Lectures on the Curry-Howard Isomorphism

Morten Heine B. Sørensen
University of Copenhagen

Paweł Urzyczyn
University of Warsaw

Preface

The Curry-Howard isomorphism states an amazing correspondence between systems of formal logic as encountered in *proof theory* and computational calculi as found in *type theory*. For instance, minimal propositional logic corresponds to simply typed λ-calculus, first-order logic corresponds to dependent types, second-order logic corresponds to polymorphic types, etc.

The isomorphism has many aspects, even at the syntactic level: formulas correspond to types, proofs correspond to terms, provability corresponds to inhabitation, proof normalization corresponds to term reduction, etc.

But there is much more to the isomorphism than this. For instance, it is an old idea—due to Brouwer, Kolmogorov, and Heyting, and later formalized by Kleene's realizability interpretation—that a constructive proof of an implication is a procedure that transforms proofs of the antecedent into proofs of the succedent; the Curry-Howard isomorphism gives syntactic representations of such procedures.

These notes give an introduction to parts of proof theory and related aspects of type theory relevant for the Curry-Howard isomorphism.

Outline

Since most calculi found in type theory build on λ-calculus, the notes begin, in Chapter 1, with an introduction to *type-free λ-calculus.* The introduction derives the most rudimentary properties of β-reduction including the Church-Rosser theorem. It also presents Kleene's theorem stating that all recursive functions are λ-definable and Church's theorem stating that β-equality is undecidable.

As explained above, an important part of the Curry-Howard isomorphism is the idea that a constructive proof of an implication is a certain procedure. This calls for some elaboration of what is meant by constructive proofs, and Chapter 2 therefore presents *intuitionistic propositional logic.* The chapter presents a natural deduction formulation of minimal and intuitionistic propositional logic. The usual semantics in terms of Heyting algebras and in terms of Kripke models are introduced—the former explained

on the basis of Boolean algebras—and the soundness and completeness results are then proved. An informal proof semantics, the so-called BHK-interpretation, is also presented.

Chapter 3 presents the *simply typed λ-calculus* and its most fundamental properties up to the subject reduction property and the Church-Rosser property. The distinction between simply typed λ-calculus à la Church and à la Curry is introduced, and the uniqueness of types property—which fails for the Curry system—is proved for the Church system. The equivalence between the two systems, in a certain sense, is also established. The chapter also proves the weak normalization property by the Turing-Prawitz method, and ends with Schwichtenberg's theorem stating that the numeric functions representable in simply typed λ-calculus are exactly the extended polynomials.

This provides enough background material for our first presentation of the *Curry-Howard isomorphism* in Chapter 4, as it appears in the context of natural deduction for minimal propositional logic and simpy typed λ-calculus. The chapter presents another formulation of natural deduction, which is often used in the proof theory literature, and which facilitates a finer distinction between similar proofs. The exact correspondence between natural deduction for minimal propositional logic and simply typed λ-calculus is then presented. The extension to product and sum types is also discussed. After a brief part on proof-theoretical applications of the weak normalization property, the chapter ends with a proof of strong normalization using the Tait-Girard method, here phrased in terms of saturated sets.

Chapter 5 presents the variation of the Curry-Howard isomorphism in which one replaces natural deduction by *Hilbert style proofs* and simply typed λ-calculus by simply typed *combinatory logic.* After type-free combinators and weak reduction—and the Church-Rosser property—the usual translations from λ-calculus to combinators, and vice versa, are introduced and shown to preserve some of the desired properties pertaining to weak reduction and β-reduction. Then combinators with types are introduced, and the translations studied in this setting. Finally Hilbert-style proofs are introduced, and the connection to combinators with types proved. The chapter ends with a part on subsystems of combinators in which relevance and linearity play a role.

Having seen two logics or, equivalently, two calculi with types, Chapter 6 then studies *decision problems* in these calculi, mainly the type checking, the type reconstruction, and the type inhabitation problem. The type reconstruction problem is shown to be P-complete by reduction to and from unification (only the reduction *to* unification is given in detail). The type inhabitation problem is shown to be PSPACE-complete by a reduction from the satisfiability problem for classical second-order propositional formulas. The chapter ends with Statman's theorem stating that equality on typed terms is non-elementary.

After introducing natural deduction systems and Hilbert-style systems, the notes introduce in Chapter 7 Gentzen's sequent calculus systems for propositional logic. Both classical and intuitionistic variants are introduced. In both cases a somewhat rare presentation—taken from Prawitz—with assumptions as sets, not sequences, is adopted. For the intuitionistic system the cut-elimination theorem is mentioned, and from this the subformula property and decidability of the logic are inferred. Two aproaches to term assignment for sequent calculus proofs are studied. In the first approach, the terms are those of the simply typed λ-calculus. For this approach, the connection between normal forms and cut-free proofs is studied in some detail. In the second approach, the terms are intended to mimic exactly the rules of the calculus, and this assignment is used to prove the cut-elimination theorem in a compact way.

The remaining chapters study variations of the Curry-Howard isomorphism for more expressive type systems and logics.

In Chapter 8 we consider the most elementary connections between natural deduction for *classical propositional logic* and simply typed λ-calculus with *control operators,* in particular, the correspondence between classical proof normalization and reduction of control operators. Kolmogorov's embedding of classical logic into intuitionistic logic is shown to induce a continuation passing style translation which eliminates control operators.

Chapter 9 is about *first-order logic.* After a presentation of the syntax for quantifiers, the proof systems and interpretations seen in earlier chapters are generalized to the first-order case.

Chapter 10 presents *dependent types,* as manifest in the calculus λP. The strong normalization property is proved by a translation to simply typed λ-calculus. A variant of λP à la Curry is introduced. By another translation it is shown that a term is typable in λP à la Curry iff it is typable in simply typed λ-calculus. While this shows that type reconstruction is no harder than in simply typed λ-calculus, the type checking problem in λP à la Curry turns out to be undecidable. The last result of the chapter shows that first-order logic can be encoded in λP.

In Chapter 11 we study *arithmetic.* The chapter introduces Peano Arithmetic (PA) and briefly recalls Gödel's theorems and the usual result stating that exactly the recursive functions can be represented in Peano Arithmetic. The notion of a provably total recursive function is also introduced. Heyting arithmetic (HA) is then introduced and Kreisel's theorem stating that provable totality in HA and PA coincide is presented. Then Kleene's realizability interpretation is introduced—as a way of formalizing the BHK-interpretation—and used to prove consistency of HA. Gödel's system **T** is then introduced and proved to be strongly normalizing. The failure of arithmetization of proofs of this property is mentioned. The result stating that the functions definable in **T** are the functions provably total in Peano Arithmetic is also presented. Finally, Gödel's *Dialectica* interpretation is

presented and used to prove consistency of HA and to prove that all functions provably total in Peano Arithmetic are definable in **T**.

Chapter 12 is about *second-order logic* and *polymorphism.* For the sake of simplicity, only second-order propositional systems are considered. Natural deduction, Heyting algebras, and Kripke models are extended to the new setting. The polymorphic λ-calculus is then presented, and the correspondence with second-order logic developed. After a part about definability of data types, a Curry version of the polymorphic λ-calculus is introduced, and Wells' theorem stating that type reconstruction and type checking are undecidable is mentioned. The strong normalization property is also proved.

The last chapter, Chapter 13, presents the λ-*cube* and *pure type systems.* First Barendregt's cube is presented, and its systems shown equivalent to previous formulations by means of a classification result. Then the cube is geneneralized to pure type systems which are then developed in some detail.

About the notes

Each chapter is provided with a number of exercises. We recommend that the reader try as many of these as possible. At the end of the notes, answers and hints are provided to some of the exercises.[1]

The notes cover material from the following sources:

- Girard, Lafont, Taylor: *Proofs and Types,* Cambridge Tracts in Theoretical Computer Science 7, 1989.
- Troelstra, Schwichtenberg: *Basic Proof Theory,* Cambridge Tracts in Theoretical Computer Science 43, 1996.
- Hindley: *Basic Simple Type Theory,* Cambridge Tracts in Theoretical Computer Science 42, 1997.
- Barendregt: Lambda Calculi with Types, pages 117–309 of *Abramsky, S. and D.M. Gabbay and T.S.E. Maibaum*, editors, *Handbook of Logic in Computer Science*, Volume II, Oxford University Press, 1992.

Either of these sources make excellent supplementary reading.

The notes are largely self-contained, although a greater appreciation of some parts can probably be obtained by readers familiar with mathematical logic, recursion theory and complexity. We recommend the following textbooks as basic references for these areas:

- Mendelson: *Introduction to Mathematical Logic*, fourth edition, Chapman & Hall, London, 1997.

[1] This part is quite incomplete due to the "work-in-progress" character of the notes.

- Jones: *Computability and Complexity From a Programming Perspective,* MIT Press, 1997.

The notes have been used for a one-semester graduate/Ph.D. course at the Department of Computer Science at the University of Copenhagen (DIKU). Roughly one chapter was presented at each lecture, sometimes leaving material out.

The notes are still in progress and should not be conceived as having been proof read carefully to the last detail. Nevertheless, we are grateful to the students attending the course for pointing out numerous typos, for spotting actual mistakes, and for suggesting improvements to the exposition.

This joint work was made possible thanks to the visiting position funded by the University of Copenhagen, and held by the second author at DIKU in the winter and summer semesters of the academic year 1997-8.

M.H.B.S. & P.U., May 1998

Contents

CHAPTER 1

Type-free λ-calculus

The λ-*calculus* is a collection of formal theories of interest in, e.g., computer science and logic. The λ-calculus and the related systems of *combinatory logic* were originally proposed as a foundation of mathematics around 1930 by Church and Curry, but the proposed systems were subsequently shown to be inconsistent by Church's students Kleene and Rosser in 1935.

However, a certain subsystem consisting of the λ-*terms* equipped with so-called β-*reduction* turned out to be useful for formalizing the intuitive notion of effective computability and led to *Church's thesis* stating that λ-*definability* is an appropriate formalization of the intuitive notion of effective computability. The study of this subsystem—which was proved to be consistent by Church and Rosser in 1936—was a main inspiration for the development of *recursion theory.*

With the invention of physical computers came also programming languages, and λ-calculus has proved to be a useful tool in the design, implementation, and theory of programming languages. For instance, λ-calculus may be considered an idealized sublanguage of some programming languages like *LISP.* Also, λ-calculus is useful for expressing semantics of programming languages as done in *denotational semantics.* According to Hindley and Seldin [55, p.43], "λ-calculus and combinatory logic are regarded as 'test-beds' in the study of higher-order programming languages: techniques are tried out on these two simple languages, developed, and then applied to other more 'practical' languages."

The λ-calculus is sometimes called *type-free* or *untyped* to distinguish it from variants in which *types* play a role; these variants will be introduced in the next chapter.

1.1. λ-terms

The objects of study in λ-calculus are λ-*terms.* In order to introduce these, it is convenient to introduce the notion of a *pre-term.*

1.1.1. Definition. Let

$$V = \{v_0, v_1, \dots\}$$

denote an infinite alphabet. The set Λ^- of *pre-terms* is the set of strings defined by the grammar:

$$\Lambda^- ::= V \mid (\Lambda^- \ \Lambda^-) \mid (\lambda V \ \Lambda^-)$$

1.1.2. Example. The following are pre-terms.

(i) $((v_0 \ v_1) \ v_2) \in \Lambda^-$;

(ii) $(\lambda v_0 \ (v_0 \ v_1)) \in \Lambda^-$;

(iii) $((\lambda v_0 \ v_0) \ v_1) \in \Lambda^-$;

(iv) $((\lambda v_0 \ (v_0 \ v_0)) \ (\lambda v_1 \ (v_1 \ v_1))) \in \Lambda^-$.

1.1.3. Notation. We use uppercase letters, e.g., K, L, M, N, P, Q, R with or without subscripts to denote arbitrary elements of Λ^- and lowercase letters, e.g., x, y, z with or without subscripts to denote arbitrary elements of V.

1.1.4. Terminology.

(i) A pre-term of form x (i.e., an element of V) is called a *variable*;

(ii) A pre-term of form $(\lambda x \ M)$ is called an *abstraction* (over x);

(iii) A pre-term of form $(M \ N)$ is called an *application* (of M to N).

The heavy use of parentheses is rather cumbersome. We therefore introduce the following, standard conventions for omitting parentheses without introducing ambiguity. We shall make use of these conventions under a no-compulsion/no-prohibition agreement—see Remark 1.1.10.

1.1.5. Notation. We use the shorthands

(i) $(K \ L \ M)$ for $((K \ L) \ M)$;

(ii) $(\lambda x \ \lambda y \ M)$ for $(\lambda x \ (\lambda y \ M))$;

(iii) $(\lambda x \ M \ N)$ for $(\lambda x \ (M \ N))$;

(iv) $(M \ \lambda x \ N)$ for $(M \ (\lambda x \ N))$.

We also omit outermost parentheses.

1.1.6. Remark. The two first shorthands concern nested applications and abstractions, respectively. The two next ones concern applications nested inside abstractions and vice versa, respectively.

To remember the shorthands, think of application as associating to the left, and think of abstractions as extending as far to the right as possible.

When abstracting over a number of variables, each variable must be accompanied by an abstraction. It is therefore convenient to introduce the following shorthand.

1.1.7. NOTATION. We write $\lambda x_1 \ldots x_n.M$ for $\lambda x_1\ \ldots \lambda x_n\ M$. As a special case, we write $\lambda x.M$ for $\lambda x\, M$.

1.1.8. REMARK. Whereas abstractions are written with a λ, there is no corresponding symbol for applications; these are written simply by juxtaposition. Hence, there is no corresponding shorthand for applications.

1.1.9. EXAMPLE. The pre-terms in Example 1.1.2 can be written as follows, respectively:

(i) $v_0\ v_1\ v_2$;

(ii) $\lambda v_0.v_0\ v_1$;

(iii) $(\lambda v_0.v_0)\ v_1$;

(iv) $(\lambda v_0.v_0\ v_0)\ \lambda v_1.v_1\ v_1$.

1.1.10. REMARK. The conventions mentioned above are used in the remainder of these notes. However, we refrain from using them—wholly or partly—when we find this more convenient. For instance, we might prefer to write $(\lambda v_0.v_0\ v_0)\ (\lambda v_1.v_1\ v_1)$ for the last term in the above example.

1.1.11. DEFINITION. For $M \in \Lambda^-$ define the set $\mathrm{FV}(M) \subseteq V$ of *free variables* of M as follows.

$$\begin{array}{lcl} \mathrm{FV}(x) & = & \{x\}; \\ \mathrm{FV}(\lambda x.P) & = & \mathrm{FV}(P)\backslash\{x\}; \\ \mathrm{FV}(P\ Q) & = & \mathrm{FV}(P) \cup \mathrm{FV}(Q). \end{array}$$

If $\mathrm{FV}(M) = \{\}$ then M is called *closed.*

1.1.12. EXAMPLE. Let x, y, z denote distinct variables. Then

(i) $\mathrm{FV}(x\ y\ z) = \{x, y, z\}$;

(ii) $\mathrm{FV}(\lambda x.x\ y) = \{y\}$;

(iii) $\mathrm{FV}((\lambda x.x\ x)\ \lambda y.y\ y) = \{\}$.

1.1.13. DEFINITION. For $M, N \in \Lambda^-$ and $x \in V$, the *substitution of N for x in M,* written $M[x := N] \in \Lambda^-$, is defined as follows, where $x \neq y$:

$$\begin{array}{lll} x[x := N] & = N; & \\ y[x := N] & = y; & \\ (P\ Q)[x := N] & = P[x := N]\ Q[x := N]; & \\ (\lambda x.P)[x := N] & = \lambda x.P; & \\ (\lambda y.P)[x := N] & = \lambda y.P[x := N], & \text{if } y \notin \mathrm{FV}(N) \text{ or } x \notin \mathrm{FV}(P); \\ (\lambda y.P)[x := N] & = \lambda z.P[y := z][x := N], & \text{if } y \in \mathrm{FV}(N) \text{ and } x \in \mathrm{FV}(P). \end{array}$$

where z is chosen as the $v_i \in V$ with minimal i such that $v_i \notin \mathrm{FV}(P) \cup \mathrm{FV}(N)$ in the last clause.

1.1.14. Example. If x, y, z are distinct variables, then for a certain variable u:

$$((\lambda x.x\ yz)\ (\lambda y.x\ y\ z)\ (\lambda z.x\ y\ z))[x := y] = (\lambda x.x\ yz)\ (\lambda u.y\ u\ z)\ (\lambda z.y\ y\ z)$$

1.1.15. Definition. Let *α-equivalence,* written $=_\alpha$, be the smallest relation on Λ^-, such that

$$\begin{array}{ll} P =_\alpha P & \text{for all } P; \\ \lambda x.P =_\alpha \lambda y.P[x := y] & \text{if } y \notin \mathrm{FV}(P), \end{array}$$

and closed under the rules:

$$\begin{array}{lcll} P =_\alpha P' & \Rightarrow & \forall x \in V: & \lambda x.P =_\alpha \lambda x.P'; \\ P =_\alpha P' & \Rightarrow & \forall Z \in \Lambda^-: & P\ Z =_\alpha P'\ Z; \\ P =_\alpha P' & \Rightarrow & \forall Z \in \Lambda^-: & Z\ P =_\alpha Z\ P'; \\ P =_\alpha P' & \Rightarrow & & P' =_\alpha P; \\ P =_\alpha P'\ \&\ P' =_\alpha P'' & \Rightarrow & & P =_\alpha P''. \end{array}$$

1.1.16. Example. Let x, y, z denote different variables. Then

(i) $\lambda x.x =_\alpha \lambda y.y$;

(ii) $\lambda x.x\ z =_\alpha \lambda y.y\ z$;

(iii) $\lambda x.\lambda y.x\ y =_\alpha \lambda y.\lambda x.y\ x$;

(iv) $\lambda x.x\ y \neq_\alpha \lambda x.x\ z$.

1.1.17. Definition. Define for any $M \in \Lambda^-$, the *equivalence class* $[M]_\alpha$ by:

$$[M]_\alpha = \{N \in \Lambda^- \mid M =_\alpha N\}$$

Then define the set Λ of *λ-terms* by:

$$\Lambda \ = \ \Lambda^-/=_\alpha \ = \ \{[M]_\alpha \mid M \in \Lambda^-\}$$

1.1.18. Warning. The notion of a pre-term and the associated explicit distinction between pre-terms and λ-terms introduced above are not standard in the literature. Rather, it is customary to call our pre-terms λ-terms, and then informally remark that α-equivalent λ-terms are "identified."

In the remainder of these notes we shall be almost exclusively concerned with λ-terms, not pre-terms. Therefore, it is convenient to introduce the following.

1.1.19. NOTATION. We write M instead of $[M]_\alpha$ in the remainder. This leads to ambiguity: is M a pre-term or a λ-term? In the remainder of these notes, M should always be construed as $[M]_\alpha \in \Lambda$, *except when explicitly stated otherwise.*

We end this section with two definitions introducing the notions of free variables and substitution on λ-terms (recall that, so far, these notions have been introduced only for pre-terms). These two definitions provide the first example of how to rigorously understand definitions involving λ-terms.

1.1.20. DEFINITION. For $M \in \Lambda$ define the set $\mathrm{FV}(M) \subseteq V$ of *free variables* of M as follows.

$$\begin{array}{lcl} \mathrm{FV}(x) & = & \{x\}; \\ \mathrm{FV}(\lambda x.P) & = & \mathrm{FV}(P)\backslash\{x\}; \\ \mathrm{FV}(P\ Q) & = & \mathrm{FV}(P) \cup \mathrm{FV}(Q). \end{array}$$

If $\mathrm{FV}(M) = \{\}$ then M is called *closed.*

1.1.21. REMARK. According to Notation 1.1.19, what we really mean by this is that we define FV as the map from Λ to subsets of V satisfying the rules:

$$\begin{array}{lcl} \mathrm{FV}([x]_\alpha) & = & \{x\}; \\ \mathrm{FV}([\lambda x.P]_\alpha) & = & \mathrm{FV}([P]_\alpha)\backslash\{x\}; \\ \mathrm{FV}([P\ Q]_\alpha) & = & \mathrm{FV}([P]_\alpha) \cup \mathrm{FV}([Q]_\alpha). \end{array}$$

Strictly speaking we then have to demonstrate there there is at most one such function (uniqueness) and that there is at least one such function (existence).

Uniqueness can be established by showing for any two functions FV_1 and FV_2 satisfying the above equations, and any λ-term, that the results of FV_1 and FV_2 on the λ-term are the same. The proof proceeds by induction on the number of symbols in any member of the equivalence class.

To demonstrate existence, consider the map that, given an equivalence class, picks a member, and takes the free variables of that. Since any choice of member yields the same set of variables, this latter map is well-defined, and can easily be seen to satisfy the above rules.

In the rest of these notes such considerations will be left implicit.

1.1.22. DEFINITION. For $M, N \in \Lambda$ and $x \in V$, the *substitution of N for x in M,* written $M\{x := N\}$, is defined as follows:

$$\begin{array}{lcll} x[x := N] & = & N; & \\ y[x := N] & = & y, & \text{if } x \neq y; \\ (P\ Q)[x := N] & = & P[x := N]\ Q[x := N]; & \\ (\lambda y.P)[x := N] & = & \lambda y.P[x := N], & \text{if } x \neq y, \text{ where } y \notin \mathrm{FV}(N). \end{array}$$

1.1.23. EXAMPLE.

(i) $(\lambda x.x\ y)[x := \lambda z.z] = \lambda x.x\ y$;

(ii) $(\lambda x.x\ y)[y := \lambda z.z] = \lambda x.x\ \lambda z.z$.

1.2. Reduction

Next we introduce reduction on λ-terms.

1.2.1. Definition. Let $\to_\beta$ be the smallest relation on Λ such that

$$(\lambda x.P)\ Q \to_\beta P[x := Q],$$

and closed under the rules:

$$\begin{array}{lll} P \to_\beta P' & \Rightarrow & \forall x \in V: \ \lambda x.P \to_\beta \lambda x.P' \\ P \to_\beta P' & \Rightarrow & \forall Z \in \Lambda: \ P\ Z \to_\beta P'\ Z \\ P \to_\beta P' & \Rightarrow & \forall Z \in \Lambda: \ Z\ P \to_\beta Z\ P' \end{array}$$

A term of form $(\lambda x.P)\ Q$ is called a *β-redex,* and $P[x := Q]$ is called its *β-contractum.* A term M is a *β-normal form* if there is no term N with $M \to_\beta N$.

There are other notions of reduction than β-reduction, but these will not be considered in the present chapter. Therefore, we sometimes omit "β-" from the notions *β-redex, β-reduction,* etc.

1.2.2. Definition.

(i) The relation $\twoheadrightarrow_\beta$ (*multi-step β-reduction*) is the transitive-reflexive closure of $\to_\beta$; that is, $\twoheadrightarrow_\beta$ is the smallest relation closed under the rules:

$$\begin{array}{lll} P \to_\beta P' & \Rightarrow & P \twoheadrightarrow_\beta P'; \\ P \twoheadrightarrow_\beta P' \ \&\ P' \twoheadrightarrow_\beta P'' & \Rightarrow & P \twoheadrightarrow_\beta P''; \\ P \twoheadrightarrow_\beta P. & & \end{array}$$

(ii) The relation $=_\beta$ (*β-equality)* is the transitive-reflexive-symmetric closure of $\to_\beta$; that is, $=_\beta$ is the smallest relation closed under the rules:

$$\begin{array}{lll} P \to_\beta P' & \Rightarrow & P =_\beta P'; \\ P =_\beta P' \ \&\ P' =_\beta P'' & \Rightarrow & P =_\beta P''; \\ P =_\beta P; & & \\ P =_\beta P' & \Rightarrow & P' =_\beta P. \end{array}$$

1.2.3. Warning. In these notes, the symbol = without any qualification is used to express the fact that two objects, e.g., pre-terms or λ-terms are identical. This symbol is very often used in the literature for β-equality.

1.2.4. Example.

(i) $(\lambda x.x\ x)\ \lambda z.z \to_\beta (x\ x)[x := \lambda z.z] = (\lambda z.z)\ \lambda y.y$;

(ii) $(\lambda z.z)\ \lambda y.y \to_\beta z[z := \lambda y.y] = \lambda y.y$;

(iii) $(\lambda x.x\ x)\ \lambda z.z \twoheadrightarrow_\beta \lambda y.y$;

(iv) $(\lambda x.x)\ y\ z =_\beta y\ ((\lambda x.x)\ z)$.

1.3. Informal interpretation

Informally, λ-terms express functions and applications of functions in a pure form. For instance, the λ-term

$$\mathbf{I} = \lambda x.x$$

intuitively denotes the function that maps any argument to itself, i.e., the identity function. This is similar to the notation $n \mapsto n$ employed in mathematics. However, $\lambda x.x$ is a *string* over an alphabet with symbols λ, x, etc. (or rather an equivalence class of such objects), whereas $n \mapsto n$ is a *function,* i.e., a certain set of pairs. The difference is the same as that between a *program* written in some language and the mathematical function it computes, e.g., addition.

As in the notation $n \mapsto n$, the name of the abstracted variable x in $\lambda x.x$ is not significant, and this is why we identify $\lambda x.x$ with, e.g., $\lambda y.y$.

Another λ-term is

$$\mathbf{K}^* = \lambda y.\lambda x.x$$

which, intuitively, denotes the function that maps any argument to a *function,* namely the one that maps any argument to itself, i.e., the identity function. This is similar to programming languages where a function may return a function as a result. A related λ-term is

$$\mathbf{K} = \lambda y.\lambda x.y$$

which, intuitively, denotes the function that maps any argument to the function that, for any argument, returns the former argument.

Since λ-terms intuitively denote functions, there is a way to invoke one λ-term on another; this is expressed by application. Thus, the λ-term

$$\mathbf{I}\,\mathbf{K}$$

expresses application of $\mathbf{I}$ to $\mathbf{K}$. Since $\mathbf{K}$ intuitively denotes a function too, $\mathbf{I}$ denotes a function which may have another function as argument. This is similar to programming languages where a procedure may receive another procedure as argument.

In mathematics we usually write application of a function, say $f(n) = n^2$, to an argument, say 4, with the argument in parentheses: $f(4)$. In the λ-calculus we would rather write this as $(f\ 4)$, or just $f\,4$, keeping Notation 1.1.5 in mind. Not all parentheses can be omitted, though; for instance,

$$(\lambda x.x)\,\mathbf{I} \qquad \lambda x.x\,\mathbf{I}$$

are not the same λ-term; the first is $\mathbf{I}$ applied to $\mathbf{I}$, whereas the second expects an argument x which is applied to $\mathbf{I}$.

Intuitively, if $\lambda x.M$ denotes a function, and N denotes an argument, then the the value of the function on the argument is denoted by the λ-term that arises by substituting N for x in M. This latter λ-term is exactly the term

$$M[x := N]$$

This is similar to common practice in mathematics; if f is as above, then $f(4) = 4^2$, and we get from the application $f(4)$ to the value 4^2 by substituting 4 for n in the body of the definition of f.

The process of calculating values is formalized by β-reduction. Indeed, $M \to_\beta N$ if N arises from M by replacing a β-redex, i.e., a part of form

$$(\lambda x.P)\ Q$$

by its β-contractum.

$$P[x := Q]$$

For instance,

$$\mathbf{I}\,\mathbf{K} = (\lambda x.x)\ \mathbf{K} \to_\beta x[x := \mathbf{K}] = \mathbf{K}$$

Then the relation $\twoheadrightarrow_\beta$ formalizes the process of computing the overall result. Also, $=_\beta$ identifies λ-terms that, intuitively, denote the same function.

Note that λ-calculus is a *type-free* formalism. Unlike common mathematical practice, we do not insist that λ-terms denote functions from certain domains, e.g., the natural numbers, and that arguments be drawn from these domains. In particular, we may have self-application as in the λ-term

$$\omega = \lambda x.x\ x$$

and we may apply *this* λ-term to *itself* as in the λ-term

$$\Omega = \omega\ \omega$$

The type-free nature of λ-calculus leads to some interesting phenomena; for instance, a λ-term may reduce to itself as in

$$\Omega = (\lambda x.x\ x)\ \omega \to_\beta \omega\ \omega = \Omega$$

Therefore, there are also λ-terms with infinite reduction sequences, like

$$\Omega \to_\beta \Omega \to_\beta \ldots$$

1.4. The Church-Rosser Theorem

Since a λ-term M may contain several β-redexes, i.e., several parts of form $(\lambda x.P)\ Q$, there may be several N such that $M \to_\beta N$. For instance,

$$\mathbf{K}\ (\mathbf{I}\,\mathbf{I}) \to_\beta \lambda x.(\mathbf{I}\,\mathbf{I})$$

and also

$$\mathbf{K}\ (\mathbf{I}\ \mathbf{I}) \rightarrow_\beta \mathbf{K}\ \mathbf{I}$$

However, the *Church-Rosser theorem,* proved below, states that if

$$M \twoheadrightarrow_\beta M_1$$

and

$$M \twoheadrightarrow_\beta M_2$$

then a single λ-term M_3 can be found with

$$M_1 \twoheadrightarrow_\beta M_3$$

and

$$M_2 \twoheadrightarrow_\beta M_3$$

In particular, if M_1 and M_2 are β-normal forms, i.e., λ-terms that admit no further β-reductions, then they must be the same λ-term, since the β-reductions from M_1 and M_2 to M_3 must be in zero steps. This is similar to the fact that when we calculate the value of an arithmetical expression, e.g.,

$$(4+2)\cdot(3+7)\cdot 11$$

the end result is independent of the order in which we do the calculations.

1.4.1. Definition. A relation $>$ on Λ satisfies the *diamond property* if, for all $M_1, M_2, M_3 \in \Lambda$, if $M_1 > M_2$ and $M_1 > M_3$, then there exists an $M_4 \in \Lambda$ such that $M_2 > M_4$ and $M_3 > M_4$.

1.4.2. Lemma. *Let $>$ be a relation on Λ and suppose that its transitive closure*[1] *is $\twoheadrightarrow_\beta$. If $>$ satisfies the diamond property, then so does $\twoheadrightarrow_\beta$.*

Proof. First show by induction on n that $M_1 > N_1$ and $M_1 > \ldots > M_n$ implies that there are $N_2, \ldots, N_n$ such that $N_1 > N_2 > \ldots > N_n$ and $M_n > N_n$.

Using this property, show by induction on m that if $N_1 > \ldots > N_m$ and $N_1 >^* M_1$ then there are $M_2, \ldots, M_m$ such that $M_1 > M_2 > \ldots > M_m$ and $N_m >^* M_m$.

[1] Let R be a relation on Λ. The *transitive closure* of R is the least relation R^* satisfying:

$$\begin{array}{lcl} PRP' & \Rightarrow & PR^*P' \\ PR^*P' \ \&\ P'R^*P'' & \Rightarrow & PR^*P'' \end{array}$$

The *reflexive closure* of R is the least relation $R^=$ satisfying:

$$\begin{array}{lcl} PRP' & \Rightarrow & PR^=P' \\ PR^=P & & \end{array}$$

Now assume $M_1 \twoheadrightarrow_\beta M_2$ and $M_1 \twoheadrightarrow_\beta M_3$. Since $\twoheadrightarrow_\beta$ is the transitive closure of $>$ we have $M_1 > \ldots > M_2$ and $M_1 > \ldots > M_3$. By what was shown above, we can find M_4 such that $M_2 > \ldots > M_4$ and $M_3 > \ldots > M_4$. Since $\twoheadrightarrow_\beta$ is the transitive closure of $>$, also $M_2 \twoheadrightarrow_\beta M_4$ and $M_3 \twoheadrightarrow_\beta M_4$. □

1.4.3. Definition. Let $\twoheadrightarrow_l$ be the relation on Λ defined by:

$$\begin{array}{lcl} P \twoheadrightarrow_l P & & \\ P \twoheadrightarrow_l P' & \Rightarrow & \lambda x.P \twoheadrightarrow_l \lambda x.P' \\ P \twoheadrightarrow_l P' \ \& \ Q \twoheadrightarrow_l Q' & \Rightarrow & P\,Q \twoheadrightarrow_l P'\,Q' \\ P \twoheadrightarrow_l P' \ \& \ Q \twoheadrightarrow_l Q' & \Rightarrow & (\lambda x.P)\,Q \twoheadrightarrow_l P'[x := Q'] \end{array}$$

1.4.4. Lemma. $M \twoheadrightarrow_l M' \ \& \ N \twoheadrightarrow_l N' \Rightarrow M[x := N] \twoheadrightarrow_l M'[x := N']$.

Proof. By induction on the definition of $M \twoheadrightarrow_l M'$. In case M' is M, proceed by induction on M. □

1.4.5. Lemma. *$\twoheadrightarrow_l$ satisfies the diamond property, i.e., for all $M_1, M_2, M_3 \in \Lambda$, if $M_1 \twoheadrightarrow_l M_2$ and $M_1 \twoheadrightarrow_l M_3$, then there exists an $M_4 \in \Lambda$ such that $M_2 \twoheadrightarrow_l M_4$ and $M_3 \twoheadrightarrow_l M_4$.*

Proof. By induction on the definition of $M_1 \twoheadrightarrow_l M_2$, using the above lemma. □

1.4.6. Lemma. *$\twoheadrightarrow_\beta$ is the transitive closure of $\twoheadrightarrow_l$.*

Proof. Clearly[2]

$$(\rightarrow_\beta)^= \ \subseteq \twoheadrightarrow_l \subseteq \twoheadrightarrow_\beta$$

Then

$$\twoheadrightarrow_\beta = \ ((\rightarrow_\beta)^=)^* \ \subseteq \twoheadrightarrow_l^* \subseteq \ (\twoheadrightarrow_\beta)^* \ = \twoheadrightarrow_\beta$$

In particular, $\twoheadrightarrow_l^* = \twoheadrightarrow_\beta$. □

1.4.7. Theorem (Church and Rosser, 1936). *For every $M_1, M_2, M_3 \in \Lambda$, if $M_1 \twoheadrightarrow_\beta M_2$ and $M_1 \twoheadrightarrow_\beta M_3$, then there exists an $M_4 \in \Lambda$ such that $M_2 \twoheadrightarrow_\beta M_4$ and $M_3 \twoheadrightarrow_\beta M_4$.*

Proof (Tait & Martin-Löf). By the above three lemmas. □

1.4.8. Corollary. *For all $M, N \in \Lambda$, if $M =_\beta N$, then there exists an $L \in \Lambda$ such that $M \twoheadrightarrow_\beta L$ and $N \twoheadrightarrow_\beta L$.*

1.4.9. Corollary. *For all $M, N_1, N_2 \in \Lambda$, if $M \twoheadrightarrow_\beta N_1$ and $M \twoheadrightarrow_\beta N_2$ and both N_1 and N_2 are in β-normal form, then $N_1 = N_2$.*

[2] Recall the relations R^* and $R^=$ defined earlier.

1.4.10. Corollary. *For all $M, N \in \Lambda$, if there are β-normal forms L_1 and L_2 such that $M \twoheadrightarrow_\beta L_1$, $N \twoheadrightarrow_\beta L_2$, and $L_1 \neq L_2$, then $M \neq_\beta N$.*

1.4.11. Example. $\lambda x.x \neq_\beta \lambda x.\lambda y.x$.

1.4.12. Remark. One can consider the lambda calculus as an equational theory, i.e., a formal theory with formulas of the form $M =_\beta N$. The preceding example establishes *consistency* of this theory, in the following sense: there exists a formula P which cannot be proved.

This may seem to be a very weak property, compared to "one cannot prove a contradiction" (where a suitable notion of "contradiction" in ordinary logic is e.g., $P \wedge \neg P$). But note that in most formal theories, where a notion of contradiction can be expressed, its provability implies provability of all formulas. Thus, consistency can be equally well defined as "one cannot prove everything".

1.5. Expressibility and undecidability

Although we have given an informal explanation of the meaning of λ-terms it remains to explain in what sense β-reduction more precisely can express computation. In this section we show that λ-calculus can be seen as an alternative formulation of recursion theory.

The following gives a way of representing numbers as λ-terms.

1.5.1. Definition.

(i) For any $n \in \mathbb{N}$ and $F, A \in \Lambda$ define $F^n(A)$ (*n-times iterated application of F to A*) by:

$$\begin{array}{lcl} F^0(A) & = & A \\ F^{n+1}(A) & = & F(F^n(A)) \end{array}$$

(ii) For any $n \in \mathbb{N}$, the *Church numeral* c_n is the λ-term

$$c_n = \lambda s.\lambda z.s^n(z)$$

1.5.2. Example.

(i) $c_0 = \lambda s.\lambda z.z$;

(ii) $c_1 = \lambda s.\lambda z.s\ z$;

(iii) $c_2 = \lambda s.\lambda z.s\ (s\ z)$;

(iv) $c_3 = \lambda s.\lambda z.s\ (s\ (s\ z))$.

1.5.3. Remark. c_n is the number n represented inside the λ-calculus.

The following shows how to do arithmetic on Church numerals.

1.5.4. Proposition (Rosser). *Let*

$$\begin{aligned} A_+ &= \lambda x.\lambda y.\lambda s.\lambda z.x\ s\ (y\ s\ z);\\ A_* &= \lambda x.\lambda y.\lambda s.x\ (y\ s);\\ A_e &= \lambda x.\lambda y.y\ x. \end{aligned}$$

Then

$$\begin{aligned} A_+\ c_n\ c_m &= c_{n+m};\\ A_*\ c_n\ c_m &= c_{n\cdot m};\\ A_e\ c_n\ c_m &= c_{n^m}\ \textit{if } m>0. \end{aligned}$$

Proof. For any $n \in \mathbb{N}$,

$$\begin{aligned} c_n\ s\ z &= (\lambda f.\lambda x.f^n(x))\ s\ z\\ &=_\beta (\lambda x.s^n(x))\ z\\ &=_\beta s^n(z) \end{aligned}$$

Thus

$$\begin{aligned} A_+\ c_n\ c_m &= (\lambda x.\lambda y.\lambda s.\lambda z.x\ s\ (y\ s\ z))\ c_n\ c_m\\ &=_\beta \lambda s.\lambda z.c_n\ s\ (c_m\ s\ z)\\ &=_\beta \lambda s.\lambda z.c_n\ s\ (s^m(z))\\ &=_\beta \lambda s.\lambda z.s^n(s^m(z))\\ &= \lambda s.\lambda z.s^{n+m}(z)\\ &= c_{n+m} \end{aligned}$$

The similar properties for multiplication and exponentiation are left as exercises. □

1.5.5. Remark. Recall that $M =_\beta N$ when, intuitively, M and N denote the same object. For instance $\mathbf{I}\,\mathbf{I} =_\beta \mathbf{I}$ since both terms, intuitively, denote the identity function.

Now consider the two terms

$$\begin{aligned} A_s &= \lambda x.\lambda s.\lambda z.s\ (x\ s\ z)\\ A'_s &= \lambda x.\lambda s.\lambda z.x\ s\ (s\ z) \end{aligned}$$

It is easy to calculate that

$$\begin{aligned} A_s\ c_n &=_\beta c_{n+1}\\ A'_s\ c_n &=_\beta c_{n+1} \end{aligned}$$

So both terms denote, informally, the successor function on Church numerals, but the two terms are not β-equal (why not?)

The following shows how to define booleans and conditionals inside λ-calculus.

1.5.6. Proposition. *Define*

$$\begin{array}{lcl} \mathbf{true} & = & \lambda x.\lambda y.x; \\ \mathbf{false} & = & \lambda x.\lambda y.y; \\ \mathbf{if}\ B\ \mathbf{then}\ P\ \mathbf{else}\ Q & = & B\,P\,Q. \end{array}$$

Then

$$\begin{array}{lcl} \mathbf{if\ true\ then}\ P\ \mathbf{else}\ Q & =_\beta & P; \\ \mathbf{if\ false\ then}\ P\ \mathbf{else}\ Q & =_\beta & Q. \end{array}$$

Proof. We have:

$$\begin{array}{rcl} \mathbf{if\ true\ then}\ P\ \mathbf{else}\ Q & = & (\lambda x.\lambda y.x)\,P\,Q \\ & =_\beta & (\lambda y.P)\,Q \\ & =_\beta & P. \end{array}$$

The proof that $\mathbf{if\ false\ then}\ P\ \mathbf{else}\ Q =_\beta Q$ is similar. □

We can also define pairs in λ-calculus.

1.5.7. Proposition. *Define*

$$\begin{array}{lcl} [P,Q] & = & \lambda x.x\,P\,Q; \\ \pi_1 & = & \lambda x.\lambda y.x; \\ \pi_2 & = & \lambda x.\lambda y.y. \end{array}$$

Then

$$\begin{array}{lcl} [P,Q]\,\pi_1 & =_\beta & P; \\ [P,Q]\,\pi_2 & =_\beta & Q. \end{array}$$

Proof. We have:

$$\begin{array}{rcl} [P,Q]\,\pi_1 & = & (\lambda x.x\,P\,Q)\,\lambda x.\lambda y.x \\ & =_\beta & (\lambda x.\lambda y.x)\,P\,Q \\ & =_\beta & (\lambda y.P)\,Q \\ & =_\beta & P. \end{array}$$

The proof that $[P,Q]\,\pi_2 =_\beta Q$ is similar. □

1.5.8. Remark. Note that we do not have $[M\ \pi_1, M\ \pi_2] =_\beta M$ for all $M \in \Lambda$; that is, our pairing operator is not *surjective.*

1.5.9. Remark. The construction is easily generalized to tuples $[M_1, \ldots, M_n]$ with projections π_i where $i \in \{1, \ldots, n\}$.

The following gives one way of expressing recursion in λ-calculus.

1.5.10. Theorem (Fixed point theorem). *For all F there is an X such that*

$$F\ X =_\beta X$$

In fact, there is a λ-term $\mathbf{Y}$ such that, for all F:

$$F\ (\mathbf{Y}\ F) =_\beta \mathbf{Y}\ F$$

Proof. Put

$$\mathbf{Y} = \lambda f.(\lambda x.f\ (x\ x))\ \lambda x.f\ (x\ x)$$

Then

$$\begin{array}{rcl} \mathbf{Y}\ F & = & (\lambda f.(\lambda x.f\ (x\ x))\ \lambda x.f\ (x\ x))\ F \\ & =_\beta & (\lambda x.F\ (x\ x))\ \lambda x.F\ (x\ x) \\ & =_\beta & F\ ((\lambda x.F\ (x\ x))\ \lambda x.F\ (x\ x)) \\ & =_\beta & F\ ((\lambda f.(\lambda x.f\ (x\ x))\ \lambda x.f\ (x\ x))\ F) \\ & = & F\ (\mathbf{Y}\ F) \end{array}$$

as required. □

1.5.11. Corollary. *Given $M \in \Lambda$ there is $F \in \Lambda$ such that:*

$$F =_\beta M[f := F]$$

Proof. Put

$$F = \mathbf{Y}\ \lambda f.M$$

Then

$$\begin{array}{rcl} F & = & \mathbf{Y}\ \lambda f.M \\ & =_\beta & (\lambda f.M)\ (\mathbf{Y}\ \lambda f.M) \\ & = & (\lambda f.M)\ F \\ & =_\beta & M[f := F] \end{array}$$

as required. □

Corollary 1.5.11 allows us to write *recursive definitions* of λ-terms; that is, we may define F as a λ-term satisfying a *fixed point equation* $F =_\beta \lambda x.M$ where the term F occurs somewhere inside M. However, there may be several terms F satisfying this equation (will these be β-equal?).

1.5.12. Example. Let C be some λ-term which expresses a condition, i.e., let $C\ c_n =_\beta \mathbf{true}$ or $C\ c_n =_\beta \mathbf{false}$, for all $n \in \mathbb{N}$. Let S define the successor function (see Remark 1.5.5). Suppose we want to compute in λ-calculus, for any number, the smallest number greater than the given one that satisfies the condition. This is expressed by the λ-term F:

$$\begin{array}{rcl} H & = & \lambda f.\lambda x.\mathbf{if}\ (C\ x)\ \mathbf{then}\ x\ \mathbf{else}\ f\ (S\ x) \\ F & = & \mathbf{Y}\ H \end{array}$$

Indeed, for example

$$\begin{array}{rcl} F\ c_4 & = & (\mathbf{Y}\ H)\ c_4 \\ & =_\beta & H\ (\mathbf{Y}\ H)\ c_4 \\ & = & (\lambda f.\lambda x.\mathbf{if}\ (C\ x)\ \mathbf{then}\ x\ \mathbf{else}\ f\ (S\ x))\ (\mathbf{Y}\ H)\ c_4 \\ & =_\beta & \mathbf{if}\ (C\ c_4)\ \mathbf{then}\ c_4\ \mathbf{else}\ (\mathbf{Y}\ H)\ (S\ c_4) \\ & = & \mathbf{if}\ (C\ c_4)\ \mathbf{then}\ c_4\ \mathbf{else}\ F\ (S\ c_4) \end{array}$$

So far we have been informal as to how λ-terms "express" certain functions. This notion is made precise as follows.

1.5.13. Definition.

(i) A *numeric function* is a map

$$f : \mathbb{N}^m \to \mathbb{N}.$$

(ii) A numeric function $f : \mathbb{N}^m \to \mathbb{N}$ is λ-*definable* if there is an $F \in \Lambda$ such that

$$F\ c_{n_1}\ \ldots\ c_{n_m} =_\beta c_{f(n_1,\ldots,n_m)}$$

for all $n_1, \ldots, n_m \in \mathbb{N}$.

1.5.14. Remark. By the Church-Rosser property, (ii) implies that, in fact,

$$F\ c_{n_1}\ \ldots\ c_{n_m} \twoheadrightarrow_\beta c_{f(n_1,\ldots,n_m)}$$

There are similar notions for partial functions—see [7].
We shall show that all recursive functions are λ-definable.

1.5.15. Definition. The class of *recursive functions* is the smallest class of numeric functions containing the *initial functions*

(i) *projections:* $U_i^m(n_1, \ldots, n_m) = n_i$ for all $1 \leq i \leq m$;

(ii) *successor:* $S^+(n) = n + 1$;

(iii) *zero:* $Z(n) = 0$.

and closed under *composition, primitive recursion,* and *minimization*:

(i) *composition:* if $g : \mathbb{N}^k \to \mathbb{N}$ and $h_1, \ldots, h_k : \mathbb{N}^m \to \mathbb{N}$ are recursive, then so is $f : \mathbb{N}^m \to \mathbb{N}$ defined by

$$f(n_1, \ldots, n_m) = g(h_1(n_1, \ldots, n_m), \ldots, h_k(n_1, \ldots, n_m)).$$

(ii) *primitive recursion:* if $g : \mathbb{N}^m \to \mathbb{N}$ and $h : \mathbb{N}^{m+2} \to \mathbb{N}$ are recursive, then so is $f : \mathbb{N}^{m+1} \to \mathbb{N}$ defined by

$$\begin{array}{rcl} f(0, n_1, \ldots, n_m) & = & g(n_1, \ldots, n_m); \\ f(n+1, n_1, \ldots, n_m) & = & h(f(n, n_1, \ldots, n_m), n, n_1, \ldots, n_m). \end{array}$$

(iii) *minimization:* if $g : \mathbb{N}^{m+1} \to \mathbb{N}$ is recursive and for all $n_1, \ldots, n_m$ there is an n such that $g(n, n_1, \ldots, n_m) = 0$, then $f : \mathbb{N}^m \to \mathbb{N}$ defined as follows is also recursive[3]

$$f(n_1, \ldots, n_m) = \mu n.g(n, n_1, \ldots, n_m) = 0$$

1.5.16. Lemma. *The initial functions are λ-definable.*

Proof. With

$$\begin{array}{lcl} \mathbf{U}_i^m & = & \lambda x_1 \ldots \lambda x_m . x_i \\ \mathbf{S}^+ & = & \lambda x . \lambda s . \lambda z . s\ (x\ s\ z) \\ \mathbf{Z} & = & \lambda x . c_0 \end{array}$$

the necessary properties hold. □

1.5.17. Lemma. *The λ-definable functions are closed under composition.*

Proof. If $g : \mathbb{N}^k \to \mathbb{N}$ is λ-definable by $G \in \Lambda$ and $h_1, \ldots, h_k : \mathbb{N}^m \to \mathbb{N}$ are λ-definable by some $H_1, \ldots, H_k \in \Lambda$, then $f : \mathbb{N}^m \to \mathbb{N}$ defined by

$$f(n_1, \ldots, n_m) = g(h_1(n_1, \ldots, n_m), \ldots, h_k(n_1, \ldots, n_m))$$

is λ-definable by

$$F = \lambda x_1 \ldots \lambda x_m . G\ (H_1\ x_1 \ldots x_m) \ldots (H_k\ x_1 \ldots x_m),$$

as is easy to verify. □

1.5.18. Lemma. *The λ-definable functions are closed under primitive recursion.*

Proof. If $g : \mathbb{N}^m \to \mathbb{N}$ is λ-definable by some $G \in \Lambda$ and $h : \mathbb{N}^{m+2} \to \mathbb{N}$ is λ-definable by some $H \in \Lambda$, then $f : \mathbb{N}^{m+1} \to \mathbb{N}$ defined by

$$\begin{array}{lcl} f(0, n_1, \ldots, n_m) & = & g(n_1, \ldots, n_m); \\ f(n+1, n_1, \ldots, n_m) & = & h(f(n, n_1, \ldots, n_m), n, n_1, \ldots, n_m), \end{array}$$

is λ-definable by $F \in \Lambda$ where

$$\begin{array}{lcl} F & = & \lambda x . \lambda x_1 \ldots \lambda x_m . x\ T\ [c_0, G\ x_1 \ldots x_n]\ \pi_2; \\ T & = & \lambda p . [\mathbf{S}^+\ (p\ \pi_1), H\ (p\ \pi_2)\ (p\ \pi_1)\ x_1 \ldots x_m]. \end{array}$$

Indeed, we have

$$\begin{array}{lcl} F\ c_n\ c_{n_1} \ldots c_{n_m} & =_\beta & c_n\ T\ [c_0, G\ c_{n_1} \ldots c_{n_m}]\ \pi_2 \\ & =_\beta & T^n([c_0, G\ c_{n_1} \ldots c_{n_m}])\ \pi_2 \end{array}$$

[3] $\mu n.g(n, n_1, \ldots, n_m) = 0$ denotes the smallest number n satisfying the equation $g(n, n_1, \ldots, n_m) = 0$.

Also,

$$\begin{array}{rcl} T\,[c_n, c_{f(n,n_1,\ldots,n_m)}] & =_\beta & [\mathbf{S}^+(c_n), H\,c_{f(n,n_1,\ldots,n_m)}\,c_n\,c_{n_1}\ldots c_{n_m}] \\ & =_\beta & [c_{n+1}, c_{h(f(n,n_1,\ldots,n_m),n,n_1,\ldots,n_m)}] \\ & =_\beta & [c_{n+1}, c_{f(n+1,n_1,\ldots,n_m)}] \end{array}$$

So

$$T^n([c_0, G\,c_{n_1}\ldots c_{n_m}]) \quad =_\beta \quad [c_n, c_{f(n,n_1,\ldots,n_m)}]$$

From this the required property follows. □

1.5.19. Lemma. *The λ-definable functions are closed under minimization.*

Proof. If $g : \mathbb{N}^{m+1} \to \mathbb{N}$ is λ-definable by $G \in \Lambda$ and for all $n_1, \ldots, n_m$ there is an n, such that $g(n, n_1, \ldots, n_m) = 0$, then $f : \mathbb{N}^m \to \mathbb{N}$ defined by

$$f(n_1, \ldots, n_m) = \mu m.g(n, n_1, \ldots, n_m) = 0$$

is λ-definable by $F \in \Lambda$, where

$$F = \lambda x_1 \ldots . \lambda x_m . H\,c_0$$

and where $H \in \Lambda$ is such that

$$H =_\beta \lambda y.\mathbf{if}\ (\mathbf{zero?}\ (G\,x_1 \ldots x_m\,y))\ \mathbf{then}\ y\ \mathbf{else}\ H\ (\mathbf{S}^+\,y).$$

Here,

$$\mathbf{zero?} = \lambda x.x\ (\lambda y.\mathbf{false})\,\mathbf{true}$$

We leave it as an exercise to verify that the required properties hold. □

The following can be seen as a form of *completeness* of the λ-calculus.

1.5.20. Theorem (Kleene). *All recursive functions are λ-definable.*

Proof. By the above lemmas. □

The converse also holds, as one can show by a routine argument. Similar results hold for partial functions as well—see [7].

1.5.21. Definition. Let $\langle \bullet, \bullet \rangle : \mathbb{N}^2 \to \mathbb{N}$ be a bijective, recursive function. The map $\# : \Lambda^- \to \mathbb{N}$ is defined by:

$$\begin{array}{lcl} \#(v_i) & = & \langle 0, i \rangle \\ \#(\lambda x.M) & = & \langle 2, \langle \#(x), \#(M) \rangle \rangle \\ \#(M\ N) & = & \langle 3, \langle \#(M), \#(N) \rangle \rangle \end{array}$$

For $M \in \Lambda$, we take $\#(M)$ to be the least possible number $\#(M')$ where M' is an alpha-representative of M. Also, for $M \in \Lambda$, we define $\ulcorner M \urcorner = c_{\#(M)}$.

1.5.22. Definition. Let $A \subseteq \Lambda$.

(i) A is *closed under* $=_\beta$ if

$$M \in A \ \&\ M =_\beta N \ \Rightarrow\ N \in A$$

(ii) A is *non-trival* if

$$A \neq \emptyset \ \&\ A \neq \Lambda$$

(iii) A is *recursive* if

$$\#A = \{\#(M) \mid M \in A\}$$

is recursive.

1.5.23. Theorem (Curry, Scott). *Let A be non-trivial and closed under $=_\beta$. Then A is not recursive.*

Proof (J. Terlouw). Suppose A is recursive. Define

$$B = \{M \mid M\ \ulcorner M \urcorner \in A\}$$

There exists an $F \in \Lambda$ with

$$\begin{array}{lcl} M \in B & \Leftrightarrow & F\ulcorner M\urcorner =_\beta c_0; \\ M \notin B & \Leftrightarrow & F\ulcorner M\urcorner =_\beta c_1. \end{array}$$

Let $M_0 \in A$, $M_1 \in \Lambda\backslash A$, and let

$$G = \lambda x.\mathbf{if}\ (\mathbf{zero?}\ (F\ x))\ \mathbf{then}\ M_1\ \mathbf{else}\ M_0$$

Then

$$\begin{array}{lcl} M \in B & \Leftrightarrow & G\ulcorner M\urcorner =_\beta M_1 \\ M \notin B & \Leftrightarrow & G\ulcorner M\urcorner =_\beta M_0 \end{array}$$

so

$$\begin{array}{lcllcl} G \in B & \Leftrightarrow & G\ulcorner G\urcorner =_\beta M_1 & \Rightarrow & G\ulcorner G\urcorner \notin A & \Rightarrow & G \notin B \\ G \notin B & \Leftrightarrow & G\ulcorner G\urcorner =_\beta M_0 & \Rightarrow & G\ulcorner G\urcorner \in A & \Rightarrow & G \in B \end{array}$$

a contradiction. □

1.5.24. Remark. The above theorem is analogous to *Rice's theorem* known in recursion theory.

The following is a variant of the halting problem. Informally it states that the formal theory of β-equality mentioned in Remark 1.4.12 is undecidable.

1.5.25. Corollary (Church). $\{M \in \Lambda \mid M =_\beta \mathbf{true}\}$ *is not recursive.*

1.5.26. Corollary. *The following set is not recursive:*

$$\{M \in \Lambda \mid \exists N \in \Lambda : M \twoheadrightarrow_\beta N\ \&\ N \textit{ is a } \beta\textit{-normal form }\}.$$

One can also infer from these results the well-known theorem due to Church stating that first-order predicate calculus is undecidable.

1.6. Historical remarks

For more on the history of λ-calculus, see e.g., [55] or [7]. First hand information may be obtained from Rosser and Kleene's eye witness statements [94, 62], and from Curry and Feys' book [24] which contains a wealth of historical information. Curry and Church's original aims have recently become the subject of renewed attention—see, e.g., [9, 10] and [50].

1.7. Exercises

1.7.1. Exercise. Show, step by step, how application of the conventions in Notation 1.1.5 allows us to express the pre-terms in Example 1.1.2 as done in Example 1.1.9.

1.7.2. Exercise. Which of the following abbreviations are correct?

1. $\lambda x.x\ y = (\lambda x.x)\ y$;
2. $\lambda x.x\ y = \lambda x.(x\ y)$;
3. $\lambda x.\lambda y.\lambda z.x\ y\ z = (\lambda x.\lambda y.\lambda z.x)\ (y\ z)$;
4. $\lambda x.\lambda y.\lambda z.x\ y\ z = ((\lambda x.\lambda y.\lambda z.x)\ y)\ z$;
5. $\lambda x.\lambda y.\lambda z.x\ y\ z = \lambda x.\lambda y.\lambda z.((x\ y)\ z)$.

1.7.3. Exercise. Which of the following identifications are correct?

1. $\lambda x.\lambda y.x = \lambda y.\lambda x.y$;
2. $(\lambda x.x)\ z = (\lambda z.z)\ x$.

1.7.4. Exercise. Do the following terms have normal forms?

1. $\mathbf{I}$, where $\lambda x.x$;
2. Ω, i.e., $\omega\ \omega$, where $\omega = \lambda x.x\ x$;
3. $\mathbf{K}\,\mathbf{I}\,\Omega$ where $\mathbf{K} = \lambda x.\lambda y.x$;
4. $(\lambda x.\mathbf{K}\,\mathbf{I}\,(x\ x))\ \lambda y.\mathbf{K}\,\mathbf{I}\,(y\ y)$;
5. $(\lambda x.z\ (x\ x))\ \lambda y.z\ (y\ y)$.

1.7.5. Exercise. A *reduction path* from a λ-term M is a finite or infinite sequence

$$M \to_\beta M_1 \to_\beta M_2 \to_\beta \dots$$

A term that has a normal form is also called *weakly normalizing* (or just *normalizing*), since at least one of its reduction paths terminate in a normal form. A term is *strongly normalizing* if *all* its reduction paths eventually terminate in normal forms, i.e., if the term has no infinite reduction paths. Which of the five terms in the preceding exercise are weakly/strongly normalizing? In which cases do different reduction paths lead to different normal forms?

1.7.6. Exercise. Which of the following are true?

1. $(\lambda x.\lambda y.\lambda z.(x\ z)\ (y\ z))\ \lambda u.u =_\beta (\lambda v.v\ \lambda y.\lambda z.\lambda u.u)\ \lambda x.x$;
2. $(\lambda x.\lambda y.x\ \lambda z.z)\ \lambda a.a =_\beta (\lambda y.y)\ \lambda b.\lambda z.z$;
3. $\lambda x.\Omega =_\beta \Omega$.

1.7.7. Exercise. Prove (without using the Church-Rosser Theorem) that for all $M_1, M_2, M_3 \in \Lambda$, if $M_1 \to_\beta M_2$ and $M_1 \to_\beta M_3$, then there exists an $M_4 \in \Lambda$ such that $M_2 \twoheadrightarrow_\beta M_4$ and $M_3 \twoheadrightarrow_\beta M_4$.

Can you extend your proof technique to yield a proof of the Church-Rosser theorem?

1.7.8. Exercise. Fill in the details of the proof Lemma 1.4.4.

1.7.9. Exercise. Fill in the details of the proof Lemma 1.4.5.

1.7.10. Exercise. Which of the following are true?

1. $(\mathbf{I}\ \mathbf{I})\ (\mathbf{I}\ \mathbf{I}) \twoheadrightarrow_l \mathbf{I}\ \mathbf{I}$;
2. $(\mathbf{I}\ \mathbf{I})\ (\mathbf{I}\ \mathbf{I}) \twoheadrightarrow_l \mathbf{I}$;
3. $\mathbf{I}\ \mathbf{I}\ \mathbf{I}\ \mathbf{I} \twoheadrightarrow_l \mathbf{I}\ \mathbf{I}\ \mathbf{I}$;
4. $\mathbf{I}\ \mathbf{I}\ \mathbf{I}\ \mathbf{I} \twoheadrightarrow_l \mathbf{I}$;

1.7.11. Exercise. Show that the fourth clause in Definition 1.4.3 cannot be replaced by

$$(\lambda x.P)\ Q \twoheadrightarrow_l P[x := Q].$$

That is, show that, if this is done, then $\twoheadrightarrow_l$ does not satisfy the diamond property.

1.7.12. Exercise. Prove Corollary 1.4.8–1.4.10.

1.7.13. Exercise. Write λ-terms (without using the notation $s^n(z)$) whose β-normal forms are the Church numerals c_5 and c_{100}.

1.7.14. Exercise. Prove that A_* and A_e satisfy the equations stated in Proposition 1.5.4.

1.7.15. Exercise. For each $n \in \mathbb{N}$, write a λ-term B_n such that

$$B_n\, c_i\, Q_1 \dots Q_n =_\beta Q_i,$$

for all $Q_1, \dots Q_n \in \Lambda$.

1.7.16. Exercise. For each $n \in \mathbb{N}$, write λ-terms $P_n, \pi_1, \dots, \pi_n$, such that for all $Q_1, \dots, Q_n \in \Lambda$:

$$(P_n\, Q_1 \dots Q_n)\, \pi_i =_\beta Q_i.$$

1.7.17. Exercise (Klop, taken from [7]). Let $\lambda x_1 x_2 \dots x_n.M$ be an abbreviation for $\lambda x_1.\lambda x_2.\dots.\lambda x_n.M$. Let

$$\begin{array}{lcl} ? & = & \lambda abcdefghijklmnopqstuvwxyzr.r\ (thisisafixedpointcombinator); \\ \$ & = & ?????????????????????????. \end{array}$$

Show that \$ is a *fixed point combinator,* i.e., that $\$\ F =_\beta F\ (\$\ F)$, holds for all $F \in \Lambda$.

1.7.18. Exercise. Define a λ-term **neg** such that

$$\begin{array}{lcl} \textbf{neg true} & =_\beta & \textbf{false}; \\ \textbf{neg false} & =_\beta & \textbf{true}. \end{array}$$

1.7.19. Exercise. Define λ-terms O and E such that, for all $n \in \mathbb{N}$:

$$Oc_m =_\beta \begin{cases} \textbf{true} & \text{if } m \text{ is odd;} \\ \textbf{false} & \text{otherwise,} \end{cases}$$

and

$$Ec_m =_\beta \begin{cases} \textbf{true} & \text{if } m \text{ is even;} \\ \textbf{false} & \text{otherwise.} \end{cases}$$

1.7.20. Exercise. Define a λ-term P such that

$$P\ c_{n+1} =_\beta c_n$$

Hint: use the same trick as in the proof that the λ-definable functions are closed under primitive recursion. (Kleene got this idea during a visit at his dentist.)

1.7.21. Exercise. Define a λ-term **eq?** such that, for all $n, m \in \mathbb{N}$:

$$\textbf{eq?}\ c_n\ c_m =_\beta \begin{cases} \textbf{true} & \text{if } m = n; \\ \textbf{false} & \text{otherwise.} \end{cases}$$

Hint: use the fixed point theorem to construct a λ-term H such that

$$\begin{aligned} H\ c_n\ c_m \quad =_\beta \quad & \textbf{if}\ (\textbf{zero?}\ c_n) \\ & \textbf{then}\ (\textbf{if}\ (\textbf{zero?}\ c_m)\ \textbf{then true else false}) \\ & \textbf{else}\ (\textbf{if}\ (\textbf{zero?}\ c_m)\ \textbf{then false else}\ (H\ (P\ c_n)\ (P\ c_m))) \end{aligned}$$

where P is as in the preceding exercise.

Can you prove the result using instead the construction in Lemma 1.5.18?

1.7.22. Exercise. Define a λ-term H such that for all $n \in \mathbb{N}$:

$$H\ c_{2n} =_\beta c_n$$

1.7.23. Exercise. Define a λ-term F such that for all $n \in \mathbb{N}$:

$$H\ c_{n^2} =_\beta c_n$$

1.7.24. Exercise. Prove Corollary 1.5.25.

CHAPTER 2

Intuitionistic logic

The classical understanding of logic is based on the notion of *truth*. The truth of a statement is "absolute" and independent of any reasoning, understanding, or action. Statements are either true or false with no regard to any "observer". Here "false" means the same as "not true", and this is expressed by the *tertium non datur* principle that "$p \vee \neg p$" must hold no matter what the meaning of p is.

Needless to say, the information contained in the claim $p \vee \neg p$ is quite limited. Take the following sentence as an example:

> *There is seven 7's in a row somewhere in the decimal representation of the number π.*

Note that it may very well happen that nobody ever will be able to determine the truth of the above sentence. Yet we are forced to accept that one of the cases must necessarily hold. Another well-known example is as follows:

> *There are two irrational numbers x and y, such that x^y is rational.*

The proof of this fact is very simple: if $\sqrt{2}^{\sqrt{2}}$ is a rational number then we can take $x = y = \sqrt{2}$; otherwise take $x = \sqrt{2}^{\sqrt{2}}$ and $y = \sqrt{2}$.

The problem with this proof is that we do not know which of the two possibilities is the right one. Again, there is very little information in this proof, because it is not *constructive*.

These examples demonstrate some of the drawbacks of classical logic, and give hints on why intuitionistic (or constructive) logic is of interest. Although the roots of constructivism in mathematics reach deeply into the XIXth Century, the principles of intuitionistic logic are usually attributed to the works of the Dutch mathematician and philosopher Luitzen Egbertus Jan Brouwer from the beginning of XXth Century. Brouwer is also the inventor of the term "intuitionism", which was originally meant to denote a

philosophical approach to the foundations of mathematics, being in opposition to Hilbert's "formalism".

Intuitionistic logic as a branch of formal logic was developed later around the year 1930. The names to be quoted here are Heyting, Glivenko, Kolmogorov and Gentzen. To learn more about the history and motivations see [26] and Chapter 1 of [107].

2.1. Intuitive semantics

In order to understand intuitionism, one should forget the classical, Platonic notion of "truth". Now our judgements about statements are no longer based on any predefined value of that statement, but on the existence of a proof or "construction" of that statement.

The following rules explain the informal constructive semantics of propositional connectives. These rules are sometimes called the *BHK-interpretation* for Brouwer, Heyting and Kolmogorov. The algorithmic flavor of this definition will later lead us to the Curry-Howard isomorphism.

- *A construction of $\varphi_1 \wedge \varphi_2$ consists of a construction of φ_1 and a construction of φ_2;*
- *A construction of $\varphi_1 \vee \varphi_2$ consists of a number $i \in \{1, 2\}$ and a construction of of φ_i;*
- *A construction of $\varphi_1 \to \varphi_2$ is a method (function) transforming every construction of φ_1 into a construction of φ_2;*
- *There is no possible construction of $\bot$ (where $\bot$ denotes falsity).*

Negation $\neg\varphi$ is best understood as an abbreviation of an implication $\varphi \to \bot$. That is, we assert $\neg\varphi$ when the assumption of φ leads to an absurd. It follows that

- *A construction of $\neg\varphi$ is a method that turns every construction of φ into a non-existent object.*

Note that the equivalence between $\neg\varphi$ and $\varphi \to \bot$ holds also in classical logic. But note also that the intuitionistic statement $\neg\varphi$ is much stronger than just "there is no construction for φ".

2.1.1. Example. Consider the following formulas:

1. $\bot \to p$;
2. $((p \to q) \to p) \to p$;
3. $p \to \neg\neg p$;

4. $\neg\neg p \to p$;
5. $\neg\neg\neg p \to \neg p$;
6. $(\neg q \to \neg p) \to (p \to q)$;
7. $(p \to q) \to (\neg q \to \neg p)$;
8. $\neg(p \wedge q) \to (\neg p \vee \neg q)$;
9. $(\neg p \vee \neg q) \to \neg(p \wedge q)$;
10. $((p \leftrightarrow q) \leftrightarrow r) \leftrightarrow (p \leftrightarrow (q \leftrightarrow r))$;
11. $((p \wedge q) \to r) \leftrightarrow (p \to (q \to r))$;
12. $(p \to q) \leftrightarrow (\neg p \vee q)$;
13. $\neg\neg(p \vee \neg p)$.

These formulas are all classical tautologies. Some of them can be easily given a BHK-interpretation, but some of them cannot. For instance, a construction for formula 3, which should be written as "$p \to ((p \to \bot) \to \bot)$", is as follows:

> Given a proof of p, here is a proof of $(p \to \bot) \to \bot$: Take a proof of $p \to \bot$. It is a method to translate proofs of p into proofs of $\bot$. Since we have a proof of p, we can use this method to obtain a proof of $\bot$.

On the other hand, formula 4 does not seem to have such a construction. (The classical symmetry between formula 3 and 4 disappears!)

2.2. Natural deduction

The language of intuitionistic propositional logic is the same as the language of classical propositional logic. We assume an infinite set PV of *propositional variables* and we define the set Φ of *formulas* by induction, represented by the following grammar:

$$\Phi ::= \bot \mid PV \mid (\Phi \to \Phi) \mid (\Phi \vee \Phi) \mid (\Phi \wedge \Phi).$$

That is, our basic connectives are: implication $\to$, disjunction $\vee$, conjunction $\wedge$, and the constant $\bot$ (false).

2.2.1. Convention. The connectives $\neg$ and $\leftrightarrow$ are abbreviations. That is,

- $\neg\varphi$ abbreviates $\varphi \to \bot$;

- $\varphi \leftrightarrow \psi$ abbreviates $(\varphi \to \psi) \wedge (\psi \to \varphi)$.

2.2.2. Convention.

1. We sometimes use the convention that implication is right associative, i.e., we write e.g. $\varphi \to \psi \to \vartheta$ instead of $\varphi \to (\psi \to \vartheta)$.

2. We assume that negation has the highest, and implication the lowest priority, with no preference between $\vee$ and $\wedge$. That is, $\neg p \wedge q \to r$ means $((\neg p) \wedge q) \to r$.

3. And of course we forget about outermost parentheses.

In order to formalize the intuitionistic propositional calculus, we define a proof system, called *natural deduction*, which is motivated by the informal semantics of 2.1.

2.2.3. Warning. What follows is a quite simplified presentation of natural deduction, which is often convenient for technical reasons, but which is not always adequate. To describe the relationship between various proofs in finer detail, we shall consider a variant of the system in Chapter 4.

2.2.4. Definition.

(i) A *context* is a finite subset of Φ. We use Γ, Δ, etc. to range over contexts.

(ii) The relation $\Gamma \vdash \varphi$ is defined by the rules in Figure 2.1. We also write $\vdash_N$ for $\vdash$.

(iii) We write Γ, Δ instead of $\Gamma \cup \Delta$, and Γ, φ instead of $\Gamma, \{\varphi\}$. We also write $\vdash \varphi$ instead of $\{\} \vdash \varphi$.

(iv) A formal *proof* of $\Gamma \vdash \varphi$ is a finite tree, whose nodes are labelled by pairs of form (Γ', φ'), which will also be written $\Gamma' \vdash \varphi'$, satisfying the following conditions:

- The root label is $\Gamma \vdash \varphi$;
- All the leaves are labelled by axioms;
- The label of each father node is obtained from the labels of the sons using one of the rules.

(v) For infinite Γ we define $\Gamma \vdash \varphi$ to mean that $\Gamma_0 \vdash \varphi$, for some finite subset Γ_0 of Γ.

(vi) If $\vdash \varphi$ then we say that φ is a *theorem* of the intuitionistic propositional calculus.

$$\Gamma, \varphi \vdash \varphi \ (\mathrm{Ax})$$

$$\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi}\ (\wedge\mathrm{I}) \qquad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi}(\wedge\mathrm{E}) \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi}$$

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi}\ (\vee\mathrm{I}) \quad \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi} \qquad \frac{\Gamma, \varphi \vdash \rho \quad \Gamma, \psi \vdash \rho \quad \Gamma \vdash \varphi \vee \psi}{\Gamma \vdash \rho}\ (\vee\mathrm{E})$$

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\ (\to \mathrm{I}) \qquad \frac{\Gamma \vdash \varphi \to \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi}(\to \mathrm{E})$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash \varphi}\ (\bot\mathrm{E})$$

Figure 2.1: Intuitionistic propositional calculus

The proof system consists of an axiom scheme, and rules. For each logical connective (except $\bot$) we have one or two *introduction* rules and one or two *elimination* rules. An introduction rule for a connective $\bullet$ tells us how a conclusion of the form $\varphi \bullet \psi$ can be derived. An elimination rule describes the way in which $\varphi \bullet \psi$ can be used to derive other formulas. The intuitive meaning of $\Gamma \vdash \varphi$ is that φ is a consequence of the assumptions in Γ.

We give example proofs of our three favourite formulas:

2.2.5. Example. Let Γ abbreviate $\{\varphi \to (\psi \to \vartheta), \varphi \to \psi, \varphi\}$.

(i)

$$\frac{\varphi \vdash \varphi}{\vdash \varphi \to \varphi}\ (\to \mathrm{I})$$

(ii)

$$\frac{\dfrac{\varphi, \psi \vdash \varphi}{\varphi \vdash \psi \to \varphi}\ (\to \mathrm{I})}{\vdash \varphi \to (\psi \to \varphi)}\ (\to \mathrm{I})$$

(iii)

$$\dfrac{(\to \mathrm{E})\dfrac{\Gamma \vdash \varphi \to (\psi \to \vartheta) \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi \to \vartheta} \qquad \dfrac{\Gamma \vdash \varphi \to \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi}(\to \mathrm{E})}{\dfrac{\dfrac{\dfrac{\Gamma \vdash \vartheta}{\varphi \to (\psi \to \vartheta), \varphi \to \psi \vdash \varphi \to \vartheta}(\to \mathrm{I})}{\varphi \to (\psi \to \vartheta) \vdash (\varphi \to \psi) \to (\varphi \to \vartheta)}(\to \mathrm{I})}{\vdash (\varphi \to (\psi \to \vartheta)) \to (\varphi \to \psi) \to (\varphi \to \vartheta)}(\to \mathrm{I})}(\to \mathrm{E})$$

2.2.6. Remark. Note the distinction between the *relation* $\vdash$, and a formal *proof* of $\Gamma \vdash \varphi$.

The following properties will be useful.

2.2.7. Lemma. *Intuitionistic propositional logic is closed under weakening and substitution, that is,* $\Gamma \vdash \varphi$ *implies* $\Gamma, \psi \vdash \varphi$ *and* $\Gamma[p := \psi] \vdash \varphi[p := \psi]$, *where* $[p := \psi]$ *denotes a substitution of* ψ *for all occurrences of a propositional variable* p.

Proof. Easy induction with respect to the size of proofs. □

2.3. Algebraic semantics of classical logic

To understand better the algebraic semantics for intuitionistic logic let us begin with classical logic. Usually, semantics of classical propositional formulas is defined in terms of the two truth values, 0 and 1 as follows.

2.3.1. Definition. Let $\mathbb{B} = \{0, 1\}$.

(i) A *valuation in* $\mathbb{B}$ is a map $v : PV \to \mathbb{B}$; such a map will also be called a 0-1 *valuation.*

(ii) Given a 0-1 valuation v, define the map $[\![\bullet]\!]_v : \Phi \to \mathbb{B}$ by:

$$\begin{array}{lcll} [\![p]\!]_v & = & v(p), & \text{for } p \in PV; \\ [\![\bot]\!]_v & = & 0; & \\ [\![\varphi \vee \psi]\!]_v & = & \max\{[\![\varphi]\!]_v, [\![\psi]\!]_v\}; & \\ [\![\varphi \wedge \psi]\!]_v & = & \min\{[\![\varphi]\!]_v, [\![\psi]\!]_v\}; & \\ [\![\varphi \to \psi]\!]_v & = & \max\{1 - [\![\varphi]\!]_v, [\![\psi]\!]_v\}. & \end{array}$$

We also write $v(\varphi)$ for $[\![\varphi]\!]_v$.

(iii) A formula $\varphi \in \Phi$ is a *tautology* if $v(\varphi) = 1$ for all valuations in $\mathbb{B}$.

Let us consider an alternative semantics, based on the analogy between classical connectives and set-theoretic operations.

2.3.2. Definition. A *field of sets (over* X*)* is a nonempty family $\mathcal{R}$ of subsets of X, closed under unions, intersections and complement (to X).

It follows immediately that $\{\}, X \in \mathcal{R}$, for each field of sets $\mathcal{R}$ over X. Examples of fields of sets are:

(i) $P(X)$;

(ii) $\{\{\}, X\}$;

(iii) $\{A \subseteq X : A \text{ finite or } -A \text{ finite}\}$ ($-A$ is the complement of A).

2.3.3. Definition. Let $\mathcal{R}$ be a field of sets over X.

(i) A *valuation in* $\mathcal{R}$ is a map $v : PV \to \mathcal{R}$.

(ii) Given a valuation v in $\mathcal{R}$, define the map $[\![\bullet]\!]_v : \Phi \to X$ by:

$$\begin{array}{lcll} [\![p]\!]_v & = & v(p) & \text{for } p \in PV \\ [\![\bot]\!]_v & = & \{\} & \\ [\![\varphi \vee \psi]\!]_v & = & [\![\varphi]\!]_v \cup [\![\psi]\!]_v & \\ [\![\varphi \wedge \psi]\!]_v & = & [\![\varphi]\!]_v \cap [\![\psi]\!]_v & \\ [\![\varphi \to \psi]\!]_v & = & (X - [\![\varphi]\!]_v) \cup [\![\psi]\!]_v & \end{array}$$

We also write $v(\varphi)$ for $[\![\varphi]\!]_v$.

2.3.4. Proposition. *The above two approaches to semantics are equivalent, i.e., the following conditions are equivalent for each field of subsets* $\mathcal{R}$ *over a nonempty set* X*:*

1. φ *is a tautology;*

2. $v(\varphi) = X$*, for all valuations* v *in* $\mathcal{R}$*.*

Proof. (1) $\Rightarrow$ (2): Suppose that $v(\varphi) \neq X$. There is an element $a \in X$ such that $a \notin v(\varphi)$. Define a 0-1 valuation w so that $w(p) = 1$ iff $a \in v(p)$. Prove by induction that for all formulas ψ

$$w(\psi) = 1 \quad \text{iff} \quad a \in v(\psi).$$

Then $w(\varphi) \neq 1$.

(2) $\Rightarrow$ (1): A 0-1 valuation can be seen as a valuation in $\mathcal{R}$ that assigns only X and $\{\}$ to propositional variables. □

2.3.5. Definition. A *Boolean algebra* is an algebraic system of the form $\mathcal{B} = \langle B, \cup, \cap, -, 0, 1\rangle$, where:

- $\cup, \cap$ are associative and commutative;
- $(a \cup b) \cap c = (a \cap c) \cup (b \cap c)$ and $(a \cap b) \cup c = (a \cup c) \cap (b \cup c)$;
- $a \cup 0 = a$ and $a \cap 1 = a$;
- $-a \cup a = 1$ and $-a \cap a = 0$.

The relation $\leq$ defined by $a \leq b$ iff $a \cup b = b$ is a partial order[1] in every Boolean algebra, and the operations $\cap, \cup$ are the *glb* and *lub* operations w.r.t. this order.

[1] A transitive, reflexive and anti-symmetric relation.

The notion of a Boolean algebra is a straightforward generalization of the notion of a field of sets. Another example of a Boolean algebra is the algebra of truth values $\langle \mathbb{B}, \max, \min, -, 0, 1 \rangle$, where $-x$ is $1 - x$.

We can generalize the above set semantics to arbitrary Boolean algebras by replacing valuations in a field of sets by valuations in a Boolean algebra in the obvious way. But in fact, every Boolean algebra is isomorphic to a field of sets, so this generalization does not change our semantics.

2.4. Heyting algebras

We will now develop a semantics for intuitionistic propositional logic.

Let Φ be the set of all propositional formulas, let $\Gamma \subseteq \Phi$ (in particular Γ may be empty) and let $\sim$ be the following equivalence relation:

$$\varphi \sim \psi \quad \text{iff} \quad \Gamma \vdash \varphi \to \psi \text{ and } \Gamma \vdash \psi \to \varphi.$$

Let $\mathcal{L}_\Gamma = \Phi/_\sim = \{[\varphi]_\sim : \varphi \in \Phi\}$, and define a partial order $\leq$ over $\mathcal{L}_\Gamma$ by:

$$[\varphi]_\sim \leq [\psi]_\sim \quad \text{iff} \quad \Gamma \vdash \varphi \to \psi.$$

That $\sim$ is an equivalence relation and that $\leq$ is a well-defined partial order is a consequence of the following formulas being provable:

- $\varphi \to \varphi$;
- $(\varphi \to \psi) \to ((\psi \to \vartheta) \to (\varphi \to \vartheta))$;

In addition, we can define the following operations over $\mathcal{L}_\Gamma$:

$$\begin{aligned} [\alpha]_\sim \cup [\beta]_\sim &= [\alpha \vee \beta]_\sim; \\ [\alpha]_\sim \cap [\beta]_\sim &= [\alpha \wedge \beta]_\sim; \\ -[\alpha]_\sim &= [\neg\alpha]_\sim. \end{aligned}$$

These operations are well-defined, because the following formulas are provable:

- $(\varphi \to \varphi') \to (\neg\varphi' \to \neg\varphi)$;
- $(\varphi \to \varphi') \to ((\psi \to \psi') \to ((\varphi \vee \psi) \to (\varphi' \vee \psi')))$;
- $(\varphi \to \varphi') \to ((\psi \to \psi') \to ((\varphi \wedge \psi) \to (\varphi' \wedge \psi')))$.

We can go on and show that operations $\cap$ and $\cup$ are the *glb* and *lub* operations w.r.t. the relation $\leq$, and that the distributivity laws

$$(a \cup b) \cap c = (a \cap c) \cup (b \cap c) \quad \text{and} \quad (a \cap b) \cup c = (a \cup c) \cap (b \cup c)$$

are satisfied.[2] The class $[\bot]_\sim$ is the least element 0 of $\mathcal{L}_\Gamma$, because $\bot \to \varphi$ is provable, and $[\top]_\sim$, where $\top = \bot \to \bot$, is the top element 1. We have $[\top]_\sim = \{\varphi : \Gamma \vdash \varphi\}$. However, there are (not unexpected) difficulties with the complement operation: We have $-a \cap a = [\bot]_\sim$ but not necessarily $-a \cup a = [\top]_\sim$.

The best we can assert about $-a$ is that it is *the greatest element such that* $-a \cap a = 0$, and we can call it a *pseudo-complement*. Since negation is a special kind of implication, the above calls for a generalization. An element c is called a *relative pseudo-complement* of a with respect to b, iff c is the greatest element such that $a \cap c \leq b$. The relative pseudo-complement, if it exists, is denoted $a \Rightarrow b$.

It is not difficult to find out that in our algebra $\mathcal{L}_\Gamma$, often called a *Lindenbaum algebra*, we have $[\varphi]_\sim \Rightarrow [\psi]_\sim = [\varphi \to \psi]_\sim$.

We have just discovered a new type of algebra, called *Heyting algebra* or *pseudo-Boolean algebra*.

2.4.1. Definition. A *Heyting algebra* is an algebraic system of the form $\mathcal{H} = \langle H, \cup, \cap, \Rightarrow, -, 0, 1\rangle$, that satisfies the following conditions:

- $\cup, \cap$ are associative and commutative;
- $(a \cup b) \cap c = (a \cap c) \cup (b \cap c)$ and $(a \cap b) \cup c = (a \cup c) \cap (b \cup c)$;
- $a \cup 0 = a$ and $a \cap 1 = a$;
- $a \cup a = a$;
- $a \cap c \leq b$ is equivalent to $c \leq a \Rightarrow b$ (where $a \leq b$ stands for $a \cup b = b$);
- $-a = a \Rightarrow 0$.

The above conditions amount to as much as saying that $\mathcal{H}$ is a distributive lattice with zero and relative pseudo-complement defined for each pair of elements. In particular, each Boolean algebra is a Heyting algebra with $a \Rightarrow b$ defined as $-a \cup b$. The most prominent example of a Heyting algebra which is not a Boolean algebra is the algebra of open sets of a topological space, for instance the algebra of open subsets of the Euclidean plane $\mathbb{R}^2$.

2.4.2. Definition.

- The symbol $\varrho(a, b)$ denotes the distance between points $a, b \in \mathbb{R}^2$;
- A subset A of $\mathbb{R}^2$ is *open* iff for every $a \in A$ there is an $r > 0$ with $\{b \in \mathbb{R}^2 : \varrho(a, b) < r\} \subseteq A$;
- If A is a subset of $\mathbb{R}^2$ then $\mathrm{Int}(A)$ denotes the *interior* of A, i.e., the union of all open subsets of A.

[2] That is, $\mathcal{L}_\Gamma$ is a *distributive lattice*.

2.4.3. Proposition. *Let* $\mathcal{H} = \langle \mathcal{O}(\mathbb{R}^2), \cup, \cap, \Rightarrow, \sim, 0, 1 \rangle$, *where*

- $\mathcal{O}(\mathbb{R}^2)$ *is the family of all open subsets of* $\mathbb{R}^2$*;*
- *the operations* $\cap$, $\cup$ *are set-theoretic;*
- $A \Rightarrow B := \mathrm{Int}(-A \cup B)$, *for arbitrary open sets* A *and* B*;*
- $0 = \{\}$ *and* $1 = \mathbb{R}^2$.
- $\sim A = \mathrm{Int}(-A)$, *where* $-$ *is the set-theoretic complement.*

Then $\mathcal{H}$ *is a Heyting algebra.*

Proof. Exercise 2.7.6. □

In fact, every Heyting algebra is isomorphic to a subalgebra of the algebra of open sets of a topological space. A comprehensive study of the algebraic semantics for intuitionistic (and classical) logic is the book of Rasiowa and Sikorski [88]. See also Chapter 13 of [108].

The semantics of intuitionistic propositional formulas is now defined as follows.

2.4.4. Definition. Let $\mathcal{H} = \langle H, \cup, \cap, \Rightarrow, -, 0, 1 \rangle$ be a Heyting algebra.

(i) A *valuation* v in a $\mathcal{H}$ is a map $v : PV \to H$.

(ii) Given a valuation v in $\mathcal{H}$, define the map $[\![\bullet]\!]_v : \Phi \to H$ by:

$$\begin{array}{lcll} [\![p]\!]_v & = & v(p) & \text{for } p \in PV \\ [\![\bot]\!]_v & = & 0 & \\ [\![\varphi \vee \psi]\!]_v & = & [\![\varphi]\!]_v \cup [\![\psi]\!]_v & \\ [\![\varphi \wedge \psi]\!]_v & = & [\![\varphi]\!]_v \cap [\![\psi]\!]_v & \\ [\![\varphi \to \psi]\!]_v & = & [\![\varphi]\!]_v \Rightarrow [\![\psi]\!]_v & \end{array}$$

As usual, we write $v(\varphi)$ for $[\![\varphi]\!]_v$.

2.4.5. Notation. Let $\mathcal{H} = \langle H, \cup, \cap, \Rightarrow, -, 0, 1 \rangle$ be a Heyting algebra. We write:

- $\mathcal{H}, v \models \varphi$, whenever $v(\varphi) = 1$;
- $\mathcal{H} \models \varphi$, whenever $\mathcal{H}, v \models \varphi$, for all v;
- $\mathcal{H}, v \models \Gamma$, whenever $\mathcal{H}, v \models \varphi$, for all $\varphi \in \Gamma$;
- $\mathcal{H} \models \Gamma$, whenever $\mathcal{H}, v \models \Gamma$, for all v;
- $\models \varphi$, whenever $\mathcal{H}, v \models \varphi$, for all H, v;
- $\Gamma \models \varphi$, whenever $\mathcal{H}, v \models \Gamma$ implies $\mathcal{H}, v \models \varphi$, for all H and v.

We say that a formula φ such that $\models \varphi$ is *intuitionistically valid* or is an *intuitionistic tautology.* It follows from the following completeness theorem that the notions of a theorem and a tautology coincide for intuitionistic propositional calculus.

2.4.6. Theorem (Soundness and Completeness). *The following conditions are equivalent*

1. $\Gamma \vdash \varphi$;
2. $\Gamma \models \varphi$.

Proof. (1) $\Rightarrow$ (2): Verify that all provable formulas are valid in all Heyting algebras (induction w.r.t. proofs).

(2) $\Rightarrow$ (1): This follows from our construction of the Lindenbaum algebra. Indeed, suppose that $\Gamma \models \varphi$, but $\Gamma \nvdash \varphi$. Then $\varphi \not\sim \top$, i.e., $[\varphi]_\sim \neq 1$ in $\mathcal{L}_\Gamma$. Define a valuation v by $v(p) = [p]_\sim$ in $\mathcal{L}_\Gamma$ and prove by induction that $v(\psi) = [\psi]_\sim$, for all formulas ψ. It follows that $v(\varphi) \neq 1$, a contradiction. □

2.4.7. Example. To see that Peirce's law $((p \to q) \to p) \to p$ is not intuitionistically valid, consider the algebra of open subsets of $\mathbb{R}^2$. Take $v(p)$ to be the whole space without one point, and $v(q) = \{\}$. (Note that $a \Rightarrow b = 1$ in a Heyting algebra iff $a \leq b$.)

Intuitionistic logic is not finite-valued: There is no single finite Heyting algebra $\mathcal{H}$ such that $\vdash \varphi$ is equivalent to $\mathcal{H} \models \varphi$. Indeed, consider the formula $\bigvee\{p_i \leftrightarrow p_j : i, j = 0, \ldots, n \text{ and } i \neq j\}$. (Here the symbol $\bigvee$ abbreviates the disjunction of all members of the set.) This formula is not valid in general (Exercise 2.7.10), although it is valid in all Heyting algebras of cardinality at most n.

But finite Heyting algebras are sufficient for the semantics, as well as one sufficiently "rich" infinite algebra.

2.4.8. Theorem.

1. *A formula φ of length n is valid iff it is valid in all Heyting algebras of cardinality at most 2^{2^n};*
2. *Let $\mathcal{H}$ be the algebra of all open subsets of a dense-in-itself*[3] *metric space V (for instance the algebra of all open subsets of $\mathbb{R}^2$). Then $\mathcal{H} \models \varphi$ iff φ is valid.*

[3]Every point x is a limit of a sequence $\{x_n\}_n$, where $x_n \neq x$, for all n.

We give only a sketch of the main ideas of the proof, for the most curious reader. See [88] for details.

For (1), suppose $\mathcal{H}, v \not\models \varphi$, and let $\varphi_1, \ldots \varphi_m$ be all subformulas of φ. We construct a small model $\mathcal{H}'$ as a distributive sublattice of $\mathcal{H}$, with 0 and 1, generated by the elements $v(\varphi_1), \ldots, v(\varphi_m)$. This lattice is a Heyting algebra (warning: this is not a Heyting subalgebra of $\mathcal{H}$), and $\mathcal{H}', v' \not\models \varphi$, for a suitable v'.

As for (2), every finite algebra can be embedded into the algebra of open subsets of some open subset of V, and this algebra is a homomorphic image of $\mathcal{H}$. Thus, every valuation in a finite algebra can be translated into a valuation in $\mathcal{H}$.

From part (1) of the above theorem, it follows that intuitionistic propositional logic is decidable. But the upper bound obtained this way (double exponential space) can be improved down to polynomial space, with help of other methods, see [103].

2.5. Kripke semantics

We now introduce another semantics of intuitionistic propositional logic.

2.5.1. Definition. A *Kripke model* is defined as a tuple of the form $\mathcal{C} = \langle C, \leq, \Vdash \rangle$, where C is a non-empty set, $\leq$ is a partial order in C and $\Vdash$ is a binary relation between elements of C (called *states* or *possible worlds*) and propositional variables, that satisfies the following monotonicity condition:

$$\text{If } c \leq c' \text{ and } c \Vdash p \text{ then } c' \Vdash p.$$

The intuition is that elements of the model represent states of knowledge. The relation $\leq$ represents extending states by gaining more knowledge, and the relation $\Vdash$ tells which atomic formulas are known to be true in a given state. We extend this relation to provide meaning for propositional formulas as follows.

2.5.2. Definition. If $\mathcal{C} = \langle C, \leq, \Vdash \rangle$ is a Kripke model, then

- $c \Vdash \varphi \vee \psi$ iff $c \Vdash \varphi$ or $c \Vdash \psi$;
- $c \Vdash \varphi \wedge \psi$ iff $c \Vdash \varphi$ and $c \Vdash \psi$;
- $c \Vdash \varphi \to \psi$ iff $c' \Vdash \psi$, for all c' such that $c \leq c'$ and $c' \Vdash \varphi$;
- $c \Vdash \bot$ never happens.

We use $\mathcal{C} \Vdash \varphi$ to mean that $c \Vdash \varphi$, for all $c \in \mathcal{C}$.

Note that the above definition implies the following rule for negation:

- $c \Vdash \neg\varphi$ iff $c' \nVdash \varphi$, for all $c' \geq c$.

and the following generalized monotonicity (proof by easy induction):

$$\text{If } c \leq c' \text{ and } c \Vdash \varphi \text{ then } c' \Vdash \varphi.$$

We now want to show completeness of Kripke semantics. For this, we transform every Heyting algebra into a Kripke model.

2.5.3. Definition. A *filter* in a Heyting algebra $\mathcal{H} = \langle H, \cup, \cap, \Rightarrow, -, 0, 1\rangle$ is a nonempty subset F of H, such that

- $a, b \in F$ implies $a \cap b \in F$;
- $a \in F$ and $a \leq b$ implies $b \in F$.

A filter F is *proper* iff $F \neq H$. A proper filter F is *prime* iff $a \cup b \in F$ implies that either a or b belongs to F.

2.5.4. Lemma. *Let F be a proper filter in $\mathcal{H}$ and let $a \notin F$. There exists a prime filter G such that $F \subseteq G$ and $a \notin G$.*

We only give a hint for the proof that can be found e.g., in [88]. Consider the family of all filters G containing F and such that $a \notin G$, ordered by inclusion. Apply Kuratowski-Zorn Lemma to show that this family has a maximal element. This is a prime filter (Exercise 2.7.12) although it is not necessarily a maximal proper filter.

2.5.5. Lemma. *Let v be a valuation in a Heyting algebra $\mathcal{H}$. There is a Kripke model $\mathcal{C} = \langle C, \leq, \Vdash \rangle$, such that $\mathcal{H}, v \models \varphi$ iff $\mathcal{C} \Vdash \varphi$, for all formulas φ.*

Proof. We take C to be the set of all prime filters in $\mathcal{H}$. The relation $\leq$ is inclusion, and we define $F \Vdash p$ iff $v(p) \in F$. By induction, we prove that, for all formulas ψ,

$$F \Vdash \psi \quad \text{iff} \quad v(\psi) \in F. \tag{2.1}$$

The only nontrivial case of this induction is when $\psi = \psi' \to \psi''$. Assume $F \Vdash \psi' \to \psi''$, and suppose that $v(\psi' \to \psi'') = v(\psi') \Rightarrow v(\psi'') \notin F$. Take the least filter G' containing $F \cup \{v(\psi')\}$. Then

$$G' = \{b : b \geq f \cap v(\psi') \text{ for some } f \in F\},$$

and we have $v(\psi'') \notin G'$, in particular G' is proper. Indeed, otherwise $v(\psi'') \geq f \cap v(\psi')$, for some $f \in F$, and thus $f \leq v(\psi') \Rightarrow v(\psi'') \in F$ — a contradiction.

We extend G' to a prime filter G, not containing $v(\psi'')$. By the induction hypothesis, $G \Vdash \psi'$. Since $F \Vdash \psi' \to \psi''$, it follows that $G \Vdash \psi''$. That is, $v(\psi'') \in G$ — a contradiction.

For the converse, assume that $v(\psi' \to \psi'') \in F \subseteq G \Vdash \psi'$. From the induction hypothesis we have $v(\psi') \in G$ and since $F \subseteq G$ we obtain $v(\psi') \Rightarrow v(\psi'') \in G$. Thus $v(\psi'') \geq v(\psi') \cap (v(\psi') \Rightarrow v(\psi'')) \in G$, and from the induction hypothesis we conclude $G \Vdash \psi''$ as desired.

The other cases are easy. Note that primality is essential for disjunction. Having shown (2.1), assume that $\mathcal{C} \Vdash \varphi$ and $\mathcal{H}, v \not\models \varphi$. Then $v(\varphi) \neq 1$ and there exist a proper filter not containing $v(\varphi)$. This filter extends to a prime filter G such that $v(\varphi) \notin G$ and thus $G \nVdash \varphi$. On the other hand, if $\mathcal{H}, v \models \varphi$, then $v(\varphi) = 1$ and 1 belongs to all filters in $\mathcal{H}$. □

2.5.6. Theorem. *The sequent $\Gamma \vdash \varphi$ is provable iff for all Kripke models $\mathcal{C}$, the condition $\mathcal{C} \Vdash \Gamma$ implies $\mathcal{C} \Vdash \varphi$.*

Proof. The left-to-right part is shown by induction (Exercise 2.7.13). For the other direction assume $\Gamma \nvdash \varphi$. Then $\mathcal{H}, v \models \Gamma$ but $\mathcal{H}, v \not\models \varphi$, for some $\mathcal{H}, v$. From the previous lemma we have a Kripke model $\mathcal{C}$ with $\mathcal{C} \Vdash \Gamma$ and $\mathcal{C} \nVdash \varphi$. □

Here is a nice application of Kripke semantics.

2.5.7. Proposition. *If $\vdash \varphi \vee \psi$ then either $\vdash \varphi$ or $\vdash \psi$.*

Proof. Assume $\nvdash \varphi$ and $\nvdash \psi$. There are Kripke models $\mathcal{C}_1 = \langle C_1, \leq_1, \Vdash_1 \rangle$ and $\mathcal{C}_2 = \langle C_2, \leq_2, \Vdash_2 \rangle$ and states $c_1 \in C_1$ and $c_2 \in C_2$, such that $c_1 \nVdash \varphi$ and $c_2 \nVdash \psi$. Without loss of generality we can assume that c_1 and c_2 are least elements of $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively, and that $C_1 \cap C_2 = \{\}$. Let $\mathcal{C} = \langle C_1 \cup C_2 \cup \{c_0\}, \leq, \Vdash \rangle$, where $c_0 \notin C_1 \cup C_2$, the order is the union of $\leq_1$ and $\leq_2$ extended by c_0 taken as the least element, and $\Vdash$ is the union of $\Vdash_1$ and $\Vdash_2$. That is,

$$c_0 \nVdash p,$$

for all variables p. It is easy to see that this is a Kripke model. In addition we have $\mathcal{C}, c_1 \Vdash \vartheta$ iff $\mathcal{C}_1, c_1 \Vdash \vartheta$, for all formulas ϑ, and a similar property holds for c_2.

Now suppose that $\vdash \varphi \vee \psi$. By soundness, we have $c_0 \Vdash \varphi \vee \psi$, and thus either $c_0 \Vdash \varphi$ or $c_0 \Vdash \psi$, by definition of $\Vdash$. Then either $c_1 \Vdash \varphi$ or $c_2 \Vdash \psi$, because of monotonicity. □

2.6. The implicational fragment

The most important logical conjective is the implication. Thus, it is meaningful to study the fragment of propositional calculus with only one connective: the implication. This is the true *minimal logic*. The natural deduction

system for the implicational fragment consists of the rules ($\to$ E), ($\to$ I) and the axiom scheme.

2.6.1. Theorem. *The implicational fragment of intuitionistic propositional calculus is complete with respect to Kripke models, i.e.,* $\Gamma \vdash \varphi$ *is provable iff for all Kripke models* $\mathcal{C}$*, the condition* $\mathcal{C} \Vdash \Gamma$ *implies* $\mathcal{C} \Vdash \varphi$*.*

Proof. The implication from left to right follows from soundness of the full natural deduction system, of which our minimal logic is a fragment. For the proof in the other direction, let us assume that $\Gamma \nvdash \varphi$. We define a Kripke model $\mathcal{C} = \langle C, \leq, \Vdash \rangle$, where

$$C = \{\Delta : \Gamma \subseteq \Delta, \text{ and } \Delta \text{ is closed under } \vdash\}.$$

That is, $\Delta \in C$ means that $\Delta \vdash \psi$ implies $\psi \in \Delta$.

The relation $\leq$ is inclusion and $\Vdash$ is $\in$, that is, $\Delta \Vdash p$ holds iff $p \in \Delta$, for all propositional variables p. By induction we show the following claim:

$$\Delta \Vdash \psi \quad \text{iff} \quad \psi \in \Delta,$$

for all implicational formulas ψ and all states Δ. The case of a variable is immediate from the definition. Let ψ be $\psi_1 \to \psi_2$ and let $\Delta \Vdash \psi$. Take $\Delta' = \{\vartheta : \Delta, \psi_1 \vdash \vartheta\}$. Then $\psi_1 \in \Delta'$ and, by the induction hypothesis, $\Delta' \Vdash \psi_1$. Thus $\Delta' \Vdash \psi_2$ and we get $\psi_2 \in \Delta'$, again by the induction hypothesis. Thus, $\Delta, \psi_1 \vdash \psi_2$, and by ($\to$ I) we get what we want.

Now assume $\psi \in \Delta$ (that is $\Delta \vdash \psi$) and take $\Delta' \geq \Delta$ with $\Delta' \Vdash \psi_1$. Then $\psi_1 \in \Delta'$, i.e., $\Delta' \vdash \psi_1$. But also $\Delta' \vdash \psi_1 \to \psi_2$, because $\Delta \subseteq \Delta'$. By ($\to$ E) we can derive $\Delta' \vdash \psi_2$, which means, by the induction hypothesis, that $\Delta' \Vdash \psi_2$. □

The completeness theorem has a very important consequence: the conservativity of the full intuitionistic propositional calculus over its implicational fragment.

2.6.2. Theorem. *Let* φ *be an implicational formula, and let* Γ *be a set of implicational formulas. If* $\Gamma \vdash \varphi$ *can be derived in the intuitionistic propositional calculus then it can be derived in the implicational fragment.*

Proof. Easy. But note that we use only one half from the two completeness theorems 2.5.6 and 2.6.1: we need only soundness of the full logic and only the other direction for the fragment. □

2.7. Exercises

2.7.1. Exercise. Find constructions for formulas (1), (3), (5), (7), (9), (11) and (13) of Example 2.1.1, and do not find constructions for the other formulas.

2.7.2. Exercise. Prove Lemma 2.2.7.

2.7.3. Exercise. Give natural deduction proofs for the formulas of Exercise 2.7.1.

2.7.4. Exercise. Show that the relation $\leq$ defined in a Boolean algebra by the condition $a \leq b$ iff $a \cup b = b$ is a partial order and that

- $a \cap b \leq a$;
- the condition $a \leq b$ is equivalent to $a \cap b = a$;
- the operations $\cup$ and $\cap$ are respectively the upper and lower bound wrt. $\leq$;
- the constants 0 and 1 are respectively the bottom and top element.

2.7.5. Exercise. Show that the relation $\leq$ defined in a Heyting algebra by the condition $a \leq b$ iff $a \cup b = b$ is a partial order and that

- $-a \cap a = 0$;
- $(a \cup b) \cap a = a$ and $a \cap b \leq a$;
- the condition $a \leq b$ is equivalent to $a \cap b = a$, and to $a \Rightarrow b = 1$;
- the operations $\cup$ and $\cap$ are respectively the upper and lower bound wrt. $\leq$;
- the constants 0 and 1 are respectively the bottom and top element.

2.7.6. Exercise. Prove Proposition 2.4.3.

2.7.7. Exercise. Fill in the details of the proof that the Lindenbaum algebra $\mathcal{L}_\Gamma$ of 2.4 is indeed a Heyting algebra.

2.7.8. Exercise. Complete the proof of the completeness theorem 2.4.6.

2.7.9. Exercise. Show that the formulas (2), (4), (6), (8) and (10) are not intuitionistically valid. (Use open subsets of $\mathbb{R}^2$ or construct Kripke models.)

2.7.10. Exercise. Show that the formula $\bigvee\{p_i \leftrightarrow p_j : i, j = 0, \ldots, n \text{ and } i \neq j\}$ is not intuitionistically valid.

2.7.11. Exercise. A filter is *maximal* iff it is a maximal proper filter. Show that each maximal filter is prime. Show also that in a Boolean algebra every prime filter is maximal.

2.7.12. Exercise. Complete the proof of Lemma 2.5.4.

2.7.13. Exercise. Complete the proof of Theorem 2.5.6, part $\Rightarrow$. (*Hint:* choose a proper induction hypothesis.)

2.7.14. Exercise. Can the proof of Theorem 2.6.1 be generalized to the full propositional calculus?

2.7.15. Exercise. A state c in a Kripke model $\mathcal{C}$ *determines* p iff either $c \Vdash p$ or $c \Vdash \neg p$. Define a 0-1 valuation v_c by $v_c(p) = 1$ iff $c \Vdash p$. Show that if c determines all propositional variables in φ then $v_c(\varphi) = 1$ implies $c \Vdash \varphi$.

2.7.16. Exercise. Let φ be a classical tautology such that all propositional variables in φ are among $p_1, \ldots, p_n$. Show that the formula $(p_1 \vee \neg p_1) \rightarrow \cdots \rightarrow (p_n \vee \neg p_n) \rightarrow \varphi$ is intuitionistically valid.

2.7.17. Exercise. Prove the Glivenko theorem: A formula φ is a classical tautology iff $\neg\neg\varphi$ is an intuitionistic tautology.

2.7.18. Warning. The Glivenko theorem does not extend to first-order logic.

CHAPTER 3

Simply typed λ-calculus

Recall from the first chapter that a λ-term, unlike the functions usually considered in mathematics, does not have a fixed *domain* and *range*. Thus, whereas we would consider the function $n \mapsto n^2$ as a function from natural numbers to natural numbers (or from integers to natural numbers, etc.) there is no corresponding requirement in λ-calculus. Or, to be more precise, there is no corresponding requirement in *type-free* λ-calculus.

Curry [23] and Church [18] also introduced versions of their systems with *types.* These systems form the topic of the present chapter.

3.1. Simply typed λ-calculus à la Curry

We begin with the simply typed λ-calculus à la Curry.

3.1.1. Definition.

(i) Let U denote a denumerably infinite alphabet whose members will be called *type variables*. The set Π of *simple types* is the set of strings defined by the grammar:

$$\Pi ::= U \mid (\Pi \to \Pi)$$

We use $\alpha, \beta, \ldots$ to denote arbitrary type variables, and $\sigma, \tau, \ldots$ to denote arbitrary types. We omit outermost parentheses, and omit other parentheses with the convention that $\to$ associates to the right.

(ii) The set C of *contexts* is the set of all sets of pairs of the form

$$\{x_1 : \tau_1, \ldots, x_n : \tau_n\}$$

with $\tau_1, \ldots, \tau_n \in \Pi$, $x_1, \ldots, x_n \in V$ (variables of Λ) and $x_i \neq x_j$ for $i \neq j$.

(iii) The *domain* of a context $\Gamma = \{x_1 : \tau_1, \ldots, x_n : \tau_n\}$ is defined by:

$$\mathrm{dom}(\Gamma) = \{x_1, \ldots, x_n\}$$

We write $x : \tau$ for $\{x : \tau\}$ and Γ, Γ' for $\Gamma \cup \Gamma'$ if $\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma') = \{\}$.

(iv) The *range* of a context $\Gamma = \{x_1 : \tau_1, \ldots, x_n : \tau_n\}$ is defined by:

$$|\Gamma| = \{\tau \in \Pi \mid (x : \tau) \in \Gamma, \text{ for some } x\}.$$

(v) The *typability* relation $\vdash$ on $C \times \Lambda \times \Pi$ is defined by:

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau} \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash M\ N : \tau}$$

where we require that $x \notin \mathrm{dom}(\Gamma)$ in the first and second rule.

(vi) The simply typed λ-calculus $\lambda{\to}$ is the triple $(\Lambda, \Pi, \vdash)$. To distinguish between this system and variants, the present one will also be called *simply typed λ-calculus à la Curry* or just $\lambda{\to}$ à la Curry.

3.1.2. Example. Let σ, τ, ρ be arbitrary types. Then:

(i) $\vdash \lambda x.x : \sigma \to \sigma$;

(ii) $\vdash \lambda x.\lambda y.x : \sigma \to \tau \to \sigma$;

(iii) $\vdash \lambda x.\lambda y.\lambda z.x\ z\ (y\ z) : (\sigma \to \tau \to \rho) \to (\sigma \to \tau) \to \sigma \to \rho$.

3.1.3. Definition. If $\Gamma \vdash M : \sigma$ then we say that M *has type* σ *in* Γ. We say that $M \in \Lambda$ is *typable* if there are Γ and σ such that $\Gamma \vdash M : \sigma$.

The set of typable terms is a subset—in fact, a proper subset—of the set of all λ-terms. In this subset, restrictions are made regarding which λ-terms may be applied to other λ-terms.

Very informally, the type variables denote some unspecified sets, and $\sigma \to \tau$ denotes the set of functions from σ to τ. Saying that M has type $\sigma \to \tau$ in Γ then intuitively means that this set of functions contains the particular function that M informally denotes. For instance, $\vdash \lambda x.x : \sigma \to \sigma$ informally states that the identity function is a function from a certain set to itself.

A context is an assumption that some elements $x_1, \ldots, x_n$ have certain types $\sigma_1, \ldots, \sigma_n$, respectively. Moreover, if x has type σ and M has type τ then $\lambda x.M$ has type $\sigma \to \tau$. This reflects the intuition that if M denotes an element of τ for each x in σ, then $\lambda x.M$ denotes a function from σ to τ. In a similar vein, if M has type $\sigma \to \tau$ and N has type σ, then $M\ N$ has type τ.

3.1.4. Warning. The idea that types denote sets should not be taken too literally. For instance, if A and B are sets then the set of functions from A to A is disjoint from the set of functions from B to B. In contrast, if σ and τ are different simple types, then a single term may have both types, at least in the version of simply typed λ-calculus presented above.

Just like λ-calculus provides a foundation of higher-order functional programming languages like LISP and Scheme, various typed λ-calculi provide a foundation of programming languages with types like Pascal, ML, and Haskell. Typed λ-calculi are also of independent interest in proof theory as we shall have frequent occasion to see in these notes.

We conclude this section with a brief review of some of the most fundamental properties of $\lambda{\to}$. The survey follows [8].

The following shows that only types of free variables of a term matter in the choice contexts.

3.1.5. Lemma (Free variables lemma). *Assume that* $\Gamma \vdash M : \sigma$. *Then:*

(i) $\Gamma \subseteq \Gamma'$ *implies* $\Gamma' \vdash M : \sigma$;

(ii) $\mathrm{FV}(M) \subseteq \mathrm{dom}(\Gamma)$;

(iii) $\Gamma' \vdash M : \sigma$ *where* $\mathrm{dom}(\Gamma') = \mathrm{FV}(M)$ *and* $\Gamma' \subseteq \Gamma$.

Proof. (i) by induction on the derivation of $\Gamma \vdash M : \sigma$. As in [8] we present the proof in some detail and omit such details in the remainder.

1. The derivation is

$$\frac{}{\Delta, x : \sigma \vdash x : \sigma}$$

where $\Gamma = \Delta, x : \sigma$, $x \notin \mathrm{dom}(\Delta)$ and $M = x$. Since $\Gamma \subseteq \Gamma'$ and Γ' is a context, $\Gamma' = \Delta', x : \sigma$ for some Δ' with $x \notin \mathrm{dom}(\Delta')$. Hence $\Delta', x : \sigma \vdash x : \sigma$, as required.

2. The derivation ends in

$$\frac{\Gamma, x : \tau_1 \vdash P : \tau_2}{\Gamma \vdash \lambda x.P : \tau_1 \to \tau_2}$$

where $x \notin \mathrm{dom}(\Gamma)$, $\sigma = \tau_1 \to \tau_2$, and $M = \lambda x.P$. Without loss of generality we can assume that $x \notin \mathrm{dom}(\Gamma')$. Then $\Gamma, x : \tau_1 \subseteq \Gamma', x : \tau_1$ so by the induction hypothesis $\Gamma', x : \tau_1 \vdash P : \tau_2$. Then we also have $\Gamma \vdash \lambda x.P : \tau_1 \to \tau_2$, as required.

3. The derivation ends in

$$\frac{\Gamma \vdash P : \tau \to \sigma \quad \Gamma \vdash Q : \tau}{\Gamma \vdash P\,Q : \sigma}$$

where $M = P\,Q$. By the induction hypothesis (twice) $\Gamma' \vdash P : \tau \to \sigma$ and $\Gamma' \vdash Q : \tau$, and then $\Gamma' \vdash P\,Q : \sigma$, as required.

(ii)-(iii) by induction on the derivation of $\Gamma \vdash M : \sigma$. □

The following shows how the type of some term must have been obtained depending on the form of the term.

3.1.6. Lemma (Generation lemma).

(i) $\Gamma \vdash x : \sigma$ *implies* $x : \sigma \in \Gamma$*;*

(ii) $\Gamma \vdash M\ N : \sigma$ *implies that there is a* τ *such that* $\Gamma \vdash M : \tau \to \sigma$ *and* $\Gamma \vdash N : \tau$*.*

(iii) $\Gamma \vdash \lambda x.M : \sigma$ *implies that there are* τ *and* ρ *such that* $\Gamma, x : \tau \vdash M : \rho$ *and* $\sigma = \tau \to \rho$*.*

Proof. By induction on the length of the derivation. □

3.1.7. Definition. The *substitution of type* τ *for type variable* α *in type* σ*,* written $\sigma[\alpha := \tau]$, is defined by:

$$\begin{array}{lcll} \alpha[\alpha := \tau] & = & \tau & \\ \beta[\alpha := \tau] & = & \beta & \text{if } \alpha \neq \beta \\ (\sigma_1 \to \sigma_2)[\alpha := \tau] & = & \sigma_1[\alpha := \tau] \to \sigma_2[\alpha := \tau] & \end{array}$$

The notation $\Gamma[\alpha := \tau]$ stands for the context $\{(x : \sigma[\alpha := \tau]) \mid (x : \sigma) \in \Gamma\}$.

The following shows that the type variables range over all types; this is a limited form of *polymorphism* [90]; we will hear much more about polymorphism later. The proposition also shows, similarly, that free term variables range over arbitrary terms.

3.1.8. Proposition (Substitution lemma).

(i) *If* $\Gamma \vdash M : \sigma$*, then* $\Gamma[\alpha := \tau] \vdash M : \sigma[\alpha := \tau]$*.*

(ii) *If* $\Gamma, x : \tau \vdash M : \sigma$ *and* $\Gamma \vdash N : \tau$ *then* $\Gamma \vdash M[x := N] : \sigma$*.*

Proof. By induction on the derivation of $\Gamma \vdash M : \sigma$ and generation of $\Gamma, x : \tau \vdash M : \sigma$, respectively. □

The following shows that reduction preserves typing.

3.1.9. Proposition (Subject reduction). *If* $\Gamma \vdash M : \sigma$ *and* $M \to_\beta N$*, then* $\Gamma \vdash N : \sigma$*.*

Proof. By induction on the derivation of $M \to_\beta N$ using the substitution lemma and the generation lemma. □

3.1.10. Remark. The similar property

$$\Gamma \vdash N : \sigma \ \& \ M \twoheadrightarrow_\beta N \ \Rightarrow \ \Gamma \vdash M : \sigma$$

is called *subject expansion* and does *not* hold in $\lambda\!\!\rightarrow$, see Exercise 3.6.2.

3.1.11. Corollary. *If* $\Gamma \vdash M : \sigma$ *and* $M \twoheadrightarrow_\beta N$, *then* $\Gamma \vdash N : \sigma$.

3.1.12. Theorem (Church-Rosser property for typable terms). *Suppose that* $\Gamma \vdash M : \sigma$. *If* $M \twoheadrightarrow_\beta N$ *and* $M \twoheadrightarrow_\beta N'$, *then there exists an* L *such that* $N \twoheadrightarrow_\beta L$ *and* $N' \twoheadrightarrow_\beta L$ *and* $\Gamma \vdash L : \sigma$.

Proof. By the Church-Rosser property for λ-terms and the subject reduction property. □

3.2. Simply typed λ-calculus à la Church

As mentioned earlier, simply typed λ-calculus was introduced by Curry [23] and Church [18]. More precisely, Curry considered types for *combinatory logic,* but his formulation was later adapted to λ-calculus [24].

There were several other important differences between the systems introduced by Church and Curry.

In Curry's system the terms are those of type-free λ-calculus and the typing relation selects among these the typable terms. For instance, $\lambda x.x$ is typable, whereas $\lambda x.x\ x$ is not.

In Church's original system, the typing rules were built into the term formation rules, as follows. Let V_σ denote a denumerable set of variables for each $\sigma \in \Pi$. Then define the set Λ_σ of simply typed terms of type σ by the clauses:

$$\begin{array}{lcl} x \in V_\sigma & \Rightarrow & x \in \Lambda_\sigma \\ M \in \Lambda_{\sigma\to\tau} \ \& \ N \in \Lambda_\sigma & \Rightarrow & M\ N \in \Lambda_\tau \\ M \in \Lambda_\tau \ \& \ x \in \Lambda_\sigma & \Rightarrow & \lambda x^\sigma.M \in \Lambda_{\sigma\to\tau} \end{array}$$

The set of all simply typed terms is then taken as the union over all simple types σ of the simply typed terms of type σ.

Instead of assuming that the set of variables is partitioned into disjoint sets indexed by the set of simple types, we can use contexts to decide the types of variables as in the system à la Curry. Also, as in the system à la Curry, we can select the typable terms among a larger set. This yields the following, more common, formulation of simply typed λ-calculus à la Church.

3.2.1. Definition.

(i) The set Λ_Π of pseudo-terms is defined by the following grammar:

$$\Lambda_\Pi ::= V \mid (\lambda x{:}\,\Pi\, \Lambda_\Pi) \mid (\Lambda_\Pi\ \Lambda_\Pi)$$

where V is the set of (λ-term) variables and Π is the set of simple types.[1] We adopt the same terminology, notation, and conventions for pseudo-terms as for λ-terms, see 1.3–1.10, mutatis mutandis.

(ii) The *typability* relation $\Vdash$ on $C \times \Lambda_\Pi \times \Pi$ is defined by:[2]

$$\frac{}{\Gamma, x:\tau \vdash^* x:\tau} \qquad \frac{\Gamma, x:\sigma \vdash^* M:\tau}{\Gamma \vdash^* \lambda x{:}\sigma.M : \sigma\to\tau} \qquad \frac{\Gamma \vdash^* M:\sigma\to\tau \quad \Gamma \vdash^* N:\sigma}{\Gamma \vdash^* M\ N:\tau}$$

where we require that $x \notin \mathrm{dom}(\Gamma)$ in the first and second rule.

(iii) The simply typed λ-calculus à la Church ($\lambda{\to}$ *à la Church,* for short) is the triple $(\Lambda_\Pi, \Pi, \Vdash)$.

(iv) If $\Gamma \vdash^* M:\sigma$ then we say that M *has type* σ *in* Γ. We say that $M \in \Lambda_\Pi$ is *typable* if there are Γ and σ such that $\Gamma \vdash^* M:\sigma$.

3.2.2. Example. Let σ, τ, ρ be arbitrary simple types. Then:

(i) $\vdash^* \lambda x{:}\sigma.x : \sigma\to\sigma$;

(ii) $\vdash^* \lambda x{:}\sigma.\lambda y{:}\tau.x : \sigma\to\tau\to\sigma$;

(iii) $\vdash^* \lambda x{:}\sigma{\to}\tau{\to}\rho.\lambda y{:}\sigma{\to}\tau.\lambda z{:}\sigma.(x\ z)\ y\ z : (\sigma{\to}\tau{\to}\rho){\to}(\sigma{\to}\tau){\to}\sigma{\to}\rho$.

Even with the formulation of $\lambda{\to}$ à la Church in Definition 3.2.1 an important difference with $\lambda{\to}$ à la Curry remains: in Church's system abstractions have *domains*, i.e. are of the form $\lambda x{:}\,\sigma.M$, whereas in Curry's system abstractions have no domain, i.e. are of the form $\lambda x.M$. Thus, in Church's system one writes

$$\lambda x{:}\sigma.x : \sigma\to\sigma,$$

whereas in Curry's system one writes

$$\lambda x.x : \sigma\to\sigma.$$

[1] Strictly speaking, we should proceed as in the case of λ-terms and define a notion of pre-pseudo-terms, then define substitution and α-equivalence on these, and finally adopt the convention that by $M \in \Lambda_\Pi$ we always mean the α-equivalence class, see 1.13–1.19. We omit the details.

[2] In this chapter it is useful to distinguish syntactically between typing in the system à la Church and the system à la Curry, and therefore we use $\vdash^*$ here. In later chapters we shall also use $\vdash$ for $\vdash^*$.

The two different systems—Curry's and Church's—represent two different paradigms in programming languages. In Church's system the programmer has to explicitly write the types for all variables used in the program as in, e.g., Pascal, whereas in Curry's approach the programmer merely writes functions, and it is then the job of the compiler or the programming environment to infer the types of variables, as e.g., in ML and Haskell.

Having introduced a new set of terms (pseudo-terms instead of λ-terms) we are obliged to introduce the notions of substitution, reduction, etc., for the new notion. This is carried out briefly below. We reuse much notation and terminology.

3.2.3. Definition. For $M \in \Lambda_\Pi$ define the set $\mathrm{FV}(M)$ of *free variables* of M as follows.

$$\begin{array}{lcl} \mathrm{FV}(x) & = & \{x\} \\ \mathrm{FV}(\lambda x{:}\sigma.P) & = & \mathrm{FV}(P)\backslash\{x\} \\ \mathrm{FV}(P\ Q) & = & \mathrm{FV}(P) \cup \mathrm{FV}(Q) \end{array}$$

If $\mathrm{FV}(M) = \{\}$ then M is called *closed.*

3.2.4. Definition. For $M, N \in \Lambda_\Pi$ and $x \in V$, the *substitution of N for x in M,* written $M[x := N]$, is defined as follows:

$$\begin{array}{lcll} x[x := N] & = & N & \\ y[x := N] & = & y & \text{if } x \neq y \\ (P\ Q)[x := N] & = & P[x := N]\ Q[x := N] & \\ (\lambda y{:}\sigma.P)[x := N] & = & \lambda y{:}\sigma.P[x := N] & \text{where } x \neq y \text{ and } y \notin \mathrm{FV}(N) \end{array}$$

3.2.5. Definition. Let $\to_\beta$ be the smallest relation on Λ_Π closed under the rules:

$$\begin{array}{lcl} (\lambda x{:}\sigma\ .\ P)\ Q \to_\beta P[x := Q] & & \\ P \to_\beta P' & \Rightarrow & \forall x \in V, \sigma \in \Pi:\ \ \lambda x{:}\sigma.P \to_\beta \lambda x{:}\sigma.P' \\ P \to_\beta P' & \Rightarrow & \forall Z \in \Lambda:\ \ P\ Z \to_\beta P'\ Z \\ P \to_\beta P' & \Rightarrow & \forall Z \in \Lambda:\ \ Z\ P \to_\beta Z\ P' \end{array}$$

A term of form $(\lambda x : \sigma\ .\ P)\ Q$ is called a *β-redex,* and $P[x := Q]$ is called its *β-contractum.* A term M is a *β-normal form* if there is no term N with $M \to_\beta N$.

3.2.6. Definition.

(i) The relation $\twoheadrightarrow_\beta$ (*multi-step β-reduction*) is the transitive-reflexive closure of $\to_\beta$;

(ii) The relation $=_\beta$ (*β-equality)* is the transitive-reflexive-symmetric closure of $\to_\beta$.

We end the section by briefly repeating the development in the preceding subsection for simply typed λ-calculus à la Church.

3.2.7. Lemma (Free variables lemma). *Let* $\Gamma \vdash^* M : \sigma$. *Then:*

(i) $\Gamma \subseteq \Gamma'$ *implies* $\Gamma' \vdash^* M : \sigma$;

(ii) $\mathrm{FV}(M) \subseteq \mathrm{dom}(\Gamma)$;

(iii) $\Gamma' \vdash^* M : \sigma$ *where* $\mathrm{dom}(\Gamma') = \mathrm{FV}(M)$ *and* $\Gamma' \subseteq \Gamma$.

Proof. See the Exercises. □

3.2.8. Lemma (Generation lemma).

(i) $\Gamma \vdash^* x : \sigma$ *implies* $x : \sigma \in \Gamma$;

(ii) $\Gamma \vdash^* M\ N : \sigma$ *implies that there is a* τ *such that* $\Gamma \vdash^* M : \tau \to \sigma$ *and* $\Gamma \vdash^* N : \tau$.

(iii) $\Gamma \vdash^* \lambda x{:}\tau.M : \sigma$ *implies that there is a* ρ *such that* $\Gamma, x : \tau \vdash^* M : \rho$ *and* $\sigma = \tau \to \rho$.

Proof. See the Exercises. □

3.2.9. Proposition (Substitution lemma).

(i) *If* $\Gamma \vdash^* M : \sigma$, *then* $\Gamma[\alpha := \tau] \vdash^* M : \sigma[\alpha := \tau]$.

(ii) *If* $\Gamma, x : \tau \vdash^* M : \sigma$ *and* $\Gamma \vdash^* N : \tau$ *then* $\Gamma \vdash^* M[x := N] : \sigma$.

Proof. See the Exercises. □

3.2.10. Proposition (Subject reduction). *If* $\Gamma \vdash^* M : \sigma$ *and* $M \to_\beta N$, *then* $\Gamma \vdash^* N : \sigma$.

Proof. See the Exercises. □

3.2.11. Theorem (Church-Rosser property). *Suppose that* $\Gamma \vdash^* M : \sigma$. *If* $M \twoheadrightarrow_\beta N$ *and* $M \twoheadrightarrow_\beta N'$, *then there exists an* L *such that* $N \twoheadrightarrow_\beta L$ *and* $N' \twoheadrightarrow_\beta L$ *and* $\Gamma \vdash^* L : \sigma$.

Proof. One way to obtain this result is to repeat for Λ_Π an argument similar to that we used for untyped terms, and then use the subject reduction property. Another method, based on so called *logical relations* can be found in [74]. □

The following two properties of simply typed λ-calculus à la Church do not hold for the Curry system. Note that (ii) implies the subject expansion property—see Remark 3.1.10.

3.2.12. PROPOSITION (Uniqueness of types).

(i) *If* $\Gamma \vdash^* M : \sigma$ *and* $\Gamma \vdash^* M : \tau$ *then* $\sigma = \tau$.

(ii) *If* $\Gamma \vdash^* M : \sigma$ *and* $\Gamma \vdash^* N : \tau$ *and* $M =_\beta N$, *then* $\sigma = \tau$.

PROOF.

(i) By induction on M.

(ii) If $M =_\beta N$ then by the Church-Rosser property, $M \twoheadrightarrow_\beta L$ and $N \twoheadrightarrow_\beta L$, for some L. By subject reduction, $\Gamma \vdash^* L : \sigma$ and $\Gamma \vdash^* L : \tau$. Now use (i). □

It is easy to see that these properties fail in $\lambda\!\!\rightarrow$ à la Curry. For instance, $\vdash \lambda x.x : \alpha \to \alpha$ and $\vdash \lambda x.x : (\alpha \to \alpha) \to (\alpha \to \alpha)$ by the derivations:

$$\frac{x : \alpha \vdash x : \alpha}{\vdash \lambda x.x : \alpha \to \alpha}$$

and

$$\frac{x : \alpha{\to}\alpha \vdash x : \alpha{\to}\alpha}{\vdash \lambda x.x : (\alpha{\to}\alpha) \to (\alpha{\to}\alpha)}$$

Although these two derivations have the same structure, their conclusions are different due to different type assumptions for x. In contrast, if the Church term M has type σ in Γ, then there is exactly one derivation of this fact, which is uniquely encoded by M.

This difference leads to some interesting problems for the Curry system. Given a term M which types can be assigned to M, if any? Is there a single best type in some sense? Such problems are studied in *type inference,* which we return to later.

Because of the above difference, $\lambda\!\!\rightarrow$ à la Curry and other similar systems are often called *type assignment* systems, in contrast to $\lambda\!\!\rightarrow$ à la Church and similar systems which are called, e.g., *typed* systems.

3.3. Church versus Curry typing

Although the simply typed λ-calculus à la Curry and Church are different, one has the feeling that essentially the same thing is going on. To some extent this intuition is correct, as we now show.

Every pseudo-term induces a type-free λ-term by erasing the domains of abstractions.

3.3.1. DEFINITION. The *erasure* map $|\bullet| : \Lambda_\Pi \to \Lambda$ is defined as follows:

$$\begin{aligned} |x| &= x; \\ |M\ N| &= |M|\ |N|; \\ |\lambda x{:}\sigma.M| &= \lambda x.|M|. \end{aligned}$$

Erasure preserves reduction and typing:

3.3.2. Proposition (Erasing). *Let $M, N \in \Lambda_\Pi$.*

(i) *If $M \to_\beta N$ then $|M| \to_\beta |N|$;*

(ii) *If $\Gamma \vdash^* M : \sigma$ then $\Gamma \vdash |M| : \sigma$.*

Proof. (i) prove by induction on M that

$$|M[x := N]| = |M|[x := |N|] \qquad (*)$$

Then proceed by induction on the derivation of $M \to_\beta N$ using $(*)$.
(ii) by induction on the derivation of $\Gamma \vdash^* M : \sigma$. □

Conversely, one can "lift" every Curry derivation to a Church one.

3.3.3. Proposition (Lifting). *For all $M, N \in \Lambda$:*

(i) *If $M \to_\beta N$ then for each $M' \in \Lambda_\Pi$ with $|M'| = M$ there is $N' \in \Lambda_\Pi$ such that $|N'| = N$, and $M' \to_\beta N'$;*

(ii) *If $\Gamma \vdash M : \sigma$ then there is an $M' \in \Lambda_\Pi$ with $|M'| = M$ and $\Gamma \vdash^* M' : \sigma$.*

Proof. By induction on the derivation of $M \to_\beta N$ and $\Gamma \vdash M : \sigma$, respectively. □

3.3.4. Warning. The above two propositions allow one to derive certain properties of Curry-style typable lambda-terms from analogous properties of Church-style typed lambda-terms, or conversely. For instance, strong normalization for one variant of $(\lambda \to)$ easily implies strong normalization for the other (Exercise 3.6.4).

However, one has to be very cautious with such proof methods, sometimes they do not work. A common mistake (cf. Exercise 3.6.5) is the following attempt to derive the Church-Rosser property for Church-style $(\lambda \to)$ from the Church-Rosser property for untyped lambda-terms:

> Assume that $M_0 \twoheadrightarrow_\beta M_1$ and $M_0 \twoheadrightarrow_\beta M_2$. Then, by Proposition 3.3.2, we have $|M_0| \twoheadrightarrow_\beta |M_1|$ and $|M_0| \twoheadrightarrow_\beta |M_2|$. By Church-Rosser property for untyped lambda-terms, we have a term P with $|M_1| \twoheadrightarrow_\beta P$ and $|M_2| \twoheadrightarrow_\beta P$. In addition, by the subject reduction property, P is typable into the desired type. It remains to apply Proposition 3.3.3, to obtain a Church-style term M_3 with $|M_3| = P$, and such that $M_2 \twoheadrightarrow_\beta M_3$ and $M_1 \twoheadrightarrow_\beta M_3$.

For an explanation why the gap in this argument cannot be easily fixed, and how it *can* be fixed, see [74, pp. 269, 559].

In the remainder, when stating properties of simply typed λ-calculus it must always be understood that the result applies to both $\lambda\to$ à la Curry and à la Church, except when explicitly stated otherwise.

3.4. Normalization

In this section we are concerned with $\lambda\!\!\rightarrow$ à la Church.

A simple type can be regarded as a finite binary tree—this is where the alternative name "finite type" comes from—where all internal nodes are labeled by arrows and all leaves are labeled by type variables. We shall often refer to properties of types expressing them as properties of this tree representation. For instance, the function $h(\tau)$ defined below is just the height of the corresponding tree.

3.4.1. Definition. Define the function $h : \Pi \to \mathbb{N}$ by:

$$\begin{array}{lcl} h(\alpha) & = & 0 \\ h(\tau \to \sigma) & = & 1 + \max(h(\tau), h(\sigma)) \end{array}$$

It is often convenient to write Church style terms (typable pseudo-terms) in such a way that types of some or all subterms are displayed by superscripts, as in e.g., $(\lambda x{:}\tau.P^{\rho})^{\tau\to\rho}R^{\tau}$. Recall that a Church style term can be typed in only one way, provided the context of free variables is known. Thus our labelling is always determined by the term and the context. But the labelling itself is not a part of syntax, just a meta-notation.

The following property is the first non-trivial property of $\lambda\!\!\rightarrow$.

3.4.2. Theorem (Weak normalization). *Suppose* $\Gamma \vdash^{*} M : \sigma$. *Then there is a finite reduction* $M_1 \to_\beta M_2 \to_\beta \ldots \to_\beta M_n \in \mathrm{NF}_\beta$.

Proof. We use a proof idea due independently to Turing and Prawitz.

Define the *height* of a redex $(\lambda x{:}\tau.P^{\rho})R$ to be $h(\tau \to \rho)$. For $M \in \Lambda_\Pi$ with $M \notin \mathrm{NF}_\beta$ define

$$m(M) = (h(M), n)$$

where

$$h(M) = \max\{h(\Delta) \mid \Delta \text{ is a redex in } M\}$$

and n is the number of redex occurrences in M of height $h(M)$. If $M \in \mathrm{NF}_\beta$ we define $h(M) = (0, 0)$.

We show by induction on lexicographically ordered pairs $m(M)$ that if M is typable in $\lambda\!\!\rightarrow$ à la Church, then M has a reduction to normal-form.

Let $\Gamma \vdash M : \sigma$. If $M \in \mathrm{NF}_\beta$ the assertion is trivially true. If $M \notin \mathrm{NF}_\beta$, let Δ be the rightmost redex in M of maximal height h (we determine the position of a subterm by the position of its leftmost symbol, i.e., the rightmost redex means the redex which *begins* as much to the right as possible).

Let M' be obtained from M by reducing the redex Δ. The term M' may in general have more redexes than M. But we claim that the number of redexes of height h in M' is smaller than in M. Indeed, the redex Δ has disappeared, and the reduction of Δ may only create new redexes of height

less than h. To see this, note that the number of redexes can increase by either copying existing redexes or by creating new ones. Now observe that if a new redex is created then one of the following cases must hold:

1. The redex Δ is of the form $(\lambda x{:}\tau.\ \ldots x P^{\rho}\ \ldots\ )(\lambda y^{\rho}.Q^{\mu})^{\tau}$, where $\tau = \rho \to \mu$, and reduces to $\ \ldots\ (\lambda y^{\rho}.Q^{\mu})P^{\rho}\ \ldots$. There is a new redex $(\lambda y^{\rho}.Q^{\mu})P^{\rho}$ of height $h(\tau) < h$.

2. We have $\Delta = (\lambda x{:}\tau.\lambda y{:}\rho.R^{\mu})P^{\tau}$, occurring in the context $\Delta^{\rho\to\mu}Q^{\rho}$. The reduction of Δ to $\lambda y{:}\rho.R_1^{\mu}$, for some R_1, creates a new redex $(\lambda y{:}\rho.R_1^{\mu})Q^{\rho}$ of height $h(\rho \to \mu) < h(\tau \to \rho \to \mu) = h$.

3. The last case is when $\Delta = (\lambda x{:}\tau.x)(\lambda y^{\rho}.P^{\mu})$, with $\tau = \rho \to \mu$, and it occurs in the context $\Delta^{\tau}Q^{\rho}$. The reduction creates the new redex $(\lambda y^{\rho}.P^{\mu})Q^{\rho}$ of height $h(\tau) < h$.

The other possibility of adding redexes is by copying. If we have $\Delta = (\lambda x{:}\tau.P^{\rho})Q^{\tau}$, and P contains more than one free occurrence of x, then all redexes in Q are multiplied by the reduction. But we have chosen Δ to be the rightmost redex of height h, and thus all redexes in Q must be of smaller heights, because they are to the right of Δ.

Thus, in all cases $m(M) > m(M')$, so by the induction hypothesis M' has a normal-form, and then M also has a normal-form. □

In fact, an even stronger property than weak normalization holds: if $\vdash^{*} M : \sigma$, then no infinite reduction $M_1 \to_\beta M_2 \to_\beta \ldots$ exists. This property is called *strong normalization* and will be proved later.

The subject reduction property together with the Church-Rosser property and strong normalization imply that reduction of any typable λ-term terminates in a normal form of the same type, where the normal form is independent of the particular order of reduction chosen.

3.5. Expressibility

As we saw in the preceding section, every simply typable λ-term has a normal-form. In fact, one can effectively find this normal-form by repeated reduction of the *leftmost* redex. (These results hold for both the à la Curry and à la Church system.) Therefore one can easily figure out whether two simply typable terms are β-equal: just reduce the terms to their respective normal-forms and compare *them.*

These results should suggest that there will be difficulties in representing all the partial recursive functions and possibly also the total recursive functions by simply typable λ-terms, as we shall now see. In the rest of this section we are concerned with simply typed λ-calculus à la Curry.

3.5.1. Definition. Let

$$\mathbf{int} = (\alpha \to \alpha) \to (\alpha \to \alpha)$$

where α is an arbitrary type variable. A numeric function $f : \mathbb{N}^n \to \mathbb{N}$ is *$\lambda\!\to$-definable* if there is an $F \in \Lambda$ with $\vdash F : \mathbf{int} \to \cdots \to \mathbf{int} \to \mathbf{int}$ ($n+1$ occurrences of **int**) such that

$$F\, c_{n_1}\, \ldots\, c_{n_m} =_\beta c_{f(n_1,\ldots,n_m)}$$

for all $n_1, \ldots, n_m \in \mathbb{N}$.

It is natural to investigate which of the constructions from Chapter 1 carry over to the typed setting. This is carried out below.

3.5.2. Lemma. *The constant and projection functions are $\lambda\!\to$-definable.*

Proof. See the Exercises. □

3.5.3. Lemma. *The function $sg : \mathbb{N} \to \mathbb{N}$ defined by $sg(0) = 0, sg(m+1) = 1$ is $\lambda\!\to$-definable.*

Proof. See the Exercises. □

3.5.4. Lemma. *Addition and multiplication are $\lambda\!\to$-definable.*

Proof. See the Exercises. □

3.5.5. Definition. The class of *extended polynomials* is the smallest class of numeric functions containing the

(i) *projections:* $U_i^m(n_1, \ldots, n_m) = n_i$ for all $1 \le i \le m$;

(ii) *constant functions:* $k(n) = k$;

(iii) *signum function:* $sg(0) = 0$ and $sg(m+1) = 1$.

and closed under *addition* and *multiplication*:

(i) *addition:* if $f : \mathbb{N}^k \to \mathbb{N}$ and $g : \mathbb{N}^l \to \mathbb{N}$ are extended polynomials, then so is $(f + g) : \mathbb{N}^{k+l} \to \mathbb{N}$

$$(f+g)(n_1, \ldots, n_k, m_1, \ldots, m_l) = f(n_1, \ldots, n_k) + g(m_1, \ldots, m_l)$$

(ii) *multiplication:* if $f : \mathbb{N}^k \to \mathbb{N}$ and $g : \mathbb{N}^l \to \mathbb{N}$ are extended polynomials, then so is $(f \cdot g) : \mathbb{N}^{k+l} \to \mathbb{N}$

$$(f \cdot g)(n_1, \ldots, n_k, m_1, \ldots, m_l) = f(n_1, \ldots, n_k) \cdot g(m_1, \ldots, m_l)$$

3.5.6. Theorem (Schwichtenberg). *The $\lambda\!\to$-definable functions are exactly the extended polynomials.*

The proof is omitted. One direction follows easily from what has been said already; the other direction is proved in [97].

If one does not insist that numbers be uniformly represented as terms of type **int**, more functions become $\lambda\!\to$-definable—see [35].

3.6. Exercises

3.6.1. Exercise. Show that the following λ-terms have no type in $\lambda\!\rightarrow$ à la Curry.

1. $\lambda x.x\ x$;
2. Ω
3. $\mathbf{K}\,\mathbf{I}\,\Omega$;
4. $\mathbf{Y}$;
5. $c_2\ \mathbf{K}$.

3.6.2. Exercise. Find terms M and M' and types σ, σ' such that $\vdash M : \sigma$, $\vdash M' : \sigma'$, $M \twoheadrightarrow_\beta M'$, and not $\vdash M : \sigma'$.

3.6.3. Exercise. Is the following true? If $M \rightarrow_\beta N$ (where $M, N \in \Lambda$) and $M', N' \in \Lambda_\Pi$ are such that $|M'| = M$, $|N'| = N$ then $M' \rightarrow_\beta N'$.

3.6.4. Exercise. Show that strong normalization for $(\lambda \rightarrow)$ à la Curry implies strong normalization for $(\lambda \rightarrow)$ à la Church, and conversely.

3.6.5. Exercise. Find the bug in the example argument in Warning 3.3.4.

3.6.6. Exercise. Consider the proof of weak normalization. Assume that a given term M is of length n including type annotations. Give a (rough) upper bound (in terms of a function in n) for the length of the normalizing sequence of reductions for M, obtained under the strategy defined in that proof. Can your function be bounded by $\exp_k(n)$, for some k? Can this be done under the assumption that the height of redexes in M is bounded by a given constant h? (Here, $\exp_0(n) = n$ and $\exp_{k+1}(n) = 2^{exp_k(n)}$.)

3.6.7. Exercise. This exercise, and the next one, are based on [35]. Define the *rank* of a type τ, denoted $rk(\tau)$, as follows:

$$\begin{array}{lcl} rk(\alpha) & = & 0 \\ rk(\tau \rightarrow \sigma) & = & \max(h(\tau)+1, h(\sigma)) \end{array}$$

Alternatively, we have

$$rk(\tau_1 \rightarrow \cdots \rightarrow \tau_n \rightarrow \alpha) = 1 + \max(h(\tau_1), \ldots, h(\tau_n)).$$

The rank of a redex $(\lambda x{:}\tau.P^\rho)R$ is $rk(\tau \rightarrow \rho)$. Then define the *depth* of a term M, denoted $d(M)$, by the conditions

$$\begin{array}{lcl} d(x) & = & 0; \\ d(MN) & = & 1 + \max(d(M), d(N)); \\ d(\lambda x{:}\sigma.M) & = & 1 + d(M). \end{array}$$

Let r be the maximum rank of a redex occurring in M, and let $d(M) = d$. Show (by induction w.r.t M) that M can be reduced in at most $2^d - 1$ steps to a term M_1 such that the maximum rank of a redex occurring in M_1 is at most $r - 1$, and $d(M_1) \leq 2^d$.

3.6.8. Exercise. Let r be the maximum rank of a redex occurring in M, and let $d(M) = d$. Use the previous exercise to prove that the normal form of M is of depth at most $\exp_r(d)$, and can be obtained in at most $\exp_r(d)$ reduction steps.

3.6.9. Exercise. Show that the constant functions and projection functions are $\lambda\!\!\rightarrow$-definable.

3.6.10. Exercise. Show that sg is $\lambda\!\!\rightarrow$-definable.

3.6.11. Exercise. Show that addition and multiplication are $\lambda\!\!\rightarrow$-definable.

CHAPTER 4

The Curry-Howard isomorphism

Having met one formalism for expressing effective functions—λ-calculus—and another formalism for expressing proofs—natural deduction for intuitionistic logic—we shall now demonstrate an amazing analogy between the two formalisms, known as the *Curry-Howard isomorphism.*

We have already seen several hints that effective functions and proofs should be intimately related. For instance, as mentioned in Chapter 2, the BHK-interpretation [17, 53, 63] states that a proof of an implication $\varphi_1 \to \varphi_2$ is a "construction" which transforms any proof of φ_1 into a proof of φ_2. What is a construction? A possible answer is that it is some kind of effective function. There are several ways to make this answer precise. In this chapter we present one such way; another one is given by Kleene's realizability interpretation, which we present later.

4.1. Natural deduction without contexts

Recall that Chapter 2 presented a so-called *natural deduction* formulation of intuitionistic propositional logic. Such systems were originally introduced by Gentzen [39]. More precisely, Gentzen introduced two kinds of systems, nowadays called *natural deduction* systems and *sequent calculus* systems, respectively. In this chapter we are concerned with the former kind; sequent calculus systems will be introduced in the next chapter.

One of the most significant studies of natural deduction systems after Gentzen's work in the 1930s appears in Prawitz' classical book [85], which is still very readable.

There is an informal way of writing natural deduction proofs. Instead of maintaining explicitly in each node of a derivation the set of assumptions on which the conclusion depends (the *context*), one writes all the assumptions at the top of the derivation with a marker on those assumptions that have been discharged by the implication introduction rule.

Since this style is quite common in the proof theory literature—at least until the Curry-Howard isomorphism became widely appreciated—we also briefly review that notation informally here. Another reason for doing so, is that the notation displays certain interesting problems concerning assumptions that are hidden in our formulation of Chapter 2.

Consider the proof tree:

$$\dfrac{\dfrac{\varphi \qquad \dfrac{(\varphi \to \psi) \wedge (\varphi \to \rho)}{\varphi \to \psi}}{\psi} \qquad \dfrac{\varphi \qquad \dfrac{(\varphi \to \psi) \wedge (\varphi \to \rho)}{\varphi \to \rho}}{\rho}}{\psi \wedge \rho}$$

First note that, as always, the proof tree is written upside-down. The leaves are the assumptions, and the root is the conclusion, so the proof tree demonstrates that one can infer $\psi \wedge \rho$ from φ and $(\varphi \to \psi) \wedge (\varphi \to \rho)$.

As usual there is an $\to$-introduction rule which discharges assumptions. Thus we are able to infer $\varphi \to \psi \wedge \rho$ from $(\varphi \to \psi) \wedge (\varphi \to \rho)$. Notationally this is done by putting brackets around the assumption in question which is then called *closed,* as opposed to the other assumptions which are called *open:*

$$\dfrac{\dfrac{\dfrac{[\varphi] \qquad \dfrac{(\varphi \to \psi) \wedge (\varphi \to \rho)}{\varphi \to \psi}}{\psi} \qquad \dfrac{[\varphi] \qquad \dfrac{(\varphi \to \psi) \wedge (\varphi \to \rho)}{\varphi \to \rho}}{\rho}}{\psi \wedge \rho}}{\varphi \to \psi \wedge \rho}$$

Note that the above step discharges *both* occurrences of φ. In general, in an $\to$-introduction step, we may discharge zero, one, or more occurrences of an assumption.

Taking this one step further we get

$$\dfrac{\dfrac{\dfrac{[\varphi] \qquad \dfrac{[(\varphi \to \psi) \wedge (\varphi \to \rho)]}{\varphi \to \psi}}{\psi} \qquad \dfrac{[\varphi] \qquad \dfrac{[(\varphi \to \psi) \wedge (\varphi \to \rho)]}{\varphi \to \rho}}{\rho}}{\psi \wedge \rho}}{\dfrac{\varphi \to \psi \wedge \rho}{(\varphi \to \psi) \wedge (\varphi \to \rho) \to \varphi \to \psi \wedge \rho}}$$

Since we may decide to discharge only *some* of the occurrences of an open assumption in an $\to$-introduction step, one sometimes adopts for readability the convention of assigning numbers to assumptions, and one then indicates in an $\to$-introduction step which of the occurrences where discharged. In the above example we thus might have the following sequence of proof trees.

First:

$$\dfrac{\dfrac{\varphi^{(1)} \qquad \dfrac{(\varphi \to \psi) \wedge (\varphi \to \rho)^{(2)}}{\varphi \to \psi}}{\psi} \qquad \dfrac{\varphi^{(1)} \qquad \dfrac{(\varphi \to \psi) \wedge (\varphi \to \rho)^{(2)}}{\varphi \to \rho}}{\rho}}{\psi \wedge \rho}$$

Then by closing both occurrences of φ:

$$\dfrac{\dfrac{[\varphi]^{(1)} \qquad \dfrac{(\varphi \to \psi) \wedge (\varphi \to \rho)^{(2)}}{\varphi \to \psi}}{\psi} \qquad \dfrac{[\varphi]^{(1)} \qquad \dfrac{(\varphi \to \psi) \wedge (\varphi \to \rho)^{(2)}}{\varphi \to \rho}}{\rho}}{\dfrac{\psi \wedge \rho}{\varphi \to \psi \wedge \rho^{(1)}}}$$

And by closing both occurrences of $(\varphi \to \psi) \wedge (\varphi \to \rho)$:

$$\dfrac{\dfrac{[\varphi]^{(1)} \qquad \dfrac{[(\varphi \to \psi) \wedge (\varphi \to \rho)]^{(2)}}{\varphi \to \psi}}{\psi} \qquad \dfrac{[\varphi]^{(1)} \qquad \dfrac{[(\varphi \to \psi) \wedge (\varphi \to \rho)]^{(2)}}{\varphi \to \rho}}{\rho}}{\dfrac{\dfrac{\psi \wedge \rho}{\varphi \to \psi \wedge \rho^{(1)}}}{(\varphi \to \psi) \wedge (\varphi \to \rho) \to \varphi \to \psi \wedge \rho^{(2)}}}$$

It is interesting to note that the notation where we indicate which assumption is discharged allows us to distinguish between certain very similar proofs. For instance, in

$$\dfrac{\dfrac{[\varphi]^{(1)} \qquad \dfrac{[(\varphi \to \psi) \wedge (\varphi \to \rho)]^{(2)}}{\varphi \to \psi}}{\psi} \qquad \dfrac{[\varphi]^{(1)} \qquad \dfrac{[(\varphi \to \psi) \wedge (\varphi \to \rho)]^{(3)}}{\varphi \to \rho}}{\rho}}{\dfrac{\dfrac{\dfrac{\psi \wedge \rho}{\varphi \to \psi \wedge \rho^{(1)}}}{(\varphi \to \psi) \wedge (\varphi \to \rho) \to \varphi \to \psi \wedge \rho^{(2)}}}{(\varphi \to \psi) \wedge (\varphi \to \rho) \to (\varphi \to \psi) \wedge (\varphi \to \rho) \to \varphi \to \psi \wedge \rho^{(3)}}}$$

and

$$\dfrac{\dfrac{[\varphi]^{(1)} \qquad \dfrac{[(\varphi \to \psi) \wedge (\varphi \to \rho)]^{(2)}}{\varphi \to \psi}}{\psi} \qquad \dfrac{[\varphi]^{(1)} \qquad \dfrac{[(\varphi \to \psi) \wedge (\varphi \to \rho)]^{(3)}}{\varphi \to \rho}}{\rho}}{\dfrac{\dfrac{\dfrac{\psi \wedge \rho}{\varphi \to \psi \wedge \rho^{(1)}}}{(\varphi \to \psi) \wedge (\varphi \to \rho) \to \varphi \to \psi \wedge \rho^{(3)}}}{(\varphi \to \psi) \wedge (\varphi \to \rho) \to (\varphi \to \psi) \wedge (\varphi \to \rho) \to \varphi \to \psi \wedge \rho^{(2)}}}$$

we discharge the two occurrences of $(\varphi \to \psi) \wedge (\varphi \to \rho)$ separately, but in different orders.

Similarly,

$$\dfrac{\dfrac{[\varphi]^{(1)}}{\varphi \to \varphi^{(1)}}}{\varphi \to \varphi \to \varphi^{(2)}}$$

and

$$\dfrac{\dfrac{[\varphi]^{(1)}}{\varphi \to \varphi^{(2)}}}{\varphi \to \varphi \to \varphi^{(1)}}$$

are two different proofs of $\varphi \to \varphi \to \varphi$. In the first proof there is first a discharge step in which the single occurrence of φ is discharged, and then a discharge step in which zero occurrences of φ are discharged. In the second proof the order is reversed.

In order to avoid confusion with assumption numbers, we require that if two assumptions φ and ψ have the same number, then φ and ψ are the same formula. Also, when we discharge the assumptions with a given number (i), we require that every assumption with this number actually occur on a branch from the node where the discharging occurs.

In general, the rules for constructing the above proof trees look as follows.

$$\dfrac{\varphi \quad \psi}{\varphi \wedge \psi} \qquad \dfrac{\varphi \wedge \psi}{\varphi} \qquad \dfrac{\varphi \wedge \psi}{\psi}$$

$$\dfrac{\varphi}{\varphi \vee \psi} \qquad \dfrac{\psi}{\varphi \vee \psi} \qquad \dfrac{\varphi \vee \psi \qquad \begin{array}{c}[\varphi]^{(i)} \\ \vdots \\ \rho\end{array} \qquad \begin{array}{c}[\psi]^{(j)} \\ \vdots \\ \rho\end{array}}{\rho^{(i,j)}}$$

$$\dfrac{\begin{array}{c}[\varphi]^{(i)} \\ \vdots \\ \psi\end{array}}{\varphi \to \psi^{(i)}} \qquad \dfrac{\varphi \to \psi \quad \varphi}{\psi}$$

$$\dfrac{\bot}{\varphi}$$

For instance, the upper left rule ($\wedge$-introduction) states that two proof trees ending in roots φ and ψ, respectively, may be joined into a single proof tree by addition of a new root $\varphi \wedge \psi$ with the former roots as children. The $\to$-introduction rule states that one may infer an implication by discharging the assumptions with the label indicated by the step.

As we shall see later in this chapter, there is an interest in proofs of a certain simple form. One arrives at such proofs from arbitrary proofs by means of *proof normalization* rules that eliminate detours in a proof. More concretely, consider the proof tree:

$$\dfrac{\dfrac{\dfrac{[\varphi]^{(1)}}{\varphi \to \varphi^{(1)}} \qquad \dfrac{[\psi]^{(2)}}{\psi \to \psi^{(2)}}}{(\varphi \to \varphi) \wedge (\psi \to \psi)}}{\varphi \to \varphi}$$

The proof tree demonstrates that $\varphi \to \varphi$ is derivable. However, it does so by first showing $\varphi \to \varphi$, then inferring $(\varphi \to \varphi) \wedge (\psi \to \psi)$, and then, finally, concluding $\varphi \to \varphi$. A more direct proof tree, which does not make an excursion via $(\varphi \to \varphi) \wedge (\psi \to \psi)$ is:

$$\dfrac{[\varphi]^{(1)}}{\varphi \to \varphi^{(1)}}$$

Note that the detour in the former proof tree is signified by an introduction rule immediately followed by the corresponding elimination rule, for example $\wedge$-introduction and $\wedge$-elimination. In fact, the above style of detour-elimination is possible whenever an introduction rule is immediately followed by the corresponding elimination rule.

As another example, consider the proof tree:

$$\dfrac{\dfrac{[\varphi]^{(3)}}{\varphi \to \varphi^{(3)}} \qquad \dfrac{\dfrac{[\varphi \to \varphi]^{(1)}}{\psi \to \varphi \to \varphi^{(2)}}}{(\varphi \to \varphi) \to \psi \to \varphi \to \varphi^{(1)}}}{\psi \to \varphi \to \varphi}$$

Here we infer $\psi \to \varphi \to \varphi$ from $\varphi \to \varphi$ and $(\varphi \to \varphi) \to \psi \to \varphi \to \varphi$. The proof of the latter formula proceeds by inferring $\psi \to \varphi \to \varphi$ from the assumption $\varphi \to \varphi$. Since we can *prove* this assumption, we could simply take this proof and replace the assumption $\varphi \to \varphi$ with the proof of this formula:

$$\dfrac{\dfrac{[\varphi]^{(3)}}{\varphi \to \varphi^{(3)}}}{\psi \to \varphi \to \varphi^{(2)}}$$

In general one considers the following proof normalization rules (sym-

metric cases omitted):

$$\dfrac{\dfrac{\dfrac{\Sigma}{\varphi}\quad\dfrac{\Pi}{\psi}}{\varphi\wedge\psi}}{\varphi}\quad\to\quad\dfrac{\Sigma}{\varphi}$$

$$\dfrac{\dfrac{\Sigma}{\psi}\quad\dfrac{\dfrac{\dfrac{[\psi]^{(i)}}{\Pi}}{\varphi}}{\psi\to\varphi^{(i)}}}{\varphi}\quad\to\quad\dfrac{\dfrac{\dfrac{\Sigma}{\psi}}{\Pi}}{\varphi}$$

$$\dfrac{\dfrac{\dfrac{\Theta}{\varphi}}{\varphi\vee\psi}\quad\dfrac{\dfrac{[\varphi]^{(i)}}{\Sigma}}{\rho}\quad\dfrac{\dfrac{[\psi]^{(j)}}{\Pi}}{\rho}}{\rho^{(i,j)}}\quad\to\quad\dfrac{\dfrac{\dfrac{\Theta}{\varphi}}{\Sigma}}{\rho}$$

The first rule states that if we, somewhere in a proof, infer φ and ψ and then use $\wedge$-introduction to infer $\varphi\wedge\psi$ followed by $\wedge$-elimination to infer φ, we might as well avoid the detour and replace this proof simply by the subproof of φ.

The second rule states that if we have a proof of φ from assumption ψ and we use this and $\to$-introduction to get a proof of $\psi\to\varphi$, and we have a proof of ψ then, instead of inferring φ by $\to$-elimination, we might as well replace this proof by the original proof of φ where we plug in the proof of ψ in all the places where the assumption ψ occurs.

The reading of the third rule is similar.

The process of eliminating proof detours of the above kind, is called *proof normalization,* and a proof tree with no detours is said to be in *normal form.* Another similar process, called *cut elimination,* eliminates detours in *sequent calculus proofs* whereas proof normalization eliminates detours in natural deduction proofs. Sequent calculus systems are introduced in the next chapter.

Proof normalization and cut elimination were studied in the 1930s by Gentzen, and his studies were continued by several researchers, perhaps most importantly by Prawitz in [85]. Nowadays, proof theory is an independent discipline of logic.

In these notes we shall not consider natural deduction proofs in the above style any further.

4.2. The Curry-Howard isomorphism

We could introduce reductions à la those of the preceding section for the natural deduction formulation of Chapter 2, but we shall not do so. The rules for that formulation are rather tedious (try the rule for $\rightarrow$!). It would be more convenient to have each proof tree denoted by some 1-dimensional expression and then state transformations on such expressions rather than on proof trees. It happens that the terms of the simply typed λ-calculus are ideal for this purpose, as we shall see in this section.

We show that any derivation in intuitionistic propositional logic corresponds to a typable λ-term à la Church, and vice versa. More precisely we show this for the *implicational fragment* of intuitionistic propositional logic.

Recall from Section 2.6 that the implicational fragment is the subsystem in which the only connective is $\rightarrow$ and in which the only rules are $(\rightarrow E)$ and $(\rightarrow I)$. This fragment is denoted IPC$(\rightarrow)$. The whole system is denoted IPC$(\rightarrow, \wedge, \vee, \bot)$ or plainly IPC.

If we take PV (the set of propositional variables) equal to U (the set of type variables), then Φ (the set of propositional formulas in the implicational fragment of intuitionistic propositional logic) and Π (the set of simply types) are identical. This will be used implicitly below.

4.2.1. Proposition (Curry-Howard isomorphism).

(i) *If* $\Gamma \vdash M : \varphi$ *then* $|\Gamma| \vdash \varphi$.[1]

(ii) *If* $\Gamma \vdash \varphi$ *then there exists* $M \in \Lambda_\Pi$ *such that* $\Delta \vdash M : \varphi$, *where* $\Delta = \{(x_\varphi : \varphi) \mid \varphi \in \Gamma\}$.

Proof. (i): by induction on the derivation of $\Gamma \vdash M : \varphi$.

(ii): by induction on the derivation of $\Gamma \vdash \varphi$. Let $\Delta = \{x_\varphi : \varphi \mid \varphi \in \Gamma\}$.

1. The derivation is
$$\Gamma, \varphi \vdash \varphi$$
We consider two subcases:
   (a) $\varphi \in \Gamma$. Then $\Delta \vdash x_\varphi : \varphi$.
   (b) $\varphi \notin \Gamma$. Then $\Delta, x_\varphi : \varphi \vdash x_\varphi : \varphi$.

2. The derivation ends in
$$\frac{\Gamma \vdash \varphi \rightarrow \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$$
By the induction hypothesis $\Delta \vdash M : \varphi \rightarrow \psi$ and $\Delta \vdash N : \varphi$, and then also $\Delta \vdash M\ N : \psi$.

[1] Recall that $|\Gamma|$ denotes the range of Γ.

3. The derivation ends in

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}$$

We consider two subcases:

(a) $\varphi \in \Gamma$. Then by the induction hypothesis $\Delta \vdash M : \psi$. By Weakening (Lemma 3.19(i)) $\Delta, x : \varphi \vdash M : \psi$, where $x \notin \mathrm{dom}(\Delta)$. Then also $\Delta \vdash \lambda x{:}\varphi \,.\, M : \varphi \to \psi$.

(b) $\varphi \notin \Gamma$. Then by the induction hypothesis $\Delta, x_\varphi : \varphi \vdash M : \psi$ and then also $\Delta \vdash \lambda x_\varphi{:}\varphi \,.\, M : \varphi \to \psi$. □

4.2.2. Remark. The correspondence displays certain interesting problems with the natural deduction formulation of Chapter 2. For instance

$$\cfrac{\cfrac{x : \varphi, y : \varphi \vdash x : \varphi}{x : \varphi \vdash \lambda y{:}\varphi \,.\, x : \varphi \to \varphi}}{\vdash \lambda x{:}\varphi \,.\, \lambda y{:}\varphi \,.\, x : \varphi \to \varphi \to \varphi}$$

and

$$\cfrac{\cfrac{x : \varphi, y : \varphi \vdash y : \varphi}{x : \varphi \vdash \lambda y{:}\varphi \,.\, y : \varphi \to \varphi}}{\vdash \lambda x{:}\varphi \,.\, \lambda y{:}\varphi \,.\, y : \varphi \to \varphi \to \varphi}$$

are two different derivations in $\lambda{\to}$ showing that both $\lambda x{:}\varphi \,.\, \lambda y{:}\varphi \,.\, x$ and $\lambda x{:}\varphi \,.\, \lambda y{:}\varphi \,.\, y$ have type $\varphi \to \varphi \to \varphi$.

Both of these derivations are projected to

$$\cfrac{\cfrac{\varphi \vdash \varphi}{\varphi \vdash \varphi \to \varphi}}{\vdash \varphi \to \varphi \to \varphi}$$

This reflects the fact that, in the natural deduction system of Chapter 2, one cannot distinguish proofs in which assumptions are discharged in different orders. Indeed, $\lambda{\to}$ can be viewed as an extension of IPC($\to$) in which certain aspects such as this distinction are elaborated.

The correspondence between derivations in IPC($\to$) and $\lambda{\to}$ can be extended to the whole system IPC by extending the simply typed λ-calculus with pairs and disjoint sums. One extends the language Λ_Π with clauses:

$$\begin{aligned} \Lambda_\Pi \quad ::= \quad \ldots \quad & \mid <\Lambda_\Pi, \Lambda_\Pi> \mid \pi_1(\Lambda_\Pi) \mid \pi_2(\Lambda_\Pi) \\ & \mid \mathrm{in}_1^{\psi\vee\varphi}(\Lambda_\Pi) \mid \mathrm{in}_2^{\psi\vee\varphi}(\Lambda_\Pi) \mid \mathrm{case}(\Lambda_\Pi; V.\Lambda_\Pi; V.\Lambda_\Pi) \end{aligned}$$

and adds typing rules:

$$\frac{\Gamma \vdash M : \psi \quad \Gamma \vdash N : \varphi}{\Gamma \vdash < M, N > : \psi \wedge \varphi} \qquad \frac{\Gamma \vdash M : \psi \wedge \varphi}{\Gamma \vdash \pi_1(M) : \psi} \qquad \frac{\Gamma \vdash M : \psi \wedge \varphi}{\Gamma \vdash \pi_2(M) : \varphi}$$

$$\frac{\Gamma \vdash M : \psi}{\Gamma \vdash \mathrm{in}_1^{\psi \vee \varphi}(M) : \psi \vee \varphi} \qquad \frac{\Gamma \vdash M : \varphi}{\Gamma \vdash \mathrm{in}_2^{\psi \vee \varphi}(M) : \psi \vee \varphi}$$

$$\frac{\Gamma \vdash L : \psi \vee \varphi \quad \Gamma, x : \psi \vdash M : \rho \quad \Gamma, y : \varphi \vdash N : \rho}{\Gamma \vdash \mathrm{case}(L; x.M; y.N) : \rho}$$

and reduction rules:

$$\begin{array}{lcl} \pi_1(< M_1, M_2 >) & \to & M_1 \\ \pi_2(< M_1, M_2 >) & \to & M_2 \end{array}$$

$$\begin{array}{lcl} \mathrm{case}(\mathrm{in}_1^{\varphi}(N); x.K; y.L) & \to & K\{x := N\} \\ \mathrm{case}(\mathrm{in}_2^{\varphi}(N); x.K; y.L) & \to & L\{y := N\} \end{array}$$

Intuitively, $\phi \wedge \psi$ is a product type, so $< M_1, M_2 >$ is a pair, and $\pi_1(M)$ is the first projection. In type-free λ-calculus these could be defined in terms of pure λ-terms (see Proposition 1.46), but this is not possible in $\lambda{\to}$. This is related to the fact that one cannot define conjunction in IPC in terms of implication (contrary to the situation in classical logic, as we shall see later).

In the same spirit, $\phi \vee \psi$ is a sum (or "variant") type. A sum type is a data type with two unary constructors. Compare this to the data type "integer list", which is usually defined as a data type with two constructors: the 0-ary constructor *Nil* and the 2-ary constructor $Cons(\bullet, \bullet)$ which takes a number and a list of numbers. In a sum we have the two unary constructors left and right injection.

Thus $\mathrm{case}(M; x.K; y.L)$ is a case-expression which tests whether M has form $\mathrm{in}_1^{\varphi}(N)$ (and then returns K with N for x) or $\mathrm{in}_2(N)$ (and then returns L with N for y), just like in a functional programming language we could have a case-expression testing whether an expression is *Nil* or $Cons(n, ns)$.

Thus, uses of the axiom of intuitionistic propositional logic are reflected by variables in the term, uses of the $\to$-elimination rule correspond to applications, and uses of the $\to$-introduction rule correspond to abstractions.

In fact, we can view $\lambda{\to}$ as a more elaborate formulation of IPC($\to$) in which the terms "record" the rules it was necessary to apply to prove the type of the term, when we view that type as a proposition. For instance, $\lambda x : \varphi \,.\, x$ has type $\varphi \to \varphi$, signifying the fact that we can prove $\varphi \to \varphi$ by first using the axiom recorded by the variable x and then using $\to$-introduction, recorded by $\lambda x : \varphi$. In short, in $\lambda{\to}$ viewed as a logic, the terms serve as a linear representation of proof trees, and are usually

called *constructions* [58]. These are also constructions in the sense of the BHK-interpretation: a construction of $\varphi \to \psi$ is a λ-term $\lambda x{:}\varphi \,.\, M$ of type $\varphi \to \psi$.

Two different propositions cannot have the same construction, since we work with Church terms. In contrast, several constructions may correspond to the same proposition. This is because the same proposition may be proven in different ways.

Thus $\lambda{\to}$ and IPC($\to$) may be viewed as different names for essentially the same thing. This means that each of the concepts and properties considered in $\lambda{\to}$ makes sense in IPC($\to$) and vice versa.

As mentioned, terms in $\lambda{\to}$ correspond to constructions in IPC($\to$). Types correspond to formulas, type constructors (sum and pair) to connectives. Asking whether there exists a term of a given type (*inhabitation*), corresponds to asking whether there exist a construction for a given proposition (*provability.*) Asking whether there exists a type for a given term (*typability*), corresponds to asking whether the construction is a construction of some formula.

What is a redex in a construction? Well, each introduction rule introduces a *constructor* (a lambda, a pair, or an injection) in the construction, and each elimination rule introduces a *destructor* (an application, a projection, or a case-expression). Now, a redex consists of a constructor immediately surrounded by the corresponding destructor. Therefore a redex in the construction represents a proof tree containing an application of an introduction rule immediately followed by an application of the corresponding elimination rule; this was what we called a detour in a proof tree. Therefore, reduction on terms corresponds to normalization of constructions. A term in normal form corresponds to a construction representing a proof tree in normal form. The subject reduction proposition states that reducing a construction of a formula yields a construction for the same formula. The Church-Rosser Theorem states that the order of normalization is immaterial. Also, it states that we managed to identify essentially identical proofs without identifying all proofs.

In summary:

$\lambda\!\!\rightarrow$	IPC($\rightarrow$)
term variable	assumption
term	construction (proof)
type variable	propositional variable
type	formula
type constructor	connective
inhabitation	provability
typable term	construction for a proposition
redex	construction representing proof tree with redundancy
reduction	normalization
value	normal construction

4.2.3. Example. Consider the following example deduction containing redundancy. The original derivation with constructions is:

$$\frac{x : \varphi \vdash x : \varphi}{\vdash \lambda x{:}\varphi \,.\, x : \varphi \to \varphi}$$

The complicated proof with constructions is:

$$\dfrac{\dfrac{\dfrac{y : \psi \vdash y : \psi}{\vdash \lambda y{:}\psi \,.\, y : \psi \to \psi} \quad \dfrac{x : \varphi \vdash x : \varphi}{\vdash \lambda x{:}\varphi \,.\, x : \varphi \to \varphi}}{\vdash < \lambda x{:}\varphi \,.\, x, \lambda y{:}\psi \,.\, y >: (\varphi \to \varphi) \wedge (\psi \to \psi)}}{\vdash \pi_1(< \lambda x{:}\varphi \,.\, x, \lambda y{:}\psi \,.\, y >) : \varphi \to \varphi}$$

The construction of the latter proof tree in fact contains a redex which upon reduction yields the construction of the former proof tree.

The perfect correspondence between reduction and normalization and the related concepts, justifies the name "isomorphism" rather than simply "bijection." In fact, reduction has been studied extensively in the λ-calculus literature, while normalization has been studied independently in proof theory.

Since the "discovery" of the isomorphism, the two worlds have merged, and some authors feel that it is exactly in the correspondence between reduction and normalization that the isomorphism is deepest and most fruitful. This point of view is supported by the fact that some typed λ-calculi have been introduced as means of studying normalization for logics, most notably Girard's System **F** introduced in his work [44] from 1971. System **F** corresponds to second order minimal propositional logic and will be discussed in Chapter 12.

As an appealing illustration of the isomorphism and an appropriate conclusion of this section, this system was independently invented at roughly the same time in computer science by Reynolds [90] in his study of polymorphism in typed functional programming languages.

In the remainder of these notes the concepts corresponding to one another under the isomorphism are used interchangeably. In particular, any system as that of the preceding subsection will be called both a logic and a λ-calculus depending on the aspects being emphasized.

4.3. Consistency from normalization

A number of properties regarding unprovability can be difficult to establish directly, but more easy to establish by semantical methods as we saw in Chapter 2. Often these semantical methods can be replaced by methods involving the weak normalization property.

The following shows that IPC($\to$) is consistent.

4.3.1. Proposition. $\not\vdash \bot$.

Proof. Assume that $\vdash \bot$. Then $\vdash M : \bot$ for some $M \in \Lambda_\Pi$. By the weak normalization theorem and the subject reduction theorem there is then an $N \in \mathrm{NF}_\beta$ such that $\vdash N : \bot$.

Now, λ-terms in normal form have form $x\ N_1 \dots N_m$ (where $N_1, \dots, N_n$ are normal-forms) and $\lambda x{:}\sigma\ .\ N'$ (where N' is in normal form). We cannot have N of the first form (then $x \in \mathrm{FV}(N)$, but since $\vdash N : \bot$, $\mathrm{FV}(N) = \{\}$). We also cannot have N of the second form (then $\bot = \sigma \to \tau$ for some σ, τ which is patently false). □

4.4. Strong normalization

As suggested by the application in the preceding section, the weak normalization property of $\lambda\!\to$ is a very useful tool in proof theory. In this section we prove the *strong normalization* property of $\lambda\!\to$ which is sometimes even more useful.

The standard method of proving strong normalization of typed λ-calculi was invented by Tait [104] for simply typed λ-calculus, generalized to second-order typed λ-calculus by Girard [44], and subsequently simplified by Tait [105].

Our presentation follows [8]; we consider in this section terms à la Curry.

4.4.1. Definition.

(i) $\mathrm{SN}_\beta = \{M \in \Lambda \mid M \text{ is strongly normalizing }\}$.

(ii) For $A, B \subseteq \Lambda$, define $A \to B = \{F \in \Lambda \mid \forall a \in A : F\ a \in B\}$.

(iii) For every simple type σ, define $[\![\sigma]\!] \subseteq \Lambda$ by:

$$\begin{array}{lcl} [\![\alpha]\!] & = & \mathrm{SN}_\beta \\ [\![\sigma \to \tau]\!] & = & [\![\sigma]\!] \to [\![\tau]\!] \end{array}$$

4.4.2. Definition.

(i) A set $X \subseteq \mathrm{SN}_\beta$ is *saturated* if

1. For all $n \geq 0$ and $M_1, \ldots M_n \in \mathrm{SN}_\beta$:

$$x\ M_1 \ldots M_n \in X$$

2. For all $n \geq 1$ and $M_1, \ldots, M_n \in \mathrm{SN}_\beta$:

$$M_0\{x := M_1\}\ M_2 \ldots M_n \in X \Rightarrow (\lambda x.M_0)\ M_1\ M_2 \ldots M_n \in X$$

(ii) $\mathbb{S} = \{X \subseteq \Lambda \mid X \text{ is saturated }\}$.

4.4.3. Lemma.

(i) $\mathrm{SN}_\beta \in \mathbb{S}$;

(ii) $A, B \in \mathbb{S} \Rightarrow A \to B \in \mathbb{S}$;

(iii) $\sigma \in \Pi \Rightarrow [\![\sigma]\!] \in \mathbb{S}$.

Proof. Exercise 4.6.3. □

4.4.4. Definition.

(i) a *valuation* is a map $\rho : V \to \Lambda$, where V is the set of term variables. The valuation $\rho(x := N)$ is defined by

$$\rho(x := N)(y) = \begin{cases} N & \text{if } x \equiv y \\ \rho(y) & \text{otherwise} \end{cases}$$

(ii) Let ρ be a valuation. Then $[\![M]\!]_\rho = M\{x_1 := \rho(x_1), \ldots, x_n := \rho(x_n)\}$, where $\mathrm{FV}(M) = \{x_1, \ldots, x_n\}$.

(iii) Let ρ be a valuation. Then $\rho \models M : \sigma$ iff $[\![M]\!]_\rho \in [\![\sigma]\!]$. Also, $\rho \models \Gamma$ iff $\rho(x) \in [\![\sigma]\!]$ for all $x : \sigma \in \Gamma$.

(iv) $\Gamma \models M : \sigma$ iff $\forall \rho : \rho \models \Gamma \Rightarrow \rho \models M : \sigma$.

4.4.5. Proposition (Soundness). $\Gamma \vdash M : \sigma \Rightarrow \Gamma \models M : \sigma$.

Proof. By induction on the derivation of $\Gamma \vdash M : \sigma$.

1. The derivation is

$$\Gamma \vdash x : \sigma \quad x : \sigma \in \Gamma$$

If $\rho \models \Gamma$, then $[\![x]\!]_\rho = \rho(x) \in [\![\sigma]\!]$.

2. The derivation ends in

$$\frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash M\ N : \tau}$$

Suppose $\rho \models \Gamma$. By the induction hypothesis $\Gamma \models M : \sigma \to \tau$ and $\Gamma \models N : \sigma$, so $\rho \models M : \sigma \to \tau$ and $\rho \models N : \sigma$, i.e., $[\![M]\!]_\rho \in [\![\sigma]\!] \to [\![\tau]\!]$ and $[\![N]\!]_\rho \in [\![\sigma]\!]$. Then $[\![M\ N]\!]_\rho = [\![M]\!]_\rho\ [\![N]\!]_\rho \in [\![\tau]\!]$, as required.

3. The derivation ends in

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}$$

Suppose $\rho \models \Gamma$. Also, suppose $N \in [\![\sigma]\!]$. Then $\rho(x := N) \models \Gamma, x : \sigma$. By the induction hypothesis $\Gamma, x : \sigma \models M : \tau$, so $\rho(x := N) \models M : \tau$, i.e., $[\![M]\!]_{\rho(x:=N)} \in [\![\tau]\!]$. Now,

$$\begin{array}{rcl} [\![\lambda x.M]\!]_\rho\ N & \equiv & (\lambda x.M)\ \{y_1 := \rho(y_1), \ldots, y_n := \rho(y_n)\}N \\ & \to_\beta & M\{y_1 := \rho(y_1), \ldots, y_n := \rho(y_n), x := N\} \\ & \equiv & [\![M]\!]_{\rho(x:=N)} \end{array}$$

Since $N \in [\![\sigma]\!] \subseteq \mathrm{SN}_\beta$ and $[\![M]\!]_{\rho(x:=N)} \in [\![\tau]\!] \in \mathbb{S}$, it follows that $[\![\lambda x.M]\!]_\rho\ N \in [\![\tau]\!]$. Hence $[\![\lambda x.M]\!]_\rho \in [\![\sigma \to \tau]\!]$. □

4.4.6. Theorem. $\Gamma \vdash M : \sigma \Rightarrow M \in \mathrm{SN}_\beta$.

Proof. If $\Gamma \vdash M : \sigma$, then $\Gamma \models M : \sigma$. For each $x : \tau \in \Gamma$, let $\rho(x) = x$. Then $x \in [\![\tau]\!]$ holds since $[\![\tau]\!] \in \mathbb{S}$. Then $\rho \models \Gamma$, and we have $M = [\![M]\!]_\rho \in [\![\sigma]\!] \subseteq \mathrm{SN}_\beta$. □

The reader may think that the above proof is more complicated than the weak normalization proof of the preceding chapter; in fact, this feeling can be made into a technical property by noting that the latter proof involves quantifying over sets, whereas the former does not.

The fact that the strong normalization property seems more difficult to prove has led to some techniques that aim at inferring strong normalization from weak normalization—see [102].

There are many applications of strong normalization, but many of these applications can be obtained already by using the weak normalization theorem. The following is a true application of strong normalization where weak normalization does not suffice.

4.4.7. Definition. Let $\to$ be a binary relation on some set L, and write $M \twoheadrightarrow M'$ if $M = M_1 \to \ldots \to M_n = M'$, where $n \geq 1$. Then

1. $\to$ satisfies CR iff for all $M_1, M_2, M_3 \in L$, $M_1 \twoheadrightarrow M_2$ and $M_1 \twoheadrightarrow M_3$ implies that there is an $M_4 \in L$ such that $M_2 \twoheadrightarrow M_4$ and $M_3 \twoheadrightarrow M_4$.

2. $\to$ satisfies WCR iff for all $M_1, M_2, M_3 \in L$, $M_1 \to M_2$ and $M_1 \to M_3$ implies that there is an $M_4 \in L$ such that $M_2 \twoheadrightarrow M_4$ and $M_3 \twoheadrightarrow M_4$.

3. $\to$ satisfies SN iff for all $M \in L$, there is no infinite reduction sequence $M \to M' \to \ldots$.

4. $\to$ satisfies WN iff for all $M \in L$, there is a finite reduction sequence $M \to M' \to \ldots \to M''$ such that M'' is a normal form (i.e., for all $N \in L$: $M'' \not\to N$).

4.4.8. Proposition (Newman's lemma). *Let $\to$ be a binary relation satisfying* SN. *If $\to$ satisfies* WCR, *then $\to$ satisfies* CR.

Proof. See the exercises. □

The following shows that the assumption about strong normalization cannot be replaced by weak normalization.

4.4.9. Proposition. *There is a binary relation $\to$ satisfying* WN *and* WCR, *but not* CR.

Proof. See the exercises. □

4.4.10. Corollary. *Let $M_1 \in \Lambda$ be typable in $\lambda{\to}$ à la Church and assume that $M_1 \twoheadrightarrow_\beta M_2$ and $M_1 \twoheadrightarrow_\beta M_3$. Then there is an M_4 such that $M_2 \twoheadrightarrow_\beta M_4$ and $M_3 \twoheadrightarrow_\beta M_4$.*

Proof. See the exercises. □

4.5. Historical remarks

The informal notion of a "construction" mentioned in the BHK-interpretation was first formalized in Kleene's *recursive realizability* interpretation [60, 61] in which proofs in *intuitionistic number theory* are interpreted as numbers, as we will see later in the notes. A proof of $\varphi_1 \to \varphi_2$ is interpreted as the *Gödel number* of a partial recursive function mapping the interpretation of any proof of φ_1 to the interpretation of a proof of φ_2.

One can see the Curry-Howard isomorphism—the correspondence between systems of formal logic and functional calculi with types, mentioned above—as a syntactic reflection of this interpretation. It shows that a certain notation system for denoting certain recursive functions coincides with a system for expressing proofs.

Curry [24] discovered that the provable formulas in a so-called *Hilbert formulation* of IPC($\to$) coincide with the inhabited types of *combinatory logic,* when one identifies function type with implication. Moreover, every proof in the logic corresponds to a term in the functional calculus, and

vice versa. Curry also noted a similar correspondence between a natural deduction formulation of IPC($\rightarrow$) and simply typed λ-calculus, and between a *sequent calculus* formulation of IPC($\rightarrow$) and a sequent calculus version of simply typed λ-calculus.

Gentzen's *Hauptsatz* [39] shows how one can transform a sequent calculus proof into another proof with no applications of the *cut rule*. Curry now proved a corresponding result for the sequent calculus version of simply typed λ-calculus. He then formulated correspondences between sequent calculus systems, natural deduction systems, and Hilbert systems (in terms of the corresponding functional calculi) and used these to infer weak normalization for β-reduction in simply typed λ-calculus and for so-called *strong reduction* in combinatory logic.

A more direct relation between reduction on terms and normalization of proofs was given by Howard in a paper from 1968, published as [58]. Prawitz had studied reduction of natural deduction proofs extensively [85]—seven years after Curry's book—and had proved weak normalization of this notion of reduction. Howard now showed that reduction of a proof in the natural deduction system for minimal implicational logic corresponds to β-reduction on the corresponding term in the simply typed λ-calculus. He also extended this correspondence to first order intuitionistic arithmetic and a related typed λ-calculus.

Howard's correspondence and the weak normalization theorem give a syntactic version of Kleene's interpretation, where one replaces recursive functions by λ-terms in normal form. For instance, any proof of $\varphi_1 \rightarrow \varphi_2$ reduces to a λ-abstraction which, when applied to a proof of φ_1, yields a proof of φ_2.

Constable [19, 20] suggested that a type or proposition φ be viewed as a specification, and any proof M of φ as a program satisfying the specification. For instance, sorting can be specified by the formula

$$\forall x\, \exists y : \mathsf{ordered}(y) \wedge \mathsf{permutation}(x, y)$$

in *predicate logic,* and a proof of the formula will be a sorting algorithm. There is a literature devoted to methods for finding efficient programs in this way.

The Curry-Howard isomorphism has evolved with the invention of numerous typed λ-calculi and corresponding natural deduction logics, see [87, 55, 8, 41]. Other names for the isomorphism include *propositions-as-types, formula-as-types,* and *proofs-as-programs.*

4.6. Exercises

4.6.1. Exercise. Give derivations of the formulas (1),(3),(5),(7),(9),(11) from Section 2.2 using the natural deduction style of Section 4.1.

4.6.2. Exercise. Give λ-terms corresponding to the derivations from Exercise 4.6.1. Use the following rule for λ-terms corresponding to the ex-falso rule:

$$\frac{\Gamma \vdash M : \bot}{\Gamma \vdash \varepsilon_\varphi(M) : \varphi} .$$

4.6.3. Exercise. Prove Lemma 4.4.3.

4.6.4. Exercise.

1. Prove Newman's Lemma.

   Hint: Prove by induction on the length of the longest reduction sequence from M that $M \twoheadrightarrow M_1$ and $M \twoheadrightarrow M_2$ implies that there is an M_3 such that $M_1 \twoheadrightarrow M_3$ and $M_2 \twoheadrightarrow M_3$.

2. Prove Proposition 4.4.9.

3. Infer from Newman's Lemma Corollary 4.4.10.

4.6.5. Exercise. Prove Proposition 4.2.1(i) in detail.

4.6.6. Exercise. A *β-reduction strategy* is a map $F : \Lambda \to \Lambda$ such that $M \to_\beta F(M)$ if $M \notin \mathrm{NF}_\beta$, and $F(M) = M$ otherwise. Informally, a reduction strategy selects from any term not in normal form a redex and reduces that. For example, F_l is the reduction strategy that always reduces the left-most redex.

A reduction strategy F is *normalizing* if, for any weakly normalizing term M, there is an i such that[2]

$$M \to_\beta F(M) \to_\beta \ldots \to_\beta F^i(M) \in \mathrm{NF}_\beta$$

That is, if the term has a normal form, then repeated application of F eventually ends in the normal form. A classical result due to Curry and Feys states that F_l is normalizing.

A reduction strategy F is *perpetual* if, for any term M which is not strongly normalizing, there is no i such that

$$M \to_\beta F(M) \to_\beta \ldots \to_\beta F^i(M) \in \mathrm{NF}_\beta$$

That is, if the term has an infinite reduction, then repeated application of F yields an infinite reduction sequence.

Define $F_\infty : \Lambda \to \Lambda$ as follows. If $M \in \mathrm{NF}_\beta$ then $F_\infty(M) = M$; otherwise[3]

$$\begin{array}{lcll}
F_\infty(x\ \vec{P}\ Q\ \vec{R}) & = & x\ \vec{P}\ F_\infty(Q)\ \vec{R} & \text{If } \vec{P} \in \mathrm{NF}_\beta, Q \notin \mathrm{NF}_\beta \\
F_\infty(\lambda x.P) & = & \lambda x.F_\infty(P) & \\
F_\infty((\lambda x.P)\ Q\ \vec{R}) & = & P\{x := Q\}\ \vec{R} & \text{If } x \in \mathrm{FV}(P) \text{ or } Q \in \mathrm{NF}_\beta \\
F_\infty((\lambda x.P)\ Q\ \vec{R}) & = & (\lambda x.P)\ F_\infty(Q)\ \vec{R} & \text{If } x \notin \mathrm{FV}(P) \text{ and } Q \notin \mathrm{NF}_\beta
\end{array}$$

[2] As usual, $F^0(M) = M$ and $F^{i+1}(M) = F(F^i(M))$.

[3] By $\vec{P}$ we denote a finite, possibly empty, sequence of terms.

Show that F_∞ is perpetual.

Let Λ_I be the set of all λ-terms M such that any part $\lambda x.P$ of M satisfies $x \in \mathrm{FV}(P)$. For instance, $\mathbf{I} \in \Lambda_I$ and $\Omega \in \Lambda_I$ but $\mathbf{K} = \lambda x.\lambda y.x \notin \Lambda_I$. Show that for any $M \in \Lambda_I$: $M \in \mathrm{WN}_\beta$ iff $M \in \mathrm{SN}_\beta$.

Hint: Compare being weakly normalizing with F_l leading to a normal form, and compare begin strongly normalizing with F_∞ leading to a normal form. What is the relation between F_l and F_∞ on $M \in \Lambda_I$?

Since Λ_I is a subset of Λ, the elements of Λ_I that have a type in $\lambda{\to}$ (à la Curry) must correspond to a subset of all proofs in IPC($\to$). Which proofs are these?