma, z{:}B \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C}$ (a) $@{:}A \rightarrow B \in \Sigma \cup \Gamma$
(b) $z \notin \mathrm{FV}(C)$

(Πl) $\dfrac{\Gamma, x{:}D \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C}$ (a) $@{:}\Pi y{:}A.B \in \Sigma \cup \Gamma$
(b) $x \notin \mathrm{FV}(C)$
(c) **G**\cut proves $\Gamma \vdash_\Sigma N{:}A$
(d) $B[N/y] \rightarrow_{\beta\eta} D$.

For simplicity, we work exclusively with $\beta\eta$-normal forms, and for such terms syntactic identity is taken up to α-congruence (change of bound variable). Consequently, often we shall write just $B[N/y]$ rather than write D such that $B[N/y] \rightarrow_{\beta\eta} D$.

As usual we refer to the variable x of the $(\rightarrow r)$ and (Πr) rules as the *eigenvariable* of the inference. We can ensure that in any derivation eigenvariables occur only in sequents above the inference at which they are introduced. $A \rightarrow B$ and $\Pi x{:}A.B$ are said to be the *principal formulae* of the operational rules. A and B are the *side formulae* of the $(\rightarrow r)$ and (Πr) rules, A and $B[N/x]$ are the side formulae of the (Πl) rule and A and B are the side formulae of the $(\rightarrow l)$ rule. **L**-*derivations* are trees of sequents regulated by the operational rules such that the sequent at the root of the tree is well-formed, and **L**-*proofs* are **L**-derivations whose leaves are axioms. □

L is not fully logicistic since an appeal is made to **G**\cut for each application of the (Πl) rule (third side condition).

In practice, derivations are constructed from the root, or *endsequent*, toward the leaves, in the spirit of Kleene [Kl68] and systems of tableaux [Sm68]. However, **L** may be considered either as a forward calculus or as a calculus of reduction operators, and we shall adopt whichever view is appropriate at any given point in our development.

We have shifted our attention from a decidable judgement (type assignment) to a semi-decidable judgement (inhabitation) since the former is uninteresting from the point of view of general theorem proving. Indeed, in the LF the type which is the succedent of a well-formed sequent is considered to be an encoding of a judgement in some logic. Thus, by searching for a proof of such a sequent we are searching for a proof of an encoded judgement.

4.2.2 The Extract-object of an L-proof

Given a proof ψ of a sequent $\Gamma \Rightarrow_\Sigma A$ in the system **L**, an object M, in normal form, such that $\Gamma \vdash_\Sigma M : A$ is provable in the system **G**\cut, can be extracted from the proof in **L** in a simple inductive manner. We call this object the *extract-object* of the **L**-proof ψ of $\Gamma \Rightarrow_\Sigma A$. We write **L** proves$^\psi$ $\Gamma \Rightarrow_\Sigma A$ to denote that ψ is an **L**-proof of $\Gamma \Rightarrow_\Sigma A$. We shall speak of the *extract-object* of a sequent when the **L**-proof is clear from the context.

DEFINITION 4.2.4 (EXTRACT-OBJECT)

Suppose **L** proves$^\psi$ $\Gamma \Rightarrow_\Sigma A$. The *extract-object* of $\Gamma \Rightarrow_\Sigma A$ is defined by induction on the structure of its **L**-proof, ψ.

- If the sequent is an axiom, then either it is

$$\frac{}{\Gamma \Rightarrow_{\Sigma,c:A,\Sigma'} A} \text{ (G\cut proves } \vdash_{\Sigma,c:A,\Sigma'} \Gamma \text{ context)},$$

in which case the extract-object is c, or is of the form

$$\frac{}{\Gamma, x{:}A, \Gamma' \Rightarrow_\Sigma A}\ (\text{G\cut proves } \vdash_\Sigma \Gamma, x{:}A, \Gamma' \text{ context}),$$

in which case the extract-object is x.

- If the last rule of **L** applied is $(\rightarrow r)$

$$\frac{\Gamma, x{:}A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma A \rightarrow B}\ (x \notin \mathrm{FV}(B)),$$

and if the extract-object of the proof of the premiss is M, then the extract-object of the proof of the conclusion is $\lambda x{:}A.M$.

- If the last rule of **L** applied is (Πr)

$$\frac{\Gamma, x{:}A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma \Pi x{:}A.B},$$

and if the extract-object of the proof of the premiss is M, then the extract-object of the proof of the conclusion is $\lambda x{:}A.M$.

- If the last rule of **L** applied is $(\rightarrow l)$

$$\frac{\Gamma \Rightarrow_\Sigma B \qquad \Gamma, z{:}C \Rightarrow_\Sigma A}{\Gamma \Rightarrow_\Sigma A}\ (@ : B \rightarrow C \in \Sigma \cup \Gamma, x \notin \mathrm{FV}(A)),$$

and if the extract-objects of the proofs of the left and right premisses are respectively M and $N(z)$, then the extract-object of the proof of the conclusion is $N(@M)$.

- If the last rule of **L** applied is (Πl)

$$\frac{\Gamma, x{:}D \Rightarrow_\Sigma A}{\Gamma \Rightarrow_\Sigma A}(@ : \Pi y{:}B.C \in \Sigma \cup \Gamma, x \notin \mathrm{FV}(A), \Gamma \vdash_\Sigma M{:}B, C[M/y] \rightarrow_{\beta\eta} D),$$

and if the extract-object of the proof of the premiss is $N(z)$, then the extract-object of the proof of the conclusion is $N(@M)$.

□

4.2.3 Soundness and Completeness of L

We prove soundness and completeness results for **L** with respect to **G**\cut. In particular, the inhabiting objects that we obtain arise as extract-objects of the appropriate sequents in **L**.

PROPOSITION 4.2.5 (SOUNDNESS OF **L**)

If **L** proves $\Gamma \Rightarrow_\Sigma A$ then **G**\cut proves $\Gamma \vdash_\Sigma M : A$, where M is the extract-object of $\Gamma \Rightarrow_\Sigma A$.

PROOF

The proof proceeds by induction on the structure of the proof in **L** of $\Gamma \Rightarrow_\Sigma A$.

For the first part of the base case, suppose the last rule of **L** applied is the $(Ax1)$ rule:

$$\frac{}{\Gamma \Rightarrow_{\Sigma,c:A,\Sigma'} A}\ (\mathbf{G}\backslash\text{cut proves } \vdash_{\Sigma,c:A,\Sigma'} \Gamma \text{ context}).$$

The extract-object of the sequent is c, and we have immediately that **G**\cut proves $\Gamma \vdash_{\Sigma,c:A,\Sigma'} c : A$.

For the second part of the base case, suppose the last rule of **L** applied is the $(Ax2)$ rule:

$$\frac{}{\Gamma, x:A, \Gamma' \Rightarrow_\Sigma A}\ (\mathbf{G}\backslash\text{cut proves } \vdash_\Sigma \Gamma, x:A, \Gamma' \text{ context}).$$

The extract-object of the sequent is x, and we have immediately that **G**\cut proves $\Gamma, x:A, \Gamma' \vdash_\Sigma x : A$.

Suppose the last rule of **L** applied is the $(\rightarrow r)$ rule:

$$\frac{\Gamma, x:A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma A \rightarrow B}\ (x \notin \mathrm{FV}(B)).$$

Suppose that the extract-object of the premiss is M. By the induction hypothesis we have that **G**\cut proves $\Gamma, x:A \vdash_\Sigma M : B$. The extract-object object of the conclusion is $\lambda x:A.\, M$, and by the $(\rightarrow r)$ rule of **G**\cut we obtain **G**\cut proves $\Gamma \vdash_\Sigma \lambda x:A.\, M :\ A \rightarrow B$.

Suppose the last rule of **L** applied is the (Πr) rule:

$$\frac{\Gamma, x{:}A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma \Pi x{:}A.B}.$$

Suppose that the extract-object of the premiss is M. By the induction hypothesis we have that **G\cut** proves $\Gamma, x{:}A \vdash_\Sigma M{:}B$. The extract-object object of the conclusion is $\lambda x{:}A.M$, and by the (Πr) rule of **G\cut** we obtain **G\cut** proves $\Gamma \vdash_\Sigma \lambda x{:}A.M{:}\ \Pi x{:}A.B$.

Suppose the last rule applied is the $(\rightarrow l)$ rule:

$$\frac{\Gamma \Rightarrow_\Sigma F \quad \Gamma, x{:}G \Rightarrow_\Sigma E}{\Gamma \Rightarrow_\Sigma E}\ (@ : F \rightarrow G \in \Sigma \cup \Gamma\ ,\ x \notin \mathrm{FV}(E)).$$

Suppose that the extract-object of the left premiss is M and the extract-object of the right premiss is $N(x)$. By the induction hypothesis we have that **G\cut** proves $\Gamma \vdash_\Sigma M{:}F$ and **G\cut** proves $\Gamma, x{:}G \vdash_\Sigma N(x){:}E$. The extract-object of the conclusion is $N[@M/x]$, and by the $(\rightarrow l)$ rule of **G\cut** we obtain **G\cut** proves $\Gamma \vdash_\Sigma N[@M/x]{:}E$.

Suppose the last rule of **L** applied is the (Πl) rule:

$$\frac{\Gamma, x{:}D \Rightarrow_\Sigma E}{\Gamma \Rightarrow_\Sigma E}\ (@ : \Pi y{:}F.G(y) \in \Sigma \cup \Gamma\ ,\ x \notin \mathrm{FV}(E)\ ,\ \Gamma \vdash_\Sigma P : F\ ,\ G[P/y] \rightarrow_{\beta\eta} D).$$

Suppose that the extract-object of the premiss is $N(x)$. By the induction hypothesis we have that **G\cut** proves $\Gamma \vdash_\Sigma P{:}F$ and **G\cut** proves $\Gamma, x{:}H \vdash_\Sigma N(x){:}E$, where $G[P/y] \rightarrow_{\beta\eta} H$. The extract-object of the conclusion is $N[@P/x]$, and by the (Πl) rule of **G\cut** we obtain **G\cut** proves $\Gamma \vdash_\Sigma N[@P/x]{:}E$. This completes the proof. □

PROPOSITION 4.2.6 (COMPLETENESS OF **L**)

If **G\cut** proves $\Gamma \vdash_\Sigma M{:}A$, where M, A and each $@ : B \in \Sigma \cup \Gamma$ are $\beta\eta$-normal forms, then **L** proves $\Gamma \Rightarrow_\Sigma A$ where M is the extract-object of $\Gamma \Rightarrow_\Sigma A$.

PROOF

The proof proceeds by induction on the structure of the proof in **G\cut** of $\Gamma \vdash_\Sigma M{:}A$.

For the first part of the base case suppose that the last rule of G\cut applied is the basic rule:

$$\frac{\vdash_\Sigma \Gamma\text{context} \qquad c{:}A \in \Sigma}{\Gamma \vdash_\Sigma c{:}A}\,.$$

We note that $\Gamma \Rightarrow_\Sigma A$ is an axiom of **L** if $c{:}A \in \Sigma \cup \Gamma$ and that the extract-object of this sequent is c.

For the second part of the base case suppose that the last rule of G\cut applied is the basic rule:

$$\frac{\vdash_\Sigma \Gamma\text{context} \qquad x{:}A \in \Gamma}{\Gamma \vdash_\Sigma x{:}A}\,.$$

We note that $\Gamma \Rightarrow_\Sigma A$ is an axiom of **L** if $x{:}A \in \Sigma \cup \Gamma$, and that the extract-object of this sequent is x.

Suppose that the last rule of G\cut applied is the equality rule for types:

$$\frac{\Gamma \vdash_\Sigma M{:}A \qquad \Gamma \vdash_\Sigma A{:}\mathrm{Type} \qquad A =_{\beta\eta} A'}{\Gamma \vdash_\Sigma M{:}A'}\,.$$

By the induction hypothesis we have that **L** proves $\Gamma \Rightarrow_\Sigma A$, so that A is a $\beta\eta$-normal form. Since A' is a $\beta\eta$-normal form and $A =_{\beta\eta} A'$ we have that $A \equiv A'$, and so we have that **L** proves $\Gamma \Rightarrow_\Sigma A'$, and the extract-object of this sequent is M.

Suppose that the last rule of G\cut applied is the $(\to r)$ rule:

$$\frac{\Gamma \vdash_\Sigma A{:}\mathrm{Type} \qquad \Gamma, x{:}A \vdash_\Sigma M{:}B \qquad x \notin \mathrm{FV}(B)}{\Gamma \vdash_\Sigma \lambda x{:}A.M : A \to B}\,.$$

By the induction hypothesis we have that **L** proves $\Gamma, x{:}A \Rightarrow_\Sigma B$, with extract-object M. By the $(\to r)$ rule of **L** we obtain **L** proves $\Gamma \Rightarrow_\Sigma A \to B$, and the extract-object of this sequent is $\lambda x{:}A\,.\,M$.

Suppose that the last rule of G\cut applied is the (Πr) rule:

$$\frac{\Gamma \vdash_\Sigma A{:}\mathrm{Type} \qquad \Gamma, x{:}A \vdash_\Sigma M{:}B}{\Gamma \vdash_\Sigma \lambda x{:}A.M : \Pi x{:}A.B}\,.$$

By the induction hypothesis we have that **L** proves $\Gamma, x{:}A \Rightarrow_\Sigma B$, with extract-object M. By the (Πr) rule of **L** we obtain **L** proves $\Gamma \Rightarrow_\Sigma \Pi x{:}A\,.\,B$, and the extract-object of this sequent is $\lambda x{:}A\,.\,M$.

Suppose that the last rule of G\cut applied is the $(\rightarrow l)$ rule:

$$\frac{\Gamma \vdash_\Sigma N:A \quad @:A \rightarrow B \in \Sigma \cup \Gamma \quad \Gamma, y:B \vdash_\Sigma M:D \quad y \notin \mathrm{FV}(D)}{\Gamma \vdash_\Sigma M[@N/y]:D}.$$

By the induction hypothesis we have that **L** proves $\Gamma \Rightarrow_\Sigma A$ and **L** proves $\Gamma, y:B \Rightarrow_\Sigma D$, with extract-objects N and $M(y)$ respectively. By the $(\rightarrow l)$ rule of **L** we obtain **L** proves $\Gamma \Rightarrow_\Sigma D$, and the extract object of this sequent is $M[@N/y]$.

Suppose that the last rule of G\cut applied is the (Πl) rule:

$$\frac{\Gamma \vdash_\Sigma N:A \quad @:\Pi x:A.B \in \Sigma \cup \Gamma \quad B[N/x] =_{\beta\eta} C \quad \Gamma, y:C \vdash_\Sigma M:D \quad y \notin \mathrm{FV}(D)}{\Gamma \vdash_\Sigma M[@N/y]:D}.$$

By the induction hypothesis we have that **L** proves $\Gamma, y:C \Rightarrow_\Sigma D$, with extract-object $M(y)$ and that G\cut proves $\Gamma \Rightarrow_\Sigma N:A$. By the (Πl) rule of **L** we obtain **L** proves $\Gamma \Rightarrow_\Sigma D$, and the extract object of this sequent is $M[@N/y]$. This completes the proof. □

From Proposition 4.2.5, Proposition 4.2.6 and the theorems of Chapter 3, we obtain:

COROLLARY 4.2.7 (EQUIVALENCE OF **L** AND **N**)

For β-normal forms M, **N** proves $\Gamma \vdash_\Sigma M:A$ if and only if **L** proves $\Gamma \Rightarrow_\Sigma A$. □

We note the admissibility in **L** of the basic structural rules: permutation, weakening and contraction.

LEMMA 4.2.8 (ADMISSIBILITY OF PERMUTATION)

If **L** proves $\Gamma, x:A, y:B, \Delta \Rightarrow_\Sigma C$, and if $x \notin \mathrm{FV}(B)$ then **L** proves $\Gamma, y:B, x:A, \Delta \Rightarrow_\Sigma C$. PROOF

By induction on the structure of proofs. We omit the details. □

LEMMA 4.2.9 (ADMISSIBILITY OF WEAKENING)

If **L** proves $\Gamma \Rightarrow_\Sigma A$, and if $\vdash_\Sigma \Gamma, x{:}B$ context, then **L** proves $\Gamma, x{:}B \Rightarrow_\Sigma A$.
PROOF

By induction on the structure of proofs. We omit the details. □

LEMMA 4.2.10 (ADMISSIBILITY OF CONTRACTION)

If **L** proves $\Gamma, x{:}A, \Theta, y{:}A, \Delta \Rightarrow_\Sigma C$ then **L** proves $\Gamma, x{:}A, \Theta, \Delta[x/y] \Rightarrow_\Sigma C[x/y]$.
PROOF

By induction on the structure of proofs. We omit the details. □

4.2.4 Uniform Proofs

In this section we define a notion of uniform proof for **L**. This notion corresponds to that introduced by Miller *et al.* for a system of hereditary Harrop formulae in [MNPS89].

DEFINITION 4.2.11 (UNIFORM **L**-PROOFS)

An **L**-proof, with **L** considered as a system of reduction operators, is said to be *uniform* if the $(\rightarrow r)$ and (Πr) operators are applied wherever it is possible to do so. □

We prove that any sequent that has an **L**-proof has a uniform **L**-proof. Of course the $(\rightarrow r)$ and (Πr) operators are sometimes applicable to axiom sequents and indeed it is possible to construct uniform proofs of such axiom sequents. We remark that, in practise, it is not very sensible to construct a uniform proof of an axiom sequent.

Before proceeding with the next two propositions we recall some notation that we introduced for Theorem 3.2.9. We denote a type (of kind Type) by a string of the form

$$A_1 r_1 A_2 r_2 \ldots r_{n-1} A_n r_n B(x_1, \ldots, x_n),$$

in which each r_i, $1 \leq i \leq n$, denotes either a Π-connective, in which case the variable associated with it is x_i, or a $\rightarrow$-connective, with association to the right: *e.g.*, if r_1 denotes a Π, r_2 denotes an $\rightarrow$ and r_3 denotes an $\rightarrow$ then $A_1 r_1 A_2 r_2 A_3 r_3 B(x_1, x_2, x_3)$ denotes the type $\Pi x_1 : A_1 \,.\, (A_2 \rightarrow (A_3 \rightarrow B(x_1)))$.

PROPOSITION 4.2.12 (UNIFORM PROOFS OF AXIOMS)

If $\Gamma \Rightarrow_\Sigma A$ is a well-formed axiom sequent, then there is a uniform **L**-proof of $\Gamma \Rightarrow_\Sigma A$.

PROOF

If $\Gamma \Rightarrow_\Sigma A$ is an axiom sequent then there is some $@ : A \in \Sigma \cup \Gamma$. We write A in the form

$$A_1 r_1 A_2 r_2 \ldots r_{n-1} A_n r_n B(x_1, \ldots, x_n)\,,$$

in which each r_i, $1 \leq i \leq n$, denotes either a Π-connective, in which case the variable associated with it is x_i, or a $\rightarrow$-connective, with association to the right. Now it is easy to see that we must prove that the sequent

$$\Gamma, x_1 : A_1, \ldots, x_n : A_n \Rightarrow_\Sigma B(x_1, \ldots, x_n)\,, \tag{4.1}$$

in which we assume without loss of generality that $x_i \notin \mathrm{Dom}(\Gamma)$ for $1 \leq i \leq n$, has a uniform proof. We do this by showing that we can recover an axiom sequent from the sequent 4.1 by the use of $(\rightarrow l)$ and (Πl) operators only. The argument is rather complex and difficult to present, so we assist the reader by including a simple example. Suppose the axiom sequent in question is of the form

$$\Gamma, x : A \rightarrow (B \rightarrow C), \Delta \Rightarrow_\Sigma A \rightarrow (B \rightarrow C)\,.$$

In order to show that this sequent has a uniform proof we must show that the sequent

$$\Gamma, x : A \rightarrow (B \rightarrow C), \Delta, y : A, z : B \Rightarrow_\Sigma C\,,$$

where y and z are new variables, has a uniform proof. We construct such a proof as follows: by applying the $(\rightarrow l)$ operator to the sequent

$$\Gamma, x : A \rightarrow (B \rightarrow C), \Delta, y : A, z : B \Rightarrow_\Sigma C$$

we obtain the subgoals

$$\Gamma, x : A \to (B \to C), \Delta, y : A, z : B \Rightarrow_\Sigma A\,,$$

which is an axiom sequent, and

$$\Gamma, x : A \to (B \to C), \Delta, y : A, z : B, v : B \to C \Rightarrow_\Sigma C\,,$$

where v is a new variable. By applying the $(\to l)$ operator, using v, to this last sequent we obtain the subgoals

$$\Gamma, x : A \to (B \to C), \Delta, y : A, z : B, v : B \to C \Rightarrow_\Sigma B\,,$$

which is an axiom sequent, and

$$\Gamma, x : A \to (B \to C), \Delta, y : A, z : B, v : B \to C, w : C \Rightarrow_\Sigma C\,,$$

where w is a new variable, which is also an axiom sequent. This concludes the example.

We now proceed with the detailed argument. Let $\Delta \equiv \Gamma, x_1 : A_1, \ldots, x_n : A_n$. For each r_i we have to apply either the $(\to l)$ operator if r_i is an $\to$ or the (Πl) operator if r_i is a Π.

The first step concerns r_1.

If r_1 is a $\to$ then by applying the $(\to l)$ operator to the sequent

$$\Delta \Rightarrow_\Sigma B(x_1, \ldots, x_n)$$

we obtain the subgoals

$$\Delta \Rightarrow_\Sigma A_1\,,$$

which is an axiom sequent, and

$$\Delta, y_2 : A_2 r_2 \ldots r_{n-1} A_n r_n B(x_1, \ldots, x_n) \Rightarrow_\Sigma B(x_1, \ldots, x_n)\,, \tag{4.2}$$

where $y_2 \notin \mathrm{Dom}(\Delta)$.

The next step divides into two cases; we apply, using the atom y_2, either the $(\to l)$ operator, if r_2 is an $\to$, or the (Πl) operator, if r_2 is a Π.

If r_2 is a $\rightarrow$ then by applying the $(\rightarrow l)$ operator, using y_2, to the sequent

$$\Delta, y_2 : A_2 r_2 \ldots r_{n-1} A_n r_n B(x_1, \ldots, x_n) \Rightarrow_\Sigma B(x_1, \ldots, x_n)$$

we obtain the subgoals

$$\Delta, y_2 : A_2 r_2 \ldots r_{n-1} A_n r_n B(x_1, \ldots, x_n) \Rightarrow_\Sigma A_2 ,$$

which is an axiom sequent, and

$$\Delta, y_2 : A_2 r_2 \ldots r_{n-1} A_n r_n B(x_1, \ldots, x_n),$$

$$y_3 : A_3 r_3 \ldots r_{n-1} A_n r_n B(x_1, \ldots, x_n) \Rightarrow_\Sigma B(x_1, \ldots, x_n) ,$$

where $y_3 \notin \mathrm{Dom}(\Delta, y_2 : \ldots)$. If r_2 is a Π then by applying the (Πl) operator, using y_2, to the sequent

$$\Delta, y_2 : A_2 r_2 \ldots r_{n-1} A_n r_n B(x_1, \ldots, x_n) \Rightarrow_\Sigma B(x_1, \ldots, x_n)$$

we obtain the subgoal

$$\Delta, y_2 : A_2 r_2 \ldots r_{n-1} A_n r_n B(x_1, \ldots, x_n) ,$$

$$y_3 : A_3 r_3 \ldots r_{n-1} A_n r_n B(x_1, \ldots, x_n) \Rightarrow_\Sigma B(x_1, \ldots, x_n) ,$$

where $y_3 \notin \mathrm{Dom}(\Delta, y_2 : (\ldots))$.

If r_1 is a Π, we proceed similarly.

We proceed in a similar manner until we have performed either a $(\rightarrow l)$ operator or (Πl) operator for each r_i for $1 \leq i \leq n$. It is easy to see that the final step yields an axiom sequent of the form

$$\Delta, y_2 : (\ldots), \ldots, y_n : (\ldots), z : B(x_1, \ldots, x_n) \Rightarrow_\Sigma B(x_1, \ldots, x_n) .$$

This completes the proof. □

PROPOSITION 4.2.13 (COMPLETENESS OF UNIFORM **L**-PROOFS)

If $\Gamma \Rightarrow_\Sigma A$ has an **L**-proof, then $\Gamma \Rightarrow_\Sigma A$ has a uniform **L**-proof.

PROOF

The proof is by induction on the structure of **L**-proofs, with **L** considered as a system of reduction operators.

First the base. If $\Gamma \Rightarrow_\Sigma A$ is an axiom sequent, then the result follows by Proposition 4.2.12.

If the first operator applied in the **L**-proof of the sequent $\Gamma \Rightarrow_\Sigma A$ is either the $(\rightarrow r)$ operator or the (Πr) operator then the result is immediate.

Suppose that the first operator applied in the **L**-proof of the sequent $\Gamma \Rightarrow_\Sigma A$ is the $\rightarrow l$ operator:

$$\frac{\Gamma \Rightarrow_\Sigma D \qquad \Gamma, x{:}C \Rightarrow_\Sigma A}{\Gamma \Rightarrow_\Sigma A} \; (@ : D \rightarrow C \in \Sigma \cup \Gamma \,,\; x \notin \mathrm{FV}(A)) \,.$$

By writing A in the form

$$A_1 r_1 A_2 r_2 \ldots r_{n-1} A_n r_n B(x_1, \ldots, x_n) \,,$$

we see that it is sufficient to show that the sequent

$$\Gamma, x_1{:}A_1, \ldots, x_n{:}A_n \Rightarrow_\Sigma B(x_1, \ldots, x_n) \,,$$

in which $x_i \notin \mathrm{Dom}(\Gamma)$ for $1 \leq i \leq n$, has a uniform proof. By the induction hypothesis we have that $\Gamma \Rightarrow_\Sigma D$ and $\Gamma, x : C \Rightarrow_\Sigma A$ have uniform **L**-proofs. Therefore we have that

$$\Gamma, x{:}C, x_1{:}A_1, \ldots, x_n{:}A_n \Rightarrow_\Sigma B(x_1, \ldots, x_n)$$

has a **L**-uniform proof. Since $x_i \notin \mathrm{FV}(C)$ for $1 \leq i \leq n$, we obtain by Lemma 4.2.8 (admissibility of permutation) that

$$\Gamma, x_1{:}A_1, \ldots, x_n{:}A_n, x{:}C \Rightarrow_\Sigma B(x_1, \ldots, x_n)$$

has a uniform **L**-proof; but by applying the $(\rightarrow l)$ operator to the sequent

$$\Gamma, x_1{:}A_1, \ldots, x_n{:}A_n \Rightarrow_\Sigma B(x_1, \ldots, x_n) \,,$$

we obtain the sequent

$$\Gamma, x_1:A_1, \ldots, x_n:A_n, x:C \Rightarrow_\Sigma B(x_1, \ldots, x_n).$$

Therefore the sequent

$$\Gamma, x_1:A_1, \ldots, x_n:A_n \Rightarrow_\Sigma B(x_1, \ldots, x_n)$$

has a uniform **L**-proof.

Suppose that the first operator applied in the **L**-proof of the sequent $\Gamma \Rightarrow_\Sigma A$ is the Πl operator:

$$\frac{\Gamma, x:D \Rightarrow_\Sigma A}{\Gamma \Rightarrow_\Sigma A},$$

$(@ : \Pi z:B.C \in \Sigma \cup \Gamma, \mathbf{G}\backslash\text{cut proves } \Gamma \vdash_\Sigma N:B, D \rightarrow_{\beta\eta} C[N/z], x \notin \mathrm{FV}(A))$.

By writing A in the form

$$A_1 r_1 A_2 r_2 \ldots r_{n-1} A_n r_n B(x_1, \ldots, x_n),$$

we see that it sufficient to show that the sequent

$$\Gamma, x_1:A_1, \ldots, x_n:A_n \Rightarrow_\Sigma B(x_1, \ldots, x_n),$$

in which $x_i \notin \mathrm{Dom}(\Gamma)$ for $1 \leq i \leq n$, has a uniform proof. By the induction hypothesis we have that $\Gamma, x:D \Rightarrow_\Sigma A$ has a uniform **L**-proof. Therefore we have that

$$\Gamma, x:D, x_1:A_1, \ldots, x_n:A_n \Rightarrow_\Sigma B(x_1, \ldots, x_n)$$

has a **L**-uniform proof. Since $x_i \notin \mathrm{FV}(D)$ for $1 \leq i \leq n$, we obtain by Lemma 4.2.8 (admissibility of permutation) that

$$\Gamma, x_1:A_1, \ldots, x_n:A_n, x:D \Rightarrow_\Sigma B(x_1, \ldots, x_n)$$

has a uniform **L**-proof; but by applying the $(\rightarrow l)$ operator to the sequent

$$\Gamma, x_1:A_1, \ldots, x_n:A_n \Rightarrow_\Sigma B(x_1, \ldots, x_n),$$

we obtain the sequent

$$\Gamma, x_1:A_1, \ldots, x_n:A_n, x:D \Rightarrow_\Sigma B(x_1, \ldots, x_n).$$

Therefore the sequent

$$\Gamma, x_1 : A_1, \ldots, x_n : A_n \Rightarrow_\Sigma B(x_1, \ldots, x_n)$$

has a uniform **L**-proof. This completes the proof. □

We remark that the extract-object of a uniform **L**-proof is a *long $\beta\eta$-normal form.* To see this, consider the example of a uniform **L**-proof of the axiom sequent $x : A \to B \Rightarrow_\Sigma A \to B$:

$$\dfrac{\dfrac{x : A \to B, y : A \Rightarrow_\Sigma A \quad x : A \to B, y : A, z : B \Rightarrow_\Sigma B}{x : A \to B, y : A \Rightarrow_\Sigma B}}{x : A \to B \Rightarrow_\Sigma A \to B}.$$

It is easy to see that the extract-object of this proof is $\lambda y : A \,.\, xy$, which is the long $\beta\eta$-normal form of x.

4.2.5 The Choice of L for Proof-search

Having established the basic properties of **L**, and before proceeding to discuss clausal form and the resolution rules, we pause to consider the choice of **L** for proof search.

L is a sequent system with a limited subformula property. An alternative choice would have been to formulate a natural deduction system for inhabitation assertions which would have given us a Π-elimination rule of the form:

$$(\Pi\mathrm{E}) \quad \dfrac{\Gamma \Rightarrow_\Sigma \Pi x : A \,.\, B}{\Gamma \Rightarrow_\Sigma B[M/x]} \qquad \text{(a) } \mathbf{N} \text{ proves } \Gamma \vdash_\Sigma M : A$$

which is similar to the usual natural deduction rule for quantifiers, and a $\to$-elimination rule of the form:

$$(\to \mathrm{E}) \ \dfrac{\Gamma \Rightarrow_\Sigma A \quad \Gamma \Rightarrow_\Sigma A \to B}{\Gamma \Rightarrow_\Sigma B}$$

which is similar to the usual natural deduction rule of *modus ponens.* The fact that the type A in the premiss is not a subformula of the conclusion means that a proof procedure based on such a calculus would have to invent the type. The limited subformula property of **L** restricts the non-determinism to the choice of term M in the (Πl) rule.

4.3 Two Resolution Rules

4.3.1 Clausal Form

In this section we introduce a *clausal form* for both types and for sequents.

This form is of particular interest because if we restrict our attention to sequents in this clausal form we are able to obtain a completeness theorem, with respect to the calculus **L**, for two calculi in which the $(\rightarrow\ l)$ and $(\Pi\, l)$ rules are replaced by rules which resemble the *resolution* rule found in first-order logic theorem proving and logic programming, [Ro65], [Bu83], [Ll84]. This form is suggested by the form of Martin-Löf's *hypothetico-general judgements* [ML85] and also by the encoding of logical rules in the LF, *cf.* Felty [Fe87].

DEFINITION 4.3.1 (CLAUSAL FORM FOR TYPES)

A type (which is well-typed in some signature and context) is defined to be *in clausal form* if it is of the following recursively defined form:

$$\Pi x_1{:}B_1 \ldots \Pi x_m{:}B_m\,.\,C_1 \rightarrow (C_2 \rightarrow (\ldots (C_n \rightarrow D)\ldots))\,, \tag{4.3}$$

where D is atomic and each B_i $(1 \leq i \leq m)$ and each C_j $(1 \leq j \leq n)$ is in clausal form. □

We are now able to define clausal form for sequents.

DEFINITION 4.3.2 (CLAUSAL FORM FOR SEQUENTS)

Let $\Gamma \Rightarrow_\Sigma A$ be a well-formed sequent. The sequent is defined to be *in clausal form* if A is in clausal form and for each $@{:}B \in \Sigma \cup \Gamma$, B is in clausal form. □

We note that any type (which is well-typed in some signature and context) can be put into clausal form. To see this, consider the type

$$E \equiv \Pi x{:}A\,\Pi y{:}B\,.\,C_1 \rightarrow (\Pi z{:}D\,.\,C_2 \rightarrow D)\,.$$

It is easy to see that any proof which uses some $@ : E$ can be replaced by a proof which uses some $@' : E'$ instead, where

$$E' \equiv \Pi x{:}A\,\Pi y{:}B\,\Pi z{:}D\,.\,C_1 \to (C_2 \to D)\,,$$

since $z \notin \mathrm{FV}(C_1)$, and that any proof which uses $@' : E'$ can be replaced by a proof which uses $@ : E$ instead.

We note that all of the types declared in the signatures given in Appendix B are in clausal form.

Note that in the definition of clausal form for types and sequents we require not only the top-level structure to have a certain form, but also each subcomponent to have the same form. This is essential because the use of $(\to r)$ and (Πr) rules introduces these subcomponents to the context, thereby making them available for use with the resolution rules, which we define below.

4.3.2 The Calculus NR1

In this section we introduce the first of our two "resolution" rules. We suppose that all types and sequents are in clausal form.

Consider the rule:

$$\frac{\Gamma \vdash_\Sigma M_1{:}A_1 \ldots \Gamma \vdash_\Sigma M_m{:}A_m \quad \Gamma \vdash_\Sigma N_1{:}D_1 \ldots \Gamma \vdash_\Sigma N_n{:}D_n}{\Gamma \vdash_\Sigma @M_1 \ldots M_m N_1 \ldots N_n : E}, \tag{4.4}$$

where $@ : \Pi x_1 : E_1 \ldots \Pi x_m : E_m\,.\,B_1 \to (B_2 \to (\ldots \to (B_n \to C)\ldots)) \in \Sigma \cup \Gamma$, $A_i[M_1/x_1, \ldots, M_{i-1}/x_{i-1}] =_{\beta\eta} E_i$ for $1 \le i \le m$, $B_i[M_1/x_1, \ldots, M_m/x_m] =_{\beta\eta} D_i$ for $1 \le i \le n$ and $C[M_1/x_1, \ldots, M_m/x_m] =_{\beta\eta} E$.

We define the system **NR1** to be that system which is obtained from the system **N** by replacing the ($\to$E) 3.1 and (ΠE) 3.5 rules by the rule 4.4.

Actually, the rule 4.4 has the additional trivial premiss

$$\Gamma, x{:}E \vdash_\Sigma x{:}E\,,$$

where $x \notin \mathrm{Dom}(\Gamma)$, but for simplicity of presentation we omit to write this premiss.

THEOREM 4.3.3 (SOUNDNESS OF **NR1**)

If **NR1** proves $\Gamma \vdash_\Sigma M : A$ then **N** proves $\Gamma \vdash_\Sigma M : A$.

PROOF

A straightforward induction on the structure of proofs. We omit the details. □

THEOREM 4.3.4 (COMPLETENESS OF **NR1**)

Let M be a β-normal form. If **N** proves $\Gamma \vdash_\Sigma M : A$ then **NR1** proves $\Gamma \vdash_\Sigma M : A$.

PROOF

The proof proceeds by induction on the structure of proofs in **N**. The only interesting cases are where the last rule of **N** applied is either the ($\to$E) rule or the (ΠE) rule. We sketch the argument.

If the last rule of **N** applied is the ($\to$E) rule, and if the conclusion of this rule is a β-normal form, then the last inference in **N** is of the form

$$\frac{\Gamma \vdash_\Sigma @M_1 \ldots M_m N_1 \ldots N_n : A \to B \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma @M_1 \ldots M_m N_1 \ldots N_n N : B},$$

where $@ : \Pi x_1 : A_1 \ldots \Pi x_m : A_m \,.\, B_1 \to (B_2 \to (\ldots \to (B_n \to (A^* \to B^*))\ldots)) \in \Sigma \cup \Gamma$ and $(A^* \to B^*)[M_1/x_1, \ldots, M_m/x_m] =_{\beta\eta} A \to B$. By induction on the structure of proofs in **N** we have by the induction hypothesis that

$$\textbf{NR1} \text{ proves } \Gamma \vdash_\Sigma N : A,$$

and by induction on the structure of proofs that

$$\textbf{NR1} \text{ proves } \Gamma \vdash_\Sigma M_i : E_i$$

where $A_i[M_1/x_1, \ldots, M_{i-1}/x_{i-1}] =_{\beta\eta} E_i$ for $1 \leq i \leq m$, and that

$$\textbf{NR1} \text{ proves } \Gamma \vdash_\Sigma N_i : D_i$$

where $B_i[M_1/x_1, \ldots, M_m/x_m] =_{\beta\eta} D_i$ for $1 \leq i \leq n$. By an argument similar to that used in the proof of Theorem 3.2.9 we can prove that

$$\textbf{NR1} \text{ proves } \Gamma, x : B^*[M_1/x_1, \ldots, M_m/x_m] \vdash_\Sigma x : B^*[M_1/x_1, \ldots, M_m/x_m],$$

where as in the proof of Theorem 3.2.9 the difficulty is to prove that

$$\textbf{NR1} \text{ proves } \Gamma \vdash_\Sigma B^*[M_1/x_1, \ldots, M_m/x_m] : \text{Type}.$$

We now obtain

$$\textbf{NR1} \text{ proves } \Gamma \vdash_\Sigma @M_1 \ldots M_m N_1 \ldots N_n N : B$$

by rule 4.4.

The case in which the last rule of **N** applied is the (ΠE) is similar to that for the ($\rightarrow$E) rule. □

Consider the rule:

$$\frac{\Gamma \Rightarrow_\Sigma D_1 \ldots \Gamma \Rightarrow_\Sigma D_n}{\Gamma \Rightarrow_\Sigma E}, \tag{4.5}$$

where $@ : \Pi x_1 : A_1 \ldots \Pi x_m : A_m . B_1 \rightarrow (B_2 \rightarrow (\ldots \rightarrow (B_n \rightarrow C)\ldots)) \in \Sigma \cup \Gamma$, **N** proves $\Gamma \vdash_\Sigma M_i : E_i$ where $A_i[M_1/x_1, \ldots, M_{i-1}/x_{i-1}] \rightarrow_{\beta\eta} E_i$ for $1 \leq i \leq m$, $B_i[M_1/x_1, \ldots, M_m/x_m] \rightarrow_{\beta\eta} D_i$ for $1 \leq i \leq n$ and $C[M_1/x_1, \ldots, M_m/x_m] \rightarrow_{\beta\eta} E$.

Actually, the rule 4.5 has the additional trivial premiss

$$\Gamma, x{:}E \Rightarrow_\Sigma E,$$

where $x \notin \text{Dom}(\Gamma)$, but for simplicity of presentation we omit to write this premiss. The importance of this premiss is now rather clear: if $n = 0$ it is the *only* premiss that is obtained.

Let **LR1** be that calculus which is obtained from the calculus **L** by replacing the ($\rightarrow l$) and (Πl) rules by the rule 4.5. It is easy to see that if the extract-object of the conclusion of the rule 4.5 is defined to be $@M_1 \ldots M_m N_1 \ldots N_n$ where, for $1 \leq i \leq n$, N_i is the extract-object of the premiss $\Gamma \Rightarrow_\Sigma D_i$, then **LR1** is sound and complete with respect to **NR1** in the following sense:

PROPOSITION 4.3.5 (SOUNDNESS AND COMPLETENESS OF **LR1**)

LR1 proves$^\psi$ $\Gamma \Rightarrow_\Sigma A$ if and only if **NR1** proves $\Gamma \vdash_\Sigma M : A$, where M is the extract-object of ψ. □

It follows immediately that **LR1** is sound and complete with respect to **L**.

4.3.3 The Calculus NR2

In this section we introduce the second of our two "resolution" rules. We suppose that all types and sequents are in clausal form.

$$\frac{\Gamma \vdash_\Sigma M_1{:}E_1 \ldots \Gamma \vdash_\Sigma M_m{:}E_m \quad \Gamma \vdash_\Sigma N_1{:}D_1 \ldots \Gamma \vdash_\Sigma N_n{:}D_n \quad \Gamma, x{:}E \vdash_\Sigma P{:}F \quad x \notin \mathrm{FV}(F)}{\Gamma \vdash_\Sigma P[@M_1 \ldots M_m N_1 \ldots N_n/x] : F}, \tag{4.6}$$

where $@ : \Pi x_1 : A_1 \ldots \Pi x_m : A_m \,.\, B_1 \to (B_2 \to (\ldots \to (B_n \to C)\ldots)) \in \Sigma \cup \Gamma$, $A_i[M_1/x_1, \ldots, M_{i-1}/x_{i-1}] =_{\beta\eta} E_i$ for $1 \leq i \leq m$ $B_i[M_1/x_1, \ldots, M_m/x_m] =_{\beta\eta} D_i$ for $1 \leq i \leq n$ and $C[M_1/x_1, \ldots, M_m/x_m] =_{\beta\eta} E$.

We define the system **NR2** to be that which is obtained from the system N by replacing the (→E) 3.1 and (ΠE) 3.5 rules by the rule 4.6.

THEOREM 4.3.6 (SOUNDNESS OF **NR2**)

If **NR2** proves $\Gamma \vdash_\Sigma M{:}A$ then N proves $\Gamma \vdash_\Sigma M{:}A$.

PROOF

A straightforward induction on the structure of proofs. We omit the details. □

THEOREM 4.3.7 (COMPLETENESS OF **NR2**)

Let M be a β-normal form. If N proves $\Gamma \vdash_\Sigma M{:}A$ then **NR2** proves $\Gamma \vdash_\Sigma M{:}A$.

PROOF

The proof proceeds by induction on the structure of proofs in N. The only interesting cases are where the last rule of N applied is either the (→E) rule or the (ΠE) rule. We sketch the argument.

If the last rule of N applied is the (ΠE) rule, and if the conclusion of this rule is a β-normal form, then the last inference in N is of the form

$$\frac{\Gamma \vdash_\Sigma @M_1 \ldots M_m N_1 \ldots N_n : \Pi x{:}A \,.\, B \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma @M_1 \ldots M_m N_1 \ldots N_n N : B[N/x]},$$

where $@ : \Pi x_1 : A_1 \ldots \Pi x_m : A_m \,.\, B_1 \to (B_2 \to (\ldots \to (B_n \to (\Pi x : A^* \,.\, B^*))\ldots)) \in \Sigma \cup \Gamma$ and $(\Pi x : A^* \,.\, B^*)[M_1/x_1, \ldots, M_m/x_m] =_{\beta\eta} \Pi x : A \,.\, B$. By induction on the structure of proofs in **N** we have by the induction hypothesis that

$$\textbf{NR2} \text{ proves } \Gamma \vdash_\Sigma N : A\,,$$

and by induction on the structure of proofs that

$$\textbf{NR2} \text{ proves } \Gamma \vdash_\Sigma M_i : E_i$$

where $A_i[M_1/x_1, \ldots, M_{i-1}/x_{i-1}] =_{\beta\eta} E_i$ for $1 \leq i \leq m$, and that

$$\textbf{NR2} \text{ proves } \Gamma \vdash_\Sigma N_i : D_i$$

where $B_i[M_1/x_1, \ldots, M_m/x_m] =_{\beta\eta} D_i$ for $1 \leq i \leq n$. By an argument similar to that used in the proof of Theorem 3.2.9 we can prove that

$$\begin{aligned} \textbf{NR2} \text{ proves } \quad & \Gamma, x : B^*[M_1/x_1, \ldots, M_m/x_m, N/x] \\ & \vdash_\Sigma x : B^*[M_1/x_1, \ldots, M_m/x_m, N/x]\,, \end{aligned}$$

where as in the proof of Theorem 3.2.9 the difficulty is to prove that

$$\textbf{NR2} \text{ proves } \Gamma \vdash_\Sigma B^*[M_1/x_1, \ldots, M_m/x_m, N/x] : \text{Type}\,.$$

We now obtain

$$\textbf{NR2} \text{ proves } \Gamma \vdash_\Sigma @M_1 \ldots M_m N_1 \ldots N_n N : B$$

by rule 4.6.

The case in which the last rule of **N** applied is the ($\to$E) is similar to that for the (ΠE) rule. □

Consider the rule:

$$\frac{\Gamma \Rightarrow_\Sigma D_1 \ldots \Gamma \Rightarrow_\Sigma D_n \quad \Gamma, x : E \Rightarrow_\Sigma F}{\Gamma \Rightarrow_\Sigma F}, \tag{4.7}$$

where $@ : \Pi x_1 : A_1 \ldots \Pi x_m : A_m \,.\, B_1 \to (B_2 \to (\ldots \to (B_n \to C)\ldots)) \in \Sigma \cup \Gamma$, **G**\cut proves $\Gamma \vdash_\Sigma M_i : E_i$ where $A_i[M_1/x_1, \ldots, M_{i-1}/x_{i-1}] \to_{\beta\eta} E_i$ for $1 \leq i \leq m$, $B_i[M_1/x_1, \ldots, M_m/x_m] \to_{\beta\eta} D_i$ for $1 \leq i \leq n$ and $C[M_1/x_1, \ldots, M_m/x_m] \to_{\beta\eta} E$.

Let **LR2** be that calculus which is obtained from the calculus **L** by replacing the $(\rightarrow l)$ and (Πl) rules by the rule 4.7. It is easy to see that if the extract-object of the conclusion of the rule 4.7 is defined to be $P[@M_1 \ldots M_m N_1 \ldots N_n/x]$ where, for $1 \leq i \leq n$, N_i is the extract-object of the premiss $\Gamma \Rightarrow_\Sigma D_i$ and where P is the extract-object of the premiss $\Gamma, x{:}E \Rightarrow_\Sigma F$, then **LR2** is sound and complete with respect to **NR2** in the following sense:

PROPOSITION 4.3.8 (SOUNDNESS AND COMPLETENESS OF **LR2**)

LR2 proves$^{\psi}$ $\Gamma \Rightarrow_\Sigma A$ if and only if **NR2** proves $\Gamma \vdash_\Sigma M : A$, where M is the extract-object of ψ. □

It follows immediately that **LR2** is sound and complete with respect to **L**.

We note that we have the obvious notion of *uniform* **LR2**-proof and remark that the strategy of constructing uniform **LR2**-proofs is complete.

4.4 Object-logic Proofs

In this section we consider the relationship between the structure of proofs of the $\lambda\Pi$-calculus with a given signature Σ, as determined by the calculus **L** and the proofs of the logic which is encoded in the LF by Σ. A full treatment of such relationships is beyond the scope of this thesis, but we are able to demonstrate that for a certain quite general class of rules in logics with a sufficiently strong adequacy theorems the resolution calculus **LR1** is able to simulate, in a certain sense, proofs in the (unencoded) object-logic.

We consider two quite general classes of natural deduction rules, namely the introduction and elimination forms described by Martin-Löf in *Hauptsatz for the Intuitionistic Theory of Iterated Inductive Definitions* [ML71].

4.4.1 Introduction Rules

Following Martin-Löf [ML71], we take the introduction rule for the symbol # to be an inference of the form:

$$\frac{P_1(\overline{x},\overline{y}) \ldots P_p(\overline{x},\overline{y})}{\#(\overline{x})} \tag{4.8}$$

where $\overline{x}$ and $\overline{y}$ respectively denote $x_1,\ldots,x_m$ and $y_1,\ldots,y_n$. We assume that this rule is encoded in the LF, via suitable types A_i, $1 \leq i \leq p$, as a constant of the form

$$\#\mathrm{I} : A_1(\overline{x},\overline{y}) \vdash (A_2(\overline{x},\overline{y}) \vdash (\ldots \vdash (A_p(\overline{x},\overline{y}))\ldots))) \vdash_{\overline{x}:\overline{B},\overline{y}:\overline{C}(\overline{x})} J(\#(\overline{x}))^3$$

where J is the appropriate basic judgement, *eg.* true, valid, *etc.* and where we assume that the encoding is natural. In particular we assume that the variables of the object-logic are identified with those of the LF.

---

[3]We use the usual notational abbreviations, $\overline{x}:\overline{A}$ for $x_1 : A_1,\ldots,x_r : A_r$ for some r, *etc.*.

We ask the following question: when can we simulate an instance of such an object-logic inference step 4.8 by a corresponding instance of a single inference rule R of some system which is equivalent to **L** so that if the succedents of the sequents of the premisses of the instance of R are exactly the encoded versions of the premisses of the instance of the object-logic inference rule then the succedent of the sequent which is the conclusion of the instance of R is exactly the encoded version of the conclusion of the instance of the object-logic rule 4.8 ?

A candidate for such a rule is the resolution rule 4.5; for the instance of this rule that is generated by the constant #I is given by

$$\frac{\Gamma \Rightarrow_\Sigma \mathrm{NF}(A_1(\overline{x},\overline{y})[\overline{M}/\overline{x}]) \quad \ldots \quad \Gamma \Rightarrow_\Sigma \mathrm{NF}(A_p(\overline{x},\overline{y})[\overline{M}/\overline{x}])}{\Gamma \Rightarrow_\Sigma \mathrm{NF}(J(\#(\overline{x}))[\overline{M}/\overline{x}])},$$

where the objects $\overline{M}$ are those which select an instance which corresponds to the required instance of the object-logic rule.

Recall the LF signature Σ_{PA} of Chapter 2 which encodes the first-order theory of Peano arithmetic. By the adequacy theorem for the encoding of this theory 2.4.1, we have that

$$\frac{\psi_1 \ldots \psi_m}{\psi} \tag{4.9}$$

is an instance of an introduction rule of Peano arithmetic if and only if there is a $\beta\eta$-normal form M of type ψ° true in Σ_{PA} the free variables of which are all of type ι or ψ_i° true. Let Γ be a context in which M is well-typed. By applying the instance of the 4.5 rule generated by the constant which encodes the rule of which 4.9 is an instance (considered as a search rule or *reduction operator* [Kl68]) to the instance of the succedent of its conclusion which is determined by ψ°, we obtain the premisses

$$\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} \psi_1^\circ \text{ true} \ \ldots \ \Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} \psi_m^\circ \text{ true}.$$

By Proposition 4.3.5 we have that

$$\mathbf{LR1} \text{ proves } \Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} \psi_i^\circ \text{ true}$$

for each $1 \leq i \leq m$ if and only is there is a $\beta\eta$-normal form $M_i : \psi_i^\circ$ true which is well-typed in Γ. But then by the adequacy theorem (Theorem 2.4.1) of Chapter

2 we know that there are proofs of each ψ_i if and only if there are normal forms of type ψ_i true. Thus we have shown that the introduction rules of the first-order theory of Peano arithmetic can be simulated in the LF by the single rule 4.5.

In the analysis above the premisses of the encoded rule were assumed to be *ground*, as is the case for $\wedge$-introduction, $\vee$-introduction *etc.*, and not *hypothetical*, as is the case in $\supset$-introduction. Hypothetical premisses are encoded as hypothetico-general judgements so that in order to obtain the simulation required it is necessary to construct uniform proofs. For example, the $\supset$I rule of first-order logic is encoded in the LF as the constant

$$\supset \mathrm{I} : (\phi\,\mathrm{true} \vdash \psi\,\mathrm{true}) \vdash_{\phi:o,\psi:o} \phi \supset \psi\,\mathrm{true}\,,$$

(see Appendix B or [AHM87]). Suppose that we have a goal of the form

$$\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} a \supset b\,\mathrm{true}\,.$$

By applying the instance of the 4.5 rule

$$\frac{\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} a\,\mathrm{true} \rightarrow b\,\mathrm{true}}{\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} a \supset b\,\mathrm{true}}\,,$$

we obtain the goal:

$$\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} a\,\mathrm{true} \rightarrow b\,\mathrm{true}\,.$$

If we apply the $(\rightarrow r)$ rule to this goal, thereby putting it in uniform form, we obtain the premiss

$$\Gamma, x{:}a\,\mathrm{true} \Rightarrow_{\Sigma_{\mathrm{PA}}} b\,\mathrm{true}\,,$$

so that the composition of these two rules gives us a rule:

$$\frac{\Gamma, x{:}a\,\mathrm{true} \Rightarrow_{\Sigma_{\mathrm{PA}}} b\,\mathrm{true}}{\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} a \supset b\,\mathrm{true}}(x \notin \mathrm{Dom}(\Gamma))\,,$$

which corresponds exactly to the $\supset$I rule because under the principle of judgements as types the hypothetical proof which constitutes the premiss of the $\supset$I rule is encoded as a hypothetical judgement. In other words, if we are performing search directly in a natural deduction presentation of first-order logic and we are faced with a goal of the form $a \supset b$ and if we wish to use the $\supset$I rule:

$$\frac{\begin{array}{c}(\phi)\\ \psi\end{array}}{\phi \supset \psi}$$

then we add the assumption ϕ to our list of assumptions and proceed to search for a proof of ψ. We are able to simulate this situation by using the instance of the 4.5 rule generated by the constant $\supset$I and by maintaining the sequents of the **LR1**-proof in uniform form.

The $\forall$I rule of first-order logic is encoded by the constant

$$\forall \mathrm{I} : (\vdash_{x:\iota} P(x)\,\mathrm{true}) \vdash_{P:\iota\vdash o} \forall(P)\,\mathrm{true}.$$

By using the instance of the 4.5 rule generated by this constant and by maintaining the sequents of the **LR1**-proof in uniform form we obtain the composite rule

$$\frac{\Gamma, x:\iota \Rightarrow_{\Sigma_{\mathrm{PA}}} p(x)\,\mathrm{true}}{\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} \forall(p)\,\mathrm{true}}(x \notin \mathrm{Dom}(\Gamma))$$

which simulates, in the sense discussed above, the $\forall$I rule of first-order logic.

The $\exists$I rule of first-order logic is encoded by the constant

$$\exists \mathrm{I} : P(x)\,\mathrm{true} \vdash_{P:\iota\vdash o, x:\iota} \exists(P)\,\mathrm{true}\,.$$

By using the instance of the 4.5 rule generated by this constant and by maintaining the sequents of the **LR1**-proof in uniform form we obtain the composite rule

$$\frac{\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} p(M)\,\mathrm{true}}{\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} \exists(p)\,\mathrm{true}}$$

where $\Gamma \vdash_{\Sigma} M : \iota$, which simulates, in the sense discussed above, the $\exists$I rule of first-order logic.

4.4.2 Elimination Rules

Following Martin-Löf [ML71], we take the elimination rule for the symbol # to be

$$\frac{J(\#(\overline{x})) \quad P_1 \quad \ldots \quad P_m}{K(\overline{x}, \overline{w})} \tag{4.10}$$

where J is a suitable basic judgement, *eg.* true, valid, *etc.* and P_i, for $1 \leq i \leq m$, abbreviates the formula

$$\bigwedge_{\overline{y_i}:\overline{B_i}} (\overline{Q_i}(\overline{x}, \overline{w}, \overline{y_i}) \vdash K(\overline{x}, \overline{w}))$$

in which we use the usual notation for hypothetical and general judgements. This rule is encoded in the LF, again via suitable types, as a constant of the form

$$\#\,\mathrm{E} : \Pi\overline{x}{:}\overline{A}\;\Pi\overline{w}{:}\overline{D}(\overline{x})\,.\,J(\#(\overline{x})) \to (\Pi\overline{y_1}{:}\overline{B_1}(\overline{x},\overline{w})\,.\,\overline{C_1}(\overline{x},\overline{w},\overline{y_1}) \to K(\overline{x},\overline{w}))$$

$$\to \ldots \to (\Pi\overline{y_m}{:}\overline{B_m}(\overline{x},\overline{w})\,.\,\overline{C_m}(\overline{x},\overline{w},\overline{y_m}) \to K(\overline{x},\overline{w}))$$

where J and K are suitable basic judgements.[4] Again, we assume that the encoding is natural, and in particular that the variables of the object logic are identified with those of the $\lambda\Pi$-calculus.

Again, we ask the following question : when can we simulate an instance of such an object-logic inference step 4.10 by a corresponding instance of a single inference rule R of some system which is equivalent to **L** so that if the succedents of the sequents of the premisses of the instance of R are exactly the encoded versions of the premisses of the instance of the object-logic inference rule then the succedent of the sequent which is the conclusion of the instance of R is exactly the encoded version of the conclusion of the instance of the object-logic rule 4.10 ?

In this case the resolution rule 4.5 is such a rule provided we work with uniform **LR1**-proofs. Suppose that we are to search for a proof of the sequent

$$\Gamma \Rightarrow_\Sigma \mathrm{NF}(K(\overline{x},\overline{w})[\overline{M}/\overline{x}])$$

for some suitable $\overline{M}$. By applying the 4.5 rule, using $\#\,\mathrm{E}$, we obtain the subgoals

$$\Gamma \Rightarrow_\Sigma J(\#(\overline{x}))[\overline{M}/\overline{x}]$$

and

$$\Gamma \Rightarrow_\Sigma P_1[\overline{M}/\overline{x}]\,,\ldots,\,\Gamma \Rightarrow_\Sigma P_m[\overline{M}/\overline{x}]\,.$$

The premisses $\Gamma \Rightarrow_\Sigma P_i[\overline{M}/\overline{x}]$ do not yet correspond to the subgoals obtained in the object-logic. This situation is recovered if we take the uniform forms of the

---

[4]We use the usual notational abbreviations, $\overline{x}{:}\overline{A}$ for $x_1 : A_1,\ldots,x_r : A_r$ for some r, *etc.*.

premisses $\Gamma \Rightarrow_\Sigma P_i[\overline{M}/\overline{x}]$, and we know that if the original premisses are provable then so are their uniform forms.

For example, in the first-order theory of Peano arithmetic the $\exists$E rule is

$$\frac{\exists x.\phi(x) \quad \begin{matrix}(\phi)\\ \psi\end{matrix}}{\psi}$$

where x is not free in any assumption on which ψ depends. By the adequacy theorem for the encoding of this theory (Theorem 2.4.1) we have that there is a proof in Peano arithmetic of the conclusion ψ from the assumptions:

- a proof in Peano arithmetic of ψ from ϕ, and
- a proof in Peano arithmetic of $\exists x.\phi(x)$

if and only if there is a normal form M of type ψ true, the free variables of which are all of type ι or $\exists x.\phi(x)$ true or ϕ true $\rightarrow \psi$ true. Thus, by an argument similar to that for introduction rules, we see that the resolution rule 4.5, in conjunction with the strategy of maintaining proofs in uniform form, is able to simulate the elimination rules of the first-order theory of Peano arithmetic.

For example, the $\supset$E rule of first-order logic which is encoded in the LF as the constant

$$\supset\mathrm{E} : \phi \supset \psi\,\mathrm{true} \vdash ((\phi\,\mathrm{true} \vdash \psi\,\mathrm{true}) \vdash \chi\,\mathrm{true}) \vdash_{\phi:o,\psi:o,\chi:o} \chi\,\mathrm{true},$$

(see Appendix B or [AHM87]). Suppose that we have a goal of the form

$$\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} c\,\mathrm{true},$$

then by applying first the instance of the 4.5 rule generated by the constant $\supset$I and then applying $(\rightarrow r)$ rule, so that both subgoals are in uniform form

$$\frac{\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} a \supset b\,\mathrm{true} \qquad \dfrac{\Gamma, x{:}a\,\mathrm{true} \rightarrow b\,\mathrm{true} \Rightarrow_{\Sigma_{\mathrm{PA}}} c\,\mathrm{true}}{\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} (a\,\mathrm{true} \rightarrow b\,\mathrm{true}) \rightarrow c\,\mathrm{true}}}{\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} c\,\mathrm{true}}$$

we obtain the composite rule which yields the subgoals

$$\Gamma \Rightarrow_{\Sigma_{\mathrm{PA}}} a \supset b\,\mathrm{true}$$

and

$$\Gamma, x{:}a\,\mathrm{true} \rightarrow b\,\mathrm{true} \Rightarrow_{\Sigma_{\mathrm{PA}}} c\,\mathrm{true},$$

which, by an argument similar to that given for the $\supset$I rule, simulates the $\supset$E rule in first-order logic.

We note that these techniques can also handle more *ad hoc* formulations of elimination rules such as *modus ponens.*

We remark that from the point of view of proof-search the given natural deduction form for elimination rules is rather unfortunate. Frequently it happens that in such rules the conclusion is simply a propostion with no necesary structure, *e.g.* as in

$$\vee\mathrm{E}\ \frac{\phi \vee \psi \qquad \begin{matrix}(\phi)\\ \chi\end{matrix} \qquad \begin{matrix}(\psi)\\ \chi\end{matrix}}{\chi}\ .$$

The encoded version of such a rule is applicable, via the rule 4.5, whenever the succedent type of a given goal is of the form a true where $a : o$ and true : $o \rightarrow$ Type.

4.5 Summary

In this chapter we have considered a sequent calculus which forms the basis for proof-search in the $\lambda\Pi$-calculus. We have considered the notion of uniform proof and have considered two resolution calculi and proved them complete. We have begun to consider the relationship between the proofs of an object-logic and the proofs in the $\lambda\Pi$-calculus of the LF-encoded version of the logic.

Chapter 5

A Unification Algorithm for the $\lambda\Pi$-calculus

This work of this chapter was completed whilst the author was visiting the Center for Study of Language and Information (CSLI), Stanford University, during the Spring quarter of 1988.

5.1 Introduction

In this chapter we present a unification algorithm for the $\lambda\Pi$-calculus.

There are three important existing works on higher-order unification, namely Pietrzykowski's algorithm for the complete-enumeration of unifiers in second-order logic, Jensen and Pietrzykowski's fundamental paper on the extension of this work to an algorithm for the complete-enumeration of unifiers in ω-order logic (simply-typed λ-calculus) [JP76] and Huet's seminal work on searching for the existence of unifiers in simply-typed λ-calculus [Hu75]. The work of Huet [Hu75] forms the basis of this chapter.

In order to give a full account of effective search and logic programming in the setting of the $\lambda\Pi$-calculus, we need a unification algorithm for $\lambda\Pi$-calculus terms. We need a unification algorithm in order to eliminate the non-deterministic choice of object in the $(\Pi\, l)$ operator. This idea is explored in Chapter 6.

In the simply-typed λ-calculus we attempt to unify terms of the same type. However in the $\lambda\Pi$-calculus we must unify not only objects, but their types as well. We must adapt the approach of [Hu75] to allow this.

For example consider the pair of terms

$$\lambda x{:}o\,.\,y(x,u) : \Pi x{:}o\,.\,f(x,u) \tag{5.1}$$

where $f : o \to o \to \text{Type}$, and

$$\lambda x{:}o\,.\,z(x,v) : \Pi x{:}o\,.\,g(x,w)\,, \tag{5.2}$$

where $g : o \to o \to \text{Type}$. The types in 5.1 and 5.2 are unified if f is substituted for g and u is substituted for w. In general we shall have to unify the types of the Π-abstracted variables as well, but in this case they are both just the constant o. Applying this substitution to the objects, we are left with the problem of unifying

$$\lambda x{:}o\,.\,y(x,u) \quad \text{and} \quad \lambda x{:}o\,.\,z(x,v)\,,$$

which now have the same type. These are unified if u is substituted for v and y is substituted for z.

The differences between the work of this chapter and that of [Hu75] may be summarized as follows:

- The need to unify not only objects but their types as well forces us to devise a procedure which reduces the problem of unifying a pair of objects (and their types) to that of unifying a set of pairs of objects. A further complication at this point is that the components of such pairs are not of equal type; although their types are *similar* in a certain technical sense to be defined.

- Considerable extra difficulty in proving that "flexible-flexible" pairs are always unifiable;

- The need to keep track of the variables that have been introduced during by the algorithm: we want to know in what context our unified terms are well-typed. This need arises when unifying "flexible-flexible" pairs and in the procedure *match*;

- The need for the procedure *match* to introduce new disagreement pairs to the disagreement set so that certain types are unified;

- The need to maintain an ordering on disagreement sets so that the unifiability of flexible-flexible disagreement sets can be guaranteed;

- Our notion of substitution is categorical — this is the natural notion in this setting: fortunately, this does not affect the applicability of the methods of [Hu75], although a little more work is required.

Thus the work of this chapter is to show that the work of [Hu75] is applicable provided we take care in handling these extra difficulties.

We first provide the necessary syntax, notation and a suitable definition of substitutions. We then proceed to give a unification algorithm in the style of Huet [Hu75], which tests for the *existence* of unifiers, and returns a complete set of "initial segments" of the unifiers of a pair of $\lambda\Pi$-calculus terms; of course this algorithm is in general non-terminating — higher-order unification is semi-decidable even in the case of simple types.

Just as in [Hu75] we first consider an algorithm for the $\lambda\Pi$-calculus with just β-equality and then consider the simplifications that arise in the presence of $\beta\eta$-equality.

Finally, we consider how to obtain an algorithm based on that of Jensen and Pietrzykowski [JP76], for the *complete enumeration* of the set of unifiers of a pair of $\lambda\Pi$-calculus terms. It is of some theoretical importance that such a procedure exists.

Elliott [El89] has also studied unification algorithms for the $\lambda\Pi$-calculus, independently, and at the same time as us. His solution is similar to ours, although his presentation diverges from the format of [Hu75] rather more in that it exploits a notion of *unification problem.* Such unification problems are used to successively approximate a solution via *rigid-rigid transformations* and *flexible-rigid transformations.*

5.2 Syntax and Substitutions

5.2.1 Syntax

We consider the type theory of the $\lambda\Pi$-calculus as defined in Chapter 2, along with the various abbreviations defined therein. Henceforth, we shall assume that we have a fixed valid $\lambda\Pi$-signature Σ, and when we write expressions such as $\Gamma \vdash_\Sigma M : A$ *etc.*, we refer to such expressions derived in the system **N** (without explicit $\rightarrow$-types).

In addition to the syntax of Chapter 2, we allow f to range over variables (where the intention is that these will have, in general, functional types), and if the object M is well-typed in some given context Γ we let $\tau_{\tau\Gamma}(M)$ denote the type of M. The relation of α-equality between terms is denoted by $=$. Recall from Chapter 2 that we write $\mathrm{NF}(U)$ to denote the β-normal form of the expression U.

5.2.2 Substitutions

DEFINITION 5.2.1 (SUBSTITUTIONS)
Let the contexts Γ and Δ be given by $\Gamma \equiv x_1 : A_1\,,\ \ldots\ ,\ x_m : A_m(x_1, \ldots, x_{m-1})$ and $\Delta \equiv y_1 : B_1\,,\ \ldots\ ,\ y_n : B_n(y_1, \ldots, y_{n-1})$.

A *substitution* is a morphism[1] $\rho : \Gamma \longrightarrow \Delta$ in which we call Δ the *domain context* and Γ the *range context*, given by a finite tuple of *substitution pairs*

$$\rho \equiv \langle\langle M_1, y_1\rangle, \ldots, \langle M_n, y_n\rangle\rangle\,,$$

such that $\Gamma \vdash_\Sigma M_i : B_i[M_1/y_1, \ldots, M_{i-1}/y_{i-1}]$ for $1 \leq i \leq m$.[2] □

---

[1] *Cf.* Appendix A and [Ca86] for the definition of the category $C(\Sigma)$ with $\beta\eta$-equality; the definition with just β-equality is similar.

[2] Note that this is the same as the definition of substitution (realization) given in Appendix A apart from the fact that here we assume in the first instance only β-equality. We take tuples of *pairs* here because this is both standard in the presentation of unification and pedagogically helpful.

The definition of substitution taken here is slightly different from that taken in [Hu75]. The reason for this is that the presence of dependent types makes the definition taken in [Hu75] rather awkward because it is difficult to define the range context in a simple manner. The categorical definition is the natural one.

However, the requirement that we must unify types means that we shall need a slightly weaker notion of substitution than this. In order to define this weaker notion of substitution we shall need the notion of *similar types.*

DEFINITION 5.2.2 (SIMILAR TYPES)
Let the types A and B be given by the expressions:

$$A \equiv \Pi x_1 : A_1 \ldots \Pi x_{n_1} : A_{n_1} . c_1 M_1 \ldots M_{q_1} ,$$

$$B \equiv \Pi y_1 : B_1 \ldots \Pi y_{n_2} : B_{n_2} . c_2 N_1 \ldots N_{q_2} ,$$

where the kinds of c_1 and c_2 have respectively q_1 and q_2 Π-abstractions. The types A and B are said to be *similar*, written $A \sim B$, if $n_1 = n_2 (= n)$, $c_1 = c_2 (= c)$, $q_1 = q_2 (= q)$[3] and for $1 \leq i \leq n$, $A_i \sim B_i$. □

We now define the notions of *pobject*, *ptype* and *pcontext* and *psubstitution* by allowing the replacement of free variables in objects, types and contexts by objects of similar, not equal, type. Recall the translations τ and $|-|$ of Chapter 2, Definition 2.2.2, of $\lambda\Pi$-calculus-terms into Curry–typable terms.

DEFINITION 5.2.3 (PCONTEXTS)
We define the notion of *pcontext* inductively as follows:

- (Valid) contexts are pcontexts;
- If $\Gamma \equiv \Gamma_1, x : A, \Gamma_2$ is a pcontext, if N, C and the pcontext Γ_1, Θ, where $\mathrm{Dom}(\Theta) \cap \mathrm{Dom}(\Gamma_2) = \emptyset$, are such that

  $$\tau(\Gamma_1, \Theta) \vdash_{\tau(\Sigma), \{\pi_\sigma | \pi_\sigma : \omega \to (\sigma \to \omega) \to \omega\}} |N| : \tau(C)$$

  is provable in Curry's type assignment system (*cf.* Lemmas 2.2.3 and 2.2.4 of Chapter 2) and if $C \sim A$, then $\Gamma_1, \Theta, \Gamma_2[N/x]$ is a pcontext. □

---

[3]Note that we have required only the given syntactic shapes; we have not required that the expressions be well-typed of kind Type (in a given context).

DEFINITION 5.2.4 (PSUBSTITUTIONS)

We define the notion of *psubstitution* inductively as follows:

- A substitution $\sigma : \Gamma \longrightarrow \Delta$ is a psubstitution;
- Let $\sigma : \Gamma_1 , x : A , \Gamma_2 \longrightarrow \Delta$ be a psubstitution. If N, C and the pcontext Γ_1, Θ, where $\mathrm{Dom}(\Theta) \cap \mathrm{Dom}(\Gamma_2) = \emptyset$, are such that

$$\tau(\Gamma_1, \Theta) \vdash_{\tau(\Sigma),\{\pi_\sigma | \pi_\sigma : \omega \to (\sigma \to \omega) \to \omega\}} |N| : \tau(C)$$

  is provable in Curry's type assignment system and if $C \sim A$ then $\sigma[N/x] : \Gamma_1, \Theta, \Gamma_2[N/x] \longrightarrow \Delta$ is a psubstitution from the *domain pcontext* Δ to the *range pcontext* $\Gamma_1, \Theta, \Gamma_2[N/x]$. □

We shall refer to the components of psubstitutions and pcontexts respectively as *pobjects* and *ptypes* and given a pobject M we shall write $\tau_\tau(M)$ for a ptype A such that, for some pcontext Γ,

$$\tau(\Gamma) \vdash_{\tau(\Sigma),\{\pi_\sigma | \pi_\sigma : \omega \to (\sigma \to \omega) \to \omega\}} |M| : \tau(A)$$

is provable in Curry's type assignment system. Lemmas 2.2.3 and 2.2.4 of Chapter 2 will guarantee the existence of *head normal forms* for these entities: recall that a λ-term is said to be in head normal form if it is of the form

$$\lambda x_1 : A_1 \ldots x_m : A_m \,.\, @ M_1 \ldots M_n \,,$$

and note that this notion extends to the ptypes of the $\lambda\Pi$-calculus in the obvious way. We denote the head normal form of the expression U by $\mathrm{HNF}(U)$.

A psubstitution is said to be in head normal form if each of its components is in head normal form. A pcontext is said to be in head normal form if each of its components is in head normal form.

DEFINITION 5.2.5 (HEADINGS; RIGID AND FLEXIBLE POBJECTS)

Consider the pobject M in head normal form given by:

$$M \equiv \lambda x_1 : A_1 \ldots \lambda x_m : A_m . @ M_1 \ldots M_p \,.$$

We call the *heading* of the pobject M the expression $\lambda x_1 : A_1 \ldots \lambda x_m : A_m \,.\, @$, and call @ the *head* of M. If @ is a constant or one of the variables $x_1, \ldots, x_n$, then M is said to be *rigid*; otherwise M is said to be *flexible*. Let A be a ptype given by

the expression: $\Pi x_1 : A_1 \ldots \Pi x_n : A_n \,.\, cN_1 \ldots N_q$. We call the *heading* of the ptype A the expression $\Pi x_1 : A_1 \ldots \Pi x_n : A_n \,.\, c$, and call c the *head* of A and $N_1 \ldots N_q$ the *matrix* of A. □

The Church–Rosser and strong normalization properties fail for pobjects and ptypes, but the next lemma summarizes the properties that we need.

LEMMA 5.2.6 (HEAD NORMAL FORMS)

Every pobject (and every ptype) has a head normal form, and every head normal form of a given pobject has the same head.

PROOF SKETCH

The proof proceeds by translating the $\lambda\Pi$-calculus objects and types into Curry-typable terms via the translations τ and $|-|$ of Chapter 2, see Definition 2.2.2, and is similar to Lemmas 2.2.3 and 2.2.4. Reductions in the Curry-typed terms are mimicked in the $\lambda\Pi$-calculus-terms. See also [HHP87] and [HHP89]. □

DEFINITION 5.2.7 (THE MEASURE π)

As in [Hu75], let $\pi(M) \in \mathbb{N}$ denote the number of applications in the pobject M (in head normal form): $\pi(\lambda x_1 : A_1 \ldots \lambda x_n : A_n \,.\, @M_1 \ldots M_p) = p + \sum_{i=1}^{p} \pi(M_i)$. □

The application of the psubstitution $\rho(\equiv \langle\langle M_1, y_1\rangle, \ldots, \langle M_n, y_n\rangle\rangle) : \Gamma(\equiv x_1 : A_1, \ldots, x_m : A_m) \longrightarrow \Delta(\equiv y_1 : B_1, \ldots, y_n : B_n)$ to a pobject M and a ptype (family) A is given by, respectively, $M\rho =_{\text{def}} \text{HNF}((\lambda y_1 : B_1 \ldots \lambda y_n : B_n \,.\, M)M_1 \ldots M_n)$ and $A\rho =_{\text{def}} \text{HNF}((\lambda y_1 : B_1 \ldots \lambda y_n : B_n \,.\, A)M_1 \ldots M_n)$.

LEMMA 5.2.8 (APPLICATION OF SUBSTITUTIONS)

If $\rho : \Gamma \longrightarrow \Delta$ is a *substitution*, and if $\Delta \vdash_\Sigma M : A$, then $\Gamma \vdash_\Sigma M\rho : A\rho$.

PROOF

An easy exercise in the manipulation of proofs. □

DEFINITION 5.2.9 (EQUIVALENCE OF PSUBSTITUTIONS)

Let $\sigma : \Gamma \longrightarrow \Delta$ and $\rho : \Gamma \longrightarrow \Delta$ be psubstitutions. We write $\sigma =_\Delta \rho$ if $\text{HNF}(\sigma) = \text{HNF}(\rho)$, defined componentwise.[4] □

---

[4]Although the subscript is technically unnecessary in this setting, we leave it in place (*cf.* [Hu75]) because we believe it is of some pedagogic value in the presentation of many of the proofs of this chapter.

We need a lemma which says that rigid pobjects and ptypes keep their heading essentially unchanged under the application of substitutions.

LEMMA 5.2.10 (PROPERTIES OF PSUBSTITUTIONS)

1. A rigid pobject keeps its head and the number of abstractions in its heading unchanged under the application of any psubstitution.

2. A ptype keeps its head and the number of abstractions in its heading unchanged under the application of any psubstitution.

PROOF

Immediate from the definitions . □

DEFINITION 5.2.11 (COMPOSITION OF PSUBSTITUTIONS)

Suppose that the pcontexts Γ and Δ are respectively given by $\Gamma \equiv x_1 : A_1, \ldots, x_m : A_m$ and $\Delta \equiv y_1 : B_1, \ldots, y_n : B_n$, and that $\rho : \Gamma \longrightarrow \Delta$ is the psubstitution given by $\rho \equiv \langle\langle M_1, y_1\rangle, \ldots, \langle M_n, y_n\rangle\rangle$ and that $\sigma : \Delta \longrightarrow \Theta$ is the psubstitution given by $\sigma \equiv \langle\langle N_1, z_1\rangle, \ldots, \langle N_p, z_p\rangle\rangle$ where Θ is the pcontext $\Theta \equiv z_1 : C_1, \ldots, z_p : C_p(z_1, \ldots, z_{p-1})$. The composite psubstitution, $\rho \circ \sigma : \Gamma \longrightarrow \Theta$, is the psubstitution given by

$$\rho \circ \sigma \equiv \langle\langle N_1[M_1/y_1, \ldots, M_n/y_n], z_1\rangle, \ldots, \langle N_p[M_1/y_1, \ldots, M_n/y_n], z_p\rangle\rangle \,. \; \square$$

DEFINITION 5.2.12 (ORDERING PSUBSTITUTIONS)

Let $\sigma : \Gamma \longrightarrow \Delta$ and $\rho : \Gamma \longrightarrow \Theta$ be psubstitutions. We say that σ is *less general than* ρ on Γ, and write $\sigma \leq_\Gamma \rho$, if and only if there exists a psubstitution $\eta : \Theta \longrightarrow \Delta$ such that $\sigma =_\Gamma \rho \circ \eta$.[5] □

DEFINITION 5.2.13 (UNIFIERS)

Suppose that $\Delta \vdash_\Sigma M : A$ and $\Delta \vdash_\Sigma N : B$. The substitution $\rho : \Gamma \longrightarrow \Delta$ is a *unifier* of $M : A$ and $N : B$ if $A\rho = B\rho$ and $M\rho = N\rho$.

We let $\mathcal{U}_\Delta(M : A, N : B)$ denote the set of all unifiers of the terms $M : A$ and $N : B$. We omit the subscript Δ where no confusion can arise by doing so. □

---

[5] Although the subscript is technically unnecessary in this setting, we leave it in place (*cf.* [Hu75]) because we believe it is of some pedagogic value in the presentation of many of the proofs of this chapter.

5.3 An Existence Algorithm

5.3.1 Outline

Wherever it is possible we shall follow the structure, style and notation of Huet's presentation of the unification algorithm for the simply-typed λ-calculus, which is found in [Hu75]. The reason for this approach is that many of the proofs and arguments needed in our work are very similar to those needed in [Hu75] and it is therefore useful to be totally explicit about this so that our necessary additional work is clearly seen.

As in [Hu75], we are interested in the existence of unifiers for $M:A$ and $N:B$. To this end, we construct a tree, called a *matching tree*, for $M:A$ and $N:B$. We need the definitions below.

DEFINITION 5.3.1 (DISAGREEMENT SETS AND DISAGREEMENT PAIRS)

A *disagreement set* is any finite set of pairs of pobjects of similar ptype (called *disagreement pairs*),

$$\{\langle M_i, N_i\rangle | \tau_\tau(M_i) \sim \tau_\tau(N_i)\ ,\ 1 \leq i \leq n\}.$$

Pairs $\langle M, N\rangle$ in which both M and N are rigid will be referred to as *rigid-rigid* pairs, and those in which both M and N are flexible will be referred to as *flexible-flexible* pairs, *etc.*.

A *simplified disagreement set* is a disagreement set that contains only flexible-rigid pairs and flexible-flexible pairs and contains at least one flexible-rigid pair.

The psubstitution σ is said to be a unifier of the disagreement set $\mathcal{D}$ if for all pairs $\langle M, N\rangle \in \mathcal{D}$, $M\sigma = N\sigma$. We let $\mathcal{U}(\mathcal{D})$ denote the set of unifiers of the disagreement set $\mathcal{D}$. □

DEFINITION 5.3.2 (MATCHING TREES)

Matching trees are trees whose nodes and arcs are labelled as follows:

- Nodes are labelled either with a pair in which the first component is either a simplified disagreement set or **succeed** or **fail** and the second component of which is a pcontext;
- Arcs are labelled with psubstitutions as follows: if $\mathcal{N}$ and $\mathcal{N}'$ are nodes and if there is an arc from $\mathcal{N}$ to $\mathcal{N}'$ then the arc is labelled by a psubstitution $\sigma : \Gamma' \longrightarrow \Gamma$ where Γ is the pcontext that labels $\mathcal{N}$ and Γ' is the pcontext that labels $\mathcal{N}'$.

Nodes labelled with either **succeed** or **fail** are called *terminal* and those labelled with simplified disagreement sets are call *non-terminal.* □

DEFINITION 5.3.3 (CURRENT PCONTEXT)

Suppose that a node $\mathcal{N}$ of a matching tree $\mathcal{M}$ is labelled by a pair in which the second component is the pcontext Γ. We say that Γ is the *current pcontext* for the node $\mathcal{N}$. □

5.3.2 Construction of a Matching Tree

The construction of a matching tree $\mathcal{M}$ for two terms $M:A$ and $N:B$ (which are well-typed in some context Δ) is based on the three functions described below:

- *reduce*, which takes as arguments two ptypes (initially A and B) and returns either a disagreement set or **fail**;
- *simplify*, which takes as argument a disagreement set, and returns a *simplified disagreement set* or **succeed** or **fail**;
- *match*, which takes as arguments a flexible pobject M, a rigid pobject N of similar ptype, a pcontext Δ and a disagreement set $\mathcal{D}_{\mathcal{N}}$; it returns a finite set $\mathcal{S}$ of psubstitutions, a pcontext and a disagreement set.

The initial node in $\mathcal{M}$ is $simplify(reduce(A,B) \cup \{\langle M,N\rangle\})$. Let $\mathcal{N}$ be the current node in $\mathcal{M}$. If $\mathcal{N}$ is labelled with (a pair in which the first component is) **succeed** or **fail** then $\mathcal{N}$ has no successor; otherwise, choose a pair $\langle M',N'\rangle$ in the set which labels $\mathcal{N}$ such that N' is rigid, and let the set of psubstitutions $\mathcal{S} = match(M',N',\Delta,\mathcal{D}_{\mathcal{N}})$, with some abuse of notation. If $\mathcal{S} = \emptyset$, replace the label of $\mathcal{N}$ by **fail**; otherwise, for every $\sigma \in \mathcal{S}$, grow an arc, labelled with σ, from $\mathcal{N}$ to the new node $\mathcal{N}'$ labelled with

$$\mathcal{D}_{\mathcal{N}'} = simplify(\{\langle M'\sigma, N'\sigma\rangle | \langle M',N'\rangle \in \mathcal{D}_{\mathcal{N}}\} \cup reduce(H_1,H_2)),$$

where $\mathcal{D}_{\mathcal{N}}$ denotes the simplified disagreement set which labels the node $\mathcal{N}$ and where H_1 and H_2 are certain ptypes generated by *match*.

5.3.3 The Procedure *reduce*

The procedure *reduce* implements[6] the function *reduce* described in Section 5.3.2 which is used to reduce the problem of unifying two $\lambda\Pi$-calculus ptypes (in head normal form) to that of unifying a finite set of pairs of $\lambda\Pi$-calculus pobjects (where in each pair the pobjects have similar ptype). Thus, given a pair of ptypes A and B, *reduce* returns a disagreement set $\mathcal{D}$.

Let the head normal forms of $M:A$ and $N:B$ be given by

$$\lambda x_1:A_1 \ldots \lambda x_{n_1}:A_{n_1}.@_1 M_1 \ldots M_{p_1} : \Pi x_1:A_1 \ldots \Pi x_{n_1}:A_{n_1}.c_1 M'_1 \ldots M'_{r_1}$$

and

$$\lambda y_1:B_1 \ldots \lambda y_{n_2}:B_{n_2}.@_2 N_1 \ldots N_{p_2} : \Pi y_1:B_1 \ldots \Pi y_{n_2}:B_{n_2}.c_2 N'_1 \ldots N'_{r_2}$$

respectively, where the kinds of c_1 and c_2 are given by

$$\Pi w_1:D_1 \ldots \Pi w_{r_1}:D_{r_1}.\mathrm{Type}$$

and

$$\Pi z_1:E_1 \ldots \Pi z_{r_2}:E_{r_2}.\mathrm{Type}$$

---

[6] As an algorithm.

respectively.

The procedure *reduce* first checks that the ptypes are similar, that is that $n_1 = n_2$ and $c_1 = c_2$ (and so $r_1 = r_2$, since $c_1 M'_1 \ldots M'_{r_1}$ and $c_2 N'_1 \ldots N'_{r_2}$ must both be of kind Type). If they are not then there is no hope of unification, *e.g.*,

$$M : \Pi\phi{:}o\ \Pi\psi{:}o.\mathrm{true}(\chi) \qquad \text{and} \qquad N : \Pi\phi{:}o.\mathrm{valid}(\chi)$$

are not unifiable. It then proceeds, recursively, to reduce each pair of ptypes arising from the headings of A and B, taking care to maintain all variable bindings; it then adds to the (putative) disagreement set, $\mathcal{D}$, each pair of pobjects arising from the matrices of A and B, again taking care to maintain all variable bindings, *cf.* the example in Section 5.1.

reduce(A, B)

Step 1

If A and B are not similar ($A \not\sim B$) then return **fail**; otherwise let $c = c_1 (= c_2)$, $r = r_1 (= r_2)$ and $n = n_1 (= n_2)$.

Step 2

If $n = 0$ then perform Step 3; otherwise, for $i = 1$ to n make the assignment:

$$\mathcal{D} := \mathcal{D} \cup (\lambda x_1 : A_1 \ldots \lambda x_{i-1} : A_{i-1}, \lambda y_1 : B_1 \ldots \lambda y_{i-1} : B_{i-1})(\mathit{reduce}(A_i, B_i)),$$

where $(\lambda x_1 : A_1 \ldots \lambda x_n : A_n, \lambda y_1 : B_1 \ldots \lambda y_n : B_n)(\mathit{reduce}(A, B))$, *etc.* denotes that all of the disagreement pairs added to $\mathcal{D}$ as a result of the application of *reduce* to A and B must be λ-abstracted with respect to $x_1, \ldots, x_n$ in the first argument, and with respect to $y_1, \ldots, y_n$ in the second argument.

Step 3

For $j = 1$ to r, make the assignment:

$$\mathcal{D} := \mathcal{D} \cup \{\langle \lambda x_1 : A_1 \ldots \lambda x_n : A_n . M'_j, \lambda y_1 : B_1 \ldots \lambda y_n : B_n . N'_j \rangle\}.$$

Step 4

Return $\mathcal{D}$. □

For example, given the pair of objects and (similar) types

$$M : \Pi x{:}A\,.\,cP \quad \text{and} \quad N : \Pi y{:}B\,.\,cQ\,,$$

where c is a constant, *reduce* returns the set

$$\{\langle \lambda x{:}A\,.\,P, \lambda y{:}B\,.\,Q\rangle\} \cup reduce(A,B)\,.$$

PROPOSITION 5.3.4 (TERMINATION OF REDUCE)

The procedure *reduce* terminates, and returns a disagreement set, $\mathcal{D}$.
PROOF

A simple inductive argument. □

PROPOSITION 5.3.5 (FAITHFULNESS OF REDUCE)

If $\sigma : \Gamma \longrightarrow \Delta$ unifies $M{:}A$ and $N{:}B$ and if $\mathcal{D} = reduce(A,B) \cup \{\langle M, N\rangle\}$, then for all $\langle M', N'\rangle \in \mathcal{D}$, $M'\sigma = N'\sigma$.
PROOF

The proof is by an analysis of the structure of the set $\mathcal{D}$. If σ unifies $M : A$ and $N{:}B$ then we must have that $A\sigma = B\sigma$, so that we must have, for $1 \leq i \leq n$, that

$$A_i\sigma = B_i\sigma$$

and, for $1 \leq j \leq r$, that

$$\lambda x_1{:}A_1 \ldots \lambda x_n{:}A_n.M'_j\sigma = \lambda y_1{:}B_1 \ldots \lambda y_n{:}B_n.N'_j\sigma\,.$$

Now, each pair (A_i, B_i), on which reduce is called, has the same general form as the pair (A,B), so that if A_i and B_i are unified by σ, then each of the pairs of ptypes arising from the headings of A_i and B_i must be unified by σ, *et seq.*. This decomposition, along with the addition of the pairs of ptypes arising from the matrices of A and B, is exactly that performed by *reduce*, so that the constructed

disagreement set, $\mathcal{D}$, contains exactly those pairs which must be unifed by σ if the types A and B are unified by σ. Therefore $M'\sigma = N'\sigma$ for all $\langle M', N' \rangle \in \mathcal{D}$. □

Before proving the adequacy of *reduce*, we introduce a function r that assigns to each pair of pobjects in *reduce*(A, B) (provided this is not **fail**) a unique level, as follows:

DEFINITION 5.3.6 (THE FUNCTION r)

Suppose that A and B are ptypes, with their head normal forms given respectively by $A \equiv \Pi x_1 : A_1 \ldots \Pi x_m : A_m \,.\, cM_1 \ldots M_n$ and $B \equiv \Pi y_1 : B_1 \ldots \Pi y_m : B_m \,.\, cN_1 \ldots N_n$. Suppose further that $A_1 \equiv \Pi x_{11} : A_{11} \ldots \Pi x_{1m_1} : A_{1m_1} \,.\, cM_{11} \ldots M_{1n_1}$ and that $B_1 \equiv \Pi y_{11} : B_{11} \ldots \Pi y_{1m_1} : B_{1m_1} \,.\, cN_{11} \ldots N_{1n_1}$, and that, similarly

$$A_{ij} \equiv \Pi x_{ij1} : A_{ij1} \ldots \Pi x_{ijm_{ij}} : A_{ijm_{ij}} \,.\, cM_{ij1} \ldots M_{ijn_{ij}}$$

and

$$B_{ij} \equiv \Pi y_{ij1} : B_{ij1} \ldots \Pi y_{ijm_{ij}} : B_{ijm_{ij}} \,.\, cN_{ij1} \ldots N_{ijn_{ij}} \,,$$

etc.. The function r assigns the level $0k$ to each of the pairs $\langle M_k, N_k \rangle$ $(1 \leq k \leq n)$; r assigns the level $1k$ to each of the pairs $\langle M_{1k}, N_{1k} \rangle$ $(1 \leq k \leq n_1)$; r assigns the level ijk to each of the pairs $\langle M_{ijk}, N_{ijk} \rangle$ $(1 \leq k \leq n_1)$, *etc.*. □

PROPOSITION 5.3.7 (ADEQUACY OF REDUCE)

Let $\Delta \vdash_\Sigma M : A$ and $\Delta \vdash_\Sigma N : B$.

1. Let $\sigma : \Gamma \longrightarrow \Delta$ be a substitution and let $\mathcal{D} = \textit{reduce}(A, B) \cup \{\langle M, N \rangle\}$. If for all $\langle M', N' \rangle \in \mathcal{D}$, $M'\sigma = N'\sigma$, and if $M\sigma = N\sigma$, then σ unifies $M : A$ and $N : B$.

2. If *reduce* returns **fail**, then $\mathcal{U}_\Delta(M : A, N : B) = \emptyset$.

PROOF

1. We observe that the levels assigned to the elements of $\mathcal{D}$ by the function r can be ordered $(\sqsupset_r)$ as follows:

$$11\ldots111 \sqsupset_r 11\ldots112 \sqsupset_r \ldots \sqsupset_r 11\ldots11 \sqsupset_r 11\ldots12 \sqsupset_r \ldots \sqsupset_r$$

$$11 \sqsupset_r 12 \sqsupset_r \ldots \sqsupset_r 01 \sqsupset_r 02 \sqsupset_r \ldots \sqsupset_r 0n \sqsupset_r 0\,,$$

with the pair $\langle M, N\rangle$ assigned the level 0.

For example, suppose that $A \equiv \Pi x_1 : A_1 \,.\, cM_1$ and $B \equiv \Pi y_1 : B_1 \,.\, cN_1$, where $A_1 \equiv \Pi x_{11} : A_{11} \,.\, dM_{11}$ and $B_1 \equiv \Pi y_{11} : B_{11} \,.\, dN_{11}$, $A_{11} \equiv eM_{111}$ and $B_{11} \equiv eN_{111}$, then the ordering can be considered to be

$$\langle M_{111}, N_{111}\rangle \sqsupset_r \langle M_{11}, N_{11}\rangle \sqsupset_r \langle M_1, N_1\rangle\,.$$

We can consider this order to be a finite segment of the natural numbers with their usual ordering, the highest number being labelled by $11\ldots111$ and the lowest being labelled by 0.

The result now follows by a simple inductive argument on the structure of A and B.

If the pairs at all levels down to and including level $j+1$ have been unified, then the components of the pair at level j have unified ptypes, but this pair is also unified, *etc.*.

2. The procedure *reduce*(A, B), for some pair (A, B) occurring in the pair of ptypes to be reduced, returns **fail** just in case A and B are not similar, so that either $n_1 \neq n_2$ or $c_1 \neq c_2$, and so by Lemma 5.2.10 $M : A$ and $N : B$ are not unifiable. □

Henceforth we shall consider $\sqsupset_r$ to be an order on disagreement pairs and drop the subscript r, so that we write $\langle M', N'\rangle \sqsupset \langle M'', N''\rangle$, *etc.*. Also, we shall write $\langle M', N'\rangle \sqsupset \langle M'', N''\rangle$ for $\langle M', N'\rangle \sqsupset \ldots \sqsupset \langle M'', N''\rangle$, *etc.*.

5.3.4 The Procedure *simplify*

The purpose of the procedure *simplify* is to remove from the input disagreement set any rigid-rigid pairs, and to check for nodes containing only flexible-flexible pairs. Let $\mathcal{D}$ be the disagreement set input to *simplify*.

simplify($\mathcal{D}$)

Step 1

If there does not exist a pair $\langle M, N\rangle \in \mathcal{D}$ such that M and N are both rigid then perform Step 2; otherwise write

$$\mathcal{D} = \{\langle M, N\rangle\} \uplus \mathcal{D}'$$

where

$$M = \lambda x_1 : A_1 \ldots \lambda x_{n_1} : A_{n_1} . @_1 M_1 \ldots M_{p_1}$$

and

$$N = \lambda y_1 : B_1 \ldots \lambda y_{n_2} : B_{n_2} . @_2 N_1 \ldots N_{p_2}.$$

We now check that the heading of M is equal to the heading of N:

if $@_1 \neq @_2$, then return **fail**, otherwise let $p = p_1 (= p_2)$, (note that M and N have similar types so that if $@_1 = @_2$ then we must have that $n_1 = n_2$ $(= n)$ and $p_1 = p_2$ $(= p)$), and make the assignment:

$$\mathcal{D} := \{\langle \tilde{M}_i, \tilde{N}_i\rangle | 1 \leq i \leq p\} \uplus \mathcal{D}'$$

where $\tilde{M}_i = \lambda x_1 : A_1 \ldots \lambda x_n : A_n . M_i$ and $\tilde{N}_i = \lambda y_1 : B_1 \ldots \lambda y_n : B_n . N_i$. Extend the order $\sqsupset$ to $\mathcal{D}$ by making the assignment

$$\sqsupset := \sqsupset \cup \{ \langle \tilde{M}_i, \tilde{N}_i\rangle \sqsupset \langle \tilde{M}_j, \tilde{N}_j\rangle \mid 1 \leq i < j \leq p \}.$$

Repeat Step 1.

Step 2

Replace in $\mathcal{D}$ each pair $\langle M, N\rangle$ where M is rigid and N is flexible by the pair $\langle N, M\rangle$.

Step 3 If there exists $\langle M, N\rangle \in \mathcal{D}$ such that N is rigid, then return the set $\mathcal{D}$; otherwise return **succeed**. □

DEFINITION 5.3.8 (THE MEASURE δ)

Let $\mathcal{D} = \{\langle M_i, N_i\rangle | 1 \leq i \leq n\}$, and define the measure $\delta(\mathcal{D}) \in \mathbb{N}$ by

$$\delta(\mathcal{D}) = n + \sum_{i=1}^{n} \pi(M_i)$$

where π is the measure given in Definition 5.2.7. □

Let $\mathcal{D}$ be a disagreement set that is input to *simplify*. We show that *simplify* terminates after a finite number of steps, and returns a simplified disagreement set possessing the properties described by the three results given below.

PROPOSITION 5.3.9 (CORRECTNESS OF AND TERMINATION OF SIMPLIFY)
Let $\mathcal{D}$ be a disagreement set.

1. If $simplify(\mathcal{D}) =$ **fail**, then $\mathcal{U}(\mathcal{D}) = \emptyset$.

2. If $simplify(\mathcal{D}) = \mathcal{D}'$, where $\mathcal{D}'$ is a disagreement set, then $\mathcal{U}(\mathcal{D}) = \mathcal{U}(\mathcal{D}')$.

Furthermore, the procedure terminates after a finite number of applications of Step 1.

PROOF

The proof of Part 1 is essentially the same as that of the corresponding result in [Hu75] (Lemma 3.4). Let $\mathcal{D}_k$, for some $k \geq 0$, be the set obtained after k successful applications of Step 1. We assume that $\mathcal{U}(\mathcal{D}_k) = \mathcal{U}(\mathcal{D})$, which is immediately true for $k = 0$. If there exists a rigid-rigid pair in $\mathcal{D}_k$, let $\langle M, N\rangle$ be the pair selected. If M and N have distinct heads then the algorithm stops and returns the answer **fail**; but if M and N have distinct headings, we know, by Lemma 5.2.10, that $\mathcal{U}(\mathcal{D}_k) = \emptyset$, and therefore that $\mathcal{U}(\mathcal{D}) = \emptyset$, which proves Part 1.

The proof of Part 2 also is essentially the same as that of the corresponding result in [Hu75] (Lemma 3.6). Suppose that M and N do not have distinct headings, and that $\sigma : \Gamma \longrightarrow \Delta$ is a unifying psubstitution. Assume that for all $v \in \mathrm{FV}(M) \cup \mathrm{FV}(N)$ and for all $1 \leq i \leq n$, $x_i \notin \mathrm{FV}(v\sigma)$ and $y_i \notin \mathrm{FV}(v\sigma)$, (otherwise we can simply rename the $x(y)_i$s by α-conversion), so that we have

$$M\sigma = \lambda x_1 : A_1 \ldots \lambda x_n : A_n . @ M_1\sigma \ldots M_p\sigma$$

and

$$N\sigma = \lambda y_1 : B_1 \ldots \lambda y_n : B_n . @ N_1\sigma \ldots N_p\sigma.$$

Now, $M\sigma = N\sigma$ if and only if for all $1 \leq i \leq p$,

$$\lambda x_1 : A_1 \ldots \lambda x_n : A_n . (M_i\sigma) = \lambda y_1 : B_1 \ldots \lambda y_n : B_n . (N_i\sigma),$$

that is, if and only if $\tilde{M}_i\sigma = \tilde{N}_i\sigma$. It is immediate that $\mathcal{D}_{k+1}$ differs from $\mathcal{D}_k$ only in that the pair $\langle M, N \rangle$ is replaced by the set of pairs $\{\langle \tilde{M}_i, \tilde{N}_i \rangle \mid 1 \leq i \leq p\}$. Therefore $\mathcal{U}(\mathcal{D}_{k+1}) = \mathcal{U}(\mathcal{D}_k) = \mathcal{U}(\mathcal{D})$. We note further that the procedure terminates after a finite number of applications of Step 1, since

$$\delta(\mathcal{D}_{k+1}) = \delta(\mathcal{D}_k) - 1.$$

If **fail** has not been returned, we proceed to Step 2 with a set $\mathcal{D}_k$ such that $\mathcal{U}(\mathcal{D}_k) = \mathcal{U}(\mathcal{D})$, and in Step 2 the only operation we perform is to swap the order of some of the pairs in $\mathcal{D}_k$, which has no effect on unification. This proves Part 2. □

We must now prove that disagreement sets in which all pairs are flexible-flexible are always unifiable. This is significantly more difficult than the corresponding result in [Hu75] (Lemma 3.5), and indeed we must delay the proof of this fact until we have proved the correctness of *match*. However, we begin by proving the result for pairs in which the components are of common type, *cf.* [Hu75] (Lemma 3.5, Part 2).

LEMMA 5.3.10 (FLEXIBLE-FLEXIBLE UNIFIABILITY FOR COMMON TYPES)

If $\mathcal{D}$ is (a subset of) a disagreement set in which each pair is flexible-flexible and such that for each pair the components of that pair are of common type then $\mathcal{U}(\mathcal{D}) \neq \emptyset$.

PROOF

We proceed in a manner similar to the elementary construction given in the proof of Lemma 3.5 of [Hu75]:

Suppose the pair $\langle Q, R\rangle$ is such that the head normal forms Q and R have the same ptype (up to α-conversion) and that they are given by

$$Q = \lambda u_1 : A_1 \ldots \lambda u_{n_1} : A_{n_1} . f Q_1 \ldots Q_{r_1}$$

and

$$R = \lambda v_1 : A_1 \ldots \lambda v_{n_1} : A_{n_2} . g R_1 \ldots R_{r_2} ,$$

where

$$\begin{aligned} \tau_\tau(f) = F \;\; = \;\; & \Pi u_{n_1+1} : A_{n_1+1} \ldots \Pi u_{r_1} : A_{r_1} \\ & \Pi u_{r_1+1} : A_{r_1+1} \ldots \Pi u_{s_1} : A_{s_1} . c P_1 \ldots P_r \end{aligned}$$

and

$$\begin{aligned} \tau_\tau(g) = G \;\; = \;\; & \Pi v_{n_2+1} : A_{n_2+1} \ldots \Pi v_{r_2} : A_{r_2} \\ & \Pi v_{r_2+1} : A_{r_2+1} \ldots \Pi v_{s_2} : A_{s_2} . c P_1 \ldots P_r \end{aligned}$$

Suppose that

$$\Delta \equiv \Delta_1, f : F, \Delta_2, g : G, \Delta_3$$

is the current pcontext. We assume, without loss of generality, that f precedes g in Δ — of course there is a trivial case in which they are equal.

Define $\zeta_\Delta(Q, R)$ to be the psubstitution

$$\zeta_\Delta(Q, R) \equiv_{\text{def}} \langle \ldots, \langle M_f, f\rangle, \ldots, \langle M_g, g\rangle, \ldots \rangle$$

where M_f is given by:

$$M_f =_{\text{def}} \lambda w_{n_1+1} : A_{n_1+1} \ldots \lambda w_{s_1} : A_{s_1} [w_{n_1+1}/v_{n_1+1}, \ldots, w_{s_1-1}/v_{s_1-1}] . w_f$$

for w_f a distinct canonical variable of appropriate type (we assume that for a given type, just one such choice of variable is available), and

$$M_g =_{\text{def}} \lambda w_{n_2+1} : A_{n_2+1} \ldots \lambda w_{s_2} : A_{s_2} [w_{n_2+1}/v_{n_2+1}, \ldots, w_{s_2-1}/v_{s_2-1}] . w_g$$

for w_g a distinct canonical variable of appropriate type.

$\zeta_\Delta(Q, R)$ unifies Q and R: to see this, consider the application of $\zeta_\Delta(Q, R)$ to Q and R:

$$Q\zeta_\Delta(Q, R) = \lambda u_1 : A_1 \ldots \lambda u_{n_1} : A_{n_1} . \lambda w_{r_1+1} : A_{r_1+1} \ldots \lambda w_{s_1} : A_{s_1} . w_f$$

and

$$R\zeta_\Delta(Q,R) = \lambda v_1 : A_1 \ldots \lambda v_{n_2} : A_{n_2} . \lambda w_{r_2+1} : A_{r_2+1} \ldots \lambda w_{s_2} : A_{s_2} . w_g .$$

Here we assume that the u_is and v_is do not clash with w_f and w_g. Since Q and R have the same type, we must have that $n_1 + s_1 - r_1 = n_2 + s_2 - r_2$, and $\tau_\tau(w_f) = \tau_\tau(w_g)$, so that we have $w_f = w_g (= w$, say). Therefore $Q\zeta_\Delta(Q,R) = R\zeta_\Delta(Q,R)$.

It remains to note that if the domain pcontext Δ of the psubstitution $\zeta_\Delta(Q,R)$ is given by:

$$\Delta_1, f:F, \Delta_2, g:G, \Delta_3 ,$$

then the range pcontext of the psubstitution is given by:

$$\Delta_1, w:cP_1 \ldots P_r, \Delta_2[M_f/f], \Delta_3[M_f/f, M_g/g] .$$

It is a straightforward matter to verify that $\zeta_\Delta(Q,R)$ is well-defined.

In order to unify a set of *flexible-flexible* pairs, such that for each pair the components of the pair are of common type, we simply compose psubstitutions of the above form. □

5.3.5 The Procedure *match*

Following [Hu75], the procedure *match* is presented in an informal algorithmic style. The procedure *match* is not very different from the procedure *MATCH* of [Hu75]. However, we must take care to maintain the current pcontext — as *match* introduces new variables — and the disagreement set — as match introduces new pairs of types that we must unify. Such new pairs arise because *match* does not construct an object whose type is the same as the variable it is to replace, as in the procedure *MATCH* of [Hu75], but yields an object whose type is merely *similar* to that of the variable it is to replace. Our procedure *reduce* guarantees only similarity of types, and *not* equality. We must take care to extend the order $\sqsupset$ to these new pairs. This ordering allows us to demonstrate that flexible-flexible pairs are always unifiable.

match$(M, N, \Delta, \mathcal{D}_{\mathcal{N}})$

Let $\mathcal{D}_{\mathcal{N}}$ be the reduced disagreement set labelling the current node $\mathcal{N}$ in the matching tree $\mathcal{M}$, and let the current pcontext be Δ. Let M be a flexible pobject and N be a rigid pobject such that $\langle M, N\rangle \in \mathcal{D}$ (and so $\tau_\tau(M) \sim \tau_\tau(N)$). We take the given pobjects M and N to have the following head normal forms:

$$M \equiv \lambda x_1 : A_1 \ldots \lambda x_m : A_m . f M_1 \ldots M_p$$

$$N \equiv \lambda y_1 : B_1 \ldots \lambda y_n : B_n . @ N_1 \ldots N_q$$

where m, n, p, $q \geq 0$, and where the ptypes of f and @ are given by:

$$f : \Pi u_1 : E_1 \ldots \Pi u_p : E_p . \Pi u_{p+1} : E_{p+1} \ldots \Pi u_r : E_r \,.\, cP_1 \ldots P_l \,,$$

and

$$@ : \Pi w_1 : F_1 \ldots \Pi w_q : F_q . \Pi w_{q+1} : F_{q+1} \ldots \Pi w_s : F_s \,.\, cQ_1 \ldots Q_l \,,$$

where c is a constant and where $cP_1 \ldots P_l$ and $cQ_1 \ldots Q_l$ are not of higher kind.

The purpose of the procedure *match* is to adjust the heading of the flexible pobject M, and to this end we shall consider psubstitutions for f of a pobject with a heading of the form $\lambda z_1 : C_1 \ldots \lambda z_t : C_t . @'$, for some t. $@'$ will be either @, in which case the most general such psubstitution will be given by the rule of *imitation*, (Case 1, below), or z_k for some $1 \leq k \leq t$, in which case the most general such psubstitution will be given by the rule of *projection*, (Case 2, below). In *imitation*, we attempt to substitute for f a pobject with the same head as N. In *projection*, we attempt to project f on to one of its arguments.

Since the number of abstractions in a pobject cannot be reduced by psubstitution, and since the number of abstractions in the heading of rigid pobjects is unchanged under psubstitution, we must require that $m \leq n$. So if $m > n$ return $\mathcal{S} = \emptyset$; otherwise $n - m \geq 0$.[7]

[7]The applicability of this argument rests on the *similarity* of the types of M and N. In [Hu75] it is the case that their types are equal, which is a stronger condition.

Case 1 : imitation

The *imitation* construction applies where either @ is a constant, or where $@ = y_i$ for some $m < i \leq n$. Following [Hu75], we distinguish two cases.

1. $n - m > 0$. In this case the pobject which we substitute for f is given uniquely by:

$$M_f =_{\text{def}} \lambda v_1 : E_1 \ldots \lambda v_p : E_p \lambda y_{m+1} : B_{m+1} \ldots \lambda y_n : B_n \,.\, @ N'_1 \ldots N'_q$$

where for $1 \leq i \leq q$, $N'_i = h_i v_1 \ldots v_p y_{m+1} \ldots y_n$ with each h_i and v_i distinct and such that each h_i and each v_i is not in $\mathrm{Dom}(\Delta)$.

For each h_i, $1 \leq i \leq q$, we require that

$$\begin{aligned} \tau_\tau(h_i) \;=\; & \Pi v_1 : E_1 \ldots \Pi v_p : E_p \\ & \Pi y_{m+1} : B_{m+1} \ldots \Pi y_n : B_n \,.\, \tau_\tau(N_i) \,. \end{aligned}$$

We must maintain the condition that $H_1 = \tau_\tau(M_f)$ and $H_2 = \tau_\tau(f)$ are to be unified. We have

$$\begin{aligned} H_1 \;=\; & \Pi v_1 : E_1 \ldots \Pi v_p : E_p \, \Pi y_{m+1} : B_{m+1} \ldots \Pi y_n : B_n \\ & \Pi w_{q+1} : F_{q+1} \ldots \Pi w_s : F_s \,.\, c Q_1 \ldots Q_l \, [N'_1/w_1, \ldots, N'_q/w_q] \end{aligned}$$

and

$$H_2 \;=\; \Pi u_1 : E_1 \ldots \Pi u_k : E_k \, \Pi u_{k+1} : E_{k+1} \ldots u_r : E_r \,.\, c P_1 \ldots P_l \,.$$

We make the assignment

$$\mathcal{D}_{\mathcal{N}'} := \mathcal{D}'_{\mathcal{N}} \cup \mathit{reduce}(H_1, H_2) \,,$$

where $\mathcal{D}'_{\mathcal{N}}$ is obtained by applying the returned psubstitution

$$\langle \ldots \langle M_f, f \rangle \ldots \rangle : \Gamma \to \Delta \,,$$

to the input disagreement set $\mathcal{D}_{\mathcal{N}}$, and where the range pcontext Γ is constructed as described below. Finally, we extend the order $\sqsupset$ to the disagreement set $\mathcal{D}_{\mathcal{N}'}$ by making the assignment $\sqsupset := \sqsupset' \cup \sqsupset_{\tau'}$, where the order $\sqsupset_{\tau'}$ is defined on $\mathit{reduce}(H_1, H_2)$ just as in Definition 5.3.6 and Proposition 5.3.7 and where the order $\sqsupset'$ is just the order $\sqsupset$ on $\mathcal{D}_{\mathcal{N}}$ under the returned psubstitution.

2. $n - m = 0$. In this case it must be that @ is a constant. The pobject M_f substituted for f is of the form:

$$M_f =_{\text{def}} \lambda v_1 : E_1 \ldots \lambda v_k : E_k . @ N'_1 \ldots N'_{q-p+k} ,$$

provided that $\max(0, p-q) \leq k \leq p$, and where $N_i = h_i v_1 \ldots v_k$, for $1 \leq i \leq q-(p-k)$, for suitable h_is, *etc.*. To ensure type-compatibility we necessarily require that

$$\tau_\tau(h_i) = \Pi v_1 : E_1 \ldots \Pi v_k : E_k \, . \, \tau_\tau(N_i) \, .$$

We must maintain the condition that

$$H_1 = \tau_\tau(\lambda v_1 : E_1 \ldots \lambda v_k : E_k . @ N'_1 \ldots N'_{q-p+k})$$

and $H_2 = \tau_\tau(f)$ are to be unified. We have

$$\begin{aligned} H_1 \;\; = \;\; & \Pi v_1 : E_1 \ldots \Pi v_k : E_k \, \Pi w_{q-p+k+1} : F_{q-p+k+1} \cdots \\ & \Pi w_s : F_s \, . \, c Q_1 \ldots Q_l \, [N'_1 / w_1, \ldots, N'_{q-p+k} / w_{q-p+k}] \end{aligned}$$

and

$$H_2 \;\; = \;\; \Pi u_1 : E_1 \ldots \Pi u_k : E_k \, \Pi u_{k+1} : F_{k+1} \ldots u_r : E_r \, . \, c P_1 \ldots P_l \, .$$

We verify that $k + s - (q - p + k) = r$ (from $m = n$ and the similarity of the types of M and N we have that $s - q = r - p$), and make the assignment

$$\mathcal{D}_{\mathcal{N}'} := \mathcal{D}'_{\mathcal{N}} \cup \mathit{reduce}(H_1, H_2) \, ,$$

where $\mathcal{D}'_{\mathcal{N}}$ is obtained by applying the returned psubstitution

$$\langle \ldots \langle M_f, f \rangle \ldots \rangle : \Gamma \rightarrow \Delta \, ,$$

to the input disagreement set $\mathcal{D}_{\mathcal{N}}$, and where the range pcontext Γ is constructed as described below. Finally, we extend the order $\sqsupset$ to the disagreement set $\mathcal{D}_{\mathcal{N}'}$ by making the assignment $\sqsupset := \sqsupset' \cup \sqsupset_{\tau'}$, where the order $\sqsupset_{\tau'}$ is defined on $\mathit{reduce}(H_1, H_2)$ just as in Definition 5.3.6 and Proposition 5.3.7 and where the order $\sqsupset'$ is just the order $\sqsupset$ on $\mathcal{D}_{\mathcal{N}}$ under the returned psubstitution.

Case 2 : projection

Following [Hu75], we distinguish three cases for the *heading* of the object that we may substitute for f. We take the union of their solutions.

1. $\lambda v_1 : E_1 \ldots \lambda v_k : E_k \,.\, v_i$, for $1 \leq k \leq p$ and $1 \leq i \leq k$.

   In this case the pobject that we substitute for f is given by:

$$M_f =_{\text{def}} \lambda v_1 : E_1 \ldots \lambda v_k : E_k . v_i N'_1 \ldots N'_{s'} ,$$

   where for $1 \leq i \leq s'$, $N'_i = h_i v_1 \ldots v_k$, with each h_i and v_i distinct and such that each h_i and v_i is not in $\mathrm{Dom}(\Delta)$, and where $\tau_\tau(h_j)$ is determined as described below.

   We have that $E_i = \Pi z_1 : G_1 \ldots \Pi z_{s''} : G_{s''} \,.\, dR_1 \ldots R_{l'}$, say. We must maintain the condition that $H_1 = \tau_\tau(M_f)$ and $H_2 = \tau_\tau(f)$ are to be unified.

   We have that

$$\begin{aligned} H_1 \;=\; & \Pi v_1 : E_1 \ldots \Pi v_k : E_k \\ & \Pi z_{s''-s'+1} : G_{s''-s'+1} \ldots \Pi z_{s''} : G_{s''} \,. \\ & dR_1 \ldots R_{l'}[N'_1/z_1, \ldots N'_{s'}/z_{s'}] \end{aligned}$$

   and

$$H_2 \;=\; \Pi u_1 : E_1 \ldots \Pi u_k : E_k \, \Pi u_{k+1} : F_{k+1} \ldots u_r : E_r \,.\, cP_1 \ldots P_l \,.$$

   For it to be possible for a solution to exist we must require (for similarity) that

$$k + s'' - s' = r \quad \text{and} \quad d = c$$

   so that we must have that $l = l'$. So in this case we take

$$\tau_\tau(h_i) = \Pi v_1 : E_1 \ldots \Pi v_k : E_k \,.\, G_i ,$$

   for $1 \leq i \leq s'$, and make the assignment

$$\mathcal{D}_{\mathcal{N}'} := \mathcal{D}'_{\mathcal{N}} \cup \mathit{reduce}(H_1, H_2) ,$$

where $\mathcal{D}'_{\mathcal{N}}$ is obtained by applying the returned psubstitution

$$\langle \ldots \langle M_f, f \rangle \ldots \rangle : \Gamma \rightarrow \Delta ,$$

to the input disagreement set $\mathcal{D}_{\mathcal{N}}$, and where the range pcontext Γ is constructed as described below. Finally, we extend the order $\sqsupset$ to the disagreement set $\mathcal{D}_{\mathcal{N}'}$ by making the assignment $\sqsupset := \sqsupset' \cup \sqsupset_{\tau'}$, where the order $\sqsupset_{\tau'}$ is defined on $reduce(H_1, H_2)$ just as in Definition 5.3.6 and Proposition 5.3.7 and where the order $\sqsupset'$ is just the order $\sqsupset$ on $\mathcal{D}_{\mathcal{N}}$ under the returned psubstitution.

2. $\lambda v_1 : E_1 \ldots \lambda v_p : E_p . \lambda y_{m+1} : B_{m+1} \ldots \lambda y_{m+k} : B_{m+k} \,.\, v_i$, for $1 \leq k \leq n - m$ and $1 \leq i \leq p$. Just as in [Hu75], this case is almost identical to the previous one.

   In this case the pobject that we substitute for f is given by:

$$M_f =_{\text{def}} \lambda v_1 : E_1 \ldots \lambda v_p : E_p \, \lambda y_{m+1} : B_{m+1} \ldots \lambda y_{m+k} : B_{m+k} \,.\, v_i N'_1 \ldots N'_{s'},$$

   where for $1 \leq i \leq s'$, $N'_i = h_i v_1 \ldots v_k$, with each h_i and v_i distinct and such that each h_i and v_i is not in $\mathrm{Dom}(\Delta)$, and where $\tau_\tau(h_j)$ and s' are determined in a manner similar to that of the previous case. We extend the disagreement set $\mathcal{D}_{\mathcal{N}}$ and the order $\sqsupset$ in a manner similar to that of the previous case.

3. $\lambda v_1 : E_1 \ldots \lambda v_p : E_p . \lambda y_{m+1} : B_{m+1} \ldots \lambda y_{m+k} : B_{m+k} \,.\, y_{m+i}$, for $1 \leq k \leq n - m$ and $1 \leq i \leq k$.

   Following the substitution of the pobject for f, we are required to unify

$$\lambda x_1 : A_1 \ldots \lambda x_m : A_m \lambda y_{m+1} : B_{m+1} \ldots \lambda y_{m+k} : B_{m+k} . y_{m+i} : B_{m+i}(\ldots)$$

   with

$$\lambda y_1 : B_1 \ldots \lambda y_m : B_m . \lambda y_{m+1} : B_{m+1} \ldots \lambda y_n : B_n . @(\ldots)$$

   and this will be rejected by *simplify* unless $k = n - m$ and $@ = y_{m+i}$, and this case has already been considered in Part 1 of the imitation case .

This concludes the description of the *imitation* and *projection* cases of the procedure *match*. For an analysis of the number of solutions calculated in the case of each of *imitation* and *projection*, the reader is referred to [Hu75].

It now only remains to construct the image pcontext of the psubstitution determined by the two cases given above.

In each of *imitation* and *projection* an object M_f is substituted for the variable f. Let $\mathcal{N}$ be the current selected node and let Δ be the current pcontext for $\mathcal{N}$, so that Δ has the form

$$\Delta \equiv \Delta_1, f{:}A, \Delta_2,$$

where A denotes the ptype of f.

Suppose that all of the free variables of the object M_f are contained in the pcontext Δ_1, Θ, where Θ contains just the new variables introduced by *match*, then the range pcontext of the psubstitution σ given by $\sigma \equiv \langle \ldots, \langle M_f, f \rangle, \ldots \rangle$ is

$$\Gamma \equiv \Delta_1, \Theta, \Delta_2[M_f/f]\,.$$

It follows immediately that the definition of a psubstitution is satisfied. □

We now proceed to prove the correctness of the procedure *match*. The proof is similar to that of the corresponding result of [Hu75], although since our psubstitutions are (ordered) tuples there are some differences. Just as in [Hu75], we introduce a complexity measure for psubstitutions. The measure is defined on the head normal forms of the pobjects constructed.

DEFINITION 5.3.11 (THE MEASURE θ)

Let the psubstitution $\xi : \Gamma \longrightarrow \Delta$ be given by

$$\xi \equiv \langle\langle M_1, y_1\rangle, \ldots, \langle M_i, y_i\rangle, \ldots, \langle M_r, y_r\rangle\rangle$$

then the measure $\theta(\xi) \in \mathbb{N}$ is given by

$$\theta(\xi) = r + \sum_{i=1}^{r} \pi(M_i)\,,$$

where the measure π is given by Definition 5.2.7. □

PROPOSITION 5.3.12 (CORRECTNESS OF MATCH)

Suppose that the terms M and N form the pair $\langle M, N\rangle$ which occurs at the node $\mathcal{N}$ which has current pcontext Δ. Suppose further that M is flexible and that N is rigid. If there exists $\rho : \Gamma \longrightarrow \Delta \in \mathcal{U}(\{\langle M, N\rangle\})$, then $match(M, N, \Delta, \mathcal{D}_{\mathcal{N}})$ returns a non-empty set $\mathcal{S}$ in which there is a psubstitution σ such that $\rho \leq_\Delta \sigma$, and we can find a psubstitution η such that $\rho =_\Delta \eta \circ \sigma$ with $\theta(\eta) < \theta(\rho)$.

PROOF

The proof is very similar to that of the corresponding result in [Hu75]. Let x_f be the head of M, and let @ be that of N. If ρ unifies M and N, then there must exist in (the tuple) ρ a pair pertaining to x_f. Suppose then that

$$x_f\rho = \lambda z_1 : B_1 \ldots \lambda z_k : B_k.@' N_1' \ldots N_l' (= M_{x_f})\,,$$

that is ρ is of the form

$$\langle \ldots, \langle M_{x_f}, x_f\rangle, \ldots\rangle\,.$$

As with the proof of the corresponding result in [Hu75], there are two cases to consider:

1. $@'$ is @ or some z_i, $1 \leq i \leq k$. In *match* we considered all possible cases of such headings, so that there must exist in $\mathcal{S} = match(M, N, \Delta, \mathcal{D}_{\mathcal{N}})$ a psubstitution σ with a component

$$\langle \lambda w_1 : B_1 \ldots \lambda w_k : B_k.@'' N_1'' \ldots N_l'', x_f\rangle$$

where $@'' = @'[w_1/z_1, \ldots, w_k/z_k]$ and $N''_j = h_j w_1 \ldots w_k$ for $1 \leq j \leq l$; such a $\sigma : \Theta \longrightarrow \Delta$ is given by

$$\langle\langle x_1, x_1\rangle, \ldots, \langle \lambda w_1 : B_1 \ldots \lambda w_k : B_k . @'' N''_1 \ldots N''_l, x_f\rangle, \ldots, \langle x_n, x_n\rangle\rangle ,$$

where the pcontext Θ is constructed from the pcontext Δ by removing the component labelled by f and adding suitable components labelled by $h_1, \ldots, h_l$.

Let $\eta : \Gamma \longrightarrow \Theta$ be the psubstitution given by

$$\eta \equiv \langle\langle M_1, x_1\rangle, \ldots, \langle P_1, h_1\rangle, \ldots, \langle P_l, h_l\rangle, \ldots, \langle M_n, x_n\rangle\rangle ,$$

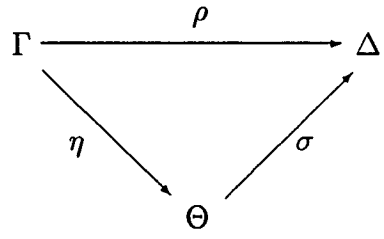
where $P_i = \lambda z_1 : B_1 \ldots \lambda z_k : B_k . N'_i$, $1 \leq i \leq l$, and where we assume that ρ is given by

$$\langle\langle M_1, x_1\rangle, \ldots, \langle M_{x_f}, x_f\rangle, \ldots, \langle M_n, x_n\rangle\rangle .$$

We note that it is a straightforward calculation to show that σ and η are well-defined, that $\rho =_\Delta \eta \circ \sigma$, and that

$$\begin{aligned} \theta(\eta) &= l - 1 + \textstyle\sum_{j=1}^{f-1} \pi(M_j) + \sum_{j=f+1}^{l} \pi(M_j) + \\ &\quad l + \textstyle\sum_{i=1}^{l} \pi(P_i) \\ &= \theta(\rho) - 1 < \theta(\rho) . \end{aligned}$$

This construction may be elucidated by the following commuting diagram of psubstitutions:



N.B. This diagram is of psubstitutions and so is not in the syntactic category that is related to the syntactic category $C(\Sigma)$ (see Appendix A) that is constructed using just β-equality. Do psubstitutions have an interesting categorical structure?

2. $@'$ is neither $@$ nor some z_i. In this case we must have that

$$M\rho = \lambda x_1 : A_1 \ldots \lambda x_n : A_n . @'' \ldots ,$$

where, for $1 \leq j \leq n$, $@''$ is distinct from $@$, x_j and z_j, so that $M\rho \neq N\rho$, and so this case can never arise. As in [Hu75], this proves that the rules of *imitation* and *projection* are sufficient for the existence algorithm. □

We are now able to prove the unifiability of simplified disagreement sets in which all the pairs are flexible-flexible.

PROPOSITION 5.3.13 (FLEXIBLE-FLEXIBLE UNIFIABILITY)

Let $simplify(\mathcal{D})$ be a simplified disagreement set that labels some node $\mathcal{N}$ of a matching tree $\mathcal{M}$ for $M : A$ and $N : B$ (well-typed in some context). If $simplify(\mathcal{D})$ = **succeed**, then $\mathcal{U}(\mathcal{D}) \neq \emptyset$.

PROOF

We must show that if *simplify* returns **succeed** then the input disagreement set is always unifiable.

simplify returns **succeed** when all pairs in the disagreement set are flexible-flexible, so we must show that such a set is always unifiable. We cannot simply apply the elementary construction of [Hu75] to flexible-flexible pairs since the components any given pair may have types which are only similar, not equal. However the ordering $\sqsupset$ on $\mathcal{D}$ allows us to construct a suitable substitution.

The first step is to note that we extended the order $\sqsupset$ to all of the elements of $\mathcal{D}$ by extending its definition each time we added new disagreement pairs to $\mathcal{D}$, in the procedures *simplify* and *match*, so that $\sqsupset$ was preserved on the existing, although modified, pairs.

We now observe that the following property of $\sqsupset$ is preserved by the procedures *simplify* and *match*: if the pair $\langle M', N' \rangle$ is such that each pair $\langle M'', N'' \rangle$ such that $\langle M'', N'' \rangle \sqsupset \langle M', N' \rangle$ is unifiable, then $\tau_\tau(M')$ and $\tau_\tau(N')$ are unifiable.

The result now follows by a simple inductive argument on $\sqsupset$. It is easy to see, by induction on the structure of ptypes, that in the pair that is assigned the highest level the two components are of common ptype. Therefore the pair at this level may be unified by the psubstitution ξ_1 which is constructed as in Lemma 5.3.10

So ξ_1 unifies the pair of highest level in the induced ordering $\sqsupset$. But now, under ξ_1, the components of the pair of level one lower in $\sqsupset$ than those unified by ξ_1 are of common ptype (ξ_1 unifies their ptypes), and so we may apply a similar construction to unify them; of course the pcontext has changed, it is now the range pcontext of the previous psubstitution.

By repeating this process we unify all the pairs of $\mathcal{D}$ and obtain a psubstitution

$$\xi_{\mathcal{D}} =_{\text{def}} \xi_d \circ \ldots \circ \xi_1 ,$$

where d is the number of levels in $\mathcal{D}$, which unifies the **succeed** node labelled with $\mathcal{D}$. □

5.3.6 Correctness of the Unification Algorithm

We now prove the correctness of the unification algorithm. This means that we prove a *soundness* theorem, that if a matching tree for two terms $M:A$ and $N:B$ has a success node, then $M:A$ and $N:B$ are unifiable, and a *completeness* theorem, that if $M:A$ and $N:B$ are unifiable, then every matching tree for $M:A$ and $N:B$ has a success node. In this setting, as opposed to that of [Hu75], we must take account of our procedure *reduce* and of the more complex *match* procedure.

THEOREM 5.3.14 (SOUNDNESS OF THE UNIFICATION ALGORITHM)

Let $\mathcal{M}$ be a matching tree for the pair of terms $M:A$ and $N:B$. For any success node $\mathcal{N}$ in $\mathcal{M}$,

$$\zeta_p \circ \sigma_{\mathcal{N}_p} : \Gamma \longrightarrow \Delta$$

unifies $M:A$ and $N:B$, where Δ is given by the construction given in *match* and where $\sigma_{\mathcal{N}_p}$ and ζ_p are given by the constructions of the proof given below.

PROOF

As usual, the proof is very similar to that of the corresponding result in [Hu75]. Suppose that we have constructed a matching tree for $reduce(A, B) \cup \{\langle M, N\rangle\}$ which possesses a success node on some branch:

$$\mathcal{N}_0 \xrightarrow{\sigma_1} \mathcal{N}_1 \xrightarrow{\sigma_2} \ldots \xrightarrow{\sigma_p} \mathcal{N}_p, \quad p \geq 0,$$

such that $\mathcal{D}_{\mathcal{N}_p} = \textbf{succeed}$. Let ζ_p be defined as a psubstitution which unifies $\mathcal{D}_{\mathcal{N}_p}$, as constructed in (the proof of) Proposition 5.3.13:

$$\zeta_p =_{\text{def}} \xi_{\mathcal{D}_{\mathcal{N}_p}}$$

$$\zeta_i =_{\text{def}} \zeta_{i+1} \circ \sigma_{i+1}, \quad 0 \leq i < p,$$

and prove by descending induction on i that ξ_i unifies $\mathcal{D}_{\mathcal{N}_i}$. The base case of this induction is contained in Proposition 5.3.13; for the induction step, assume that ξ_{i+1} unifies $\mathcal{D}_{\mathcal{N}_{i+1}}$, and let

$$\overline{\mathcal{D}_{\mathcal{N}_{i+1}}} = \{\langle M'\sigma_{i+1}, N'\sigma_{i+1}\rangle \mid \langle M', N'\rangle \in \mathcal{D}_{\mathcal{N}_i}\} \cup reduce(H_1, H_2)$$

By Proposition 5.3.9 (Part 2) and the induction hypothesis we have that since

$$\mathcal{D}_{\mathcal{N}_{i+1}} = simplify(\overline{\mathcal{D}_{\mathcal{N}_{i+1}}}),$$

ξ_{i+1} unifies $\overline{\mathcal{D}_{\mathcal{N}_{i+1}}}$; that is, for every $\langle M', N'\rangle \in \mathcal{D}_{\mathcal{N}_i}$, and every $\langle M'', N''\rangle \in reduce(H_1, H_2)$

$$M'\sigma_{i+1}\xi_{i+1} = N'\sigma_{i+1}\xi_{i+1} \qquad \text{and} \qquad M''\sigma_{i+1}\xi_{i+1} = N''\sigma_{i+1}\xi_{i+1}.$$

Therefore ξ_i unifies $\mathcal{N}_i$. This completes the induction step.

In particular we now get that $\xi_0 = \xi_p \circ \sigma_p \circ \ldots \circ \sigma_1$ unifies

$$\mathcal{D}_{\mathcal{N}_0} = simplify(reduce(A, B) \cup \{\langle M, N\rangle\})$$

so that by Proposition 5.3.9 (Part 2), ξ_0 unifies

$$reduce(A, B) \cup \{\langle M, N\rangle\}.$$

By setting $\sigma_{\mathcal{N}_p} =_{\text{def}} \sigma_p \circ \ldots \circ \sigma_1$ and applying Proposition 5.3.7, the result follows.

□

THEOREM 5.3.15 (COMPLETENESS OF THE UNIFICATION ALGORITHM)

If $\mathcal{U}(M:A, N:B) \neq \emptyset$, then every matching tree for the pair of terms $M:A$ and $N:B$ possesses a success node.

PROOF

Again, the proof is very similar to that of the corresponding result in [Hu75]. We assume that $M:A$ and $N:B$ are unifiable by the psubstitution $\rho : \Gamma \longrightarrow \Delta$, and consider an arbitrary matching tree for

$$reduce(A, B) \cup \{\langle M, N\rangle\}.$$

As in [Hu75], we construct a branch

$$\mathcal{N}_0 \xrightarrow{\sigma_1} \mathcal{N}_1 \xrightarrow{\sigma_2} \ldots \xrightarrow{\sigma_i} \mathcal{N}_i$$

of this tree, together with a sequence of psubstitutions $\xi_0, \ldots, \xi_i$ such that for all $i \geq 0$, $\xi_i \in \mathcal{U}(\mathcal{D}_{\mathcal{N}_i})$ or $\mathcal{D}_{\mathcal{N}_i} = \textbf{succeed}$, and of strictly decreasing complexity. As in [Hu75], this is proved by ascending induction on i. The base case of the induction is immediate, taking $\xi_0 = \rho$, since if

$$\mathcal{D}_{\mathcal{N}_0} = simplify(reduce(A, B) \cup \{\langle M, N\rangle\}) \neq \textbf{succeed}$$

then by Proposition 5.3.9 (Part 2), $\rho \in \mathcal{U}(\mathcal{D}_{\mathcal{N}_0})$. For the induction step we assume that $\xi_i \in \mathcal{U}(\mathcal{D}_{\mathcal{N}_i})$. Either $\mathcal{D}_{\mathcal{N}_i} = \textbf{succeed}$, in which case we stop the construction, or we are considering in $\mathcal{D}_{\mathcal{N}_i}$ a pair $\langle M', N'\rangle$ such that N' is rigid. By Proposition 5.3.9 (Part 1) $\mathcal{D}_{\mathcal{N}_i}$ cannot be **fail** since by the induction hypothesis $\xi_i \in \mathcal{U}(\mathcal{D}_{\mathcal{N}_i})$. By Proposition 5.3.12, $match(M_1, M_2, \Delta_{\mathcal{N}_i}, \mathcal{D}_{\mathcal{N}_i})$, where $\Delta_{\mathcal{N}_i}$ is the current pcontext for $\mathcal{N}_i$, returns some σ such that there exists η with

$$\xi_i =_{\Delta_{\mathcal{N}_i}} \eta \circ \sigma,$$

so that there exists in the tree a branch $\mathcal{N}_i \xrightarrow{\sigma} \mathcal{N}'$. Choose $\xi_{i+1} = \eta$, $\sigma_{i+1} = \sigma$ and $\mathcal{N}_{i+1} = \mathcal{N}'$. By construction, $\mathcal{D}_{\mathcal{N}_{i+1}} = simplify(\overline{\mathcal{D}_{\mathcal{N}_{i+1}}})$, where

$$\overline{\mathcal{D}_{\mathcal{N}_{i+1}}} = \{\langle M'\sigma, N'\sigma\rangle \mid \langle M', N'\rangle \in \mathcal{D}_{\mathcal{N}_i}\} \cup reduce(H_1, H_2).$$

By the induction hypothesis, $M'\xi_i = N'\xi_i$ for all $\langle M', N' \rangle \in \mathcal{D}_{\mathcal{N}_i}$, so that

$$M'\sigma_{i+1}\xi_{i+1} = N'\sigma_{i+1}\xi_{i+1},$$

since $\Delta_{\mathcal{N}_i}$ contains all the free variables of M' and N'. Since ξ_{i+1} unifies each such $\langle M'\sigma_{i+1}, N'\sigma_{i+1} \rangle$ it follows that $\sigma_{i+1}\xi_{i+1}$ unifies $reduce(H_1, H_2)$. This shows that ξ_{i+1} unifies $\overline{\mathcal{D}_{\mathcal{N}_{i+1}}}$, so that by Proposition 5.3.9 (Part 2), either $\mathcal{D}_{\mathcal{N}_{i+1}} = \mathbf{succeed}$ or ξ_i unifies $\mathcal{D}_{\mathcal{N}_{i+1}}$. By Proposition 5.3.12, we have that $\theta(\xi_{i+1}) < \theta(\xi_i)$. This concludes the induction step. Therefore our construction yields a **succeed** node of complexity at most $\theta(\rho)$ and the result follows by Proposition 5.3.7. □

5.3.7 Complete Unification

Following [Hu75], we investigate the extent to which we are able to enumerate all unifiers of two terms $M : A$ and $N : B$, which are well-typed in some given context Δ, using our existence algorithm. As in [Hu75], we aim to strengthen the completeness theorem. In fact we have already done all of the necessary work, and indeed the arguments of Section 4.3 of [Hu75] go through almost unaltered.

DEFINITION 5.3.16 (COMPLETE SETS OF UNIFIERS)

Suppose that $\Delta \vdash_\Sigma M{:}A$ and $\Delta \vdash_\Sigma N{:}B$. A *complete set of unifiers* of $M{:}A$ and $N{:}B$ on Δ is any set $\mathcal{S}$ of psubstitutions such that:

1. $\mathcal{S} \subseteq \mathcal{U}(M{:}A, N{:}B)$;
2. For all $\rho \in \mathcal{U}(M{:}A, N{:}B)$, there exists $\sigma \in \mathcal{S}$ such that $\rho \leq_\Delta \sigma$. □

Now, in [Hu75], Huet defines the notion of a *complete matching tree.* A complete matching tree is distinguished from a matching tree in that the third argument given to his procedure $MATCH$ at each node $\mathcal{N}$ is given by a recursive construction which ensures that this set contains all the new variables introduced on the branch leading to $\mathcal{N}$. In our case however, by construction, the pcontext given as the third argument to *match* already has this property, so that we need no further definitions. Indeed, the proof of Theorem 4.4 of [Hu75] now goes straight through.

THEOREM 5.3.17 (COMPLETE UNIFICATION)

Let $\mathcal{M}$ be any complete matching tree for the pair $M:A$ and $N:B$ on Δ. For any $\sigma \in \mathcal{U}(M:A, N:B)$ there exists a **succeed** node $\mathcal{N}$ in $\mathcal{M}$ such that $\sigma \leq_\Delta \sigma_\mathcal{N}$.

PROOF

As for the corresponding result in [Hu75], we adapt the proof of our earlier completeness theorem. To this end we add the condition $\rho =_\Delta \xi_i \circ \sigma_{\mathcal{N}_i}$ to the induction hypothesis.

The base case is trivially obtained; $\sigma_{\mathcal{N}_0}$ is the identity, $\xi_0 = \rho$. For the induction step, we have that $\xi_i =_{\Delta_{\mathcal{N}_i}} \xi_{i+1} \circ \sigma_{i+1}$, and we note that, since this property is true of $\sigma_1, \sigma_2, \ldots, \sigma_i$, for all of the pairs $\langle M, x\rangle$ of $\sigma_{\mathcal{N}_i}$, $\mathrm{FV}(M) \subset \mathrm{Dom}(\Delta_{\mathcal{N}_i})$. This implies that

$$\xi_i \circ \sigma_{\mathcal{N}_i} =_{\Delta_{\mathcal{N}_i}} \xi_{i+1} \circ \sigma_{i+1} \circ \sigma_{\mathcal{N}_i} = \xi_{i+1} \circ \sigma_{\mathcal{N}_{i+1}},$$

and therefore, since $\mathrm{Dom}(\Delta) \subset \mathrm{Dom}(\Delta_{\mathcal{N}_i})$, that $\rho =_\Delta \xi_{i+1}\sigma_{\mathcal{N}_{i+1}}$. This concludes the induction step. Therefore for the node $\mathcal{N}_p$ labelled with **succeed**, we have that $\rho =_\Delta \xi_p \circ \sigma_{\mathcal{N}_p}$. Hence result. □

5.3.8 η-reduction

Just as in [Hu75], the addition of η-reduction enables us to simplify the unification algorithm. In particular the procedure *match* is greatly simplified. As in [Hu75], the following simplifications in *match* are possible:

- We have in *match* that $m = n$, so that we can eliminate Case 1 in *imitation* and Case 2 in *projection*;
- In Case 2 in *imitation* and in Case 1 in *projection* we can restrict to the solution $k = p$.

We note that in this case we need to consider *long head normal forms* in Lemma 5.2.6

5.4 Complete-enumeration Algorithms

In contrast to [Hu75], the work of Jensen and Pietrzykowski [JP76] provides an algorithm which enumerates all of the unifiers of a pair of simply-typed λ-terms. This algorithm enumerates all of the unifiers instead of reporting that unifiers exist (with the possibility of constructing certain instances). This is achieved by taking approximation rules of *elimination*, *iteration* and *identification* in addition to the rules of *imitation* and *projection*. The reader is referred to [JP76] for a detailed explanation of these rules.

If we take this algorithm as our basis, we conjecture that it is possible to obtain an algorithm for the complete enumeration of all of the unifiers of a pair of well-typed $\lambda\Pi$-terms in a remarkably simple manner. We outline the steps which are necessary in order to enumerate all of the unifiers of a pair of terms which are well-typed in some given context.

1. Suppose the well-typed terms are given by $\Delta \vdash_\Sigma M:A$ and $\Delta \vdash_\Sigma N:B$.

2. Apply the procedure *reduce* to the given pair of well-typed terms.

3. Enumerate all unifiers of the pairs of objects resulting from the reduction of A and B and by utilizing the ordering given in the proof of Proposition 5.3.7 (Part 2) and the modified Jensen and Pietrzykowski algorithm; thereby enumerate all unifiers of A and B.

4. Each time a unifying substitution, σ, for A and B is calculated, apply it to the objects M and N, and proceed to apply the modified Jensen and Pietrzykowski algorithm to the resulting pair of terms (which now have the same type); thereby enumerate all the unifiers of $M\sigma$ and $N\sigma$.

5. If ρ unifies $M\sigma$ and $N\sigma$, then $\sigma\rho$ unifies $M:A$ and $N:B$.

Although of little practical value, the existence of such a procedure is important if we wish to state a sufficiently strong completeness theorem for $\lambda\Pi$-calculus search and computation procedures.

5.5 Summary

In this chapter we have shown how to extend the unification algorithm for the simply-typed λ-calculus to the $\lambda\Pi$-calculus. Further work based on this chapter falls into four main groups:

1. A study of decidable subproblems;

2. Extensions of the algorithm to cope with more complicated systems of dependent types;

3. An algebraic study of unification in *Generalized Algebraic Theories* [Ca86], perhaps in the manner suggested by the work of Burstall, Rydeheard and Stell [RS87];

4. Heuristic improvements to the algorithm for efficiency.

Chapter 6

Effective Search in the $\lambda\Pi$-calculus

The main part of the work of this chapter was performed at the B.P. Research Centre, Sunbury-on-Thames, during the Autumn of 1988 in collaboration with Lincoln Wallen of the University of Oxford [PW90].

6.1 Introduction

So far we have reduced the amount of non-determinism in proof search in the $\lambda\Pi$-calculus by:

- Using the calculus **L** as the basis for proof search;
- Considering uniform proofs; and
- Considering the resolution rules.

In this chapter we show how to remove a further source of non-determinism in proof-search — that due to the choice of object for use with the (Πl) rule, by introducing a metacalculus **U** for **L** in which the (Πl) rule has the form:

$$(\Pi l)\quad \frac{\Gamma, x{:}B[\alpha/y] \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C} \qquad \begin{array}{l}\text{(a) } @ : \Pi y{:}A.B \in \Sigma \cup \Gamma \\ \text{(b) } x \notin \mathrm{Dom}(\Gamma).\end{array}$$

This rule has no appeal to the external calculus G\cut, but introduces a free or *universal* variable into the proof, denoted here by α. Universal variables are distinct from the usual *eigenvariables* that are bound by the derivation (and explicitly declared in contexts). The following *closure condition* for **U** is used to compensate for the omission of object information in the (Πl) rule:

$$\Gamma \Rightarrow_\Sigma A \qquad \text{(a)}\ (\exists\sigma)\ B\sigma \equiv A\sigma, @{:}B \in \Sigma \cup \Gamma\,,$$

where σ is an *instantiation* of the universal variables of the sequent, and $B\sigma$ denotes the normal form of the term resulting from the application of σ (as a substitution) to B. The calculation of instantiations can be performed by the *unification algorithm* for the $\lambda\Pi$-calculus given in Chapter 5.

A **U**-proof is a pair $\langle \psi, \sigma \rangle$ consisting of a **U**-derivation ψ and an instantiation σ such that $\psi\sigma$ (the application of σ as a substitution to ψ) is an **L**-proof. Not every instantiation that closes the leaves of a given **U**-derivation will yield an **L**-proof when applied to that **U**-derivation. It is sufficient to check that the instantiation can be *well-typed* in the derivation to ensure that the result is an **L**-proof. If Γ is the context and A the type of the universal variable α when introduced into the **U**-derivation, then the well-typing condition for α amounts to:

$$\Gamma\sigma \vdash_\Sigma \alpha\sigma{:}\, A\sigma$$

ensuring that side condition (c) of the (Πl) rule of **L** — the condition omitted from the (Πl) rule of **U** — is nevertheless satisfied in $\psi\sigma$. The unconstrained choice of object in the (Πl) rule of **L** is replaced by a highly constrained choice in **U**.[1] This wholesale reduction in the search space is analogous, of course, to that obtained by Robinson in the context of the predicate calculus [Ro65].

---

[1]The inference system that corresponds to the calculation of instantiations by unification does indeed have a *subterm property*. The soundness and completeness result for **U** is a form of *Herbrand Theorem* for the theory. The unification algorithm searches amongst the terms of a "Herbrand universe" defined by each leaf sequent. This aspect is not explored in detail here, see Chapter 7.

The well-typing of an instantiation depends on the structure of the derivation from which it is calculated. However, even if it fails to be well-typed in that derivation it may be well-typed in some permutation of the derivation, since rule applications (or reductions) can sometimes be permuted whilst leaving the endsequent (*i.e.,* root) of the derivation and its leaves unchanged. The degree to which this can be done is summarized in the form of a *Permutation Theorem,* in the sense of Kleene [Kl52] and Curry [Cu52], and underlies a system, called **R**, which is a metacalculus for **U**.

The rules of **R** are just the rules of **U** (so the derivations are the same) but the condition for instantiations to yield a proof is weakened. An **R**-proof is again a pair consisting of a derivation ψ and an instantiation σ under which ψ is closed, but now we require only that there exist perhaps another (closed) derivation ψ^* in which the instantiation is well-typed; *i.e.,* $\psi^*\sigma$ is an **L**-proof. Of course the crucial computational question is whether the existence of at least one suitable ψ^*, given ψ and σ, can be determined as the search progresses. We show that this is indeed the case using a *reduction ordering:* a notion which was introduced by Bibel [Bi81] for classical connectives and extended by Wallen in [Wa89] to various non-classical connectives.[2] An **R**-proof therefore corresponds to an equivalence class of **U**-proofs of the same endsequent consisting of all permutation variants of the original derivation in which the calculated instantiation is well-typed.

---

[2]The condition is equivalent to an enhanced "occurs-check" in the unification algorithm if a suitable notion of *Skolem function* were introduced. The suitable notion is *not* the obvious one that the reader might suppose from experience of classical quantifiers, not least because the logic under investigation here is intuitionistic; actually, the logic — lacking negation — is minimal, but the distinction is not important in this context. In general, the theoretical diversion via Skolemization is unnecessary and can be difficult to justify semantically, even in simple type theory (*cf.* [Mi83]). Our approach follows Herbrand's Theorem (which is finitary) rather than the Skolem-Herbrand-Gödel Theorem (which is not).

6.2 A Metacalculus for L

We introduce a new syntactic class of *universal* variables denoted by lowercase Greek letters $\alpha, \beta, \ldots$, and extend the syntactic category of objects to include them thus:

$$\textit{Objects} \quad M \ ::= \ c \mid \alpha \mid x \mid \lambda x{:}A.M \mid MN.$$

Notice that universal variables cannot appear λ-bound. By virtue of this extension, entities of all syntactic classes may now contain universal variables as subterms. When we wish to emphasize that a syntactic entity does not contain universal variables we shall refer to it as being *ground.*

We define a calculus for sequents, which is to be read as a system of reduction operators, by dropping the axiom schemata of **L** and modifying the (Πl) rule as follows:

DEFINITION 6.2.1 (THE CALCULUS **U**)

The rules of **U** consist of the $(\rightarrow r)$, $(\rightarrow l)$ and (Πr) rules of **L**, with the side conditions modified so that contexts are extended only with new variables when the rules are read as reduction operators, together with the new form of (Πl) rule, as given below:

$$(\rightarrow r) \qquad \frac{\Gamma, x{:}A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma A \rightarrow B} \qquad \text{(a) } x \notin \mathrm{Dom}(\Gamma),\ x \notin \mathrm{FV}(B)$$

$$(\Pi r) \qquad \frac{\Gamma, x{:}A \Rightarrow_\Sigma B}{\Gamma \Rightarrow_\Sigma \Pi x{:}A.B} \qquad \text{(a) } x \notin \mathrm{Dom}(\Gamma)$$

$$(\rightarrow l) \qquad \frac{\Gamma \Rightarrow_\Sigma A \qquad \Gamma, x{:}B \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C} \qquad \begin{array}{l}\text{(a) } x \notin \mathrm{Dom}(\Gamma) \\ \text{(b) } @{:}\, A \rightarrow B \in \Sigma \cup \Gamma\end{array}$$

$$(\Pi l) \qquad \frac{\Gamma, x{:}B[\alpha/y] \Rightarrow_\Sigma C}{\Gamma \Rightarrow_\Sigma C} \qquad \begin{array}{l}\text{(a) } @ : \Pi y{:}A.B \in \Sigma \cup \Gamma \\ \text{(b) } x \notin \mathrm{Dom}(\Gamma).\end{array}$$

In the (Πl) rule we assume that α is not introduced elsewhere in the derivation. □

DEFINITION 6.2.2 (U-DERIVATION)

U-*derivations* are trees regulated by the rules of Definition 6.2.1 such that the sequent at the root of the tree is well-formed. In applications of the (Πl) rule we call Γ the *typing context* of α and A the *type* of α. □

Note that we have not yet defined **U**-*proofs*; as yet there are no axiom schemata.

The system **U** thus consists of four operational rules. Notice that the (Πl) rule no longer contains an external appeal to **G**\cut or a choice of term, but is otherwise identical to the (Πl) rule of **L**.

DEFINITION 6.2.3 (INSTANTIATION)

An *instantiation* is a mapping from universal variables to objects. The capture-avoiding application of instantiations to all of the constructs of the language is defined in the obvious way. The notation that we use for instantiations is the same as that which we use for substitutions in Chapter 5. □

The following notion compensates for the absence of axiom schemata in **U**:

DEFINITION 6.2.4 (CLOSURE)

A sequent $\Gamma \Rightarrow_\Sigma A$ is said to be *closed under* an instantiation σ just in case, for some declaration $@{:}B \in \Sigma \cup \Gamma$ such that $B\sigma \rightarrow_{\beta\eta} D$ and $A\sigma \rightarrow_{\beta\eta} C$, $D \equiv C$. A **U**-derivation is said to be *closed under* σ just in case all of its leaf sequents are closed under σ. □

We are interested in instantiations that are *well-typed* in the following sense:

DEFINITION 6.2.5 (WELL-TYPING)

An instantiation σ is said to be *well-typed* in an **U**-derivation just in case for every universal variable α of the derivation, with typing context Γ and type A, we have:

$$\mathbf{G}\backslash\text{cut proves } \Gamma\sigma \vdash_\Sigma \alpha\sigma{:}A\sigma\,. \quad \square$$

We are now in a position to define a notion of proof for **U**.

DEFINITION 6.2.6 (U-PROOF)

An **U**-*proof* is a pair $\langle \psi, \sigma \rangle$, where ψ is an **U**-derivation and σ an instantiation, such that (1) ψ is closed under σ, and (2) σ is well-typed in ψ. □

THEOREM 6.2.7 (U-PROOFS YIELD L-PROOFS)

If $\langle \psi, \sigma \rangle$ is an **U**-proof then $\psi\sigma$ is an **L**-proof.
PROOF

By induction on the structure of **U**-derivations. The closure condition (1) ensures that the leaves of $\psi\sigma$ are **L**-axioms. The well-typing condition (2) ensures that the image under σ of each instance of a (Πl) rule (of **U**) in ψ satisfies the side conditions on the (Πl) rule of **L**. The remaining operational rules are common to the two systems. □

We remark that it is immediate that any **L**-proof arises as a **U**-proof: this is the converse of Theorem 6.2.7.

We shall postpone discussion of proof-search in **U** until we have introduced a further refinement. One could stop here. However, using the unification algorithm developed in Chapter 5 (based on that of [Hu75]) to calculate instantiations when a putative leaf has been reached, then checking the well-typing condition using **G**\cut (recall that **G**\cut is a decidable system for well-typing). The search space induced by **U** is a proper subspace of that induced by **L** since the choice of object at (Πl) reductions in the former is constrained by the syntactic content of the leaf sequents (*cf.* [Ro65]). The main reason for considering a further refinement is that **U** distinguishes between derivations that are *intrinsically* identical in a sense made precise below. Consequently the search space induced by **U** still contains a major source of redundancy.

6.3 A Metacalculus for U

The content of the typing contexts of universal variables in a **U**-derivation depends on the structure of the derivation. For example, the two **U**-derivations below differ in the order in which the (Πl) and (Πr) rule have been applied (remember the figures should be read from endsequent to premisses):

$$\begin{array}{ll} (\Pi r) & \dfrac{\Gamma, z{:}B(\alpha), x{:}A \Rightarrow_\Sigma B(x)}{\Gamma, z{:}B(\alpha) \Rightarrow_\Sigma \Pi x{:}A.B(x)} \\ (\Pi l) & \dfrac{\Gamma, z{:}B(\alpha) \Rightarrow_\Sigma \Pi x{:}A.B(x)}{\Gamma \Rightarrow_\Sigma \Pi x{:}A.B(x)} \end{array} \qquad \begin{array}{ll} & \dfrac{\Gamma, x{:}A, z{:}B(\alpha) \Rightarrow_\Sigma B(x)}{\Gamma, x{:}A \Rightarrow_\Sigma B(x)} \\ & \dfrac{\Gamma, x{:}A \Rightarrow_\Sigma B(x)}{\Gamma \Rightarrow_\Sigma \Pi x{:}A.B(x)} \end{array} \begin{array}{l} (\Pi l) \\ (\Pi r) \end{array}$$

in which the principal formula of the (Πl) reduction in each derivation is the declaration $@ : \Pi y{:}A.B(y)$ assumed to be an element of the context Γ. We also assume that $x, z \notin \mathrm{Dom}(\Gamma)$. The instantiation σ that maps α to x closes both derivations. The typing context of α in the first derivation is Γ, while in the second it is $\Gamma, x{:}A$. Since $\alpha\sigma = x$, in order to check the well-typing condition for α we must show $\Gamma \vdash_\Sigma x{:}A$ for the first derivation and $\Gamma, x{:}A \vdash_\Sigma x{:}A$ for the second. Consequently σ is well-typed in the second derivation but not in the first (since $x \notin \mathrm{Dom}(\Gamma)$).

Our second refinement is to introduce a calculus, **R**, in which the existence of the **U**-derivation on the left, together with the closing instantiation, is sufficient to infer the existence of the **U**-proof on the right. That is, we investigate conditions under which rule instances may be *permuted* whilst leaving the endsequent and leaves of the derivation essentially unchanged.

DEFINITION 6.3.1 (**R**-DERIVATION)

The rules of **R** are exactly the rules of **U**; consequently the derivations of **R** are exactly those of **U**. (The notion of **R**-proof however differs from that of **U**-proof; see below.) □

Let ψ be an **R**-derivation of a given endsequent and let $\mathcal{F}_\psi$ denote the collection of inferences that comprise ψ[3]. We use $\mathcal{F}_\psi(\Xi)(\subseteq \mathcal{F}_\psi)$ to denote the inferences of a given type Ξ, for Ξ one of the following: $(\rightarrow l)$, $(\rightarrow r)$, (Πl) or (Πr). Let σ be an instantiation for ψ.

DEFINITION 6.3.2 (RELATIONS ON $\mathcal{F}_\psi$)

The following binary relations are defined on $\mathcal{F}_\psi$:

1. $R <_\psi R'$ if and only if a side formula of R is the principal formula of R'.[4]

2. $R \ll_\psi R'$ if and only if R occurs below[5] R' in ψ.

3. $R \sqsubset_\sigma R'$ if and only if the universal variable or eigenvariable introduced by R is a free variable of $\alpha\sigma$, where α is the universal variable introduced by R'. □

Notice that $\ll_\psi$ decomposes into sixteen subrelations: $\ll_\psi^{\Xi,\Omega} \subseteq \mathcal{F}_\psi(\Xi) \times \mathcal{F}_\psi(\Omega)$ for Ξ, Ω amongst $(\rightarrow l)$, $(\rightarrow r)$, (Πl) and (Πr). The relation $\ll_\psi$ and its subrelations are called the *skeletal orderings* of the derivation ψ. Notice also that $<_\psi$ is a subrelation of $\ll_\psi$.

---

[3]That is, individual *occurrences* of rules in ψ.

[4]More accurately: if and only if a side formula of R is a subformula [Kl68] or "descendent" of the principal formula of R': this is defined just as in [Kl68] or [Gi89] for first-order terms: the *immediate subformulae* of $A \rightarrow B$ are A and B and the *immediate subformulae* of $\Pi x : A \,.\, B$ are (the normal forms of) the $B[M/x]$, for suitable M. We distinguish the "occurrences" of a formula in a derivation [Kl68].

[5]That is, nearer to the endsequent (§4.2.1).

DEFINITION 6.3.3 (REDUCTION ORDERING)

The *reduction ordering* $\lhd_{\psi,\sigma}$ induced by an **R**-derivation ψ and a instantiation σ is defined by:

$$\lhd_{\psi,\sigma} \;=_{\text{def}}\; (<_{\psi} \cup \prec_{\psi} \cup \sqsubset_{\sigma})^{+},$$

where $+$ indicates transitive closure and the relation $\prec_{\psi}$ is defined by:

$$\prec_{\psi} \;=_{\text{def}}\; \ll_{\psi} \setminus \bigcup_{\Xi}(\ll_{\psi}^{\Pi l,\Xi} \cup \ll_{\psi}^{\Xi,\Pi l});$$

Ξ ranges over the operational rules of **R**. □

The presence of a relation in $\lhd_{\psi,\sigma}$ indicates that that relationship between specific inferences may not be altered by permutation. Consequently, the definition of $\prec_{\psi}$ fixes the relative positions of all rule applications in ψ *except* (Πl) rule applications (since they are removed from $\ll_{\psi}$ to form $\prec_{\psi}$).

DEFINITION 6.3.4 (COMPATIBILITY AND DEGREE)

1. (Compatibility) A derivation is said to be *compatible* with an instantiation just in case the reduction ordering induced is irreflexive.

2. (Degree) The *degree* of a compatible derivation and an instantiation is the number of pairs of inferences in the derivation whose skeletal order is inconsistent with the reduction ordering. That is, $\langle R, R' \rangle$ for which both $R \ll_{\psi} R'$ (R is below R' in ψ) and $R' \lhd_{\psi,\sigma} R$. If a derivation is compatible with an instantiation with degree n, we say it is *n-compatible.* □

THEOREM 6.3.5 (PERMUTATION THEOREM)

If ψ is compatible with σ, then there is a 0-compatible **R**-derivation ψ^* of the same endsequent. Moreover, if ψ is closed under σ, so is ψ^*. We say that ψ^* is a *permutation* of ψ.

PROOF

By induction on the degree of ψ. We interchange (Πl) inferences with other inferences to reduce the degree. One such case was given as an example at the start of this section ((Πl) over (Πr)). One must check that the leaves of the permuted derivation contain the same declarations (though in a different order).

In order to simplify the presentation of the argument we distinguish two cases:

1. In which we consider the exchange of (Πl) rule with the binary rule $(\rightarrow l)$;

2. In which we consider the exchange of the (Πl) rule with each of the unary rules $(\rightarrow r)$, (Πr) and (Πl) itself.

The first case divides into two subcases:

- R' is $(\rightarrow l)$ and R is (Πl). Here we must assume that the (Πl) occurs on one of the branches of the proof, and then add an instance of (Πl), with the same existential variable, to the other branch immediately below the $(\rightarrow l)$. This addition introduces no new inconsistencies and preserves closure. If we did not introduce this copy of the inference, the reordering of these two rules would introduce many new inconsistencies, thereby destroying our inductive argument. Consequently, we may assume that, locally, the original derivation tree is of the form:

$$\dfrac{\dfrac{\Gamma, x{:}C(\alpha) \Rightarrow_\Sigma D}{\Gamma \Rightarrow_\Sigma D} \qquad \dfrac{\Gamma, y{:}E, x{:}C(\alpha) \Rightarrow_\Sigma A}{\Gamma, y{:}E \Rightarrow_\Sigma A}}{\Gamma \Rightarrow_\Sigma A},$$

  where there is some $@ : D \rightarrow E \in \Sigma \cup \Gamma$ and some $@' : \Pi z{:}B\,.\,C(z) \in \Sigma \cup \Gamma$. (The substitution ordering, $\sqsubset_\sigma$, ensures that $@' \neq y$.) Locally, the reordered derivation is of the form:

$$\dfrac{\dfrac{\Gamma, x{:}C(\alpha) \Rightarrow_\Sigma D \qquad \Gamma, x{:}C(\alpha), y{:}E \Rightarrow_\Sigma D}{\Gamma, x{:}C(\alpha) \Rightarrow_\Sigma A}}{\Gamma \Rightarrow_\Sigma A}.$$

  We must show that if each branch of the original derivation leads to a closed leaf under σ, then each branch of the reordered derivation leads to a closed leaf under σ. It is immediate that this holds for the left hand branch of

the derivation, since the reordering leaves the bottom, left hand sequent unchanged. As for the right hand branch, we note that the components of the context of the bottom, right hand sequent are altered only in order. Therefore the right hand branch of the reordered derivation is closed under σ. Since $\sqsubset_\sigma$ is a subrelation of $\lhd_{\psi,\sigma}$ and because $\lhd_{\psi,\sigma}$ is irreflexive, y is not a subterm of $C(\alpha)$. Therefore the reordering of the context is consistent with the reordered derivation. Therefore the degree of the reordered derivation is one less than that of the original derivation;

- R' is (Πl) and R is $(\rightarrow l)$. Locally, the original derivation is of the form:

$$\dfrac{\dfrac{\Gamma, x{:}C(\alpha) \Rightarrow_\Sigma D \qquad \Gamma, x{:}C(\alpha), y{:}E \Rightarrow_\Sigma D}{\Gamma, x{:}C(\alpha) \Rightarrow_\Sigma A}}{\Gamma \Rightarrow_\Sigma A}\,.$$

where there is some $@ : D \rightarrow E \in \Sigma \cup \Gamma$ and some $@' : \Pi z{:}B\,.\,C(z) \in \Sigma \cup \Gamma$. (The substitution ordering, $\sqsubset_\sigma$, ensures that $@' \neq y$.) Locally, the reordered derivation is of the form:

$$\dfrac{\dfrac{\Gamma, x{:}C(\alpha) \Rightarrow_\Sigma D}{\Gamma \Rightarrow_\Sigma D} \qquad \dfrac{\Gamma, y{:}E, x{:}C(\alpha) \Rightarrow_\Sigma A}{\Gamma, y{:}E \Rightarrow_\Sigma A}}{\Gamma \Rightarrow_\Sigma A}\,,$$

We must show that if each branch of the original derivation leads to a closed leaf under σ, then each branch of the reordered derivation leads to a closed leaf under σ. This follows by an argument similar to the one given above.

The exchange of (Πl) with the unary operators divides into five subcases (we give the details of the proof in just two cases):

- R' is $(\rightarrow r)$ and R is (Πl). Locally, the original derivation is of the form:

$$\dfrac{\dfrac{\Gamma, y{:}D(\alpha), x{:}A \Rightarrow_\Sigma B}{\Gamma, y{:}D(\alpha) \Rightarrow_\Sigma A \rightarrow B}}{\Gamma \Rightarrow_\Sigma A \rightarrow B}\,,$$

where there is some $@ : \Pi z : C\,.\,D(z) \in \Sigma \cup \Gamma$. Locally, the reordered derivation is of the form:

$$\dfrac{\dfrac{\Gamma, x{:}A, y{:}D(\alpha) \Rightarrow_\Sigma B}{\Gamma, x{:}A \Rightarrow_\Sigma B}}{\Gamma \Rightarrow_\Sigma A \rightarrow B}\,.$$

We note that components of the bottom sequent are altered only in order, so that if the original derivation is closed under σ, then so is the reordered derivation. Since $\sqsubset_\sigma$ is a subrelation of $\lhd_{\psi,\sigma}$ and because $\lhd_{\psi,\sigma}$ is irreflexive, y is not a subterm of A. Therefore the reordering of the context is consistent with the reordered derivation. Therefore the degree of the reordered derivation is one less than that of the original derivation;

- R' is (Πr) and R is (Πl). We omit the argument;
- R' is (Πl) and R is $(\rightarrow r)$. We omit the argument;
- R' is (Πl) and R is (Πr). We omit the argument;
- R' is (Πl) and R is (Πl). Locally, the original derivation is of the form:

$$\dfrac{\dfrac{\Gamma, x{:}C(\alpha), y{:}D(\beta) \Rightarrow_\Sigma A}{\Gamma, x{:}C(\alpha) \Rightarrow_\Sigma A}}{\Gamma \Rightarrow_\Sigma A},$$

for suitable instances of (Πl). Locally, the reordered derivation, which has degree 1 less than that of the original derivation, is of the form:

$$\dfrac{\dfrac{\Gamma, y{:}D(\beta), x{:}C(\alpha) \Rightarrow_\Sigma A}{\Gamma, y{:}D(\beta) \Rightarrow_\Sigma A}}{\Gamma \Rightarrow_\Sigma A}.$$

Once again we note that the components of the bottom sequent have altered only in order, so that if the original derivation is closed under σ, then so is the reordered derivation. Since $\sqsubset_\sigma$ is a subrelation of $\lhd_{\psi,\sigma}$ and because $\lhd_{\psi,\sigma}$ is irreflexive, x is not a subterm of $D(\beta)$. Therefore the reordering of the context is consistent with the reordered derivation. Therefore the degree of the reordered derivation is one less than that of the original derivation.

This completes the proof. □

We have as a corollary to the construction performed in the proof of the Permutation Theorem:

COROLLARY 6.3.6 (PRESERVATION OF $\lhd_{\psi,\sigma}$)

$\lhd_{\psi^*,\sigma} = \lhd_{\psi,\sigma}$.

PROOF

The only relationships changed in the permutation are those excluded from $\lhd_{\psi,\sigma}$.

□

The following lemma shows that the 0-compatibility of a derivation and instantiation is a necessary condition for the well-typing of the latter in the former.

LEMMA 6.3.7 (0-COMPATIBILITY YIELDS WELL-TYPING)

If σ is well-typed in ψ then ψ is 0-compatible with σ.

PROOF

An inconsistency between the skeletal ordering of ψ and the reduction ordering arises from an inconsistency between $\sqsubset_\sigma$ and the skeletal ordering. Since $\sqsubset_\sigma \subseteq \mathcal{F}_\psi(\Xi) \times \mathcal{F}_\psi(\Pi l)$, it must arise from a (Πl) inference, introducing α say, being nearer the root of the derivation than a (Πl) or (Πr) inference that gives rise to a variable, v say, free in $\alpha\sigma$. But then v cannot be declared in the typing context of α since it is introduced above α, and therefore σ is not well-typed, contradicting our hypothesis. Therefore there can be no inconsistencies and ψ is 0-compatible with σ. □

We give a simple example of an **R**-derivation and a closure instantiation which fails to be well-typed in the given derivation, but which is well-typed in a reordering of that derivation.

Let $\Sigma \equiv_{\text{def}} A : \text{Type}, B : A \to \text{Type}, C : \text{Type}, p : A \to \text{Type}, f : B(a) \to B(a), a : A$ and let $\Gamma \equiv_{\text{def}} x_1 : \Pi x_2 : A \,.\, \Pi x_3 : B(x_2) \,.\, p x_3$.

We search for a proof of the endsequent $\Gamma \Rightarrow_\Sigma \Pi x_4 : B(a) \,.\, C \to p(f x_4)$. Consider the following **R**-derivation, ψ :

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, x_6 : \Pi x_3 : B(\alpha) \,.\, p x_3, x_7 : p\beta, x_4 : B(a), x_5 : C \Rightarrow_\Sigma p(f x_4)}{\Gamma, x_6 : \Pi x_3 : B(\alpha) \,.\, p x_3, x_7 : p\beta, x_4 : B(a) \Rightarrow_\Sigma C \to p(f x_4)}\,(\to r)}{\Gamma, x_6 : \Pi x_3 : B(\alpha) \,.\, p x_3, x_7 : p\beta \Rightarrow_\Sigma \Pi x_4 : B(a) \,.\, C \to p(f x_4)}\,(\Pi r)}{\Gamma, x_6 : \Pi x_3 : B(\alpha) \,.\, p x_3 \Rightarrow_\Sigma \Pi x_4 : B(a) \,.\, C \to p(f x_4)}\,(\Pi l)}{\Gamma \Rightarrow_\Sigma \Pi x_4 : B(a) \,.\, C \to p(f x_4)}\,(\Pi l)$$

The leaf sequent is closed by the instantiation $\sigma \equiv \langle\langle a, \alpha\rangle, \langle fx_4, \beta\rangle\rangle$, but fx_4 is not well-typed in $\Gamma, x_6 : \Pi x_3 : B(a) \,.\, Px_3$. However, if we reorder the rules in the derivation so that the (Πl)s are used after the $(\rightarrow r)$ and (Πr) we obtain the derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, x_4 : B(a), x_5 : C, x_6 : \Pi x_3 : B(\alpha) \,.\, px_3, x_7 : p\beta \Rightarrow_\Sigma p(fx_4)}{\Gamma, x_4 : B(a), x_5 : C, x_6 : \Pi x_3 : B(\alpha) \,.\, px_3 \Rightarrow_\Sigma p(fx_4)}\,(\Pi l)}{\Gamma, x_4 : B(a), x_5 : C \Rightarrow_\Sigma p(fx_4)}\,(\Pi l)}{\Gamma, x_4 : B(a) \Rightarrow_\Sigma C \rightarrow p(fx_4)}\,(\rightarrow r)}{\Gamma \Rightarrow_\Sigma \Pi x_4 : B(a) \,.\, C \rightarrow p(fx_4)}\,(\Pi r)$$

The leaf sequent of this derivation also is closed by the instantiation σ and fx_4 is well-typed in the context $\Gamma, x_4 : B(a), x_5 : C, x_6 : \Pi x_3 : B(a) \,.\, px_3$. It is a simple matter to verify that this reordering is consistent with the reduction ordering.

We stress that in general the situation is rather more complicated than is illustrated by the example given above; in general we must consider the migration of (Πl) rules over (Πl) rules as in the reordering of

$$\dfrac{\dfrac{\Gamma, x : C(\alpha), y : D(\beta) \Rightarrow_\Sigma A}{\Gamma, x : C(\alpha) \Rightarrow_\Sigma A}}{\Gamma \Rightarrow_\Sigma A}\,,$$

to yield

$$\dfrac{\dfrac{\Gamma, y : D(\beta), x : C(\alpha) \Rightarrow_\Sigma A}{\Gamma, y : D(\beta) \Rightarrow_\Sigma A}}{\Gamma \Rightarrow_\Sigma A}\,.$$

From a computational point of view testing for compatibility is a simple matter given a derivation and an instantiation: it is an acyclicity check in a directed graph. Compatibility is not, however, a *sufficient* test for well-typing: this arises because the (sub)formula ordering fails to exclude certain permutations which introduce non-well-formedness in the context. For example, suppose that for the (non-well-formed) sequent

$$\Gamma, z : \mathrm{NF}(B([M(y)/\alpha])), x : A, y : \mathrm{NF}(C[N/\beta]), \Delta \Rightarrow_\Sigma \mathrm{NF}(D[M(y)/\alpha, N/\beta])\,,$$

in which the type of N depends on x, compatiblity requires that we reorder the derivation (ψ, say) so that the component $y : \mathrm{NF}(C[N/\beta])$ is moved so that it precedes the component $z : \mathrm{NF}(B[M(y)/\alpha])$:

$$\Gamma, y : \mathrm{NF}(C[N/\beta]), z : \mathrm{NF}(B[M(y)/\alpha]), x : A, \Delta \Rightarrow_\Sigma \mathrm{NF}(D[M(y)/\alpha, N/\beta])\,.$$

The expression fails to be a well-formed context because x appears after y, the type of which depends on x. The (sub)formula ordering $<_\psi$ and the substitution ordering $\sqsubset_\sigma$ fail, in general, to prohibit this reordering. Thus we can achieve 0-compatiblity and fail to achieve well-typing.

The Permutation Theorem gives us the existence of 0-compatible derivations in which we might test for well-typing of the instantiation incrementally (*i.e.*, as it is found) but this involves repeatedly constructing permutations using the constructive proof of the theorem. This is inelegant and computationally expensive.

Another alternative would be to ignore well-typing until a closed, compatible derivation and instantiation have been found, and then utilize the permutation theorem once and check well-typing. We reject this option on the grounds that typing constraints reduce the search space of the unification algorithm drastically.

We develop instead a computationally tractable test on a derivation and instantiation that, if passed, guarantees the well-typing of the instantiation in all 0-compatible permutations of the derivation. Our ability to define such a notion is a corollary of the normalization (cut elimination) result for $\lambda\Pi$-calculus [HHP87] with its attendant (partial) subformula property, just as the results obtained in [Bi81] and [Wa89] for other logics rely on metatheorems of this sort.

Henceforth we treat contexts as ordered structures or DAGs[6] rather than sequences since the dependencies between declarations form such an order. Consequently the implicit union, denoted above by a comma, such as in "$\Gamma, x{:}A$", should be understood as an order preserving union of the order (DAG) Γ and the singleton order $x{:}A$. The latter will be higher in the resulting order than the declarations of the free variables in A, and incomparable with the other maximal elements of Γ. This assumption simplifies our discussion.

The reduction ordering $\lhd_{\psi,\sigma}$ is defined, for given ψ and σ, on pairs of inferences in the derivation ψ. In our subsequent development it will be helpful to consider it to be defined on (eigen- or universal) variables. Accordingly we shall write

---

[6]Directed Associative Graphs.

$w \lhd_{\psi,\sigma} v$ just in case $R(w) \lhd_{\psi,\sigma} R(v)$, where we write $R(u)$ for the inference with which the variable u is associated: u is associated with the inference R just in case u is an eigen- or universal variable introduced to the derivation by R (considered as a reduction operator).

The following notions are introduced for an **R**-derivation ψ of a well-formed endsequent. We use u, v, w possibly subscripted to denote universal and eigenvariables of ψ. Let $T(v)$ denote the type of the variable v in ψ.

DEFINITION 6.3.8 (INTRINSIC TYPING CONTEXT)

The *intrinsic typing context* $I(v)$ for each *eigen- or universal* variable v of ψ is defined inductively on the structure of the endsequent as follows:

$$I(v) \ =_{\text{def}} \biguplus_{w \in \text{FV}(T(v))} (I(w), w{:}T(w)).$$

$\biguplus$ denotes order preserving union of orders; $\text{FV}(M)$ denotes the set of free variables of the term M. □

$I(v)$ is well-defined since the endsequent is well-formed. Indeed, we have:

LEMMA 6.3.9 (WELL-FORMEDNESS IN $I(v)$)

$I(v) \vdash_\Sigma T(v)$: Type.

PROOF

By construction and the well-formedness of the endsequent (see Chapter 2 for the notion of a well-formed context). □

Let ψ be compatible with the instantiation σ. We give an inductive definition of the intrinsic typing context and type of a variable of ψ under a compatible instantiation σ. The induction is on the (well-founded) reduction ordering $(\lhd_{\psi,\sigma})$ over the domain of σ.

DEFINITION 6.3.10 ($I_\sigma(v)$ AND $T_\sigma(v)$)

Base. For all $v \in \mathrm{FV}(\psi)$, define $I_\epsilon(v) = I(v)$ and $T_\epsilon(v) = T(v)$. (ϵ is the empty instantiation.)

Step. Given $v \in \mathrm{Dom}(\sigma)$, we assume that we have defined $I_\sigma(w)$ and $T_\sigma(w)$ for all $w \in \mathrm{Dom}(\sigma)$ such that $w \lhd_{\psi,\sigma} v$ (Inductive Hypothesis). Let $w_1, w_2, \ldots, w_n$ be an enumeration of those variables declared in $I_\epsilon(v)$. By definition of $\lhd_{\psi,\sigma}$ and $I(v)$, we have $w_i \lhd_{\psi,\sigma} v$ for $0 < i < n+1$. Define

$$\begin{aligned} D_0(v) &= I_\epsilon(v) \vdash_\Sigma T_\epsilon(v)\text{:Type} \\ D_{k+1}(v) &= \mathrm{CUT}(\ I_\sigma(w_k) \vdash_\Sigma w_k\sigma\text{:}T_\sigma(w_k)\ ,\ D_k(v)\ ), \qquad 0 \le k < n. \end{aligned}$$

If $D_n(v)$ is the assertion: $\Delta \vdash_\Sigma C$:Type, then define

$$\begin{aligned} I_\sigma(v) &=_{\mathrm{def}} \Delta \\ T_\sigma(v) &=_{\mathrm{def}} C. \quad \square \end{aligned}$$

The "CUT" operation in the above definition is determined by the admissible rule of transitivity (see Chapter 2). That is, $D_{k+1}(v)$ is defined in terms of $D_k(v)$ by the following inference figure:

$$\frac{I_\sigma(w_k) \vdash_\Sigma w_k\sigma\text{:}T_\sigma(w_k) \qquad D_k(v)}{D_{k+1}(v)}\ \mathrm{cut}^7.$$

The cut rule is being used to effect substitution of the values (under σ) of universal variables throughout the judgement starting from the "uninstantiated" intrinsic typing context and type. The definition is well-formed since the context of the left premiss of each cut is a subcontext of the right premiss. (This follows from the construction of $I(v)$.) The cut above then serves to eliminate the declaration $w_k\text{:}T_\sigma(w_k)$ from the context of $D_k(v)$, replacing w_k by $w_k\sigma$ throughout the rest of the assertion.

The enumeration taken is irrelevant since independent cuts commute. Consider

$$\frac{I_\sigma(u_2) \vdash_\Sigma u_2\sigma\text{:}T_\sigma(u_2) \qquad \dfrac{I_\sigma(u_1) \vdash_\Sigma u_1\sigma\text{:}T_\sigma(u_1) \qquad D_k(v)}{D_{k+1}(v)}}{D_{k+2}(v)}$$

---

[7] Actually, this is by a trivial variant of the cut rule.

and

$$\dfrac{I_\sigma(u_1) \vdash_\Sigma u_1\sigma{:}T_\sigma(u_1) \qquad \dfrac{I_\sigma(u_2) \vdash_\Sigma u_2\sigma{:}T_\sigma(u_2) \qquad D'_k(v)}{D'_{k+1}(v)}}{D'_{k+2}(v)}.$$

In the first derivation $w_k = u_1$ and $w_{k+1} = u_2$. In the second, $w_k = u_2$ and $w_{k+1} = u_1$. If u_1 and u_2 are assumed independent (*i.e.*, unrelated via $\lhd_{\psi,\sigma}$), we have $u_i \notin \mathrm{Dom}(I_\sigma(u_j))$, $i \neq j$. Hence substitution of the value $u_1\sigma$ for u_1 does not interfere with substitution of the value $u_2\sigma$ for u_2, and $D_{k+2} = D'_{k+2}$.

We can now state the desired well-typing condition for σ in ψ.

DEFINITION 6.3.11 (INTRINSIC WELL-TYPING)

σ is said to be *intrinsically well-typed* in ψ just in case for all universal variables α of ψ, we have: $I_\sigma(\alpha) \vdash_\Sigma \alpha\sigma{:}T_\sigma(\alpha)$. □

The importance of the definition is summarized by:

PROPOSITION 6.3.12 (INTRINSIC WELL-TYPING AND COMPATIBILITY)

If σ is intrinsically well-typed in ψ, then it is intrinsically well-typed in all compatible permutations of ψ. In particular it is well-typed in 0-compatible permutations.

PROOF

Reference to the definition will show that the intrinsic well-typing of σ in ψ does not depend on the (Πl) structure of ψ. (In fact we deliberately forbade such dependence by our definition of $\lhd_{\psi,\sigma}$.) Hence the conditions are unaffected by permutations allowed by the reduction ordering $\lhd_{\psi,\sigma}$, which is itself unaltered by permutation (Corollary 6.3.6). For a 0-compatible permutation ψ^* of ψ, the intrinsic typing context for a variable is a subcontext of the typing context in $\psi^*\sigma$. Since "Thinning" is admissible (Chapter 2), σ is well-typed in ψ^*. □

In a similar vein, we state without proof the following:

PROPOSITION 6.3.13 (COMPATIBILITY AND INTRINSIC WELL-TYPING)

If σ is well-typed in a 0-compatible derivation ψ, it is intrinsically well-typed in ψ. □

We can now define a computationally acceptable notion of **R**-proof.

DEFINITION 6.3.14 (**R**-PROOF)

An **R**-*proof* is a pair $\langle \psi, \sigma \rangle$ such that (1) ψ is closed under σ; (2) ψ is compatible with σ, and (3) σ is intrinsically well-typed in ψ. □

THEOREM 6.3.15 (**R**-PROVABLE IF AND ONLY IF **U**-PROVABLE)

For well-formed sequents $\Gamma \Rightarrow_\Sigma A$,

$$\mathbf{R} \text{ proves } \Gamma \Rightarrow_\Sigma A \quad \text{if and only if} \quad \mathbf{U} \text{ proves } \Gamma \Rightarrow_\Sigma A.$$

PROOF

Suppose $\langle \psi, \sigma \rangle$ is an **R**-proof of $\Gamma \Rightarrow_\Sigma A$. The Permutability Theorem gives us a permutation ψ^* of ψ, closed under (hypothesis 1), and compatible with (hypothesis 2), the instantiation σ. Hypothesis (3), via Proposition 6.3.12, ensures that σ is well-typed in ψ^*. Hence $\langle \psi^*, \sigma \rangle$ is an **U**-proof.

Conversely, let $\langle \psi, \sigma \rangle$ be a **U**-proof of $\Gamma \Rightarrow_\Sigma A$. By definition ψ is closed under σ and σ is well-typed in ψ. Compatibility follows from Lemma 6.3.7, and intrinsic well-typing from Proposition 6.3.13. □

The reduction ordering, as formulated, is insufficiently subtle to yield well-typing as a consequence of 0-compatibility. The notions of intrinsic typing context and intrinsic well-typing are necessary in order to correct this situation and indeed these constructions could be built in to the definitions of $<_\psi$ and $\sqsubset_\sigma$ in order to yield well-typing as a consequence of 0-compatibility. (We believe that our presentation is clearer.) Thus these constructions may be considered to be those required in order to modify $\lhd_{\psi,\sigma}$ from being simply "logical" to being "type-theoretic".

6.4 Some remarks

Instantiations are generated by a unification algorithm acting on putative axiom sequents. They are first checked for compatibility (occurs-check) and then for intrinsic well-typing. The incremental nature of intrinsic typing means that the unification algorithm can use the typing information to constrain its search. New values for previously uninstantiated variables may be used to eliminate those variables from the typing contexts of the remaining ones. No permutations need be calculated.

Some remarks on the use of unification to calculate instantiations are in order. The unification algorithm of Chapter 5 takes as input two typing assertions which have N-proofs. However, it can be used to calculate instantiations for the universal variables that occur in a leaf of a derivation ψ by assuming that their types are those required by ψ. The procedures of the algorithm then calculate objects in the normal way. This process is justifiable by the reordering and well-typing arguments of this chapter. For example, suppose that the sequent:

$$\Gamma_1, \Gamma_2, x{:}A(\alpha), \Delta \Rightarrow_\Sigma B(\beta)\,,$$

in which we assume that α and β are the only universal variables and in which we assume that α occurs only in the type-expression $T(\beta)$ and in B and that β occurs only in B, is closed under the ground instantiation σ, *i.e.* $\mathrm{NF}(A(\alpha)\sigma) \equiv \mathrm{NF}(B(\beta)\sigma)$. If σ is well-typed in the reordering of the derivation which yields the sequent:

$$\Gamma_1, x{:}A(\alpha), \Gamma_2, \Delta \Rightarrow_\Sigma B(\alpha, \beta)\,,$$

then we could have considered the input to unification to be the pair of sequents:

$$\Gamma_1, x_\alpha{:}T(\alpha), x{:}A[x_\alpha/\alpha], \Delta, x_\beta{:}T(\beta)[x_\alpha/\alpha] \Rightarrow_\Sigma B[x_\alpha/\alpha, x_\beta/\beta]$$

and

$$\Gamma_1, x_\alpha{:}T(\alpha), x{:}A[x_\alpha/\alpha], \Delta, x_\beta{:}T(\beta)[x_\alpha/\alpha] \Rightarrow_\Sigma A[x_\alpha/\alpha]\,,$$

where x_α and x_β are new (non-universal) variables. (Strictly speaking, unification requires inhabitation assertions as input, but since we know that these sequents are provable in **L**, then we know that there are objects, whose inhabitation of the succedent types is provable in **N**.) Then we have that unification has calculated a substitution ρ with domain context $\Gamma_1, x_\alpha : T(\alpha), x : A(x_\alpha), \Delta, x_\beta : T(\beta)[x_\alpha/\alpha]$ and range context $\Gamma_1, x : \mathrm{NF}(A(\alpha)\sigma), \Delta$ such that

$$\mathbf{L} \text{ proves } \Gamma_1, x : \mathrm{NF}(A(\alpha)\sigma), \Delta \Rightarrow_\Sigma \mathrm{NF}(A[x_\alpha/\alpha]\rho),$$

$$\mathbf{L} \text{ proves } \Gamma_1, x : \mathrm{NF}(A(\alpha)\sigma), \Delta \Rightarrow_\Sigma \mathrm{NF}(B[x_\alpha/\alpha, x_\beta/\beta]\rho)$$

and $\mathrm{NF}(A[x_\alpha/\alpha]\rho)$ and $\mathrm{NF}(B[x_\alpha/\alpha, x_\beta/\beta]\rho)$ have a common $\beta\eta$-reduct. (We constrain the unification algorithm to calculate substitutions for those variables which arise from non-universal variables (x_α and x_β) only.)

It is a straightforward but tedious argument to show that this use of the unification algorithm is correct. Of course such a use of unification may not return a ground instantiation. In such situations we attempt to find ground instantiations by performing further search (this point is discussed further in Chapter 7).

We have been somewhat cautious in this development and allowed only (Πl) rules to migrate. As a consequence the basic structure of a derivation is largely fixed. The next step is to remove the ordering constraints induced by the propositional structure of the logic, perhaps using unification here as was done in [Wa89] for first-order intuitionistic logic. The final result would be a *matrix method* in the style of Bibel [Bi81] or Andrews [An81].

6.5 The Resolution Rule LR2

The methods of this Chapter are applicable to the *resolution* rule of the calculus **LR2** of Chapter 4. We illustrate this by considering the calculus **LR2**, for types in clausal form, which replaces the $(\rightarrow l)$ and (Πl) rules by the rule:

$$\frac{\Gamma \Rightarrow_\Sigma E_1 \quad \ldots \quad \Gamma \Rightarrow_\Sigma E_n \quad \Gamma, x{:}E \Rightarrow_\Sigma F}{\Gamma \Rightarrow_\Sigma F},$$

where $@ : \Pi x_1 : B_1 \ldots \Pi x_m : B_m \,.\, C_1 \rightarrow (C_2 \rightarrow (\ldots (C_n \rightarrow D)\ldots)) \in \Sigma \cup \Gamma$, **LR2** proves $\Gamma \vdash_\Sigma M_i{:}B_i[M_1/x_1, \ldots, M_{i-1}/x_{i-1}]$ for $1 \leq i \leq m$, $C_j[M_1/x_1, \ldots, M_m/x_m] \rightarrow_{\beta\eta} E_j$ for $1 \leq j \leq n$, $x \notin \mathrm{Dom}(\Gamma)$ and $D[M_1/x_1, \ldots, M_m/x_m] \rightarrow_{\beta\eta} E$.

The corresponding metacalculus replaces this rule by the rule:

$$\frac{\Gamma \Rightarrow_\Sigma C_1(\overline{\alpha}) \ldots \Gamma \Rightarrow_\Sigma C_n(\overline{\alpha}) \quad \Gamma, x{:}D(\overline{\alpha}) \Rightarrow_\Sigma F}{\Gamma \Rightarrow_\Sigma F},$$

where $@ : \Pi x_1 : B_1 \ldots \Pi x_m : B_m \,.\, C_1 \rightarrow (C_2 \rightarrow (\ldots (C_n \rightarrow D)\ldots)) \in \Sigma \cup \Gamma$, $x \notin \mathrm{Dom}(\Gamma)$, $C_i(\overline{\alpha})$ denotes $C_i[\alpha_1/x_1, \ldots, \alpha_m/x_m]$ for $1 \leq i \leq n$, $D(\overline{\alpha})$ denotes $D[\alpha_1/x_1, \ldots, \alpha_m/x_m]$ and each α_i is distinct and not introduced elsewhere in the derivation.

Essentially, an application of the resolution rule of the calculus **LR2** combines several applications of the $(\rightarrow l)$ and (Πl) rules in a single rule, and consequently the reduction ordering for the metacalculus is defined in a manner that is similar to that for the metacalculus **U**.

6.6 Summary

In this chapter we have shown how to remove the non-deterministic choice of objects from uses, in proof-search, of the (Πl) rule as a reduction operator. We have extended the methods of Wallen [Wa89] and Bibel [Bi81] to the $\lambda\Pi$-calculus, and we have indicated that further work in this area might be to develop a matrix method for proof-search in this (and similar) languages.

Chapter 7

Computation and Its Semantics

7.1 Introduction

In Chapter 6 we considered the effective computation of proofs of ground endsequents. In this chaper we consider how the admission of universal variables (which we might now also call *logical variables* or *program variables*) into endsequents yields a simple notion of *logic programming*. We provide an operational semantics for this notion of logic programming.

We proceed to construct a denotational semantics for logic programs. The basic framework for this construction is the categorical models for the $\lambda\Pi$-calculus described in Appendix A, based on the work of Cartmell [Ca86].

There is a "standard pattern" for the construction of models of logic programs (*cf.* [Ll84], [Mi89]), which we follow, and which can be summarized as follows:

- Identify the general notion of model for the given logic;
- Construct a term model by performing a least fixed point construction, which is determined by computation steps, on a collection of *Herbrand interpretations* — in our case maps from $|C(\Sigma)|$ to $|\mathcal{F}|$ (see Appendix A);

Unfortunately, this least fixed point construction does not determine a model of the $\lambda\Pi$-calculus in the sense of Appendix A. However we observe that we can obtain a sensible denotational semantics for logic programs, in terms of the Yoneda functor, which generalizes the least fixed point construction and which is a model of the $\lambda\Pi$-calculus.

7.2 Logic Programs and Operational Semantics

7.2.1 Programs

We define non-ground endsequents — sequents which contain universal variables which were not introduced as the result of a (Πl) reduction in the calculus U. A universal variable that is introduced by a (Πl) comes along with some typing information, and it is possible to compute an intrinsic typing context for it. For universal variables that occur in an endsequent, we must supply both of these entities and *impose* a well-formedness condition. We replace the typing information with a *declaration.*

DEFINITION 7.2.1 (DECLARATION)

Let $\alpha_1, \ldots, \alpha_m$ be universal variables. A *declaration for the typing expressions of* $\alpha_1, \ldots, \alpha_m$ is an expression of the form $\alpha_1 : T(\alpha_1), \ldots, \alpha_m : T(\alpha_m)$. □

DEFINITION 7.2.2 (NON-GROUND SEQUENT)

A *non-ground sequent* is an ordered quintuple $\langle \Sigma, \delta, \Gamma, A, \mathcal{I} \rangle$, which we write as $(\delta, \mathcal{I}) . \Gamma \Rightarrow_\Sigma A$, where: Σ is a valid $\lambda\Pi$-signature; $\delta \equiv \alpha_1 : T(\alpha_1), \ldots, \alpha_m : T(\alpha_m)$ is a declaration of the typing expressions for the universal variables $\alpha_1, \ldots, \alpha_m$, upon which the context and type expressions $\Gamma \equiv x_1 : A_1, \ldots, x_n : A_n$ and A depend; and $\mathcal{I}$ is a set of intrinsic typing contexts, one for each of the α_i $(1 \leq i \leq m)$. $(\delta, \mathcal{I}) . \Gamma$ is said to be a *non-ground context.* □

DEFINITION 7.2.3 (WELL-FORMED NON-GROUND SEQUENT)

Let Σ be valid $\lambda\Pi$-signature and let $(\delta, \mathcal{I}) . \Gamma \Rightarrow_\Sigma A$ be a non-ground sequent. $(\delta, \mathcal{I}) . \Gamma \Rightarrow_\Sigma A$ is said to be *well-formed* just in case $I(v) \vdash_\Sigma T(v)$: Type for each variable v that occurs in $(\delta, \mathcal{I}) . \Gamma \Rightarrow_\Sigma A$. The expression $(\delta, \mathcal{I}) . \Gamma$ is said to be a *well-formed non-ground context.* □

We shall omit the $(\delta, \mathcal{I})$ prefix from sequents where such information is unambiguous. We refer to a well-formed non-ground context as a *program*; ground programs correspond therefore to (ground) contexts.

We summarize the logic programming interpretation of non-ground endsequents. Consider the well-formed non-ground sequent:

$$(\delta, \mathcal{I}) \,.\, \Gamma(\alpha_1, \ldots, \alpha_m) \Rightarrow_\Sigma A(\alpha_1, \ldots, \alpha_m) \,.$$

This sequent may be interpreted as a *logic program together with a query* in the following sense:

- Σ determines a *language* — in the LF, an encoded logic;
- $\Gamma(\alpha_1, \ldots, \alpha_m)$ determines a list of *program clauses* written in the language Σ — programs are either *ground* or *non-ground*;
- $A(\alpha_1, \ldots, \alpha_m)$ determines a *query* written in the language Σ.

The universal variables $\alpha_1, \ldots, \alpha_m$ are *program variables* or *logical variables*; these correspond to the logical variables of the programmming language PROLOG [Ko74], [CM84]. The sequent represents a request to compute an instantiation ρ for the universal variables of the sequent such that **L** proves $\Gamma\rho \Rightarrow_\Sigma A\rho$. Such an instantiation is an *answer instantiation.*

We give a program to calculate the *nth* Fibonacci number (see [Mi89]), where Σ is the LF signature of Peano arithmetic [HHP87]. The context Γ is:

$$\begin{aligned} &fib{:}\iota \to (\iota \to o), x{:}fib(0,0)\text{true}, y{:}fib(1,1)\text{true}, \\ &\quad z{:}\Pi N_1{:}\iota.\Pi N_2{:}\iota.\Pi F_1{:}\iota.\Pi F_2{:}\iota.\Pi F{:}\iota.\Pi N{:}\iota. \\ &\qquad = (N_1, N-1)\text{true} \to = (N_2, N-2)\text{true} \to = (F, F_1 + F_2)\text{true} \\ &\qquad\quad \to fib(N_1, F_1)\text{true} \to fib(N_2, F_2)\text{true} \to fib(N, F)\text{true} \end{aligned}$$

The query A is $fib(n, \alpha)$true where α is an universal variable. The resulting sequent $\Gamma \Rightarrow_\Sigma A$ is a request to compute an instantiation for α, the *nth* Fibonnacci number.

7.2.2 Operational Semantics

The operational semantics of Σ-$\lambda\Pi$-logic programs is provided by extending the search procedures of Chapter 6 to include the case of non-ground endsequents.

In fact we have very little work to do. Our syntactic treatment of universal variables for non-ground endsequents is, manifestly, independent of whether they are introduced during a derivation or are present in the endsequent.

THEOREM 7.2.4 (SOUNDNESS AND COMPLETENESS OF THE OPERATIONAL SEMANTICS)

Let Σ be a valid $\lambda\Pi$-signature and let $(\delta, \mathcal{I}) \,.\, \Gamma \Rightarrow_\Sigma A$ be a well-formed non-ground sequent. **R** proves $(\delta, \mathcal{I}) \,.\, \Gamma \Rightarrow_\Sigma A$ if and only if there is a ground instantiation σ of the universal variables of δ such that **R** proves $\Gamma\sigma \Rightarrow_\Sigma A\sigma$.

PROOF SKETCH

We remark that **R**-derivations are well-defined for well-formed non-ground endsequents and that the notions of closure, compatibility and intrinsic well-typing are well-defined for such derivations. Consequently, **R**-proofs are well-defined for such endsequents. Intrinsic well-typing for the endsequent ensures well-typing of σ. □

We remark that a mechanical implementation of a logic programming system should exploit the resolution calculus **LR1** of Chapter 4. The appropriate "reordering calculus" for this calculus can be considered to be given by a *matrix method.*[1] Such a method is beyond the scope of this thesis.

As an example of the evaluation of a program and a query consider the signature $\Sigma \equiv A : \text{Type}, B : A \to \text{Type}, C : \text{Type}, p : A \to \text{Type}, a : A, f : B(a) \to B(a)$ and context (ground program) $\Gamma \equiv x_1 : \Pi x_2 : A \,.\, \Pi x_3 : B(x_2) \,.\, p x_3$.

We search for a proof of the non-ground endsequent

$$(\delta, \mathcal{I}) \,.\, \Gamma \Rightarrow_\Sigma \Pi x_4 : B(a) \,.\, C \to p(f\alpha)\,,$$

[1] *Cf.* the last paragraph of Section 6.4.

where $\delta \equiv \alpha{:}B(a)$ and $\mathcal{I} = \{I(\alpha)\}$, where $I(\alpha) \equiv \Gamma$.

Consider the following **R**-derivation, ψ:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, x_4{:}B(a), x_5{:}C, x_6 : \Pi x_3{:}B(\beta)\,.\,px_3, x_7 : p\gamma \Rightarrow_\Sigma p(f\alpha)}{\Gamma, x_4{:}B(a), x_5{:}C, x_6 : \Pi x_3{:}B(\beta)\,.\,px_3 \Rightarrow_\Sigma p(f\alpha)}\,(\Pi l)}{\Gamma, x_4{:}B(a), x_5{:}C \Rightarrow_\Sigma p(f\alpha)}\,(\Pi l)}{\Gamma, x_4{:}B(a) \Rightarrow_\Sigma C \to p(f\alpha)}\,(\to r)}{\Gamma \Rightarrow_\Sigma \Pi x_4{:}B(a)\,.\,C \to p(f\alpha)}\,(\Pi r)\,.$$

The leaf sequent is closed by the instantiation $\langle\langle x_4, \alpha\rangle, \langle a, \beta\rangle, \langle fx_4, \gamma\rangle\rangle$, so that the answer instantiation for α is $\langle x_4, \alpha\rangle$.

Note that unification alone may not always be sufficient to calculate closure instantiations. For example, suppose that the non-ground endsequent

$$(\delta, \mathcal{I})\,.\,\Gamma, x{:}c\alpha, \Delta \Rightarrow_\Sigma c\beta$$

is well-formed. Unification produces a closure instantiation by the identification of α and β. In order to obtain a proof we must find a ground instantiation for α by solving the further subgoal $I(\alpha) \Rightarrow_\Sigma T(\alpha)$.

7.2.3 Some Remarks

- A weakness of this work is that we have ignored computation rules and termination criteria. We should like to develop such rules and criteria.

- Further, we should like to develop an *abstract interpreter* for our programs with a view to the mechanical implementation of these ideas.

- We remark that our work includes a notion of *modularity* which is similar to the notion of modularity introduced by Miller [Mi89]. We should like to develop this idea.

- The formulation of the $\lambda\Pi$-calculus used by Harper [Ha87] and Salvesen [Sa89] includes equality judgements. The extension of our techniques to such a system would yield a combination of logic programming and functional programming for the $\lambda\Pi$-calculus. This would require the extension of the

unification algorithm to handle equality judgements, *cf.* [Pl72]. We should like to develop this idea.

7.3 The Denotational and Model-theoretic Semantics of Σ-$\lambda\Pi$-logic Programs

7.3.1 Introduction

In this section we present the construction of a set-theoretic interpretation of Σ-$\lambda\Pi$-logic programs.

The "execution" of a program consists of the application of the rules of **L** considered as reduction operators, so the form of the $(\rightarrow r)$ and (Πr) rules implies that in order to determine whether $\Gamma \Rightarrow_\Sigma A$ it might be necessary to determine that $\Gamma, \Delta \Rightarrow_\Sigma B$ for some extension Δ of Γ. In semantical terms this means that the interpretation of a program Γ depends on the interpretation of larger programs. We solve this problem by constructing a single interpretation which interprets all (ground) programs simultaneously. This construction is effected by defining a satisfaction relation $\models_\Sigma$ for each signature Σ which resembles the forcing relation in the theory of Kripke models [Tr73]. Our construction is similar to that of Miller [Mi89].

We show that, subject to a suitable notion of forcing, our construction can be characterized in model-theoretic terms via the Yoneda functor. However, this characterization requires that we abandon the value category $\mathcal{F}$ in favour of the presheaf topos $\mathcal{S}^{C(\Sigma)}$, where $\mathcal{S}$ is the category of (small) sets and $C(\Sigma)$ is the syntactic category given in Appendix A.

7.3.2 Construction of the Interpretation

We define the *Herbrand universe* for a signature Σ in the usual way. Here it is the set of all morphisms of the form $\langle x_1, \ldots, x_m, M \rangle$ which realize $\Gamma, x:A$ in terms of Γ and extension $\Gamma, x:A$ of Γ which are well-formed in Σ. Informally, this is the set of all objects M which can be constructed in some context Γ over the signature Σ.

DEFINITION 7.3.1 (HERBRAND UNIVERSE)

Let Σ be a valid $\lambda\Pi$-signature and let N proves $\vdash_\Sigma \Gamma$ context. The *Herbrand universe of* Σ, $\mathcal{H}_\Sigma$ is defined by:

$$\mathcal{H}_\Sigma =_{\text{def}} \bigcup_{\{\Gamma | \mathbf{N} \text{ proves } \vdash_\Sigma \Gamma \text{ context}\}} \bigcup_{\{A | \mathbf{N} \text{ proves } \Gamma \vdash_\Sigma A: \text{Type}\}} \{\sigma \in \hom_{C(\Sigma)}(\Gamma, \Gamma, x:A) | \sigma \circ p(\Gamma, x:A) = 1_\Gamma\},$$

where 1_Γ denotes the identity morphism on Γ, *cf.* [Ma71]. □

A *Herbrand interpretation* is, essentially, just a mapping which assigns to each valid context Γ over a signature Σ a subset of the Herbrand universe. Formally, it is a mapping from the objects of $C(\Sigma)$ to the objects of $\mathcal{F}$: this additional structure indicates the extent to which the denotational semantics that we construct is a model of the $\lambda\Pi$-calculus: it reflects the fact that a Herbrand interpretation can be considered to be an ordered family of mappings indexed by contexts in which the ordering is that given by the tree structure of $C(\Sigma)$. The ordering encodes the sense in which the interpretation of a given context includes interpretations of its subcontexts, *e.g.* an interpretation of $\Gamma, x:A$ includes an interpretation of Γ. However, as we shall see in Definition 7.3.2 and in Section 7.3.3, it is not possible to define Herbrand interpretations on the objects and maps of $C(\Sigma)$ (substitutions) so that the denotational semantics that we construct is an $\mathcal{F}$-valued Σ-$\lambda\Pi$-*model.*

DEFINITION 7.3.2 (Σ-$\lambda\Pi$-HERBRAND INTERPRETATION)

Let Σ be a valid $\lambda\Pi$-signature. A Σ-$\lambda\Pi$-*Herbrand interpretation* is a mapping $I : C(\Sigma) \longrightarrow \mathcal{F}$ such that:

- $I(\langle\rangle) \subseteq \bigcup_{\{A | \mathbf{N} \text{ proves } \vdash_\Sigma A:\text{Type}\}} \{\sigma \in \hom_{C(\Sigma)}(\langle\rangle, x:A)\} (\subseteq \mathcal{H}_\Sigma)$;
- For all $\langle\rangle \blacktriangleleft \Gamma \in |C(\Sigma)|$, if $I(\Gamma) = \langle I(\Delta), F_I\rangle$ where $\Delta \triangleleft \Gamma$, then $F_I(w) \subseteq \mathcal{H}_\Sigma$ for all $w \in \text{DEN}(I(\Delta))$, and $l_{C(\Sigma)}(\Gamma) = l_{\mathcal{F}}(\Gamma)$, where $l_{C(\Sigma)}$ is the level function in $C(\Sigma)$ and $l_{\mathcal{F}}$ is the level function in $\mathcal{F}$. □

Note that in order for it to be possible to extend the definition of a Σ-$\lambda\Pi$-Herbrand interpretation to be a Σ-$\lambda\Pi$-Herbrand model one of the things that we should have

had to require is that $I(\langle\rangle) =_{\text{def}} \epsilon$. We reject such a condition on the grounds that it fails to admit the construction of a Σ-$\lambda\Pi$-Herbrand interpretation that represents the computations that a program (context) can successfully perform *with respect to the underlying signature*. Although we could avoid this difficulty by working with a formulation of the $\lambda\Pi$-calculus without signatures [HHP89], [Sa90], the problem that arises in Section 7.3.3 is more fundamental.

The collection of Herbrand interpretations forms a lattice under the following definitions of partial order, join and meet.

DEFINITION 7.3.3 (LATTICE OF HERBRAND INTERPRETATIONS)

Let Σ be a valid $\lambda\Pi$-signature and let I_1 and I_2 be Σ-$\lambda\Pi$-Herbrand interpretations.

- We define the partial order $\sqsubseteq$ by: $I_1 \sqsubseteq I_2$ if and only if for all $\Gamma \in |C(\Sigma)|$ $\text{DEN}(I_1(\Gamma)) \subseteq \text{DEN}(I_2(\Gamma))$.

- We define the join of I_1 and I_2 inductively by:

  - $(I_1 \sqcup I_2)(\langle\rangle) =_{\text{def}} I_1(\langle\rangle) \cup I_2(\langle\rangle)$;
  - For all $\epsilon \blacktriangleleft \Gamma \in |C(\Sigma)|$, $(I_1 \sqcup I_2)(\Gamma) =_{\text{def}} \langle (I_1 \sqcup I_2)(\Delta), F_{(I_1 \sqcup I_2)} \rangle$, where $\Delta \triangleleft \Gamma$ and $F_{(I_1 \sqcup I_2)}(w) =_{\text{def}} F_{I_1}(w) \cup F_{I_2}(w)$ for all $w \in \text{DEN}((I_1 \sqcup I_2)(\Delta))$.

- We define the meet of I_1 and I_2 inductively by:

  - $(I_1 \sqcap I_2)(\langle\rangle) =_{\text{def}} I_1(\langle\rangle) \cap I_2(\langle\rangle)$;
  - For all $\epsilon \blacktriangleleft \Gamma \in |C(\Sigma)|$, $(I_1 \sqcap I_2)(\Gamma) =_{\text{def}} \langle (I_1 \sqcap I_2)(\Delta), F_{(I_1 \sqcap I_2)} \rangle$, where $\Delta \triangleleft \Gamma$ and $F_{(I_1 \sqcap I_2)}(w) =_{\text{def}} F_{I_1}(w) \cap F_{I_2}(w)$ for all $w \in \text{DEN}((I_1 \sqcap I_2)(\Delta))$.

There is a bottom interpretation which assigns, essentially, the empty set to all contexts.

DEFINITION 7.3.4 (BOTTOM INTERPRETATION)

Let Σ be a valid $\lambda\Pi$-signature and let **N** proves $\vdash_\Sigma \Gamma$ context. The *bottom* Σ-$\lambda\Pi$-*Herbrand interpretation* $\perp$ is a bottom Σ-$\lambda\Pi$-Herbrand interpretation that is defined inductively by:

- $\perp(\langle\rangle) =_{\text{def}} \emptyset$;
- For all $\epsilon \blacktriangleleft \Gamma \in |C(\Sigma)|$, $\perp(\Gamma) =_{\text{def}} \langle \perp(\Delta), F_\perp \rangle$ where $F_\perp(w) =_{\text{def}} \emptyset$ for all $w \in \text{DEN}(\perp(\Delta))$. □

Under these definitions the collection of Herbrand interpretations over a signature Σ forms a complete lattice.

LEMMA 7.3.5 (COMPLETE LATTICE)

Let Σ be a valid Σ-$\lambda\Pi$-signature. The lattice of Σ-$\lambda\Pi$-Herbrand interpretations is complete.

PROOF

This is an immediate consequence of the fact that the lattice of subsets of the powerset of the union over valid contexts of Herbrand universes is complete. □

We define a satisfaction predicate $\models_\Sigma$: such a predicate is essential for our least fixed point construction. In this construction the contexts Γ can be considered to be possible worlds, the ordering on these worlds being that determined by the tree structure of $C(\Sigma)$, and a Herbrand interpretation considered to be an ordered collection of interpretations indexed by contexts. Consequently, the assertion I , $\Gamma \models_\Sigma M : A$ is to be read as "the inhabitation of the type A by the object M is satisfiable in the interpretation I at the context Γ". Henceforth we adopt the notation $[\![\Gamma]\!]^I$ for $I(\Gamma)$.

DEFINITION 7.3.6 (SATISFACTION)

Let Σ be a valid $\lambda\Pi$-signature and let I be a Σ-$\lambda\Pi$-Herbrand interpretation. We define the satisfaction predicate $\models_\Sigma$ as follows:

- If $[\![\Gamma]\!]^I = \epsilon$ then $I\ ,\ \Gamma \models_\Sigma M : A$ if and only if there is a substitution $\sigma \equiv \langle x_1, \ldots, x_m, M\rangle \in \hom_{C(\Sigma)}(\Gamma, \Gamma, x{:}A)$;

- $I\ ,\ \Gamma \models_\Sigma M : A$ $(A \equiv cM_1 \ldots M_m)$ if and only if there is some pair $\langle\langle u_1, u_2\rangle, \sigma\rangle \in \mathrm{DEN}([\![\Gamma]\!]^I)$ such that $\sigma \equiv \langle x_1, \ldots, x_m, M\rangle \in \hom_{C(\Sigma)}(\Gamma, \Gamma, x{:}A)$;

- $I\ ,\ \Gamma \models_\Sigma M{:}A \to B$ if and only if *either* there is some pair $\langle\langle u_1, u_2\rangle, \sigma\rangle \in \mathrm{DEN}([\![\Gamma]\!]^I)$ such that $\sigma \equiv \langle x_1, \ldots, x_m, M\rangle \in \hom_{C(\Sigma)}(\Gamma, \Gamma, x : A \to B)$, *or*

$$I\ ,\ \Gamma, x{:}A \models_\Sigma N{:}B\,,$$

where $M \equiv \lambda x{:}A\,.\,N$;

- $I\ ,\ \Gamma \models_\Sigma M{:}\Pi x{:}A\,.\,B$ if and only if *either* there is some pair $\langle\langle u_1, u_2\rangle, \sigma\rangle \in \mathrm{DEN}([\![\Gamma]\!]^I)$ such that $\sigma \equiv \langle x_1, \ldots, x_m, M\rangle \in \hom_{C(\Sigma)}(\Gamma, \Gamma, y : \Pi x{:}A\,.\,B)$, *or*

$$I\ ,\ \Gamma, x{:}A \models_\Sigma N{:}B\,,$$

where $M \equiv \lambda x{:}A\,.\,N$. □

The last two clauses Definition 7.3.6 ensure that the extension of programs during execution due to the form of the $(\to r)$ and (Πr) rules is captured by the satisfaction relation, *cf.* [Mi89].

We wish to build a single Herbrand interpretation I such that $I\ ,\ \Gamma \models_\Sigma M : A$ if and only if G\cut proves $\Gamma \vdash_\Sigma M : A$. Such an interpretation arises as the least fixed point of a mapping T_Σ, given below, from Herbrand interpretations to Herbrand interpretations.

There is a strong computational motivation for the definition of T_Σ. Roughly, T_Σ applied to the Herbrand interpretation I takes the collection of (realizers of) all of the types which are satisfied by I and adds to this collection all of the

(realizers of) the types which can be obtained by one further inference of **L**. The inhabiting (extract-)object encodes the computation of each type from the given context. Thus T_Σ may be considered to provide a denotational semantics for the computations that a context or program Γ can perform. Note however that T_Σ encodes no information about the search strategy employed for the execution of a program — it is simply assumed to be complete. Schmidt [Sch86] provides a denotational semantics for a propositional version of PROLOG which interprets the backtracking mechanism.

DEFINITION 7.3.7 (THE OPERATOR T_Σ)

Let Σ be a valid $\lambda\Pi$-signature and let N proves $\vdash_\Sigma \Gamma$ context. We assume that $\Gamma \equiv x_1 : A_m, \ldots, x_m : A_m$. Let $[\Sigma, \Gamma] =_{\text{def}} \{\sigma \mid \sigma \equiv \langle x_1, \ldots, x_m, @\rangle : \Gamma \longrightarrow \Gamma, x : A\}$, where $@ : A \in \Sigma \cup \Gamma$. Let I be a Σ-$\lambda\Pi$-Herbrand interpretation.

$T_\Sigma(I)$ is a Herbrand interpretation which is defined inductively on valid contexts by:

- $[\![\langle\rangle]\!]^{T_\Sigma(I)} =_{\text{def}} [\Sigma, \langle\rangle]$;
- $[\![\Gamma]\!]^{T_\Sigma(I)} =_{\text{def}} \langle [\![\Delta]\!]^{T_\Sigma(I)}, F_{T_\Sigma(I)} \rangle$, where $\Delta \lhd \Gamma$ and where
  $F_{T_\Sigma(I)}(w) =_{\text{def}} F_I(w) \cup \{\sigma \mid \sigma \in [\Sigma, \Gamma]\}$
  $\cup \{\sigma \mid \sigma \equiv \langle x_1, \ldots, x_m, P\rangle : \Gamma \longrightarrow \Gamma, y : A$
  where
  - $@ : B \to C \in \Sigma \cup \Gamma$;
  - $I\ , \Gamma \models_\Sigma N : B$;
  - $I\ , \Gamma, x : C \models_\Sigma Q : A$; and
  - $Q(@N) \to_{\beta\eta} P\ \}$

  $\cup \{\sigma \mid \sigma \equiv \langle x_1, \ldots, x_m, P\rangle : \Gamma \longrightarrow \Gamma, y : A$
  where
  - $@ : \Pi x : B\,.\,C \in \Sigma \cup \Gamma$;
  - $I\ , \Gamma \models_\Sigma N : B$;
  - $Q(@N) \to_{\beta\eta} P$; and
  - $I\ , \Gamma, y : D \models_\Sigma Q : A$ where $C[N/x] \to_{\beta\eta} D\ \}$.

□

The reader is referred to [Mi89] for a similar construction in the setting of a sublanguage of intuitionistic first-order logic.

In order to obtain the least fixed point of T_Σ we must prove several properties of the complete lattice of Herbrand interpretations and T_Σ, *cf.* [Mi89], [Ll84].

LEMMA 7.3.8 (T_Σ PRESERVES HERBRAND INTERPRETATIONS)

Let Σ be a valid $\lambda\Pi$-signature. If I is a Σ-$\lambda\Pi$-Herbrand interpretation, then so is $T_\Sigma(I)$.

PROOF

This is an immediate consequence of the definition of T_Σ. □

LEMMA 7.3.9 (SATISFACTION RESPECTS $\sqsubseteq$)

Let Σ be a valid Σ-$\lambda\Pi$-signature and let I_1 and I_2 be Σ-$\lambda\Pi$-Herbrand interpretations such that $I_1 \sqsubseteq I_2$. If $I_1\ ,\ \Gamma \models_\Sigma M : A$ then $I_2\ ,\ \Gamma \models_\Sigma M : A$.

PROOF

The proof proceeds by induction on the structure of A.

If $A \equiv cM_1 \ldots M_m$ and if $I_1\ ,\ \Gamma \models_\Sigma M : A$ then there is some pair $\langle u, \sigma \rangle \in \mathrm{DEN}([\![\Gamma]\!]^{I_1})$ such that $\sigma \equiv \langle x_1, \ldots, x_m, M \rangle \in \hom_{C(\Sigma)}(\Gamma, \Gamma, x : A)$. But by hypothesis we have that $\mathrm{DEN}([\![\Gamma]\!]^{I_1}) \sqsubseteq \mathrm{DEN}([\![\Gamma]\!]^{I_2})$ and therefore $I_2\ ,\ \Gamma \models_\Sigma M : A$.

If $A \equiv B \to C$ and if $I_1\ ,\ \Gamma \models_\Sigma M : B \to C$ then by the definition of the predicate $\models_\Sigma$ we have either that there is some suitable $\langle u, \sigma \rangle \in \mathrm{DEN}([\![\Gamma]\!]^{I_1})$, in which case by an argument similar to that for the previous case we have that $I_2\ ,\ \Gamma \models_\Sigma M : A$; or that $I_1\ ,\ \Gamma, x : B \models_\Sigma N : C$, where $M \equiv \lambda x : B . N$ in which case by the induction hypothesis we have that $I_2\ ,\ \Gamma, x : B \models_\Sigma N : C$ and so by the definition of the predicate $\models_\Sigma$ we have $I_2\ ,\ \Gamma \models_\Sigma M : B \to C$.

If $A \equiv \Pi x : B . C$ then the argument is similar to that for the previous case. This completes the proof. □

LEMMA 7.3.10 (SATISFACTION AT FINITE LEVEL)

Let Σ be a valid Σ-$\lambda\Pi$-signature and let $I_1 \sqsubseteq I_2 \sqsubseteq I_3 \sqsubseteq \ldots$ be a sequence of Σ-$\lambda\Pi$-Herbrand interpretations. Suppose that

$$\bigsqcup_{i=1}^{\infty} I_i\ ,\ \Gamma \models_\Sigma M : A .$$

Then there exists $k \geq 1$ such that

$$I_k\,,\,\Gamma \models_\Sigma M{:}A\,.$$

PROOF

The proof proceeds by induction on the structure of the type A. Suppose we have that $\bigsqcup_{i=1}^{\infty} I_i\,,\,\Gamma \models_\Sigma M{:}A$.

If $A \equiv cM_1 \ldots M_m$ then there is some pair $\langle u, \sigma\rangle \in \mathrm{DEN}([\![\Gamma]\!]^{\bigsqcup_{i=1}^{\infty} I_i})$ such that $\sigma \equiv \langle x_1, \ldots, x_m, M\rangle \in \hom_{C(\Sigma)}(\Gamma, \Gamma, x : A)$. But $F_{\bigsqcup_{i=1}^{\infty} I_i}(w) = \bigcup_{i=1}^{\infty} F_{I_i}(w)$, for each w, so that there exists k such that $I_k\,,\,\Gamma \models_\Sigma M{:}A$.

If $A \equiv B \to C$ then by the definition of the predicate $\models_\Sigma$ we have either that there is some pair $\langle u, \sigma\rangle \in \mathrm{DEN}([\![\Gamma]\!]^{\bigsqcup_{i=1}^{\infty} I_i})$ such that $\sigma \equiv \langle x_1, \ldots, x_m, M\rangle \in \hom_{C(\Sigma)}(\Gamma\,,\,\Gamma, x : A \to B)$, in which case by an argument similar to that for the previous case we have that there exists k such that $I_k\,,\,\Gamma \models_\Sigma M : A$; or that $\bigsqcup_{i=1}^{\infty} I_i\,,\,\Gamma, x : B \models_\Sigma N : C$, where $M \equiv \lambda x : B\,.\,N$. By the induction hypothesis we know that there exists a k such that $I_k\,,\,\Gamma, x : B \models_\Sigma N : C$, so that by the definition of the predicate $\models_\Sigma$ we have that $I_k\,,\,\Gamma \models_\Sigma M : B \to C$.

If $A \equiv \Pi x{:}B\,.\,C$ then the argument is similar to that for the previous case. □

LEMMA 7.3.11 (MONOTONICITY OF T_Σ)

The operator T_Σ is monotonic. Let Σ be a valid Σ-$\lambda\Pi$-signature and let I_1 and I_2 Σ-$\lambda\Pi$-Herbrand interpretations. If $I_1 \sqsubseteq I_2$, then $T_\Sigma(I_1) \sqsubseteq T_\Sigma(I_2)$.

PROOF

Suppose that $I_1 \sqsubseteq I_2$. We proceed by induction on the length of Γ.

If $\Gamma \equiv \langle\rangle$ then we are done. Otherwise, suppose that $\langle u, \sigma\rangle \in \mathrm{DEN}([\![\Gamma]\!]^{T_\Sigma(I_1)})$, where $u \in \mathrm{DEN}([\![\Delta]\!]^{T_\Sigma(I_1)})$ and $\sigma \in F_{T_\Sigma(I_1)}(u)$ where $\Delta \lhd \Gamma$. By the induction hypothesis we have that $u \in \mathrm{DEN}([\![\Delta]\!]^{T_\Sigma(I_2)})$. We must now proceed by case analysis on σ.

If $\sigma \in [\Sigma, \Gamma]$ then we have immediately that $\langle u, \sigma \rangle \in \mathrm{DEN}([\![\Gamma]\!]^{T_\Sigma(I_2)})$, since $I_1 \sqsubseteq I_2$.

If $\sigma \equiv \langle x_1, \ldots, x_m, P \rangle \in \hom_{C(\Sigma)}(\Gamma, \Gamma, x:A)$, where there is some $@ \in B \to C \in \Sigma \cup \Gamma$ such that $I_1\ , \Gamma \models_\Sigma N{:}B$, $I_1\ , \Gamma, y{:}C \models_\Sigma Q{:}A$ and $Q(@N) \to_{\beta\eta} P$, by Lemma 7.3.9 we then have that $I_2\ , \Gamma \models_\Sigma N{:}B$ and $I_2\ , \Gamma, y{:}C \models_\Sigma Q{:}A$, and so $\langle u, \sigma \rangle \in \mathrm{DEN}([\![\Gamma]\!]^{T_\Sigma(I_2)})$.

If σ arises via some $@ : \Pi x{:}A\,.\,C \in \Sigma \cup \Gamma$ then the argument is similar to that given in the previous case. This completes the proof. □

LEMMA 7.3.12 (CONTINUITY OF T_Σ)

The operator T_Σ is continuous. Let Σ be a valid Σ-$\lambda\Pi$-signature and let $I_1 \sqsubseteq I_2 \sqsubseteq I_3 \sqsubseteq \ldots$ be a sequence of Σ-$\lambda\Pi$-Herbrand interpretations, then

$$\bigsqcup_{i=1}^{\infty} T_\Sigma(I_i) = T_\Sigma(\bigsqcup_{i=1}^{\infty} I_i).$$

PROOF

We prove this equality by proving inclusion in two directions.

Since $I_j \sqsubseteq \bigsqcup_{i=1}^{\infty} I_i$ for all $j \geq 1$, by Lemma 7.3.11 we obtain $T_\Sigma(I_j) \sqsubseteq T_\Sigma(\bigsqcup_{i=1}^{\infty} I_i)$ for all $j \geq 1$. But j is arbitrary, so $\bigsqcup_{j=1}^{\infty} T_\Sigma(I_j) \sqsubseteq T_\Sigma(\bigsqcup_{i=1}^{\infty} I_i)$.

To prove the converse inclusion we must show that

$$\mathrm{DEN}([\![\Gamma]\!]^{T_\Sigma(\bigsqcup_{i=1}^{\infty} I_i)}) \subseteq \mathrm{DEN}([\![\Gamma]\!]^{\bigsqcup_{i=1}^{\infty} T_\Sigma(I_i)})$$

for all Γ such that $\mathbf{N}$ proves $\vdash_\Sigma \Gamma$ context. We proceed by induction on the length of Γ.

If $\Gamma \equiv \langle\rangle$ then we are done. Otherwise, suppose that $\langle u, \sigma \rangle \in \mathrm{DEN}([\![\Gamma]\!]^{T_\Sigma(\bigsqcup_{i=1}^{\infty} I_i)})$. where $u \in \mathrm{DEN}([\![\Delta]\!]^{T_\Sigma(\bigsqcup_{i=1}^{\infty} I_i)})$ and $\sigma \in F_{T_\Sigma(\bigsqcup_{i=1}^{\infty} I_i)}(u)$, where $\Delta \lhd \Gamma$. By the induction hypothesis we have that $u \in \mathrm{DEN}([\![\Delta]\!]^{\bigsqcup_{i=1}^{\infty} T_\Sigma(I_i)})$. We must now proceed by case analysis on σ.

If $\sigma \in [\Sigma, \Gamma]$ then we have immediately that $\langle u, \sigma \rangle \in \mathrm{DEN}([\![\Gamma]\!]^{\bigsqcup_{i=1}^{\infty} T_\Sigma(I_i)})$.

If $\sigma \in \hom_{C(\Sigma)}(\Gamma, \Gamma, x{:}A)$, where there is some $@ \in B \to C \in \Sigma \cup \Gamma$ such that $\bigsqcup_{i=1}^{\infty} I_i$, $\Gamma \models_\Sigma N{:}B$ and $\bigsqcup_{i=1}^{\infty} I_i$, $\Gamma, y{:}C \models_\Sigma Q{:}A$, then we have by Lemma 7.3.10 that there is a k such that I_k , $\Gamma \models_\Sigma N{:}B$ and I_k , $\Gamma, y{:}C \models_\Sigma Q{:}A$. Therefore we have that $\langle u, \sigma \rangle \in \mathrm{DEN}([\![\Gamma]\!]^{T_\Sigma(I_k)})$, and since $F_{\bigsqcup_{i=1}^{\infty} T_\Sigma(I_i)}(w) = \bigcup_{i=1}^{\infty} F_{T_\Sigma(I_i)}$, we obtain $\langle u, \sigma \rangle \in \mathrm{DEN}([\![\Gamma]\!]^{\bigsqcup_{i=1}^{\infty} T_\Sigma(I_i)})$.

If σ arises via some $@ : \Pi x{:}A\,.\,C \in \Sigma \cup \Gamma$ then the argument is similar to that given in the previous case. This completes the proof. □

The continuity of T_Σ yields the least fixed point of T_Σ as the supremum of the ordinal powers of T_Σ applied to the bottom interpretation, $\bot$.

LEMMA 7.3.13 (LEAST FIXED POINT OF T_Σ)

Let Σ be a valid $\lambda\Pi$-signature. The least fixed point of the operator T_Σ exists and is defined by:

$$T_\Sigma^\omega(I_\bot) =_{\mathrm{def}} \bigsqcup_{i=1}^{\infty} T_\Sigma^i(I_\bot)\,.$$

PROOF

By the Knaster-Tarski fixed point theorem, [Ta55], [AvE82]. □

We are now able to prove the soundness and completeness of **G**\cut (and consequently of **L**) with respect to the interpretation T_Σ^ω. We remark that although the well-formedness conditions are stated with respect to the system **N**, by Theorems 3.2.6 and 3.2.7, no difficulties arise.

THEOREM 7.3.14 (SOUNDNESS)

Let Σ be a valid $\lambda\Pi$-signature and let **N** proves $\Gamma \vdash_\Sigma A{:}\,\mathrm{Type}$. We have that if **G**\cut proves $\Gamma \vdash_\Sigma M{:}A$ then $T_\Sigma^\omega(\bot)$, $\Gamma \models_\Sigma M{:}A$.

PROOF

The proof is by induction on the structure of **G**\cut-proofs.

If $\Gamma \vdash_\Sigma M : A$ is an axiom sequent then there is a morphism

$$\sigma \equiv \langle x_1, \ldots, x_m, @_i \rangle : \Gamma \longrightarrow \Gamma, x{:}A \in [\Sigma, \Gamma]\,,$$

where $@ \in \Sigma \cup \Gamma$, and so there is some pair $\langle u, \sigma \rangle \in \mathrm{DEN}([\![\Gamma]\!]^{T^1_\Sigma(\bot)}) \subseteq \mathrm{DEN}([\![\Gamma]\!]^{T^\omega_\Sigma(\bot)})$.

Suppose the last rule of G\cut applied is the $(\to r)$ rule:

$$\frac{\Gamma \vdash_\Sigma A\colon \mathrm{Type} \qquad \Gamma, x{:}A \vdash_\Sigma N{:}B}{\Gamma \vdash_\Sigma \lambda x{:}A\,.\,N : A \to B}\ (x \notin \mathrm{FV}(B))\,.$$

By the induction hypothesis we have that $T^\omega_\Sigma(\bot)\ ,\ \Gamma, x{:}A \models_\Sigma N{:}B$, and so by the definition of the predicate $\models_\Sigma$ we have that $T^\omega_\Sigma(\bot)\ ,\ \Gamma \models_\Sigma \lambda x{:}A\,.\,N : A \to B$.

Suppose the last rule of G\cut applied is the (Πr) rule:

$$\frac{\Gamma \vdash_\Sigma A\colon \mathrm{Type} \qquad \Gamma, x{:}A \vdash_\Sigma N{:}B}{\Gamma \vdash_\Sigma \lambda x{:}A\,.\,N : \Pi x{:}A\,.\,B}\,.$$

By the induction hypothesis we have that $T^\omega_\Sigma(\bot)\ ,\ \Gamma, x{:}A \models_\Sigma N{:}B$, and so by the definition of the predicate $\models_\Sigma$ we have that $T^\omega_\Sigma(\bot)\ ,\ \Gamma \models_\Sigma \lambda x{:}A\,.\,N : \Pi x{:}A\,.\,B$.

Suppose the last rule of G\cut applied is the $(\to l)$ rule:

$$\frac{\Gamma \vdash_\Sigma N{:}B \qquad \Gamma, x{:}C \vdash_\Sigma Q{:}A}{\Gamma \vdash_\Sigma Q(@N){:}A}\ (@ : B \to C \in \Sigma \cup \Gamma, x \notin \mathrm{FV}(A))\,.$$

By the induction hypothesis we have that $T^\omega_\Sigma(\bot)\ ,\ \Gamma \models_\Sigma N{:}B$ and that $T^\omega_\Sigma(\bot)\ ,\ \Gamma, x{:}B \models_\Sigma Q{:}A$, and so by the definitions of T_Σ and $\models_\Sigma$ we have that

$$T_\Sigma(T^\omega_\Sigma)(\bot)\ ,\ \Gamma \models_\Sigma Q(@N){:}A\,.$$

But T^ω_Σ is a fixed point, so that we obtain $T^\omega_\Sigma(\bot)\ ,\ \Gamma \models_\Sigma Q(@N){:}A$.

The case in which the last rule of G\cut applied is (Πl) is similar to the previous argument.

Suppose the last rule of G\cut applied is the equality rule:

$$\frac{\Gamma \vdash_\Sigma M{:}A \quad \Gamma \vdash_\Sigma A\colon \mathrm{Type} \qquad A =_{\beta\eta} A'}{\Gamma \vdash_\Sigma M{:}A'}\,.$$

If $T^\omega_\Sigma(\bot)\ ,\ \Gamma \models_\Sigma N : A$ then it is immediate from the definitions of T_Σ and the predicate $\models_\Sigma$ that $T^\omega_\Sigma(\bot)\ ,\ \Gamma \models_\Sigma M{:}A'$. This completes the proof. □

THEOREM 7.3.15 (COMPLETENESS)

Let Σ be a valid $\lambda\Pi$-signature and suppose that **N** proves $\Gamma \vdash_\Sigma A\colon \mathrm{Type}$. If $T^\omega_\Sigma(\bot)\ , \Gamma \models_\Sigma M\colon A$ then **G\cut** proves $\Gamma \vdash_\Sigma M\colon A$.

PROOF

Let Γ, M and A be such that $T^k_\Sigma(\bot)\ , \Gamma \models_\Sigma M : A$ for some positive integer k, and assume that this k is the smallest such k. If A contains $n \geq 0$ occurences of $\rightarrow$ and Π attach to the ordered pair $\langle \Gamma, A\rangle$ the ordinal measure $\omega \cdot (k-1) + n$.

We prove by induction on this measure that for all contexts Γ and types A if the measure of $\langle \Gamma, A\rangle$ is α then **G\cut** proves $\Gamma \vdash_\Sigma M\colon A$.

The base of the induction is where $\alpha = 0\,(= \omega \cdot 0 + 0)$, so it must be that $T^1_\Sigma(\bot)\ , \Gamma \models_\Sigma M : A$ and A is atomic: then there is some $\langle x_1, \ldots, x_m, @\rangle \in [\Sigma, \Gamma]$, in which case it is immediate that **G\cut** proves $\Gamma \vdash_\Sigma @\colon A$.

The induction step divides into the two cases where $\omega \cdot \alpha + \beta$ is a successor ordinal and where $\omega \cdot \alpha + \beta$ is a limit ordinal.

$\omega \cdot \alpha + \beta$ is a limit ordinal just in case $\beta = 0$ and $\alpha > 0$, so that we have $T^{\alpha+1}_\Sigma(\bot)\ , \Gamma \models_\Sigma M\colon A$ and A is atomic. By the definition of the operator T_Σ there are three possibilities:

1. There is some $\langle x_1, \ldots, x_m, @\rangle \in [\Sigma, \Gamma]$, in which case it is immediate that **G\cut** proves $\Gamma \vdash_\Sigma M\colon A$;

2. There is some $@ : B \rightarrow C \in \Sigma \cup \Gamma$, with $T^\alpha_\Sigma(\bot)\ , \Gamma \models_\Sigma N\colon B$ and $T^\alpha_\Sigma(\bot)\ , \Gamma, x\colon C \models_\Sigma Q\colon A$. By the induction hypothesis we have that **G\cut** proves $\Gamma \vdash_\Sigma N\colon B$ and **G\cut** proves $\Gamma, x\colon C \vdash_\Sigma Q\colon A$, so that by the $(\rightarrow l)$ rule we obtain **G\cut** proves $\Gamma \vdash_\Sigma Q(@N)\colon A$;

3. There is some $@ : \Pi x : B.C \in \Sigma \cup \Gamma$, with $T^\alpha_\Sigma(\bot)\ , \Gamma \models_\Sigma N : B$ and $T^\alpha_\Sigma(\bot)\ , \Gamma, y\colon D \models_\Sigma Q(@N)\colon A$, where $C[N/x] \rightarrow_{\beta\eta} D$. By the induction hypothesis we have that **G\cut** proves $\Gamma \vdash_\Sigma N : B$ and **G\cut** proves

$\Gamma, y : D \vdash_\Sigma Q(@N) : A$, so that by the (Πl) rule we obtain **G**\cut proves $\Gamma \vdash_\Sigma Q(@N) : A$.

$\omega \cdot \alpha + \beta$ is a successor ordinal just in case $\beta > 0$ in which case we have that $T_\Sigma^{\alpha+1}(\bot)$, $\Gamma \models_\Sigma M : A$ and A is not atomic. There are two possibilities, one for each of the possible top-level connectives of A.

If $A \equiv B \to C$ then by the definition of the predicate $\models_\Sigma$ we have either that there is some pair $\langle u, \sigma \rangle \in \mathrm{DEN}([\![\Gamma]\!]^{T^{\alpha+1}(\bot)})$ such that $\sigma \equiv \langle x_1, \ldots, x_m, M \rangle \in \hom_{\mathcal{C}(\Sigma)}(\Gamma, \Gamma, x : B \to C)$, in which case it follows from Proposition A.4.7 that **G**\cut proves $\Gamma \vdash_\Sigma M : B \to C$; or that $T_\Sigma^{\alpha+1}(\bot)$, $\Gamma, x : B \models_\Sigma N : C$, where $M \equiv \lambda x : B \,.\, N$. The type C has fewer connectives than the type $B \to C$, so by the induction hypothesis we have that **G**\cut proves $\Gamma, x : B \vdash_\Sigma N : C$. Therefore by the $(\to r)$ rule we obtain **G**\cut proves $\Gamma \vdash_\Sigma \lambda x : B \,.\, N : B \to C$.

If $A \equiv \Pi x : B \,.\, C$ then by the definition of the predicate $\models_\Sigma$ we have either that there is some pair $\langle u, \sigma \rangle \in \mathrm{DEN}([\![\Gamma]\!]^{T^{\alpha+1}(\bot)})$ such that $\sigma \equiv \langle x_1, \ldots, x_m, M \rangle \in \hom_{\mathcal{C}(\Sigma)}(\Gamma, \Gamma, y : \Pi x : B \,.\, C)$, in which case it follows from Proposition A.4.7 that **G**\cut proves $\Gamma \vdash_\Sigma M : \Pi x : B \,.\, C$; or that $T_\Sigma^{\alpha+1}(\bot)$, $\Gamma, x : B \models_\Sigma N : C$, where $M \equiv \lambda x : B \,.\, N$. The type C has fewer connectives than the type $\Pi x : B \,.\, C$, so by the induction hypothesis we have that **G**\cut proves $\Gamma, x : B \vdash_\Sigma N : C$. Therefore by the (Πr) rule we obtain **G**\cut proves $\Gamma \vdash_\Sigma \lambda x : B \,.\, N : \Pi x : B \,.\, C$.

It remains only to note that if $T_\Sigma^{\omega}(\bot)$, $\Gamma \models_\Sigma M : A$ then by Lemma 7.3.10 there exists k such that $T_\Sigma^{k}(\bot)$, $\Gamma \models_\Sigma M : A$, so that each pair $\langle \Gamma, A \rangle$ such that $T_\Sigma^{\omega}(\bot)$, $\Gamma \models_\Sigma M : A$ for some M has a measure. This completes the proof. □

7.3.3 Model-theoretic Characterization of the Semantics

It is unfortunate that it is not possible to extend the definition of the interpretation $T_\Sigma^{\omega}(\bot)$ to substitutions so as to yield an $\mathcal{F}$-valued Σ-$\lambda\Pi$-model.[2] To

[2]See also the remarks after Definition 7.3.2.

see this suppose that $\sigma : \Gamma \longrightarrow \Delta$ is a substitution in $C(\Sigma)$. $[\![\sigma]\!]^{T^{\omega}_{\Sigma}(\perp)}$ would have to be a function from $\mathrm{DEN}([\![\Gamma]\!]^{T^{\omega}_{\Sigma}(\perp)})$ to $\mathrm{DEN}([\![\Delta]\!]^{T^{\omega}_{\Sigma}(\perp)})$. Suppose that $\langle t, u\rangle \in \mathrm{DEN}([\![\Gamma]\!]^{T^{\omega}_{\Sigma}(\perp)})$. $[\![\sigma]\!]^{T^{\omega}_{\Sigma}(\perp)}(\langle t, u\rangle)$ must be a pair $\langle v, w\rangle \in \mathrm{DEN}([\![\Delta]\!]^{T^{\omega}_{\Sigma}(\perp)})$. u is of the form $\langle x_1, \ldots, x_m, M\rangle : \Gamma \longrightarrow \Gamma, x{:}A$ and w is of the form $\langle y_1, \ldots, y_n, N\rangle : \Delta \longrightarrow \Delta, y{:}B$, so that we must require that $M = N\sigma$. However, it is easy to see that not all M which occur via some such u arise in this way. The problem is that in order for such a function to be well-defined $T^{\omega}_{\Sigma}(\perp)$ would have to contravariant, and the category $\mathcal{F}^{\mathrm{op}}$ is not a contextual category with products.

However, all is not lost. Consider the Yoneda functor, which is a full embedding[3], $\mathrm{Y} : D \longrightarrow [D^{\mathrm{op}}, \mathcal{S}]$, where D is a small category, which can be defined by

- $r \mapsto \hom_D(r, -) : D^{\mathrm{op}} \longrightarrow \mathcal{S}$, for objects r of D, and
- $f : r \longrightarrow s \mapsto \hom_D(s, -) \longrightarrow \hom_D(r, -)$, for maps f of D.[4]

It is a straightforward matter to prove that if D is a contextual category with products then so is $\mathrm{Y}D$, the contexual and product structure being inherited from D. The reader is referred to [Be88] for a discussion of the (co)completeness of functor categories.

Applying the Yoneda functor to the (small) category $C(\Sigma)$ we obtain a set-theoretic contextual category with products $[C(\Sigma)^{\mathrm{op}}, \mathcal{S}]$. It follows that Y is a $[C(\Sigma)^{\mathrm{op}}, \mathcal{S}]$-valued Σ-$\lambda\Pi$-model. Furthermore, if $\langle\langle u_1, u_2\rangle, v\rangle \in \mathrm{DEN}([\![\Gamma]\!]^{T^{\omega}_{\Sigma}(\perp)})$ then $v \in \mathrm{Y}(\Gamma)(\Gamma, x{:}A)$ and $u_2 \in \mathrm{Y}(\Delta)(\Delta, y{:}B)$ for some A and B, where $\Delta \triangleleft \Gamma$, *etc.*.

Of course $\mathrm{Y}(\Gamma)(\Theta)$ also contains maps that are not present in $\mathrm{DEN}([\![\Gamma]\!]^{T^{\omega}_{\Sigma}(\perp)})$. However, these maps are simply of the form $\rho : \Gamma \longrightarrow \Theta$ and are such that if $\mathbf{G}\backslash\mathrm{cut}\ \Theta \vdash_{\Sigma} N{:}B$ then $\mathbf{G}\backslash\mathrm{cut}\ \Gamma \vdash_{\Sigma} \mathrm{NF}(N\rho){:}\mathrm{NF}(B\rho)$. Therefore these maps may

---

[3]A functor is said to be a *full embedding* if it is full, faithful and injective on objects [LS86].

[4]Here we consider the hom-functor $\hom_D(-,-)$ to be of the form $D \times D^{\mathrm{op}} \longrightarrow \mathcal{S}$, *etc.*.

be considered to model the fact that Γ can perform all of the computations that Θ can perform. Indeed, if we define a satisfaction relation $\Vdash$ by $\mathrm{Y}, \Gamma \Vdash_\Sigma P : C$ if and only if there is some $\sigma \in \mathrm{Y}(\Gamma)(\Delta)$ such that $\mathrm{Y}, \Delta \models\!\!\models_\Sigma Q : D$ where $Q\sigma \to_{\beta\eta} P$ and $D\sigma \to_{\beta\eta} C$, where the predicate $\models\!\!\models_\Sigma$ is defined by:

- $\mathrm{Y}\ , \Delta \models\!\!\models_\Sigma M : A (\equiv cM_1 \ldots M_m)$ if and only if there is a map $\langle x_1, \ldots, x_m, M \rangle \in \mathrm{Y}(\Delta)(\Delta, x : A)$;

- $\mathrm{Y}\ , \Delta \models\!\!\models_\Sigma M : A \to B$ if and only if *either* there is a map $\langle x_1, \ldots, x_m, M \rangle \in \mathrm{Y}(\Delta)(\Delta, x : A \to B)$, *or*

$$\mathrm{Y}\ , \Delta, x : A \models\!\!\models_\Sigma N : B\,,$$

  where $M \equiv \lambda x : A\,.\,N$;

- $\mathrm{Y}\ , \Delta \models\!\!\models_\Sigma M : \Pi x : A\,.\,B$ if and only if *either* there is a map $\langle x_1, \ldots, x_m, M \rangle \in \mathrm{Y}(\Delta)(\Delta, y : \Pi x : A\,.\,B)$, *or*

$$\mathrm{Y}\ , \Delta, x : A \models\!\!\models_\Sigma N : B\,,$$

  where $M \equiv \lambda x : A\,.\,N$;

then we obtain the result that, for $\beta\eta$-normal forms,

$$\mathbf{G}\backslash\mathrm{cut} \text{ proves } \Gamma \vdash_\Sigma M : A \text{ iff } \mathrm{Y}/=_{\beta\eta}, \Gamma \Vdash_\Sigma M : A\,,$$

where $\mathrm{Y}/=_{\beta\eta}$ is the quotient of Y by $\beta\eta$-equality defined by

$$(\mathrm{Y}/=_{\beta\eta})(\Gamma)(\Delta) =_{\mathrm{def}} \{\sigma \in \mathrm{Y}(\Gamma)(\Delta) |\ \sigma \text{ is in } \beta\eta\text{-normal form }\}.$$

For an exposition of the relationship between the Yoneda functor, *geometric theories*[5] and *generic models* in the setting of topos theory the reader is referred to [Be88] and [Jo77].

---

[5]There is a sense in which contexts are a generalization of the notion of *geometric theory*, but this is beyond the scope of this thesis.

7.3.4 The Semantics of Non-ground Programs

So far we have only given a semantics for ground programs.

The difficulty in giving a semantics to non-ground programs is that non-ground contexts are not objects of the category $C(\Sigma)$. However, a solution is found by noting that such a context $\Gamma(\alpha_1, \ldots, \alpha_m)$ represents a set of objects of $C(\Sigma)$, namely the set of (well-formed) contexts which are instantiations[6] of $\Gamma(\alpha_1, \ldots, \alpha_m)$. Thus we define the interpretation of a non-ground program to be the union of the interpretations of these instantiations.

DEFINITION 7.3.16 (SEMANTICS OF NON-GROUND PROGRAMS)

Let Σ be a valid $\lambda\Pi$-signature, let $(\delta, \mathcal{I}) \,.\, \Gamma$ be a program and let I be the set of all instantiations σ of $\alpha_1, \ldots, \alpha_m$ which are well-typed in some **L**-derivation in which (the normal form of) $\Gamma(\alpha_1, \ldots, \alpha_m)\sigma$ is the context of the endsequent.

1. The interpretation of the program $(\delta, \mathcal{I}) \,.\, \Gamma(\alpha_1, \ldots, \alpha_m)$, as determined by $T_\Sigma^\omega(\bot)$, is given by:

$$[\![(\delta, \mathcal{I}) \,.\, \Gamma(\overline{\alpha})]\!]^{T_\Sigma^\omega(\bot)} =_{\text{def}} \bigcup_{\sigma \in \mathrm{I}} \mathrm{DEN}([\![\mathrm{NF}(\Gamma(\overline{\alpha})\sigma)]\!]^{T_\Sigma^\omega(\bot)}),$$

   in our usual abbreviated notation;

2. The interpretation of the program $(\delta, \mathcal{I}) \,.\, \Gamma(\alpha_1, \ldots, \alpha_m)$, as determined by Y, is given by:

$$[\![(\delta, \mathcal{I}) \,.\, \Gamma(\overline{\alpha})]\!]^{\mathrm{Y}} =_{\text{def}} \bigcup_{\sigma \in \mathrm{I}} \bigcup_{\Delta} \mathrm{Y}(\mathrm{NF}(\Gamma(\overline{\alpha})\sigma))(\Delta),$$

   in our usual abbreviated notation. □

---

[6] Recall that *instantiations* are defined in Chapter 6.

7.4 Perpetual Processes

Lloyd [Ll84] defines a *perpetual process* to be: " ... a program which does not terminate and yet which is doing useful computation, in some sense."

Lloyd proceeds to give a semantics to non-terminating computations performed by (classical first-order, Horn clause) logic programs[7] by defining the notion of *complete Herbrand interpretation.* Complete Herbrand interpretations are Herbrand interpretations which take values in the *complete Herbrand universe*, the complete Herbrand universe of a program being obtained from the Herbrand universe by taking the topological completion of the Herbrand universe under the *term ultrametric* [Ll84]. We expect that it will be possible to extend these ideas to our setting.

7.5 Summary

In this chapter we have extended the work of Chapter 6 to provide an operational semantics for an appropriate notion of logic programming. We have provided a semantics for such logic programs in the categorical framework for the model theory of the $\lambda\Pi$-calculus which is discussed in Appendix A. We have discussed some possible further developments of this work.

[7]Lloyd considers "pure" PROLOG programs.

Chapter 8

Conclusions and Further Work

We have developed the proof theory of the $\lambda\Pi$-calculus by considering a proof system that resembles a Gentzen-style sequent calculus [Ge34], and proved a cut elimination theorem for normal forms. This extends the work of Gentzen [Ge34] to the $\lambda\Pi$-calculus and is related to the work of Martin-Löf for the simply-typed λ-calculus [ML71a] [ML71b] and to the work of Howard [How80]. Further work suggested by this work includes the extension of these results to appropriate fragments of Generalized Type Systems [Ba89].

We have provided a theory of search and logic programming for the $\lambda\Pi$-calculus, and hence for any logic which is adequately encoded in the LF.

At the heart of any theory of logic programming there must lie a theory of search. We first developed such a theory for the $\lambda\Pi$-calculus. Search procedures in logical systems are inherently non-deterministic and the principal difficulty in developing such a theory — indeed the main reason for the theory to exist — is the desire to reduce this non-determinism as much as possible. To this end three steps in our development were significant:

- The formulation of a sequent calculus of types: the form of the (Πl) rule is such that in search it introduces much less non-determinism than the (ΠE) rule of the (linearized) natural deduction system;

- The development of a unification algorithm for the $\lambda\Pi$-calculus. Much further work is suggested by our work on unification:

- The identification of possibly decidable subproblems *e.g.* first-order terms;
- The extension of our existing work to consider more exotic type systems, *e.g.* the Calculus of Constructions and other systems in Barendregt's cube of λ-calculi, Generalized Type Systems, recursive types, *etc.*.
- Improvements to the efficiency of the existing algorithm, in particular, the elimination of redundancy, *cf.* [Hu75];
- The study of unification in an algebraic setting (*i.e.* in the category of Generalized Algebraic Theories, **GAT**), in the manner of Rydeheard and Stell [RS87];
- The extension of the algorithm to handle equality judgements [Ha87];
- The development of a proof-theoretic characterization of unification;

- The extension of the work on proof-reordering and the "occurs check" of Bibel [Bi81] and Wallen [Wa87] to the $\lambda\Pi$-calculus, thereby allowing the use of unification in search to eliminate non-deterministic term choices in a manner which identifies classes of derivations which are essentially equivalent in terms of their closure conditions. Further work suggested by this falls into three groups:
  - Extensions to different (more expressive) type systems. Calculi in which universal variables have internal characterizations ?
  - The development of matrix methods [Wa87] for the $\lambda\Pi$-calculus.
  - The study of the search spaces that are induced for the LF's object logics by search at the level of the framework.

This work provides the necessary theory of search. Further improvements were made by exploiting a clausal form for hypothetico-general judgements in order to obtain a notion of resolution, and by considering the notion of uniform proof, which is closely related to Miller's notion of uniform proof [MNPS89].

We then extended the notion of universal variable so that such variables were allowed to occur in endsequents. We were then able to draw a close analogy be-

tween the various components of a non-ground endsequent and a PROLOG program together with a query. By extending our work on reordering and the "occurs check" to non-ground endsequents (a simple step) we obtained an operational semantics for our notion of logic programming.

We then provided a denotational semantics for our notion of logic programming by performing a least fixed point construction in the manner of Miller [Mi89] over a collection of Herbrand interpretations. We were able to characterize this semantics in model-theoretic terms via the Yoneda functor.

Further work suggested by these sections includes:

- The model theory of the LF: the category of models for the $\lambda\Pi$-calculus discussed in Appendix A provides a possible foundation for a study of the embedding of models of object-logics of the LF in models of $\lambda\Pi$-calculus, perhaps via the development of the notion of an *institution* [GB84] to include contexts as well as signatures;

- A study of program specification in the category **GAT**;

- A study of modularity, in the manner of Miller, [Mi89];

- A study of these notions for Generalized type Systems.

Appendix A

The Model Theory of the $\lambda\Pi$-calculus

A.1 Introduction

We present the basic model theory of the $\lambda\Pi$-calculus in the manner of "functorial semantics" first introduced by Lawvere in *Functorial semantics of algebraic theories* [La63]. The development of this idea which is appropriate for the $\lambda\Pi$-calculus is that of contextual categories which was introduced by Cartmell [Ca78], [Ca86] in order to provide a functorial semantics for the type theories of Martin-Löf [ML82]. This work has been extended by Streicher [St88] to the Calculus of Constuctions of Coquand and Huet [Co85], [CH85]. See the work of Hyland and Pitts [HP87] for a discussion of topos-theoretic models of the Calculus of Constructions. A general discussion of models for systems of dependent types may be found in [Eh88].

The basic idea of functorial semantics is very general and rather simple. Given a syntactic system, *e.g.* a collection of inference rules, and a signature or theory Σ, *e.g.* a collection of constants, we construct a *syntactic category* out of this syntax. In the case of the $\lambda\Pi$-calculus this category has contexts as objects and substitutions (or realizations) — tuples of object-level terms which satisfy certain conditions — as morphisms. Such a syntactic category inherits from the syntactic system out of which it is constructed a certain syntactic (or logical) structure. In the case of the $\lambda\Pi$-calculus this structure is that of the dependent function space (or product), which appears syntactically as Π, and its interaction with the structure of contexts. *Models* of the given system are then functors from the constructed syntactic category to an appropriate *value category* which preserve the appropriate structure of the syntactic category. Value categories are usually

set-theoretic and indeed in much of categorical logic are taken to be the category $\mathcal{S}$ of (small) sets and functions [Ma71]. In the case of the $\lambda\Pi$-calculus the value category corresponding to $\mathcal{S}$ is the category $\mathcal{F}$ of sets, families of sets, families of families of sets *etc.*, which we define later, and which is used for the construction of the term model.

We assume a basic knowledge of category theory [Ma71] and categorical logic [Go79], [LS86] without further comment. A categorical semantics for Martin-Löf type theory which is more closely related to the idea of a *hyperdoctrine* (see [Se77], [Se83]) is given by Seely in [Se84].

A.2 Contextual Categories

We present definition and basic theory of contextual categories as developed by Cartmell [Ca78], [Ca86] and further developed and presented by Streicher [St88]. For this we shall need to define the notion of a *tree structure* on a set.

DEFINITION A.2.1 (TREE STRUCTURE)

Let N be a set (of nodes). A *tree structure* on N is a pair $\triangleleft = \langle p, l \rangle$ where $p : N \longrightarrow N$ (the predecessor function) and $l : N \longrightarrow \mathbb{N}$ (the level function) are such that:

1. For all $x, y \in N$, if $l(x) = l(y) = 0$ then $x = y$ and $p(x) = x$.

2. For all $x \in N$ and $n \in \mathbb{N}$, if $l(x) = n + 1$ then $l(p(x)) = n$. □

DEFINITION A.2.2 (CONTEXTUAL CATEGORIES)

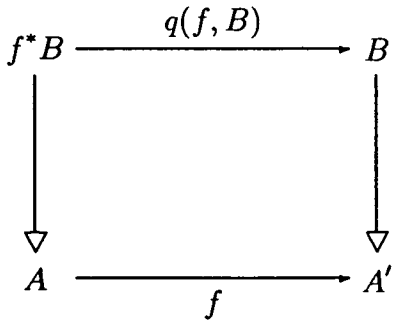
A *contextual category* consists of:

1. A category $\mathcal{C}$ with terminal object ϵ.

2. A tree structure, $\triangleleft = \langle p, l \rangle$, on the objects of $\mathcal{C}$ such that the terminal object ϵ is the unique least element, under the ordering determined by l, of the tree. We write $A \triangleleft B$ if the objects A and B of $\mathcal{C}$ are such that $p(B) = A$.

3. For all A, $B \in |\mathcal{C}|$ such that $A \triangleleft B$, a morphism $p(B) : B \longrightarrow A$ in $\mathcal{C}$. This morphism will also be written as $B \longrightarrow\!\!\triangleright A$.

4. For all A, $A' \in |\mathcal{C}|$, for all $f : A \longrightarrow A'$ in $\mathcal{C}$ and for all $B \in |\mathcal{C}|$ such that $A' \triangleleft B$, an object $f^*B \in |\mathcal{C}|$ and a morphism $q(f, B) : f^*B \longrightarrow B$ such that the diagram

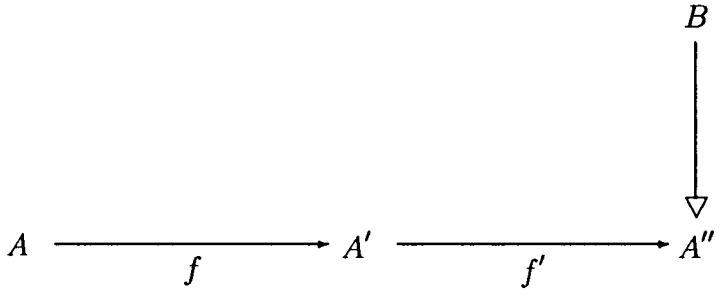


is a pullback diagram in $\mathcal{C}$.

We also require two coherence conditions.

- For all $A, B \in |\mathcal{C}|$ such that $A \lhd B$, $1_A^* B = B$ and $q(1_A, B) = 1_B$, where 1_A is the identity morphism on A, *etc.*.

- If



in $\mathcal{C}$, then $(f \circ f')^* B = f^*(f'^* B)$ and $q(f \circ f', B) = q(f, f'^* B) \circ q(f', B)$. □

If $g : B \longrightarrow B'$ is another morphism over A', then we write $f^* g$ for the unique morphism m such that $m \circ q(f, B') = q(f, B) \circ g$ and $p(f^* B) = m \circ p(f^* B')$.

DEFINITION A.2.3 (THE RELATION $\blacktriangleleft$)

$A \blacktriangleleft B$ if $A \lhd \ldots \lhd B$. □

The morphisms of Part 3 of Definition A.2.2 can be composed to give a morphism $p(B, A)$, where $A \blacktriangleleft B$.

DEFINITION A.2.4 (MULTI-STEP p)

If $A \blacktriangleleft B$ in the contextual category $\mathcal{C}$, then we define the morphism $p(B, A) : B \longrightarrow A$ in $\mathcal{C}$, which will also be written as $B \longrightarrow\!\!\blacktriangleright A$, by

$$p(B, A) = p(B) \circ p(X_n) \circ \ldots \circ p(X_1)$$

where $X_1, \ldots, X_n$ is the unique sequence of objects of $\mathcal{C}$ such that $A \triangleleft X_1 \triangleleft \ldots \triangleleft X_n \triangleleft B$ in $\mathcal{C}$. In the case $A = B$, $p(B, A) = 1_A$. □

We can obtain a pullback for any morphism of the form $p(B, A)$ along any morphism with codomain A by repeatedly forming the pullback along each $p(X_i)$. Thus, if $A' \blacktriangleleft B$ in $\mathcal{C}$ and if $f : A \longrightarrow A'$ is a morphism in $\mathcal{C}$ then we have a pullback for the morphism $p(B, A')$ along f given by the diagram

$$\begin{array}{ccc} q(\ldots q(f, X_1) \ldots X_n)^* B & \xrightarrow{q(q(\ldots q(f, X_1) \ldots X_n), B)} & B \\ \downarrow & & \downarrow \\ A & \xrightarrow[f]{} & A' \end{array}$$

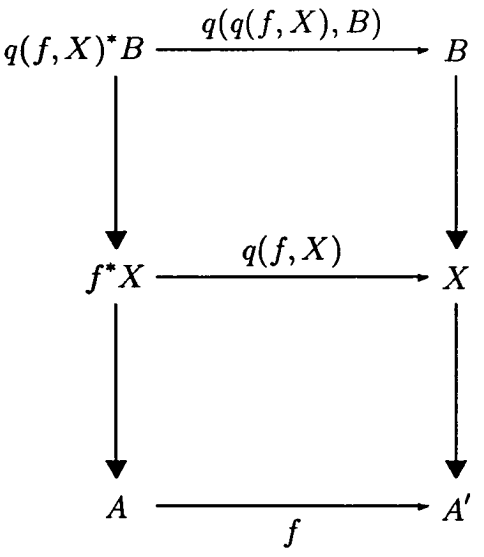
We extend the * and q notation to these new pullback diagrams, so that for $f : A \longrightarrow A'$ and $A' \blacktriangleleft B$

$$\begin{array}{ccc} f^* B & \xrightarrow{q(f, B)} & B \\ {\scriptstyle p(f^* B, A)} \downarrow & & \downarrow {\scriptstyle p(B, A')} \\ A & \xrightarrow[f]{} & A' \end{array}$$

is the canonical pullback diagram.

LEMMA A.2.5 (COMPOSITION OF PULLBACKS)

If $f : A \longrightarrow A'$ and $A' \blacktriangleleft X \blacktriangleleft B$ in the contextual category $\mathcal{C}$ then $f^* B = q(f, X)^* B$ and $q(f, B) = q(q((f, X), B))$, *i.e.*,



□

DEFINITION A.2.6 (THE CONTEXTUAL CATEGORY OF FAMILIES OF SETS, $\mathcal{F}$)
The category $\mathcal{F}$ of families of sets is defined as a construction on the category of (small) sets and functions, $\mathcal{S}$, as follows:

- Objects, level and denotation, DEN:
  - ϵ is the unique context of level 0, and $\mathrm{DEN}(\epsilon) = \{\emptyset\}$.
  - A context A, of level $n+1$, is a pair $\langle B, F\rangle$ where B is of level n and
    $$F : \mathrm{DEN}(B) \longrightarrow \mathcal{S}$$
    is a family of sets indexed by the elements of DEN(B) and DEN(A) is defined by
    $$\mathrm{DEN}(A) = \mathrm{DEN}(\langle B, F\rangle) = \{\langle x, y\rangle \mid x \in \mathrm{DEN}(B) \text{ and } y \in F(x)\}.$$
- Morphisms:
  - If A and B are objects of $\mathcal{F}$ then the morphisms from A to B are the functions from $\mathrm{DEN}(A)$ to $\mathrm{DEN}(B)$ (which are morphisms in $\mathcal{S}$).

The reader is referred to the work of Cartmell [Ca78], [Ca86] or Streicher [St88] for the details of the straightforward proof of the next proposition.

PROPOSITION A.2.7 ($\mathcal{F}$ IS A CONTEXTUAL CATEGORY)

The category $\mathcal{F}$ is structured as a contextual category as follows:

- The unique terminal object is ϵ.
- If $A = \langle B, F\rangle$ is an object of $\mathcal{F}$ then $p(A)$ is the projection on the first component, *i.e.*, for $x \in \mathrm{DEN}(B)$ and $y \in F(x)$, $p(A)(\langle x, y\rangle) = x$. The level function is that given by Definition A.2.6. Thus the tree structure is determined.
- If $A = \langle B, F\rangle$ is an object of $\mathcal{F}$ and if $f : C \longrightarrow B$ is a morphism then $f^*A = \langle C, F \circ f\rangle$.
- If $A = \langle B, F\rangle$ is an object of $\mathcal{F}$ and if $f : C \longrightarrow B$ is a morphism then $q(f, A)$ is the function from $\mathrm{DEN}(f^*A)$ to $\mathrm{DEN}(A)$ such that for all $x \in \mathrm{DEN}(C)$ and $y \in F(f(x))$, $q(f, A)(\langle x, y\rangle) = \langle f(x), y\rangle$. □

The structure preserving morphisms (or homomorphisms) between contextual categories are *contextual functors.*

DEFINITION A.2.8 (CONTEXTUAL FUNCTORS)

Let $\mathcal{C}$ and $\mathcal{C}'$ be contextual categories. A *contextual functor* $F : \mathcal{C} \longrightarrow \mathcal{C}'$ is a functor $F : \mathcal{C} \longrightarrow \mathcal{C}'$ such that:

1. $F(\epsilon) = \epsilon$, and if $A \triangleleft B$ in $\mathcal{C}$, then $F(A) \triangleleft F(B)$ in $\mathcal{C}'$.
2. For all objects A of $\mathcal{C}$, $F(p(A)) = p(F(A))$.
3. For all f and B such that f^*B is defined in $\mathcal{C}$, $F(f^*B) = F(f)^*F(B)$ and $F(q(f, B)) = q(F(f), F(B))$. □

A.3 Contextual Categories with Products

The definition of a contextual category allows us to consider families of types, but does not include any notion of the "product of a family of types".

The appropriate type-theoretic intuition for the product $B(x)$ of a family of types indexed by $x \in A$ is as the type of all functions f whose domain is the type A and whose value $f(a)$ on an object $a \in A$ is an object of $B(a)$.

With this in mind, we proceed to define the notion of a *contextual category with products.*

The content of the next definition is due to Cartmell [Ca78], but the notation $sections_{\mathcal{C}}(B)$ is due to Streicher [St88]. We adopt Streicher's more helpful notation. This definition provides a simple mechanism for imposing a simple coherence condition in the definition of a contextual category with products.

DEFINITION A.3.1 (THE SET $sections_{\mathcal{C}}(B)$)

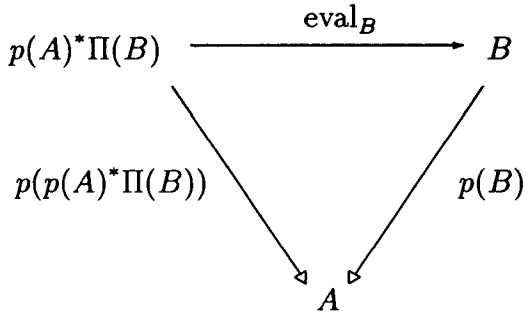
Let $\mathcal{C}$ be a contextual category and let A, B be objects of $\mathcal{C}$ such that $A \triangleleft B$.

$$sections_{\mathcal{C}}(B) =_{\text{def}} \{f \in \hom_{\mathcal{C}}(A, B) \mid p(B) \circ f = 1_A\}. \ \square$$

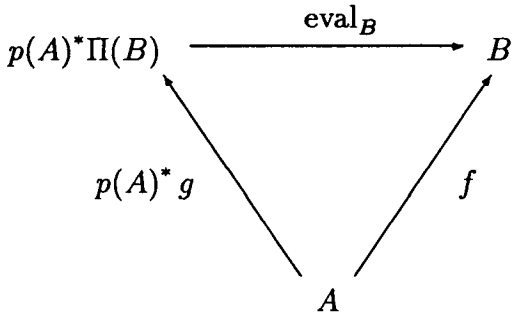
DEFINITION A.3.2 (CONTEXTUAL CATEGORIES WITH PRODUCTS)

Let $\mathcal{C}$ be a contextual category. $\mathcal{C}$ is a *contextual category with products* if it satisfies:

1. If $C \triangleleft A \triangleleft B$ in $\mathcal{C}$ there is an object $\Pi(B)$ of $\mathcal{C}$ such that $C \triangleleft \Pi(B)$, and a morphism $\text{eval}_B : p(A)^*\Pi(B) \longrightarrow B$ such that the diagram



commutes, and such that $\Pi(B)$ and eval_B have the property that for every morphism $f \in \mathit{sections}_{\mathcal{C}}(B)$ there is a unique morphism $g \in \mathit{sections}_{\mathcal{C}}(\Pi(B))$ such that the diagram



commutes. Following Streicher [St88], we write $g = \mathit{curry}(f)$ and $f = \mathit{uncurry}(g)$.

2. If $m : P \longrightarrow Q$ is a morphism in $\mathcal{C}$ and $Q \triangleleft A \triangleleft B$, then

   - $m^*\Pi(B) = \Pi(m^*B)$ and
   - $m^*\mathrm{eval}_B = \mathrm{eval}_{m^*B}$. □

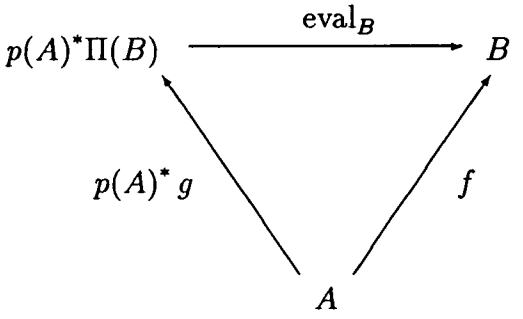
   This clause is the appropriate formulation of the *Beck-Chevalley condition* (see [Se77], [Se83]).[1] □

We define the notion of a contextual functor which preserves products.

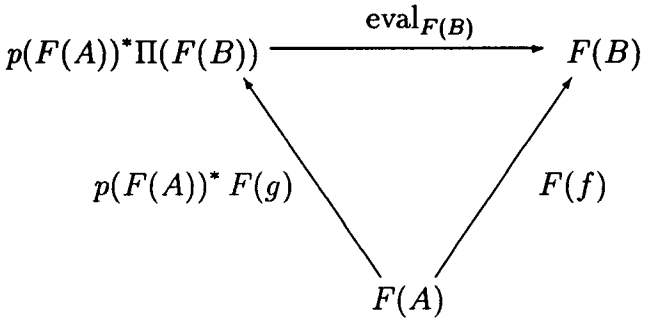
DEFINITION A.3.3 (PRODUCT-PRESERVING CONTEXTUAL FUNCTORS)

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be contextual categories with products and let $F : \mathcal{C}_1 \longrightarrow \mathcal{C}_2$ be a contextual functor. F is said to *preserve products* if whenever

[1] In fact, it is possible to reformulate the statement of this condition so that it more closely resembles the usual presentation of the Beck-Chevalley condition, see [St88].



is a product diagram in $\mathcal{C}_1$,



is a product diagram in $\mathcal{C}_2$. □

The reader is referred to [Ca78] and [St88] for the proof of the next proposition.

PROPOSITION A.3.4 ($\mathcal{F}$ IS A CONTEXTUAL CATEGORY WITH PRODUCTS)

The category $\mathcal{F}$ is structured as a contextual category with products as follows:

Let A be an object of $\mathcal{F}$ such that $B = \langle\langle A, F\rangle, G\rangle$.

- We define the object $\Pi(B)$ by $\Pi(B) =_{\text{def}} \langle A, P\rangle$, where $P(x)$ is defined by:

$$P(x) =_{\text{def}} \{f : F(x) \longrightarrow \bigcup\{G(\langle x, y\rangle) | y \in F(x)\} | \text{ for all } y \in F(x), f(y) \in G(\langle x, y\rangle)\}.$$

- The morphism $\mathrm{eval}_B : p(A)^*\Pi(B) \longrightarrow B$ is defined by $\mathrm{eval}_B(\langle x, y\rangle, f)) = \langle x, f(y)\rangle$ for each $x \in \mathrm{DEN}(A)$, $y \in F(x)$ and $f \in P(x)$. □

If in the definition of a contextual category with products we drop the requirement that g be unique, then we speak of a *contextual category with weak products.*

A.4 The Functorial Semantics of the $\lambda\Pi$-calculus

In this section we present the functorial semantics of the $\lambda\Pi$-calculus. We begin by defining the interpretation of the $\lambda\Pi$-calculus in a contextual category with products, and state the correctness of such interpretations. This shows that contextual categories with products are the correct structures in which to interpret the $\lambda\Pi$-calculus. We then construct a syntactic category $C(\Sigma)$ from the syntax of the $\lambda\Pi$-calculus, and illustrate that $C(\Sigma)$ is a contextual category with products which correponds exactly to the syntax of the $\lambda\Pi$-calculus. If $\mathcal{V}$ is a contextual category with products we are then able to define a $\mathcal{V}$-valued model of the $\lambda\Pi$-calculus to be a product-preserving contextual functor from $C(\Sigma)$ to $\mathcal{V}$. We construct a closed term model, for which the appropriate value category is $\mathcal{F}$.

We note that since the category $\mathcal{F}$ has products, rather than weak products, it is able to interpret $\beta\eta$-equality: a suitable value category with weak products would interpret β-equality (see [Ca78], §3.4). In the subsequent text we do not distinguish between expressions that are identical up to change of bound variables. We abbreviate **N** proves $\Gamma \vdash_\Sigma M : A$ by $\Gamma \vdash_\Sigma M : A$, *etc.*.

A.4.1 Algebraic Interpretation

It is clear that it is not possible to define the set of well-formed $\lambda\Pi$-calculus contexts, types and objects independently of the set of provable judgements, such as $\Gamma \vdash_\Sigma M : A$. Consequently, the definition of an algebraic interpretation of the $\lambda\Pi$-calculus in a contextual category with products must be as a partial function which is later shown to be defined for all appropriate provable judgements.

The definition of the interpretation of the $\lambda\Pi$-calculus in a contextual category with products is facilitated by the definition of a measure on expressions as follows:

DEFINITION A.4.1 (DEPTH OF EXPRESSIONS)

Let Σ be a valid $\lambda\Pi$-signature, let $\Gamma \equiv x_1 : A_1, \ldots, x_m : A_m$ (not necessarily a *well-formed* context). By structural induction on (not necessarily *well-formed*) expressions in the usual notation we define a depth function mapping (not necessarily *well-formed*) expressions to $\mathbb{N}$ as follows (*cf.* [St88]):

$$
\begin{array}{lcl}
depth(c) & = & 1 \\
depth(x) & = & 1 \\
depth(\Pi x\!:\!A\,.\,B) & = & 1 + depth(A) + depth(B) \\
depth(\lambda x\!:\!A\,.\,M) & = & 1 + depth(A) + depth(M) \\
depth(MN) & = & 1 + depth(M) + depth(N) + depth(A) + depth(B), \\
 & & \text{where } M : \Pi x\!:\!A\,.\,B \text{ and } N\!:\!A \\
depth(\Gamma) & = & \sum_{i=1}^{m} depth(A_i)\,.
\end{array}
$$

□

DEFINITION A.4.2 (ALGEBRAIC INTERPRETATION)

Let Σ be a valid $\lambda\Pi$-signature, let $\Gamma \equiv x_1 : A_1, \ldots, x_m : A_m$ and let $c : A \in \Sigma$. Let C be a contextual category with products. Let $\simeq$ be the usual Kleene equality, [Kl52]. We define a *C-valued Σ-algebraic interpretation (partial) function* $\{\!\{-\}\!\}^{\mathcal{A}}$ by induction on the level of its argument, where we define $level(\Gamma) = depth(\Gamma)$, $level(\Gamma, x : A) = level(\Gamma) + depth(A)$ and $level(\langle x_1, \ldots, x_m, M\rangle) = depth(\Gamma) + depth(M)$, as follows (*cf.* [St88]):

- $\{\!\{\langle\rangle\}\!\}^{\mathcal{A}} \simeq \epsilon$;
- $\{\!\{\Gamma\}\!\}^{\mathcal{A}} \simeq u$ where u is an object of C of level m;
  $\{\!\{\Gamma, x\!:\!A\}\!\}^{\mathcal{A}} \simeq v$ where v is an object of C of level $m+1$ such that $u \triangleleft v$;
- $\{\!\{\langle x_1, \ldots, x_m, c\rangle\}\!\}^{\mathcal{A}} \simeq f : \{\!\{\Gamma\}\!\}^{\mathcal{A}} \longrightarrow \{\!\{\Gamma, x : A\}\!\}^{\mathcal{A}}$ where the map f is such that $f \circ p(\{\!\{\Gamma, x\!:\!A\}\!\}^{\mathcal{A}}) = 1_{\{\!\{\Gamma\}\!\}^{\mathcal{A}}}$;
- $\{\!\{\langle x_1, \ldots, x_m, x_i\rangle\}\!\}^{\mathcal{A}}(1 \leq i \leq m) \simeq p(A_n, A_i)^*\Delta(A_i)$ where $\{\!\{\Gamma\}\!\}^{\mathcal{A}} \simeq A_m \triangleright \ldots \triangleright A_1 \triangleright 1$ and $\Delta(A_i)$ is the unique map $g : A_i \longrightarrow p(A_i)^*A_i$ such that $p(p(A_i)^*A_i) \circ g = q(p(A_i), A_i) \circ g = 1_{A_i}$;

- $\{\!|\Gamma, x:\Pi x{:}A\,.\,B|\!\}^{\mathcal{A}} \simeq \Pi(\{\!|\Gamma, x{:}A, y{:}B|\!\}^{\mathcal{A}})$;

- $\{\!|\langle x_1,\ldots,x_m, MN\rangle|\!\}^{\mathcal{A}} \simeq h_2^*(\mathrm{eval}(F) \circ p(E)^*(h_1))$ if $\{\!|\Gamma|\!\}^{\mathcal{A}} \triangleleft E \triangleleft F$ and $D = \Pi(F)$, where we have that the map $\{\!|\langle x_1,\ldots,x_m, M\rangle|\!\}^{\mathcal{A}} \simeq h_1 : \{\!|\Gamma|\!\}^{\mathcal{A}} \longrightarrow D$, $\{\!|\langle x_1,\ldots,x_m, N\rangle|\!\}^{\mathcal{A}} \simeq h_2 : \{\!|\Gamma|\!\}^{\mathcal{A}} \longrightarrow E$ and $\{\!|\Gamma, x{:}A, y{:}B|\!\}^{\mathcal{A}} \simeq F$. □

The reader is referred to [St88] for the proof of Proposition A.4.3. In fact, [St88] proves the correctness of an interpretation of the Calculus of Constructions [Co85] in *doctrines of constructions*, which are contextual categories with products and some further structure. The proof of Proposition A.4.3 is a subproof of his.

PROPOSITION A.4.3 (CORRECTNESS OF ALGEBRAIC INTERPRETATION)

Let Σ be a valid $\lambda\Pi$-signature and let $\mathcal{A}$ be a *C-valued Σ-algebraic interpretation function*. Let $\Gamma \equiv x_1{:}A_1,\ldots,x_m{:}A_m$.

1. If $\vdash_\Sigma \Gamma$ context then $\{\!|\Gamma|\!\}^{\mathcal{A}}$ is defined.

2. If $\Gamma \vdash_\Sigma A{:}\mathrm{Type}$ then $\{\!|\Gamma, x{:}A|\!\}^{\mathcal{A}}$ is defined.

3. If $\Gamma \vdash_\Sigma M{:}A$ then $\{\!|\Gamma|\!\}^{\mathcal{A}}$ is defined; if the map $\{\!|\langle x_1,\ldots x_m, M\rangle|\!\}^{\mathcal{A}} = f$ and $\{\!|\Gamma, x{:}A|\!\}^{\mathcal{A}} = P$ then $f : \{\!|\Gamma|\!\}^{\mathcal{A}} \longrightarrow \{\!|\Gamma, x{:}A|\!\}^{\mathcal{A}}$. □

A.4.2 A Syntactic Category of Contexts and Substitutions

We define a syntactic category in which the objects are valid $\lambda\Pi$-calculus contexts over a given valid signature Σ, and in which the morphisms are certain tuples of $\lambda\Pi$-calculus objects over Σ. This category corresponds exactly to the syntax of the $\lambda\Pi$-calculus over Σ.

DEFINITION A.4.4 (THE CATEGORY $C(\Sigma)$)

Let Σ be a given valid $\lambda\Pi$-calculus signature, and let $\Gamma \equiv x_1 : A_1,\ldots,x_m : A_m$ and $\Delta \equiv y_1 : B_1,\ldots,y_n : B_n$ be valid $\lambda\Pi$-calculus contexts over Σ. A *substitution*

for Δ *with respect to* Γ is a tuple $\langle M_1, \ldots, M_n \rangle$ such that for each $1 \leq i \leq n$, $\Gamma \vdash_\Sigma M_i : B_i[M_1/y_1, \ldots, M_{i-1}/y_{i-1}]$.[2]

- The objects of $C(\Sigma)$ are valid $\lambda\Pi$-calculus contexts over Σ.

- The morphisms of $C(\Sigma)$ are substitutions $\langle M_1, \ldots, M_n \rangle : \Gamma \longrightarrow \Delta$, where Γ and Δ are contexts.

  - The identity morphism on the object $\Gamma \equiv x_1 : A_1, \ldots, x_m : A_m$ is the substitution $\langle x_1, \ldots, x_m \rangle$.

  - Let Γ and Δ and the substitution $\langle M_1, \ldots, M_n \rangle$ be defined as above. Let $\Theta \equiv z_1 : C_1, \ldots, z_p : C_p$ be a context, and let $\langle N_1, \ldots, N_p \rangle$ be a morphism $\Delta \longrightarrow \Theta$. Composition in $C(\Sigma)$ is defined by

$$\langle M_1, \ldots, M_n \rangle \circ \langle N_1, \ldots, N_p \rangle =_{\text{def}}$$

$$\langle N_1[M_1/y_1, \ldots, M_n/y_n], \ldots, N_p[M_1/y_1, \ldots, M_n/y_n] \rangle .$$

This completes the definition of the category $C(\Sigma)$. □

The reader is referred to the work of Cartmell [Ca78], [Ca86] and Streicher [St88] for the proofs of the next two theorems.

THEOREM A.4.5 ($C(\Sigma)$ IS A CONTEXTUAL CATEGORY)

The category $C(\Sigma)$ is structured as a contextual category as follows:

- The least element ϵ is the empty context, $\langle\rangle$.

- Let $\Gamma \equiv x_1 : A_1, \ldots, x_m : A_m$ and $\Delta \equiv \Gamma, x_{m+1} : A_{m+1}, \ldots, x_{m+n} : A_{m+n}$ be contexts. The canonical morphism $p(\Delta, \Gamma) : \Delta \longrightarrow \Gamma$ is defined by $p(\Delta, \Gamma) \equiv_{\text{def}} \langle x_1, \ldots, x_m \rangle$.

---

[2] Cartmell [Ca78], [Ca86] calls substitutions *realizations*.

- Let $\Gamma \equiv x_1 : A_1, \ldots, x_m : A_m$, $\Gamma' \equiv y_1 : B_1, \ldots, y_n : B_n$, and $\Delta \equiv y_1 : B_1, \ldots, y_n : B_n, y_{n+1} : B_{n+1}, \ldots, y_{n+p} : B_{n+p}$ and suppose that $f \equiv \langle M_1, \ldots, M_n \rangle : \Gamma \longrightarrow \Gamma'$ is a substitution for Γ' with respect to Γ, then:

  - $f^*B \equiv_{\text{def}} x_1 : A_1, \ldots, x_m : A_m, y_{n+1} : B_{n+1}[M_1/y_1, \ldots, M_n/y_n], \ldots, y_{n+p} : B_{n+p}[M_1/y_1, \ldots, M_n/y_n]$, and
  - $q(f, B) \equiv_{\text{def}} \langle M_1, \ldots, M_n, y_{n+1}, \ldots, y_{n+p} \rangle$. □

THEOREM A.4.6 ($C(\Sigma)$ IS A CONTEXTUAL CATEGORY WITH PRODUCTS)

Let Σ be a valid $\lambda\Pi$-signature. The category $C(\Sigma)$ is structured as a contextual category with products as follows:

If $\Gamma \equiv x_1 : A_1, \ldots, x_m : A_m, x : A, y : B$ is a valid context over Σ, then:

- $\Pi(\Gamma) \equiv_{\text{def}} x_1 : A_1, \ldots, x_m : A_m, x : \Pi x : A \,.\, B$;
- $\text{eval}_\Gamma \equiv_{\text{def}} \langle x_1, \ldots, x_m, x, yx \rangle : \Delta, x : A, y : \Pi x : A \,.\, B \longrightarrow \Delta$, where $\Delta \equiv x_1 : A_1, \ldots, x_m : A_m$. □

The next theorem summarizes the equivalence of the logical and algebraic descriptions of the syntax of the $\lambda\Pi$-calculus; the reader is referred to [Ca86] and [St88] for its proof.

THEOREM A.4.7 (CORRECTNESS OF $C(\Sigma)$)

Let Σ be a valid $\lambda\Pi$-signature. Let $\Gamma \equiv x_1 : A_1, \ldots, x_m : A_m$ and $\Delta \equiv y_1 : B_1, \ldots, y_n : B_n$ be valid Σ-$\lambda\Pi$-contexts. $\langle M_1, \ldots, M_n \rangle : \Gamma \longrightarrow \Delta$ is a morphism in $C(\Sigma)$ if and only if

$$\Gamma \vdash_\Sigma M_i : B_i[M_1/y_1, \ldots, M_{i-1}/y_{i-1}]$$

for each $1 \leq i \leq n$. □

A.4.3 Models of the $\lambda\Pi$-calculus

We are now able to present the basic model theory of the $\lambda\Pi$-calculus in the manner of functorial semantics. The category of $\mathcal{V}$-valued models of the $\lambda\Pi$-calculus, where $\mathcal{V}$ is a contextual category with products, is the full subcategory $[C(\Sigma), \mathcal{V}]_{\mathrm{con},\Pi}$ of product-preserving contextual functors of the functor category $[C(\Sigma), \mathcal{V}]$.

DEFINITION A.4.8 ($\mathcal{V}$-VALUED Σ-$\lambda\Pi$-MODELS)

Let Σ be a valid $\lambda\Pi$-signature and let $\mathcal{V}$ be a contextual category with products. A *$\mathcal{V}$-valued Σ-$\lambda\Pi$-model* is an object $\mathcal{M}$ of the functor category $[C(\Sigma), \mathcal{V}]_{\mathrm{con},\Pi}$ of product-preserving contextual functors from the syntactic category $C(\Sigma)$ to $\mathcal{V}$. □

We write $[\![\Gamma]\!]^{\mathcal{M}}$ and $[\![\sigma]\!]^{\mathcal{M}}$ to denote respectively the images under $\mathcal{M}$ of a context Γ and a substitution σ.

DEFINITION A.4.9 (MORPHISMS OF $\mathcal{V}$-VALUED Σ-$\lambda\Pi$-MODELS)

Let Σ be a valid $\lambda\Pi$-signature and let $\mathcal{M}_1$ and $\mathcal{M}_2$ be $\mathcal{V}$-valued Σ-$\lambda\Pi$-models. A *morphism of $\mathcal{V}$-valued Σ-$\lambda\Pi$-models* (henceforth *Σ-$\lambda\Pi$-morphism* when no confusion is possible) is a natural transformation $m : \mathcal{M}_1 \stackrel{\cdot}{\longrightarrow} \mathcal{M}_2$. □

The construction of a term model is possible, thereby proving the existence of Σ-$\lambda\Pi$-models. The appropriate value category for this construction is $\mathcal{F}$, and the closed term model K_Σ is initial in this category of models [Ca86].

DEFINITION A.4.10 (THE CLOSED TERM MODEL K_Σ)

Let Σ be a valid $\lambda\Pi$-signature. The *closed term Σ-$\lambda\Pi$-model* K_Σ is defined inductively on contexts and substitutions as follows:

- Let Γ be a context.
  For $\Gamma \equiv x{:}A$, $[\![x{:}A]\!]^{K_\Sigma} =_{\mathrm{def}} \{\ \sigma \in \hom_{C(\Sigma)}(\langle\rangle, x{:}A),\ \sigma \text{ in } \beta\eta\text{-normal form}\ \}$.
  For $\langle\rangle \lhd \ldots \lhd \Delta \lhd \Gamma$, $[\![\Gamma]\!]^{K_\Sigma} =_{\mathrm{def}} \langle [\![\Delta]\!]^{K_\Sigma}, F_{K_\Sigma}\rangle$ where $\Delta \lhd \Gamma$ and
  $F_{K_\Sigma}(\langle u, v\rangle) =_{\mathrm{def}} \{\ \sigma \in \hom_{C(\Sigma)}(\langle\rangle, \Gamma) | \sigma \circ p(\Gamma) = v,\ \sigma \text{ in } \beta\eta\text{-normal form}\ \}$.

- Let $\sigma : \Gamma \longrightarrow \Delta$ be a substitution.
  $[\![\sigma]\!]^{K_\Sigma}$ is a function from $\mathrm{DEN}([\![\Gamma]\!]^{K_\Sigma})$ to $\mathrm{DEN}([\![\Delta]\!]^{K_\Sigma})$.
  $[\![\sigma]\!]^{K_\Sigma}(\langle t, u\rangle) =_{\mathrm{def}} \langle v, w\rangle$ where $w = u \circ \sigma$ and $t = v \circ \sigma$ (where this latter composition is defined inductively, pairwise). □

It is a straightforward calculation to verify that K_Σ is an $\mathcal{F}$-valued Σ-$\lambda\Pi$-model, and we omit the argument.

Appendix B

Example Encodings

B.1 Example Encodings

In this appendix we give a collection of encodings of logics in the LF and their adequacy theorems. The presentations of these encodings and theorems are taken directly from the University of Edinburgh report: ECS-CS-87-31, *Using Typed Lambda Calculus to Implement Formal Systems on a Machine* by Arnon Avron, Furio Honsell and Ian Mason [AHM87], to which the reader is referred for more discussion of the given examples. We thank the authors for allowing us to quote this work. We have taken the liberty of correcting typographical and grammatical errors.

B.1.1 Kleene's Three-valued Logic

The Signature Σ_{Kl}

- Syntactic Categories
  - o : Type
- Operations
  - $\neg : o \to o$
  - $\wedge : o \to o \to o$
  - $\vee : o \to o \to o$

- Judgement
  - $\mathrm{T} : o \rightarrow$ Type
- Axioms and Rules
  - $\wedge\mathrm{I} : \prod_{\phi,\psi:o} . \mathrm{T}(\phi) \rightarrow \mathrm{T}(\psi) \rightarrow \mathrm{T}(\phi \wedge \psi)$
  - $\vee\mathrm{I}_l : \prod_{\phi,\psi:o} . \mathrm{T}(\phi) \rightarrow \mathrm{T}(\phi \vee \psi)$
  - $\vee\mathrm{I}_r : \prod_{\phi,\psi:o} . \mathrm{T}(\psi) \rightarrow \mathrm{T}(\phi \vee \psi)$
  - $\neg\neg\mathrm{I} : \prod_{\phi:o} . \mathrm{T}(\phi) \rightarrow \mathrm{T}(\neg\neg\phi)$
  - $\neg\wedge\mathrm{I}_l : \prod_{\phi,\psi:o} . \mathrm{T}(\neg\phi) \rightarrow \mathrm{T}(\neg(\phi \wedge \psi))$
  - $\neg\wedge\mathrm{I}_r : \prod_{\phi,\psi:o} . \mathrm{T}(\neg\psi) \rightarrow \mathrm{T}(\neg(\phi \wedge \psi))$
  - $\neg\vee\mathrm{I} : \prod_{\phi,\psi:o} . \mathrm{T}(\neg\phi) \rightarrow \mathrm{T}(\neg\psi) \rightarrow \mathrm{T}(\neg(\phi \vee \psi))$
  - $\neg\mathrm{E} : \prod_{\phi,\psi:o} . \mathrm{T}(\phi) \rightarrow \mathrm{T}(\neg\phi) \rightarrow \mathrm{T}(\psi)$
  - $\wedge\mathrm{E}_l : \prod_{\phi,\psi:o} . \mathrm{T}(\phi \wedge \psi) \rightarrow \mathrm{T}(\phi)$
  - $\wedge\mathrm{E}_r : \prod_{\phi,\psi:o} . \mathrm{T}(\phi \wedge \psi) \rightarrow \mathrm{T}(\psi)$
  - $\neg\neg\mathrm{E} : \prod_{\phi:o} . \mathrm{T}(\neg\neg\phi) \rightarrow \mathrm{T}(\phi)$
  - $\neg\vee\mathrm{E}_l : \prod_{\phi,\psi:o} . \mathrm{T}(\neg(\phi \vee \psi)) \rightarrow \mathrm{T}(\neg\phi)$
  - $\neg\vee\mathrm{E}_r : \prod_{\phi,\psi:o} . \mathrm{T}(\neg(\phi \vee \psi)) \rightarrow \mathrm{T}(\neg\psi)$
  - $\vee\mathrm{E} : \prod_{\phi,\psi,\vartheta:o} . (\mathrm{T}(\phi) \rightarrow \mathrm{T}(\vartheta)) \rightarrow (\mathrm{T}(\psi) \rightarrow \mathrm{T}(\vartheta)) \rightarrow (\mathrm{T}(\phi \vee \psi) \rightarrow \mathrm{T}(\vartheta))$
  - $\neg\wedge\mathrm{E} : \prod_{\phi,\psi,\vartheta:o} .$

    $$(\mathrm{T}(\neg\phi) \rightarrow \mathrm{T}(\vartheta)) \rightarrow (\mathrm{T}(\neg\psi) \rightarrow \mathrm{T}(\vartheta)) \rightarrow (\mathrm{T}(\neg(\phi \wedge \psi)) \rightarrow \mathrm{T}(\vartheta))$$

THEOREM B.1.1 (ADEQUACY AND FAITHFULNESS 1)

The following hold:

$$\phi_1, \ldots, \phi_n \vdash_{\mathrm{KI}} \psi$$

if and only if there exists a term t such that:

$$p_1 : o, \ldots, p_n : o \vdash_{\Sigma_{\mathrm{KI}}} t : \mathrm{T}(\phi_1) \rightarrow \cdots \rightarrow \mathrm{T}(\phi_n) \rightarrow \mathrm{T}(\psi)$$

Where $p_1, \ldots, p_n$ are the atomic variables occuring in $\phi_i, 1 \leq i \leq n$ and ψ (we use the same symbol for denoting a formula and the corresponding term in LF). Moreover, there is a compositional bijection between proofs in the natural deduction system and proof terms such as t above. □

B.1.2 First-order Logic with a Choice Operator

The Signature $\Sigma_{\epsilon 1^o}$

- Syntactic Categories
  - ι : Type
  - o : Type
- Operations
  - $= \; : \iota \to \iota \to o$
  - $\neg : o \to o$
  - $\supset : o \to o \to o$
  - $\epsilon : (\iota \to o) \to \iota$
  - $\exists : (\iota \to o) \to o$
  - $\forall : (\iota \to o) \to o$
- Judgement
  - $\mathrm{T} : o \to$ Type
- Axioms and Rules
  - $\mathrm{E}_0 : \prod_{x:\iota} . \; \mathrm{T}(x = x)$
  - $\mathrm{E}_1 : \prod_{x,y:\iota} . \; \prod_{t:\iota \to \iota} . \; \mathrm{T}(x = y) \to \mathrm{T}(t(x) = t(y))$
  - $\mathrm{E}_2 : \prod_{x,y:\iota} . \; \prod_{\Phi:\iota \to o} . \; \mathrm{T}(x = y) \to \mathrm{T}(\Phi(x)) \to \mathrm{T}(\Phi(y))$
  - $\neg\mathrm{I} : \prod_{\phi,\psi:o} . \; (\mathrm{T}(\phi) \to \mathrm{T}(\psi)) \to (\mathrm{T}(\phi) \to \mathrm{T}(\neg\psi)) \to \mathrm{T}(\neg\phi)$

- $\wedge \mathrm{I} : \prod_{\phi,\psi:o} . \mathrm{T}(\phi) \to \mathrm{T}(\psi) \to \mathrm{T}(\phi \wedge \psi)$
- $\vee \mathrm{I}_l : \prod_{\phi,\psi:o} . \mathrm{T}(\phi) \to \mathrm{T}(\phi \vee \psi)$
- $\vee \mathrm{I}_r : \prod_{\phi,\psi:o} . \mathrm{T}(\psi) \to \mathrm{T}(\phi \vee \psi)$
- $\wedge \mathrm{E}_l : \prod_{\phi,\psi:o} . \mathrm{T}(\phi \wedge \psi) \to \mathrm{T}(\phi)$
- $\wedge \mathrm{E}_r : \prod_{\phi,\psi:o} . \mathrm{T}(\phi \wedge \psi) \to \mathrm{T}(\psi)$
- $\neg\neg \mathrm{E} : \prod_{\phi:o} . \mathrm{T}(\neg\neg\phi) \to \mathrm{T}(\phi)$
- $\vee \mathrm{E} : \prod_{\phi,\psi,\vartheta:o} . (\mathrm{T}(\phi) \to \mathrm{T}(\vartheta)) \to (\mathrm{T}(\psi) \to \mathrm{T}(\vartheta)) \to (\mathrm{T}(\phi \vee \psi) \to \mathrm{T}(\vartheta))$
- $\supset \mathrm{E} : \prod_{\phi_1,\phi_2:o} . \mathrm{T}(\phi_1 \supset \phi_2) \to \mathrm{T}(\phi_1) \to \mathrm{T}(\phi_2)$
- $\supset \mathrm{I} : \prod_{\phi,\psi:o} . (\mathrm{T}(\phi) \to \mathrm{T}(\psi)) \to \mathrm{T}(\phi \supset \psi)$
- $\epsilon \mathrm{I} : \prod_{\Phi:\iota\to o} . \mathrm{T}(\exists(\Phi)) \to \mathrm{T}(\Phi(\epsilon(\Phi)))$
- $\exists \mathrm{E} : \prod_{\Phi:\iota\to o} . \prod_{\phi:o} . \mathrm{T}(\exists(\Phi)) \to (\prod_{x:\iota} . \mathrm{T}(\Phi(x)) \to \mathrm{T}(\phi)) \to \mathrm{T}(\phi)$
- $\exists \mathrm{I} : \prod_{\Phi:\iota\to o} . \prod_{t:\iota} . \mathrm{T}(\Phi(t)) \to \mathrm{T}(\exists(\Phi))$
- $\forall \mathrm{E} : \prod_{\Phi:\iota\to o} . \prod_{t:\iota} . \mathrm{T}(\forall(\Phi)) \to \Phi(t)$
- $\forall \mathrm{I} : \prod_{\Phi:\iota\to o} . (\prod_{t:\iota} . \mathrm{T}(\Phi(t))) \to \mathrm{T}(\forall(\Phi))$

THEOREM B.1.2 (ADEQUACY AND FAITHFULNESS 2)

We state the adequacy only for a monadic language, the general case follows exactly the same pattern. Letting

$$\Gamma = \{x_1 : \iota, \ldots x_n : \iota, X_1 : \iota \to o, \ldots X_m : \iota \to o\}$$

the following hold:

1. $\Gamma \vdash M : \iota$ if and only if $\Phi_\Gamma(M)$ is a well-formed term of first-order logic with a choice operator whose only free individual variables are among $x_1, \ldots, x_n$ and whose unary relations are among the $X_1, \ldots, X_m$.

2. $\Gamma \vdash M : o$ if and only if $\Phi_\Gamma(M)$ is a well-formed formula of first-order logic with a choice operator whose only free individual variables are among $x_1, \ldots, x_n$ and whose unary relations are among the $X_1, \ldots, X_m$.

3. $\Gamma \cup \{y_1 : \mathrm{True}(\phi_1), \ldots, y_k : \mathrm{True}(\phi_k)\} \vdash M : \mathrm{True}(\phi)$

iff

$$\Phi_\Gamma(\phi_1), \ldots, \Phi_\Gamma(\phi_k) \vdash \Phi_\Gamma(\phi).$$

where Φ_Γ is a bijective function

$$\Phi_\Gamma : \Xi_\Gamma(i) \cup \Xi_\Gamma(o) \rightarrow \epsilon 1^\circ$$

to be defined shortly. $\epsilon 1^\circ$ denotes the collection of terms and formulas of first-order logic with a choice operator whose only free individual variables are among $x_1, \ldots, x_n$ and whose unary relations are among the $X_1, \ldots, X_m$. $\Xi_\Gamma(\tau)$ is the set of long $\beta\eta$-normal forms of type τ in the context Γ. Finally

$$\Phi_\Gamma(M) = \begin{cases} x & \text{if } M \equiv x \\ \Phi_\Gamma(M') = \Phi_\Gamma(N) & \text{if } M \equiv (M' = N) \\ \epsilon(\Phi_{\Gamma,x:\iota}(P[x])) & \text{if } M \equiv \epsilon(\lambda x : \iota.P[x]) \\ \neg\Phi_\Gamma(M') & \text{if } M \equiv \neg M \\ \Phi_\Gamma(M') \supset \Phi_\Gamma(N) & \text{if } M \equiv M' \supset N \\ \forall x.\Phi_{\Gamma,x:\iota}(M'[x]) & \text{if } M \equiv \forall(\lambda x : \iota.M[x]) \\ \exists x.\Phi_{\Gamma,x:\iota}(M'[x]) & \text{if } M \equiv \exists(\lambda x : \iota.M[x]) \\ X(\Phi_\Gamma(M')) & \text{if } M \equiv X(M'). \end{cases} \quad \square$$

B.1.3 Hilbert Style Modal Logics

The Signature $\Sigma_{H\Box}$

The resulting signature in the particular case of S4 is :

- Syntactic Categories
  - o : Type
- Operations
  - $\neg : o \rightarrow o$

- $\supset : o \to o \to o$
- $\Box : o \to o$

- Judgements
  - $\text{True} : o \to \text{Type}$
  - $\text{Valid} : o \to \text{Type}$

- Axioms and Rules
  - $\text{C} : \prod_{\phi:o} . \text{Valid}(\phi) \to \text{True}(\phi)$
  - $\text{A}_1 : \prod_{\phi_1,\phi_2:o} . \text{Valid}(\phi_1 \supset (\phi_2 \supset \phi_1))$
  - $\text{A}_2 : \prod_{\phi_1,\phi_2,\phi_3:o} . \text{Valid}(\phi_1 \supset (\phi_2 \supset \phi_3) \supset (\phi_1 \supset \phi_2) \supset (\phi_1 \supset \phi_3))$
  - $\text{A}_3 : \prod_{\phi_1,\phi_2:o} . \text{Valid}((\neg\phi_1 \supset \neg\phi_2) \supset (\phi_2 \supset \phi_1))$
  - $\text{A}_4 : \prod_{\phi:o} . \text{Valid}(\Box\phi \supset \phi)$
  - $\text{A}_5 : \prod_{\phi_1,\phi_2:o} . \text{Valid}(\Box(\phi_1 \supset \phi_2) \supset (\Box\phi_1 \supset \Box\phi_2))$
  - $\text{A}_6 : \prod_{\phi:o} . \text{Valid}(\Box\phi \supset \Box\Box\phi)$
  - $\text{MP}_T : \prod_{\phi_1,\phi_2:o} . \text{True}(\phi_1 \supset \phi_2) \to \text{True}(\phi_1) \to \text{True}(\phi_2)$
  - $\text{MP}_V : \prod_{\phi_1,\phi_2:o} . \text{Valid}(\phi_1 \supset \phi_2) \to \text{Valid}(\phi_1) \to \text{Valid}(\phi_2)$
  - $\text{Nec} : \prod_{\phi:o} . \text{Valid}(\phi) \to \text{Valid}(\Box\phi)$

THEOREM B.1.3 (ADEQUACY AND FAITHFULNESS 3)

The following hold:

1. There is a compositional surjection between proofs in the Hilbert-type system of S4 that $\phi_1, \ldots, \phi_n \vdash_t \psi$ and terms t such that:

$$p_1 : o, \ldots, p_n : o \vdash_{\Sigma_{H\Box}} t : \text{True}(\phi_1) \to \cdots \to \text{True}(\phi_n) \to \text{True}(\psi)$$

   Where $p_1, \ldots, p_n$ are the atomic variables of ϕ.

2. There is a compositional bijection between proofs in the Hilbert-type system of S4 that $\phi_1, \ldots, \phi_n \vdash_v \psi$ and terms t such that:

$$p_1 : o, \ldots, p_n : o \vdash_{\Sigma_{H\Box}} t : \text{Valid}(\phi_1) \to \cdots \to \text{Valid}(\phi_n) \to \text{Valid}(\psi)$$

Where $p_1, \ldots, p_n$ are the atomic variables of ϕ.

The above can be strengthened as follows:

- There is a compositional surjection between LF objects of type:

$$\text{Valid}(\phi_1) \to \cdots \to \text{Valid}(\phi_n) \to \text{True}(\psi_1) \to \cdots \text{True}(\psi_m) \to \text{True}(\vartheta)$$

and proofs that $\psi_1, \ldots, \psi_m \vdash_t \vartheta$ in the Hilbert-type system which is obtained by adding $\phi_1, \ldots, \phi_n$ as axioms to S4. □

B.1.4 Natural Deduction Style S4

The Signature $\Sigma_{\text{ND}\Box}$

- Syntactic Categories
  - o : Type
- Operations
  - $\bot$: o
  - $\supset : o \to o \to o$
  - $\Box : o \to o$
- Judgements
  - Taut : $o \to$ Type
  - Valid : $o \to$ Type
- Axioms and Rules

- $C : \prod_{\phi:o}. \text{Taut}(\phi) \rightarrow \text{Valid}(\phi)$
- $R : \prod_{\phi_1,\phi_2:o}.(\text{Taut}(\phi_1) \rightarrow \text{Valid}(\phi_2)) \rightarrow (\text{Valid}(\phi_1) \rightarrow \text{Valid}(\phi_2))$
- $\supset I_V : \prod_{\phi_1,\phi_2:o}. (\text{Valid}(\Box\phi_1) \rightarrow \text{Valid}(\phi_2)) \rightarrow \text{Valid}(\Box\phi_1 \supset \phi_2)$
- $\bot\ E : \prod_{\phi:o} \text{Taut}(\bot) \rightarrow \text{Taut}(\phi)$
- $\neg\neg E : \prod_{\phi:o} \text{Taut}((\phi \supset \bot) \supset \bot) \rightarrow \text{Taut}(\phi)$
- $\supset I_T : \prod_{\phi_1,\phi_2:o}. (\text{Taut}(\phi_1) \rightarrow \text{Taut}(\phi_2)) \rightarrow \text{Taut}(\phi_1 \supset \phi_2)$
- $\supset E_T : \prod_{\phi_1,\phi_2:o}. \text{Taut}(\phi_1 \supset \phi_2) \rightarrow \text{Taut}(\phi_1) \rightarrow \text{Taut}(\phi_2)$
- $\Box I : \prod_{\phi:o}. \text{Valid}(\phi) \rightarrow \text{Valid}(\Box\phi)$
- $\Box E : \prod_{\phi:o}. \text{Valid}(\Box\phi) \rightarrow \text{Valid}(\phi)$

THEOREM B.1.4 (ADEQUACY AND FAITHFULNESS 4)

Given a proof of a formula ϕ in Prawitz's system, there is an uniform way of constructing a corresponding term t of the LF (in the above signature) of type $\text{Valid}(\phi)$, in which the only free variables are those corresponding to the atomic variables of ϕ. Moreover, this construction provides a compositional surjection between assumptions free proofs in Prawitz and this sort of LF terms. □

B.1.5 The Classical Lambda Calculus

The Signature Σ_Λ

- Syntactic Categories
  - o : Type
- Operations
  - $\Lambda : o \rightarrow (o \rightarrow o)$
  - $\text{App} : o \rightarrow o \rightarrow o$
- Judgements

- $\bowtie : o \to o \to$ Type
- $= : o \to o \to$ Type

- Axioms and Rules
  - $\mathrm{E}_0 : \prod_{x:o} . \; x \bowtie x$
  - $\mathrm{E}_1 : \prod_{x,y:o} . \; x \bowtie y \to y \bowtie x$
  - $\mathrm{E}_2 : \prod_{x,y,z:o} . \; x \bowtie y \to y \bowtie z \to x \bowtie z$
  - $\mathrm{E}_3 : \prod_{x,y,x',y':o} . \; x \bowtie y \to x' \bowtie y' \to \mathrm{App}(x,x') \bowtie \mathrm{App}(y,y')$
  - $\beta : \prod_{x:o\to o} . \; \prod_{y:o} . \; \mathrm{App}(\Lambda(x),y) \bowtie xy$
  - $\xi : \prod_{x,y:o\to o} . \; (\prod_{z:o} . \; xz \bowtie yz) \to \Lambda(x) \bowtie \Lambda(y)$
  - $\mathrm{Con} : \prod_{x,y:o} . \; x \bowtie y \to x = y$
  - $\mathrm{E}_4 : \prod_{x:o} . \; x = x$
  - $\mathrm{E}_5 : \prod_{x,y:o} . \; x = y \to y = x$
  - $\mathrm{E}_6 : \prod_{x,y,z:o} . \; x = y \to y = z \to x = z$
  - $\mathrm{E}_7 : \prod_{x,y,x',y':o} . \; x = y \to x' = y' \to \mathrm{App}(x,x') = \mathrm{App}(y,y')$
  - $\beta : \prod_{x:o\to o} . \; \prod_{y:o} . \; \mathrm{App}(\Lambda(x),y) = xy$

THEOREM B.1.5 (ADEQUACY AND FAITHFULNESS 5)

The following hold:

1. $x_1 : o, \ldots, x_n : o \vdash M : o \quad \text{iff} \quad \Phi_\Gamma(M) \in \Lambda$
2. $x_1 : o, \ldots, x_n : o, \eta_1 : M_1 \bowtie N_1, \ldots, \eta_m : M_m \bowtie N_m \vdash P : M \bowtie N$

   iff

   $$\Phi_\Gamma(M_1) = \Phi_\Gamma(N_1), \ldots, \Phi_\Gamma(M_m) = \Phi_\Gamma(N_m) \vdash \Phi_\Gamma(M) = \Phi_\Gamma(N)$$

where $M \in \Xi_\Gamma(o)$, $\Xi_\Gamma(o)$ is the set of normal forms of type o in the context Γ,

$$\Gamma = x_1 : o, \ldots, x_n : o$$

and

$$\Phi_\Gamma : \Xi_\Gamma(o) \longrightarrow \Lambda[x_1, \ldots, x_n]$$

is a bijective function defined as follows

$$\Phi_\Gamma(M) = \begin{cases} x & \text{if } M = x \\ \Phi_\Gamma(M')\Phi_\Gamma(N) & \text{if } M = \mathrm{App}(M', N) \\ \lambda x.\Phi_{\Gamma, x:o}(M'[x]) & \text{if } M = \Lambda(\lambda x.M'[x]). \end{cases} \square$$

B.1.6 Call-by-value Lambda Calculus

The Signature Σ_{Λ_v}

- Syntactic Categories
  - o : Type
  - v : Type
- Operations
  - $! : v \to o$
  - $\Lambda_v : (v \to o) \to v$
  - $\mathrm{App} : o \to o \to o$
- Judgement
  - $= \ : o \to o \to$ Type
- Axioms and Rules
  - $\mathrm{E}_0 : \prod_{x:o} . \ x = x$
  - $\mathrm{E}_1 : \prod_{x,y:o} . \ x = y \to y = x$
  - $\mathrm{E}_2 : \prod_{x,y,z:o} . \ x = y \to y = z \to x = z$

- $\mathrm{E}_3 : \prod_{x,y,x',y':o} . \ x = y \rightarrow x' = y' \rightarrow \mathrm{App}(x, x') = \mathrm{App}(y, y')$
- $\beta_v : \prod_{x:v\rightarrow o} . \ \prod_{y:v} . \ \mathrm{App}(\Lambda_v(x)!, y!) = xy$
- $\xi_v : \prod_{x,y:v\rightarrow o} . \ (\prod_{z:v} . \ xz = yz) \rightarrow \Lambda_v(x)! = \Lambda_v(y)!$
- $\eta_v : \prod_{x:v} . \ \Lambda_v(\lambda y : v \ . \ \mathrm{App}(x!, y!) = x!$

THEOREM B.1.6 (ADEQUACY AND FAITHFULNESS 6)

The following hold:

1. $x_1 : v, \ldots, x_n : v \vdash M : o \quad \text{iff} \quad \Phi_\Gamma(M) \in \Lambda_v$
2. $x_1 : v, \ldots, x_n : v, \eta_1 : M_1 = N_1, \ldots, \eta_m : M_m = N_m \vdash P : M = N$

iff

$$\Phi_\Gamma(M_1) = \Phi_\Gamma(N_1), \ldots, \Phi_\Gamma(M_m) = \Phi_\Gamma(N_m) \vdash \Phi_\Gamma(M) = \Phi_\Gamma(N)$$

where $M \in \Xi_\Gamma(o)$, $\Xi_\Gamma(o)$ is the set of normal forms of type o in the context Γ,

$$\Gamma = x_1 : v, \ldots, x_n : v$$

and

$$\Phi_\Gamma : \Xi_\Gamma(o) \longrightarrow \Lambda_v[x_1, \ldots, x_n]$$

is a bijective function defined as follows

$$\Phi_\Gamma(M) = \begin{cases} x & \text{if } M = x! \\ \Phi_\Gamma(M')\Phi_\Gamma(N) & \text{if } M = \mathrm{App}(M', N) \\ \lambda x.\Phi_{\Gamma,x:v}(M'[x]) & \text{if } M = \Lambda_v(\lambda x.M'[x])! \end{cases} . \ \square$$

B.1.7 The Lambda I Calculus

The Signature Σ_{Λ_I}

- Syntactic Categories
  - o: Type

- Operations
  - $\bot$: o
  - $\Lambda_I : \prod_{x:o\to o} . \ x(\bot) = \bot \to o$
  - App : $o \to o \to o$
- Judgement
  - $= \ : o \to o \to$ Type
- Axioms and Rules
  - $\mathrm{E}_0 : \prod_{x:o} . \ x = x$
  - $\mathrm{E}_1 : \prod_{x,y:o} . \ x = y \to y = x$
  - $\mathrm{E}_2 : \prod_{x,y,z:o} . \ x = y \to y = z \to x = z$
  - $\mathrm{E}_3 : \prod_{x,y,x',y':o} . \ x = y \to x' = y' \to \mathrm{App}(x, x') = \mathrm{App}(y, y')$
  - $\bot_r$: $\prod_{x:o} . \ \mathrm{App}(x, \bot) = \bot$
  - $\bot_l$: $\prod_{x:o} . \ \mathrm{App}(\bot, x) = \bot$
  - $\bot_\Lambda$: $\bot = \Lambda_I(\lambda x : o \ . \ \bot, \mathrm{E}_0(\bot))$
  - $\beta_I : \prod_{x:o\to o} . \ \prod_{y:o} . \ \prod_{t:x(\bot)=\bot} . \ \mathrm{App}(\Lambda_I(x,t), y) = xy$
  - $\xi_I : \prod_{x,y:o\to o} . \ \prod_{t_1:x(\bot)=\bot} . \ \prod_{t_2:y(\bot)=\bot} .$

$$(\prod_{z:o} . \ x(z) = y(z)) \to \Lambda_I(x, t_1) = \Lambda_I(y, t_2)$$

THEOREM B.1.7 (ADEQUACY AND FAITHFULNESS 7)

The following hold:

1. $x_1 : o, \ldots, x_n : o \vdash M : o$ iff $\Phi_\Gamma(M) \in \Lambda_I$
2. $x_1 : o, \ldots, x_n : o, \eta_1 : M_1 = N_1, \ldots, \eta_m : M_m = N_m \vdash P : M = N$

iff

$$\Phi_\Gamma(M_1) = \Phi_\Gamma(N_1), \ldots, \Phi_\Gamma(M_m) = \Phi_\Gamma(N_m) \vdash \Phi_\Gamma(M) = \Phi_\Gamma(N)$$

where $M \in \Xi_\Gamma(o)$, $\Xi_\Gamma(o)$ is the set of normal forms of type o in the context Γ, in which $\perp$ occurs only within the second argument to the Λ_I-operator.

$$\Gamma = x_1 : o, \ldots, x_n : o$$

and

$$\Phi_\Gamma : \Xi_\Gamma(o) \longrightarrow \Lambda_I[x_1, \ldots, x_n]$$

is a surjective function defined as follows

$$\Phi_\Gamma(M) = \begin{cases} x & \text{if } M = x \\ \Phi_\Gamma(M')\Phi_\Gamma(N) & \text{if } M = \mathrm{App}(M', N) \\ \lambda x.\Phi_{\Gamma, x:o}(M'[x]) & \text{if } M = \Lambda_I(\lambda x : o.M'[x], t)\,. \end{cases} \square$$

B.1.8 Linear Lambda Calculus

The Signature Σ_{Λ_L}

In the following presentation of the signature we make, for typographical reasons, the following definition. We let

$$L = \lambda x : o \to o. \prod_{z,w:o} . \; x(z \vee w) = x(z) \vee x(w).$$

- Syntactic Categories
  - o : Type
- Operations
  - $\perp$: o
  - $\vee : o \to o \to o$
  - $\Lambda_L : \prod_{x:o\to o} . \; x(\perp) = \perp \to L(x) \to o$
  - $\mathrm{App} : o \to o \to o$
- Judgement
  - $=\ \ : o \to o \to$ Type

- Axioms and Rules

  - $\mathrm{E}_0 : \prod_{x:o} . \; x = x$
  - $\mathrm{E}_1 : \prod_{x,y:o} . \; x = y \to y = x$
  - $\mathrm{E}_2 : \prod_{x,y,z:o} . \; x = y \to y = z \to x = z$
  - $\mathrm{E}_3 : \prod_{x,y,x',y':o} . \; x = y \to x' = y' \to \mathrm{App}(x, x') = \mathrm{App}(y, y')$
  - $\bot_r$: $\prod_{x:o} . \; \mathrm{App}(x, \bot) = \bot$
  - $\bot_l$: $\prod_{x:o} . \; \mathrm{App}(\bot, x) = \bot$
  - $\bot_\Lambda$:$\bot = \Lambda_L(\lambda x : o \; . \; \bot, \mathrm{E}_0(\bot), \lambda y : o . \lambda z : o . \vee_{\mathrm{id}} (\bot))$
  - $\vee_{\mathrm{id}} : \prod_{x:o} . \; x = x \vee x$
  - $\vee_\bot : \prod_{x:o} . \; x \vee \bot = x$
  - $\vee_{\mathrm{sym}} : \prod_{x,y:o} . \; x \vee y = y \vee x$
  - $\vee_{\mathrm{assoc}} : \prod_{x,y,z:o} . \; (x \vee y) \vee z = x \vee (y \vee z)$
  - $\vee_= : \prod_{x,y,z:o} . \; x = y \to (x \vee z) = (y \vee z)$
  - $\vee_l : \prod_{x,y,z:o} . \; \mathrm{App}(x, y \vee z) = \mathrm{App}(x, y) \vee \mathrm{App}(x, z)$
  - $\vee_r : \prod_{x,y,z:o} . \; \mathrm{App}(x \vee y, z) = \mathrm{App}(x, z) \vee \mathrm{App}(y, z)$
  - $\vee_\Lambda : \prod_{x,y:o\to o} \cdot \prod_{t_1:x(\bot)=\bot} \cdot \prod_{t_2:y(\bot)=\bot} \cdot \prod_{t_3:L(x)} \cdot \prod_{t_4:L(y)} \cdot$
    $\prod_{t_5:y(\bot)\vee z(\bot)=\bot} \cdot \prod_{t_6:L(\lambda z:o.x(z)\vee y(z))} \cdot$
    $\Lambda_L(x, t_1, t_3) \vee \Lambda_L(y, t_2, t_4) = \Lambda_L(\lambda z : o \; . \; x(z) \vee y(z), t_5, t_6)$
  - $\beta_L : \prod_{x:o\to o} \cdot \prod_{y:o} \cdot \prod_{t_1:x(\bot)=\bot} \cdot \prod_{t_2:L(x)} \cdot \mathrm{App}(\Lambda_L(x, t_1, t_2), y) = xy$
  - $\xi_L : \prod_{x,y:o\to o} \cdot \prod_{t_1:x(\bot)=\bot} \cdot \prod_{t_2:y(\bot)=\bot} \cdot \prod_{t_3:L(x)} \cdot \prod_{t_4:L(y)} \cdot$
    $(\prod_{z:o} . \; x(z) = y(z)) \to \Lambda_L(x, t_1, t_3) = \Lambda_L(y, t_2, t_4)$

THEOREM B.1.8 (ADEQUACY AND FAITHFULNESS 8)

The following hold:

1. $\vdash M : o$ iff $\Phi_\emptyset(M) \in \Lambda_L^\circ$

2. $\eta_1 : M_1 = N_1, \ldots, \eta_m : M_m = N_m \vdash P : M = N$

iff

$$\Phi_\emptyset(M_1) = \Phi_\emptyset(N_1), \ldots, \Phi_\emptyset(M_m) = \Phi_\emptyset(N_m) \vdash \Phi_\emptyset(M) = \Phi_\emptyset(N)$$

where $M \in \Xi_\Gamma(o)$, $\Xi_\Gamma(o)$ is the set of normal forms of type o in the context Γ, in which $\perp$ occurs only within the second argument and $\vee$ occurs only in the third arguement to the Λ_L-operator.

$$\Gamma = x_1 : o, \ldots, x_n : o$$

and

$$\Phi_\Gamma : \Xi_\Gamma(o) \longrightarrow \Lambda_L^*[x_1, \ldots, x_n]$$

is a surjective function defined as follows

$$\Phi_\Gamma(M) = \begin{cases} x & \text{if } M = x \\ \Phi_\Gamma(M')\Phi_\Gamma(N) & \text{if } M = \mathrm{App}(M', N) \\ \lambda x.\Phi_{\Gamma, x:o}(M'[x]) & \text{if } M = \Lambda_L(\lambda x : o.M'[x], t, s) \,. \end{cases} \quad \square$$

The Signature Σ_{HL}

- Syntactic Categories
  - l : Type
  - ι : Type
  - o : Type
  - w : Type
  - h : Type
- Operations
  - $! : l \to \iota$
  - $0 : \iota$
  - $1 : \iota$

- $+ : \iota \to \iota \to \iota$
- $= \ : \iota \to \iota \to o$
- $\neg : o \to o$
- $\supset : o \to o \to o$
- $\forall : (\iota \to o) \to o$
- $\ldots := \ldots : l \to \iota \to w$
- $\ldots ; \ldots : w \to w \to w$
- $\mathrm{if} : \prod_{e:o} . \ \mathtt{QF}(e) \to w \to w \to w$
- $\mathrm{while} : \prod_{e:o} . \ \mathtt{QF}(e) \to w \to w$
- $\{\ldots\} \ldots \{\ldots\} : o \to w \to o \to h$

- Judgements
  - $\vdash_o : o \to \ \mathrm{Type}$
  - $\vdash_h : h \to \ \mathrm{Type}$
  - $\neq \ : l \to l \to \ \mathrm{Type}$
  - $\sharp_\iota : l \to \iota \to \ \mathrm{Type}$
  - $\sharp_o : l \to o \to \ \mathrm{Type}$
  - $\mathtt{QF} : o \to \ \mathrm{Type}$

- Axioms and Rules
  - $\sharp_0 : \prod_{x:l} . \ x \sharp_\iota 0$
  - $\sharp_1 : \prod_{x,y:l} . \ x \neq y \to x \sharp_\iota y!$
  - $\sharp_2 : \prod_{x:l} . \ \prod_{t_1,t_2:\iota} . \ x \sharp_\iota t_1 \to x \sharp_\iota t_2 \to x \sharp_\iota (t_1 + t_2)$
  - $\sharp_3 : \prod_{x:l} . \ \prod_{t_1,t_2:\iota} . \ x \sharp_\iota t_1 \to x \sharp_\iota t_2 \to x \sharp_o (t_1 = t_2)$
  - $\sharp_4 : \prod_{x:l} . \ \prod_{\phi:o} . \ x \sharp_o \phi \to x \sharp_o \neg \phi$
  - $\sharp_5 : \prod_{x:l} . \ \prod_{\phi_1,\phi_2:o} . \ x \sharp_o \phi_1 \to x \sharp_o \phi_2 \to x \sharp_o (\phi_1 \supset \phi_2)$
  - $\sharp_6 : \prod_{\phi:\iota\to o} . \ \prod_{x,y:l} . \ (x \neq y \to x \sharp_o \Phi(y!)) \to x \sharp_o \forall \Phi$

- $\mathtt{QF}_0 : \Pi_{t_1:\iota} \ . \ \Pi_{t_2:\iota}.\mathtt{QF}(t_1 = t_2)$,
- $\mathtt{QF}_2 : \Pi_{\phi_1:o}.\mathtt{QF}(\phi_1) \to \mathtt{QF}(\neg\phi_1)$
- $\mathtt{QF}_3 : \Pi_{\phi_1:o}\Pi_{\phi_2:o} \quad \mathtt{QF}(\phi_1) \to \mathtt{QF}(\phi_2) \to \mathtt{QF}(\phi_1 \supset \phi_2)$
- Ass : $\Pi_{x:l} \cdot \Pi_{t:\iota} \cdot \Pi_{\phi:\iota\to o} \cdot x\sharp_o \forall\Phi \to (\vdash_h \{\Phi(t)\}x := t\{\Phi(x!)\})$
- Seq : $\Pi_{\phi_0,\phi_1,\phi_2:o}\Pi_{w_1,w_2:w}$

$$\vdash_h \{\phi_0\}w_1\{\phi_1\} \to \quad \vdash_h \{\phi_1\}w_2\{\phi_2\} \to \quad \vdash_h \{\phi_0\}w_1; w_2\{\phi_2\}$$

- If : $\Pi_{\phi:o}\Pi_{\phi_1:o}\Pi_{\phi_2:o}\Pi_{w_1:w}\Pi_{w_2:w}\Pi_{p:\mathtt{QF}(\phi)}$

$$\vdash_h \{\phi_1 \wedge \phi\}w_1\{\phi_2\} \to \vdash_h \{\phi_1 \wedge \neg\phi\}w_2\{\phi_2\} \to \vdash_h \{\phi_1\}\mathrm{if}(\phi, p, w_1, w_2)\{\phi_2\}$$

- While : $\Pi_{\phi:o}\Pi_{\phi_1:o}\Pi_{w_1:w}\Pi_{p:\mathtt{QF}(\phi)}$

$$\vdash_h \{\phi_1 \wedge \phi\}w_1\{\phi_1\} \to \quad \vdash_h \{\phi_1\}\mathrm{while}(\phi, p, w_1)\{\neg\phi_1\}$$

- Con : $\Pi_{\phi_1:o}\Pi_{\phi_1':o}\Pi_{\phi_2:o}\Pi_{\phi_2':o}\Pi_{w_1:w}$

$$\vdash_o \phi_1 \Rightarrow \phi_1' \to \quad \vdash_o \phi_2' \Rightarrow \phi_2 \to \quad \vdash_h \{\phi_1'\}w_1\{\phi_2'\} \to \quad \vdash_h \{\phi_1\}w_1\{\phi_2\}$$

THEOREM B.1.9 (ADEQUACY AND FAITHFULNESS 9)

Let Γ_n^m be the following context, for m, n integers:

$$\Gamma_n^m = \{y_0 : \iota, \ldots, y_m : \iota, x_0 : l, \ldots, x_n : l, z_{i,j} : x_i \neq x_j, z_{i,j}^* : x_j \neq x_i\}_{0\le i\le j\le n}.$$

In the above LF signature and in the context Γ_n^m we have the following facts concerning syntax:

- l : All well-formed long $\beta\eta$-normal forms of type l are LF variables of type l, and hence are among the $x_0, \ldots, x_n$.
- There are compositional bijections τ_K, for each syntactic category K, which map long $\beta\eta$-normal forms of type K to:
  - well-formed terms of the assertion language built up from the set of identifiers $\bar{x}$ and the logical variables $\bar{y}$, in the case when $K = \iota$.

- formulas of the assertion language built up from the set of identifiers $\bar{x}$ (which if they occur must occur free) and the logical variables $\bar{y}$, in the case when $K = o$.
- while programs of τ built up from the set of identifiers $\bar{x}$ and the logical variables $\bar{y}$ (which do not occur in the left hand side of any assignment statement), in the case when $K = w$.
- asserted programs (*i.e.* Hoare triples) built up from the set of identifiers $\bar{x}$ and the logical variables $\bar{y}$ (here again no variable from $\bar{y}$ can occur in the left hand side of any assignment statement), in the case of $K = h$.

- There is a compositional bijection between proofs of a Hoare triple $\{p\}S\{q\}$ from assumptions $r_0, \ldots, r_s$ (in the assertion language) and assumptions $\{p_0\}S_0\{q_0\}, \ldots, \{p_t\}S_t\{q_t\}$ (concerning asserted programs) and well-formed $\beta\eta$-normal forms of type

$$\vdash_h \{p\}S\{q\}$$

in the above signature and in the context Γ where

$$\Gamma = \Gamma_n^m \cup \{w_j : (\vdash_o r_j), v_i : (\vdash_h \{p_i\}S_i\{q_i\})\}_{0 \leq j \leq s, 0 \leq i \leq t},$$

and Γ_n^m is adequate for the syntax of objects involved. □

B.1.9 Two-register Machine Hoare Logic

The Signature Σ_{HL^2}

- Syntactic Categories
  - l : Type
  - ι : Type
  - o : Type
  - w : Type
  - h : Type

- Operations
  - $! : l \to \iota$
  - $0 : \iota$
  - $+ : \iota \to \iota \to \iota$
  - $= \ : \iota \to \iota \to o$
  - $\neg : o \to o$
  - $\supset : o \to o \to o$
  - $\forall : (\iota \to o) \to o$
  - $\ldots := \ldots : l \to \iota \to w$
  - $\ldots ; \ldots : w \to w \to w$
  - $\{\ldots\}\ldots\{\ldots\} : o \to w \to o \to h$
- Judgements
  - $\vdash$: $(l \to l \to h) \to$ Type
- Axioms and Rules
  - $\mathrm{Ass}_1 : \Pi_{t:l\to l\to\iota} \cdot \Pi_{\phi:\iota\to\iota\to o} \cdot$

    $$\vdash \lambda x : l \ . \ \lambda y : l.\{\Phi(t(x,y),y!)\}x := t(x,y)\{\Phi(x!,y!)\}$$
  - $\mathrm{Ass}_2 : \Pi_{t:l\to l\to\iota} \cdot \Pi_{\phi:\iota\to\iota\to o} \cdot$

    $$\vdash \lambda y : l \ . \ \lambda x : l.\{\Phi(t(x,y),y!)\}x := t(x,y)\{\Phi(x!,y!)\}$$
  - $\mathrm{Seq} : \Pi_{\phi_0,\phi_1,\phi_2:l\to l\to o} \cdot \Pi_{w_1,w_2:l\to l\to w} \cdot$

    $$(\vdash \lambda x : l \ . \ \lambda y : l.\{\phi_0(x,y)\}w_1(x,y)\{\phi_1(x,y)\}) \to$$
    $$(\vdash \lambda x : l \ . \ \lambda y : l.\{\phi_1(x,y)\}w_2(x,y)\{\phi_2(x,y)\}) \to$$
    $$(\vdash \lambda x : l \ . \ \lambda y. : l.\{\phi_0(x,y)\}w_1(x,y);w_2(x,y)\{\phi_2(x,y)\})$$

The adequacy and faithfulness property for this signature follows the usual pattern.

Bibliography

[Ac77] Aczel, P. *An Introduction to Inductive Definitions.* Handbook of Mathematical Logic. North–Holland, 1977.

[An71] Andrews, P. *Resolution in Type Theory.* J. Symbolic Logic 36(3)(1971), pp. 414-432.

[An81] Andrews, P. *Theorem-proving via general matings.* J. Assoc. Comp. Mach. 28(2)(1981), pp. 193-214.

[Av87a] Avron, A. *Simple Consequence Relations.* University of Edinburgh report ECS-LFCS-87-30, 1987.

[Av87b] Avron, A. *Internalizing S4 and S5 in the* LF. Proc. Workshop on General Logic, Edinburgh, 1987. University of Edinburgh report ECS-LFCS-88-52, 1987.

[Av87c] Avron, A. *The Foundations and Proof Theory of Three–valued Logics.* University of Edinburgh report ECS-LFCS-88-48, 1987.

[Av87d] Avron, A. *The Semantics and Proof Theory of Linear Logic.* University of Edinburgh report ECS-LFCS-87-27, 1987.

[Av87e] Avron, A. *Gentzenizing Schroeder–Heister's Natural Extension of Natural Deduction.* University of Edinburgh report ECS-LFCS-87-20, 1987.

[AHM87] Avron, A., Honsell, F., Mason, I. *Using Typed Lambda Calculus to Implement Formal Systems on a Machine.* University of Edinburgh report ECS-LFCS-87-31, 1987.

[Ba89] Barendregt, H. *Introduction to Generalized Type Systems.* Preprint, 1989.

[BR83] Barendregt, H. Rezus, A. *Semantics for Classical Automath and Related Systems.* Information and Control, 1983.

[Be88] Bell, J.L. *Toposes and Local Set Theories.* Oxford University Press, 1988.

[Bi81] Bibel, W. *Computationally Improved Versions of Herbrand's Theorem.* In Proc. Herbrand Symposium, Logic Colloquium '81 (editor J.P. Stern). North–Holland, 1982.

[Bir48] Birkhoff, G. *Lattice Theory.* American Mathematical Society, 1948.

[Bu83] Bundy, A. *The Computer Modelling of Mathematical Reasoning.* Academic Press, 1983.

[BW84] Barr, M., Wells, C. *Toposes, Triples and Theories.* Springer-Verlag, 1984.

[Ca78] Cartmell, J. *Generalized algebraic theories and contextual categories.* D.Phil. thesis. University of Oxford, 1978.

[Ca86] Cartmell, J. *Generalized Algebraic Theories and Contextual Categories.* Annals of Pure and Applied Logic 32(1986), pp. 209-243.

[Ch40] Church, A. *A Formulation of the Simple Theory of Types.* J. Symbolic Logic 5(1940), pp.56-68.

[CM84] Clocksin, W., Mellish, C. *Programming in Prolog.* Springer 1984.

[CM85] Charniak, E., McDermott, D. *Introduction to Artificial Intelligence.* Addison–Wesley, 1985.

[Co85] Coquand, Th. *Une théorie des constructions.* Thèse de Troisième Cycle. Université de Paris VII, 1985.

[CH85] Coquand, Th., Huet, G. *Constructions: a higher–order proof system for mechanizing mathematics.* In: EUROCAL '85: European Conf. on Computer Algebra. Springer–Verlag, 1985.

[Cu52] Curry, H.B. *The permutability of rules in the classical inferential calculus.* J. Symbolic Logic 17(1952), pp. 245-248.

[vDa80] van Dalen, D. *Logic and Structure.* Springer–Verlag, 1980.

[vD80] van Daalen, D. T. *The Language Theory of AUTOMATH.* Ph.D thesis. Technical University of Eindhoven, 1980.

[De79] Devlin, K.J. *Fundamentals of Contemporary Set Theory.* Springer–Verlag, 1979.

[DHR86] Dezani, M., Honsell, F., Ronchi della Rocca, S. *Models for the Theories of functions Depending Strictly on all Their Arguments.* Abstract, J. Symbolic Logic 51(3), 1986.

[Eh88] Ehrhard, T. *Une sémantique catégorique des types dépendents. Application au Calcul des Constructions.* Thèse, Univ. Paris VII, 1988.

[El89] Elliot, C. *Higher-order unification with dependent function types.* In: Proc. Rewriting Techniques and Applications, Chapel Hill, NC, 1989.

[Fe87] Felty, A. *Implementing Theorem Provers in Logic Programming.* University of Pennsylvania report, MS-CIS-87-109, 1987.

[Fi87] Fitting, M. *Computability Theory, Semantics, and Logic Programming.* Oxford University Press, 1987.

[Ge34] Gentzen, G. *Investigations into Logical Deduction.* In: [Sz69].

[Ge34] Gentzen, G. *Untersuchungen über das logische Schliessen.* Mathematische Zeitschrift 39(1934), pp. 176-210, 405-431.

[Gi87] Girard, J–Y. *Linear Logic.* Th. Comp. Sci. 50, 1987.

[Gi89] Girard, J–Y. *Proofs and Types.* Cambridge University Press, 1989.

[Har89] Harland, J. *Ph.D thesis.* Forthcoming, University of Edinburgh, 1990.

[He30] Herbrand, J. *Recherches sur la théorie de la démonstration.* Thèse. Univesité de Paris, 1930. In : Ecrits Logique de Jacques Herbrand, PUF Paris, 1968.

[Ha87] Harper, R. *An Equational Formulation of* LF. University of Edinburgh report ECS-LFCS-88-67, 1987.

[HHP87] Harper, R., Honsell, F., Plotkin, G. *A Framework for Defining Logics.* Proc. 2nd. LICS. Ithaca, NY, June 1987.

[HHP89] Harper, R., Honsell, F., Plotkin, G. *A Framework for Defining Logics.* Submitted to the J. Assoc. Comp. Mach., 1989.

[HS86] Hindley, J.R., Seldin, J.P. *Introduction to Combinators and λ-calculus.* London Mathematical Society Student Texts 1. Cambridge University Press, 1986.

[Hod77] Hodges, W. *Logic.* Penguin, 1987.

[Hon87a] Honsell, F. *The Metatheory of the* LF *Type System.* Proc. Workshop on General Logic, Edinburgh 1987. University of Edinburgh report ECS-LFCS-88-52.

[Hon87b] Honsell, F. *Personal Communication.* Edinburgh, 1987.

[Hor51] Horn, A. *On sentences which are the direct union of algebras.* J. Symbolic Logic 16(1951).

[How80] Howard, W.A. *The formulae-as-types notion of construction.* In: To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism (editors J.P. Seldin & J.R. Hindley). Academic Press, 1980.

[Hu75] Huet, G. *A Unification Algorithm for Typed λ-calculus.* Th. Comp. Sci. 1(1975), pp. 27-57.

[Hu76] Huet, G. Résolution d'Équations dans des Languages d'Ordre 1, 2, ..., ω. Thèse de Doctorat d'État. Université de Paris VII, 1976.

[Hu86] Huet, G. *Formal Structures for Deduction and Computation.* Post-graduate Lecture Notes, Carnegie-Mellon University, First Edition, May 1986.

[HP87] Hyland, J.M.E., Pitts, A. *The Calculus of Constructions: Categorical Semantics and Topos-theoretic Models.* Proc. Boulder, 1987.

[JP76] Jensen, D. Pietrzykowski, T. *Mechanizing ω-order Type Theory Through Unification.* Th. Comp. Sci. 3(1976), pp. 123-171.

[Jo77] Johnstone, P.T. *Topos Theory.* Academic Press, 1977.

[Kl52] Kleene, S.C. *Permutability of Inferences in Gentzen's Calculi LK and LJ.* Mem. Amer. Math. Soc., 10(1952), pp. 1-26.

[Kl68] Kleene, S.C. *Mathematical Logic.* Wiley and Sons, 1968.

[Kl52] Kleene, S.C. *Introduction to Metamathematics.* North–Holland, 1952.

[Ko69] Kowalski, R. *Search Strategies for Theorem Proving.* Machine Intelligence 5. Edinburgh University Press, 1969.

[Ko79] Kowalski, R. *Logic for Problem Solving.* North–Holland, 1979.

[La63] Lawvere, F.W. *Functorial semantics of algebraic theories.* Proc. Nat. Acad. Sci. 50(1963), pp. 869-873.

[LS86] Lambek, J., Scott, P. *Introduction to Higher–order Categorical Logic.* Cambridge University Press, 1986.

[Ll84] Lloyd, J.W. *Foundations of Logic Programming.* Springer 1984.

[Ma71] MacLane, S. *Categories for the Working Mathematician.* Springer–Verlag, 1971.

[ML71a] Martin–Löf, P. *Hauptsatz for the Intuitionistic Theory of Iterated Inductive Definitions.* Proc. Second Scandinavian Logic Symposium. North–Holland, 1971.

[ML71b] Martin–Löf, P. *Hauptsatz for the Theory of Species.* Proc. Second Scandinavian Logic Symposium. North–Holland, 1971.

[ML73] Martin–Löf, P. *An intuitionistic theory of types: predicative part.* In: H.E. Rose and Shepherdson, editors, Logic Colloquium '73. North–Holland, 1973.

[ML82] Martin–Löf, P. *Constructive Mathematics and Computer Programming.* 6th. Annual Congress for Logic Methodology and Philosophy of Science. North–Holland, 1982.

[ML85] Martin–Löf, P. *On the meanings of the logical constants and the justifications of the logical laws.* Technical Report 2, Scuola di Specializzazione in Logica Matematica, Università di Siena, 1985.

[Ma87] Mason, I. *Hoare's Logic in the* LF. University of Edinburgh report, ECS-LFCS-87-32, 1987.

[Me82] Meyer, A. *What is a Model of the Lambda Calculus ?* Information and Control 52(1982), pp. 87-122.

[MR86] Meyer, A., Reinhold, M. *'Type' is not a type: preliminary report.* In: Proceedings of the 13th. ACM Symposium on the Principles of Programming Languages, 1986.

[Mi83] Miller, D. *Proofs in Higher-order Logic.* Ph.D thesis. Carnegie-Mellon University, 1983.

[Mi87] Miller, D. *A Compact Representation of Proofs.* University of Pennsylvania, MS-CIS-87-30, 1987.

[Mi89] Miller, D. *A Logical Analysis of Modules in Logic Programming.* J. Logic Programming, 6(1&2)(1989), pp. 79-108.

[MN86] Miller, D., Nadathur, G. *Higher–order Logic Programming.* University of Pennsylvania report, MS-CIS-86-17, 1986.

[MNPS89] Miller, D., Nadathur, G., Pfenning, F., Scedrov, A. *Uniform Proofs as a Foundation for Logic Programming.* University of Pennsylvania, MS-CIS-89-36.

[Mil84] Milner, R. *The use of machines to assist in rigorous proof.* Phil. Trans. R. Soc. Lond. 312, 1984.

[Na87] Nadathur, G. *A Higher–order Logic as the Basis for Logic Programming.* University of Pennsylvania, MS-CIS-87-48.

[NM88] Nadathur, G., Miller, D. *An Overview of λProlog.* Duke University report, CS-1988-14, 1988.

[NPS86] Nordström, B., Petersson, K., Smith, J. *An Introduction to Martin–Löf's Type Theory.* University of Göteborg, 1986.

[Pa86] Paulson, L.C. *Natural Deduction Proof as Higher–order Resolution.* J. Logic Programming 3(1986), pp. 237-258.

[Pl72] Plotkin, G. *Building-in Equational Theories.* Machine Intelligence 7, pp. 73-90. Elsevier, New York, 1972.

[Pl75] Plotkin, G. *Call–by–name, Call–by–value and the λ–calculus.* Th. Comp. Sci. 1(1975), pp. 125-129.

[Pl87] Plotkin, G. *Towards Search Spaces for the Logical Framework.* Proc. Workshop on General Logic, Edinburgh 1987. University of Edinburgh report ECS-LFCS-88-52.

[Po89] Pollack, R. *Ph.D thesis.* Forthcoming, University of Edinburgh.

[Pr65] Prawitz, D. *Natural Deduction: A Proof–theoretical Study.* Almqvist & Wiksell, Stockholm, 1965.

[PW90] Pym, D., Wallen, L. *Investigations into Proof-search in a System of First-order Dependent Function Types.* Proc. 10th International Conference on Automated Deduction, Kaiserslautern, FRG. July 1990, Springer-Verlag. (Also published as: University of Edinburgh report, ECS-LFCS-90-111.)

[Ro65] Robinson, J. *A machine–oriented logic based on the resolution principle.* J. Assoc. Comp. Mach. 12, 1965.

[RS87] Rydeheard, D., Stell, J. *Foundations of Equational Deduction: A Categorical Treatment of Equational Proofs and Unification Algorithms.* Proc. Summer Conf. on Category Theory and Computer Science. University of Edinburgh, 1987.

[Sa90] Salvesen, A. Manuscript (to appear as an LFCS report), University of Edinburgh, 1990.

[SH84] Schroeder–Heister, P. *A Natural Extension of Natural Deduction.* J. Symbolic Logic, 49(4)(1984), pp. 1284-1299.

[Sc77] Schütte, K. *Proof Theory.* Springer, 1977.

[Se77] Seely, R.A.G. *Hyperdoctrines and natural deduction.* Ph.D thesis. University of Cambridge, 1977.

[Se83] Seely, R.A.G. *Hyperdoctrines, natural deduction and the Beck condition.* Zeitschr. f. math. Logik und Grundlagen d. Math. 29(1983).

[Se84] Seely, R.A.G. *Locally cartesian closed categories and type theory.* Math. Proc. Cam. Phil. Soc., 95(1984), pp. 33-48.

[Sh67] Schoenfield, J.R. *Mathematical Logic.* Addison–Wesley, Reading, Massachusetts, 1967

[Sch86] Schmidt, D. *Denotational Semantics.* Allyn and Bacon, 1986.

[Sm73] Smorynski, C. *Applications of Kripke models.* Chapter V of [Tr73].

[Ste72] Stenlund, S. *Combinators, λ-terms and Proof Theory.* D. Reidel, Dordrecht, Holland, 1972.

[St88] Streicher, T. *Correctness and Completeness of a Categorical Semantics of the Calculus of Constructions.* Ph.D thesis. Passau, 1988.

[Sz69] Szabo, M.E. *The Collected Papers of Gerhard Gentzen.* North–Holland, 1969.

[Sz78] Szabo, M.E. *Algebra of Proofs.* North–Holland, 1978.

[Tä77] Tärnlund, S–Å. *Horn Clause Computability.* BIT 17 (1977).

[Ta86] Taylor, P. *Recursive Domains, Indexed Category Theory and Polymorphism.* Ph.D thesis. University of Cambridge, 1986.

[Tr73] Troelstra, A.S. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis.* Lecture Notes in Mathematics 344. Springer, 1973.

[Wa87] Wallen, L. A. *Automated proof–search in non-classical logics.* Ph.D thesis. University of Edinburgh, 1987.

List of Definitions